Computational Biology Courses

02-201 Programming for Scientists
Fall and Spring: 10 units
Provides a practical introduction to programming for students with little or no prior programming experience who are interested in science. Fundamental scientific algorithms will be introduced, and extensive programming assignments will be based on analytical tasks that might be faced by scientists, such as parsing, simulation, and optimization. Principles of good software engineering will also be stressed. The course will introduce students to the Go programming language, an industry-supported, modern programming language, the syntax of which will be covered in depth. Other assignments will be given in other programming languages such as Python and Java to highlight the commonalities and differences between languages. No prior programming experience is assumed, and no biology background is needed. Analytical skills and mathematical maturity are required. Course not open to CS majors.

02-223 Personalized Medicine: Understanding Your Own Genome
Fall: 9 units
Do you want to know how to discover the tendencies hidden in your genome? Since the first draft of a human genome sequence became available at the start of this century, the cost of genome sequencing has decreased dramatically. Personal genome sequencing will likely become a routine part of medical exams for patients for prognostic and diagnostic purposes. Personal genome information will also play an increasing role in lifestyle choices, as people take into account their own genetic tendencies. Commercial services such as 23andMe have already taken first steps in this direction. Computational methods for mining large-scale genome data are being developed to unravel the genetic basis of diseases and assist doctors in clinics. This course introduces students to biological, computational, and ethical issues concerning use of personal genome information in health maintenance, medical practice, biomedical research, and policymaking. We focus on practical issues, using individual genome sequences (such as that of Nobel prize winner James Watson) and other population-level genome data. Without requiring any background in biology or CS, we begin with an overview of topics from genetics, molecular biology, stats, and machine learning relevant to the modern personal genome era. We then cover computational biology, including biological modeling and image analysis. It includes homework assignments requiring modification of scripts to perform computational analyses. The modeling component includes computer models of population dynamics, biochemical kinetics, cell pathways, neuron behavior, and stochastic simulations. The imaging component includes basics of machine vision, morphological image analysis, image classification and image-derived models. Lectures and examinations are joint with 03-250, but recitations are separate. Recitations for this course are intended primarily for computational biology majors as well as computer science, statistics or engineering majors at the undergraduate or graduate level who have had significant prior experience with computer science or programming. Students may not take both 02-250 and 03-250 for credit.
Prerequisites: 15-110 or 15-112 or 02-201

02-261 Quantitative Cell and Molecular Biology Laboratory
Fall and Spring: 9 units
This is an introductory laboratory-based course designed to teach basic biological laboratory skills used in exploring the quantitative nature of biological systems and the reasoning required for performing research in computational biology. Over the course of the semester, students will design and perform multiple modern experiments and quantitatively analyze the results of these experiments. During this course students will also have an opportunity to use techniques learned during the course to experimentally answer an open question. Designing the experiments will require students to think critically about the biological context of the experiments as well as the necessary controls to ensure interpretable experimental results. During this course students will gain experience in many aspects of scientific research, including: sequencing DNA, designing and performing PCR for a variety of analyses, maintaining cell cultures, taking brightfield and fluorescent microscopy images, developing methods for automated analysis of cell images, communicating results to peers and colleagues. As space is limited, laboratory sections will be small. Additional sections will be added to accommodate all students on the waitlist.
Course Outline: (1) 3-hour lab per week, (1) 1-hour lecture per week.

02-317 Algorithms in Nature
Intermittent: 9 units
Computer systems and biological processes often rely on networks of interacting entities to reach joint decisions, coordinate and respond to inputs. There are many similarities in the goals and strategies of biological and computational systems which suggest that each can learn from the other. These include the distributed nature of the networks (in biology molecules, cells, or organisms often operate without central control), the ability to successfully handle failures and attacks on a subset of the nodes, modularity and the ability to reuse certain components or sub-networks in multiple applications and the use of stochasticity in biology and randomized algorithms in computer science. In this course we will start by discussing classic biologically motivated algorithms including neural networks (inspired by the brain), genetic algorithms (sequence evolution), non-negative matrix factorization (signal processing in the brain), and search optimization (ant colony formation). We will then continue to discuss more recent bi-directional studies that have relied on biological processes to solve routing and synchronization problems, discover Maximal Independent Sets (MIS), and design robust and fault tolerant networks. In the second part of the class students will read and present new research in this area. Students will also work in groups on a final project in which they develop and test a new biologically inspired algorithm. No prior biological knowledge required.
Prerequisites: 15-210 and 15-251
Course Website: http://www.algorithmsinnature.org
02-319 Genomics and Epigenetics of the Brain
Fall: 9 units
This course will provide an introduction to genomics, epigenetics, and their application to problems in neuroscience. The rapid advances in genomic technology are in the process of revolutionizing how we conduct molecular biology research. These new techniques have given us an appreciation for the role that epigenetics modifications of the genome play in gene regulation, development, and inheritance. In this course, we will cover the biological basis of genomics and epigenetics, the basic computational tools to analyze genomic data, and the application of those tools to neuroscience. Through programming assignments and reading primary literature, the material will also serve to demonstrate important concepts in neuroscience, including the diversity of neural cell types, neural plasticity, the role that epigenetics plays in behavior, and how the brain is influenced by neuropsychiatric disorders. Although the course focuses on neuroscience, the material is accessible and applicable to a wide range of topics in biology. 02-250 is a suggested prerequisite.
Prerequisites: (03-151 or 03-121) and 03-220 and (15-110 or 02-201 or 15-121)

02-402 Computational Biology Seminar
Fall and Spring: 3 units
This course consists of weekly invited presentations on current computational biology research topics by leading scientists. Students will be expected to digest what they have learned in the seminar by writing short summaries on each speaker’s topic.

02-403 Special Topics in Bioinformatics and Computational Biology
Intermittent: 6 units
A decade ago, mass spectrometry (MS) was merely a qualitative research technique allowing the analysis of samples regarding the presence of specific biomolecules. However, as MS has turned quantitative, more sophisticated experiments can be performed, such as the recording of signal transduction kinetics and the analysis of the composition of protein complexes and organelles. This makes MS-based proteomics a powerful method to study spatiotemporal protein dynamics. The development of relative quantification approaches, which generally use 2H, 13C or 15N isotope labels, has especially led to an increase in quantitation accuracy and set off numerous new experimental approaches to study protein regulation. In this mini-course, we will cover mass spectrometry principles, discuss classical as well as current primary literature addressing method development and quantitative analysis, and highlight state-of-the-art biological studies that employ MS. A combination of lectures, student presentations, and written exercises will establish a thorough knowledge of current bio-analytical MS approaches.
Prerequisites: (02-250 Min. grade C or 03-250 Min. grade C) and 03-121 Min. grade C

02-421 Algorithms for Computational Structural Biology
Intermittent: 12 units
Some of the most interesting and difficult challenges in computational biology and bioinformatics arise from the determination, manipulation, or exploitation of molecular structures. This course will survey these challenges and present a variety of computational methods for addressing them. Topics will include: molecular dynamics simulations, computer-aided drug design, and computer-aided protein design. The course is appropriate for both students with backgrounds in computer science and those in the life sciences.

02-425 Computational Methods for Proteomics and Metabolomics
Spring: 9 units
Proteomics and metabolomics are the large scale study of proteins and metabolites, respectively. In contrast to genomes, proteomes and metabolomes vary with time and the specific stress or conditions an organism is under. Applications of proteomics and metabolomics include determination of protein and metabolite functions (including in immunology and neurobiology) and discovery of biomarkers for disease. These applications require advanced computational methods to analyze experimental measurements, create models from them, and integrate with information from diverse sources. This course specifically covers computational mass spectrometry, structural proteomics, proteogenomics, metabolomics, genome mining and metagenomics. Prerequisites: 02-250 or 02-604.
Prerequisites: 02-250 or 02-604

02-450 Automation of Biological Research: Robotics and Machine Learning
Fall: 9 units
Biology is increasingly becoming a “big data” science, as biomedical research has been revolutionized by automated methods for generating large amounts of data on diverse biological processes. Integration of data from many types of experiments is required to construct detailed, predictive models of cell, tissue or organism behaviors, and the complexity of the systems suggests that these models need to be constructed automatically. This requires iterative cycles of acquisition, analysis, modeling, and experimental design, since it is not feasible to do all possible biological experiments. This course will cover a range of automated biological research methods and a range of computational methods for automating the acquisition and interpretation of the data (especially active learning, proactive learning, compressed sensing and model structure learning). Grading will be based on class participation, homeworks, and a final project. The course is designed for undergraduate or graduate students with a wide variety of backgrounds. The course is intended to be self-contained but students may need to do some additional work to gain fluency in core concepts. Students should have a basic knowledge of biology, statistics, and programming. Experience with Machine Learning is useful but not mandatory.
Prerequisites: 15-122 and 10-401
Course Website: https://sites.google.com/site/automatonofbiologicalresearch/

02-499 Independent Study in Computational Biology
Fall and Spring
The student will, under the individual guidance of a faculty member, read and digest process papers or a textbook in an advanced area of computational biology not offered by an existing course at Carnegie Mellon. The student will demonstrate their mastery of the material by a combination of one or more of the following: oral discussions with the faculty member; exercises set by the faculty member accompanying the readings; and a written summary synthesizing the material that the student learned. Permission required.

02-500 Undergraduate Research in Computational Biology
Fall and Spring
This course is for undergraduate students who wish to do supervised research for academic credit with a Computational Biology faculty member. Interested students should first contact the Professor with whom they would like to work. If there is mutual interest, the Professor will direct you to the Academic Programs Coordinator and Asst Dept Head for Education.

02-510 Computational Genomics
Fall and Spring
Dramatic advances in experimental technology and computational analysis are fundamentally transforming the basic nature and goal of biological research. The emergence of new frontiers in biology, such as evolutionary genomics and systems biology, is demanding new methodologies that can confront quantitative issues of substantial computational and mathematical sophistication. In this course we will discuss classical approaches and latest methodological advances in the context of the following biological problems: 1) sequence analysis, focusing on gene finding and motifs detection, 2) analysis of high throughput molecular data, such as gene expression data, including normalization, clustering, pattern recognition and classification, 3) model, share and regulatory evolution, focusing on phylogenetic inference and regulatory network evolution, 4) population genetics, focusing on how genomes within a population evolve through recombination, mutation, and selection to create various structures in modern genomes and 5) systems biology, concerning how to combine diverse data types to make mechanistic inferences about biological processes. From the computational side this course focuses on modern machine learning methodologies for computational problems in molecular biology and genetics, including probabilistic modeling, inference and learning algorithms, data integration, time series analysis, active learning, etc. This course may be taken for 12 units, which requires completion of a course project, or for 9 units, which does not.
Prerequisites: 15-122 Min. grade C and (15-151 Min. grade C or 21-127 Min. grade C or 21-128 Min. grade C)
02-512 Computational Methods for Biological Modeling and Simulation
Fall: 9 units
This course covers a variety of computational methods important for modeling and simulation of biological systems. It is intended for graduates and advanced undergraduates with either biological or computational backgrounds who are interested in developing computer models and simulations of biological systems. The course will emphasize practical algorithms and algorithm design methods drawn from various disciplines of computer science and applied mathematics that are useful in biological applications. The general topics covered will be models for optimization problems, simulation and sampling, and parameter tuning. Course work will include problem sets with significant programming components and independent or group final projects.
Prerequisites: 02-201 or 15-112 or 15-110

02-514 String Algorithms
Fall: 12 units
Provides an in-depth look at modern algorithms used to process string data, particularly those relevant to genomics. The course will cover the design and analysis of efficient algorithms for processing enormous collections of strings. Topics will include string search; inexact matching; string compression; string data structures such as suffix trees, suffix arrays, and searchable compressed indices; and the Burrows-Wheeler transform. Applications of these techniques in biology will be presented, including genome assembly, transcript assembly, whole-genome alignment, gene expression quantification, read mapping, and search of large sequence databases. No knowledge of biology is assumed, and the topics covered will be of use in other fields involving large collections of strings. Programming proficiency is required.
Prerequisite: 15-251

02-518 Computational Medicine
Spring: 12 units
Modern medical research increasingly relies on the analysis of large patient datasets to enhance our understanding of human diseases. This course will focus on the computational problems that arise from studies of human diseases and the translation of research to the bedside to improve human health. The topics to be covered include computational strategies for advancing personalized medicine, pharmacogenomics for predicting individual drug responses, metagenomics for learning the role of the microbiome in human health, mining electronic medical records to identify disease phenotypes, and case studies in complex human diseases such as cancer and asthma. We will discuss how machine learning methodologies such as regression, classification, clustering, semi-supervised learning, probabilistic modeling, and time-series modeling are being used to analyze a variety of datasets collected by clinicians. Class sessions will consist of lectures, discussions of papers from the literature, and guest presentations by clinicians and other domain experts. Grading will be based on homework assignments and a project. 02-250 is a suggested pre-requisite.
Prerequisites: 10-401 or 10-601

02-530 Cell and Systems Modeling
Fall: 12 units
This course will introduce students to the theory and practice of modeling biological systems from the molecular to the organism level with an emphasis on intracellular processes. Topics covered include kinetic and equilibrium descriptions of biological processes, systematic approaches to model building and parameter estimation, analysis of biochemical circuits modeled as differential equations, modeling the effects of noise using stochastic methods, modeling spatial effects, and modeling at higher levels of abstraction or scale using logical or agent-based approaches. A range of biological models and applications will be considered including gene regulatory networks, cell signaling, and cell cycle regulation. Weekly lab sessions will provide students hands-on experience with methods and models presented in class. Course requirements include regular class participation, bi-weekly homework assignments, a take-home exam, and a final project.
Prerequisites: The course is designed for graduate and upper-level undergraduate students with a wide variety of backgrounds. The course is intended to be self-contained but students may need to do some additional work to gain fluency in core concepts. Students should have a basic knowledge of calculus, differential equations, and chemistry as well as some previous exposure to molecular biology and biochemistry. Experience with programming and numerical computation is useful but not mandatory. Laboratory exercises will use MATLAB as the primary modeling and computational tool augmented by additional software and tools.
Prerequisites: (03-121 or 03-151 or 33-121) and (03-231 or 03-232) and 21-112 and 09-105

02-601 Programming for Scientists
Fall and Spring: 12 units
Provides a practical introduction to programming for students with little or no prior programming experience who are interested in science. Fundamental scientific algorithms will be introduced, and extensive programming assignments will be based on analytical tasks that might be faced by scientists, such as parsing, simulation, and optimization. Principles of good software engineering will also be stressed, and students will have the opportunity to design their own programming project on a scientific topic of their choice. The course will introduce students to the Go programming language, an industry-supported, modern programming language, the syntax of which will be covered in depth. Other assignments will be given in other programming languages such as Python and Java to highlight the commonalities and differences between languages. No prior programming experience is assumed, and no biology background is needed. Analytical skills and mathematical maturity are required. Course not open to CS majors.

02-602 Professional Issues in Computational Biology
Fall and Spring: 1 unit
This course gives MS in Computational Biology students an opportunity to develop professional skills necessary for a successful career in computational biology. This course will include assistance with resume writing, interview preparation, presentation skills, and job search techniques. This course will also include opportunities to network with computational biology professionals and academic researchers. This course will meet once a week. This course is pass/fail only. Grading scheme will be discussed on first day of class.

02-604 Fundamentals of Bioinformatics
Spring: 12 units
How do we find potentially harmful mutations in your genome? How can we reconstruct the Tree of Life? How do we compare similar genes from different species? These are just three of the many central questions of modern biology that can only be answered using computational approaches. This 12-unit course will delve into some of the fundamental computational ideas used in biology and let students apply existing resources that are used in practice every day by thousands of biologists. The course offers an opportunity for students who possess an introductory programming background to become more experienced coders within a biological setting. As such, it presents a natural next course for students who have completed 02-250. 02-250 is a suggested pre-requisite for undergraduates.

02-613 Algorithms and Advanced Data Structures
Fall and Spring: 12 units
The objective of this course is to study algorithms for general computational problems, with a focus on the principles used to design those algorithms. Effective data structures will be discussed to support these algorithmic concepts. Topics include: Run time analysis, divide-and-conquer algorithms, dynamic programming algorithms, network flow algorithms, linear and integer programming, large-scale search algorithms and heuristics, efficient data storage and query, and NP-completeness. Although this course may have a few programming assignments, it is primarily not a programming course. Instead, it will focus on the design and analysis of algorithms for general classes of problems. This course is not open to CS graduate students who should consider taking 15-451 instead. 02-250 is a suggested prerequisite for undergraduates.

02-651 New Technologies and Future Markets
Fall: 12 units
This course focuses on technological trends and how these trends can help shape or disrupt new and existing markets. Students will learn to identify, analyze, and synthesize emerging trends and perform detailed research on these trends influencing trends and create markets. By understand the drivers behind these trends students will be able to identify key market opportunity inflection points in biotechnology as well as the relationship between business processes and information technology (IT). Students will also learn to assess some information technologies and the potential of applying them to solve problems and create commercially viable solutions. The course is designed for the student interested in finding new venture opportunities on the cutting edge of technology and finding and evaluating the opportunities for further development. For MS Biotechnology Innovation and Computation students only.
Prerequisite: 11-695
02-654 Biotechnology Enterprise Development
Fall: 12 units
In this course students learn how to develop a biotech start-up, create a Minimum Viable Product (MVP), business model and strategy for the product. Students will learn about business modeling, customer development, customer validation, proposal, product branding, and marketing for their product. The course will require students to spend most time to validate their start up concept and prototypes with potential customers and adapt to critical feedback and revise their respective value propositions accordingly. Students learn to balance technical product development with customer requirements, business strategy and budget constraints. This course provides real world, hands-on learning on what it is like to start a company. Different business modeling will be covered. By understand customer discovery and validation concepts will aid students to effectively modify their original concepts to meet market demands. Student teams will learn how to revise, improve their prototype by the end of the term. This is a fast paced course in which students are expected to spend most of the time outside of the classroom to interact with potential customers to validate, test, verify, and integrate essentials elements for their start-up business proposal. Up to now, students have been learning some technologies and methods for solving problems in the life science industry and build a prototype for their start-up. However, a new venture proposal is not a collection of isolated bits. It should be thorough validated via customer’s inputs and market needs to tell a single story of how the venture will reach its end goals. Final deliverable is creation and presentation of a well explicated, business proposal in addition to a product prototype corresponding to the business proposal. Prerequisites: 11-695 and 02-651

02-699 Independent Study in Computational Biology
Fall and Spring
The student will, under the individual guidance of a faculty member, read and digest process papers or a textbook in an advanced area of computational biology not offered by an existing course at Carnegie Mellon. The student will demonstrate their mastery of the material by a combination of one or more of the following: oral discussions with the faculty member; exercises set by the faculty member accompanying the readings; and a written summary synthesizing the material that the student learned. Permission required.

02-700 M.S. Thesis Research
Fall and Spring
This course is for M.S. students who wish to do supervised research for academic credit with a Computational Biology faculty member. Interested students should first contact the Professor with whom they would like to work. If there is mutual interest, the Professor will direct you to the Academic Programs Coordinator, who will enroll you in the course.

02-701 Current Topics in Computational Biology
Fall and Spring: 3 units
The course consists of weekly presentations by students and faculty on current topics in computational biology.

02-702 Computational Biology Seminar
Fall and Spring: 3 units
This course consists of weekly invited presentations on current computational biology research topics by leading scientists. Attendance is mandatory for a passing grade.

02-703 Special Topics in Bioinformatics and Computational Biology
Intermittent: 6 units
A decade ago, mass spectrometry (MS) was merely a qualitative research technique allowing the analysis of samples regarding the presence of specific biomolecules. However, as MS has turned quantitative, more sophisticated experiments can be performed, such as the recording of signal transduction kinetics and the analysis of the composition of protein complexes and organelles. This makes MS-based proteomics a powerful method to study spatiotemporal protein dynamics. The development of relative quantification approaches, which generally use 2H, 13C or 15N isotope labels, has especially led to an increase in quantification accuracy and set off numerous new experimental approaches to study protein regulation. In this mini-course, we will cover mass spectrometry principles, discuss classical as well as current primary literature addressing method development and quantitative analysis, and highlight state-of-the-art biological studies that employ MS. A combination of lectures, student presentations, and written exercises will establish a thorough knowledge of current bio-analytical MS approaches.

02-710 Computational Genomics
Spring: 12 units
Dramatic advances in experimental technology and computational analysis are fundamentally transforming the basic nature and goal of biological research. The emergence of new frontiers in biology, such as evolutionary genomics and systems biology is demanding new methodologies that can confront quantitative issues of substantial computational and mathematical sophistication. In this course we will discuss classical approaches and latest methodological advances in the context of the following biological problems: 1) sequence analysis, focusing on gene finding and motifs detection, 2) analysis of high throughput molecular data, such as gene expression data, including normalization, clustering, pattern recognition and classification, 3) molecular and regulatory evolution, focusing on phylogenetic inference and regulatory network evolution, 4) population genetics, focusing on how genomes within a population evolve through recombination, mutation, and selection to create various structures in modern genomes and 5) systems biology, concerning how to combine diverse data types to make mechanistic inferences about biological processes. From the computational side this course focuses on modern machine learning methodologies for computational problems in molecular biology and genetics, including probabilistic modeling, inference and learning algorithms, data integration, time series analysis, active learning, etc.

02-711 Computational Molecular Biology and Genomics
Spring: 12 units
An advanced introduction to computational molecular biology, using an applied algorithms approach. The first part of the course will cover established algorithmic methods, including pairwise sequence alignment and dynamic programming, multiple sequence alignment, fast database search heuristics, hidden Markov models for molecular motifs and phylogeny reconstruction. The second part of the course will explore emerging computational problems driven by the newest genomic research. Course work includes four to six problem sets, one midterm and final exam. Prerequisites: (03-151 or 03-121) and 15-122

02-712 Computational Methods for Biological Modeling and Simulation
Fall: 12 units
This course covers a variety of computational methods important for modeling and simulation of biological systems. It is intended for graduates and advanced undergraduates with either biological or computational backgrounds who are interested in developing computer models and simulations of biological systems. The course will emphasize practical algorithms and algorithm design methods drawn from various disciplines of computer science and applied mathematics that are useful in biological applications. The general topics covered will be models for optimization problems, simulation and sampling, and parameter tuning. Course work will include problem sets with significant programming components and independent or group final projects. Prerequisites: (15-110 or 15-112) and (02-613 or 02-201)

02-714 String Algorithms
Fall: 12 units
Provides an in-depth look at modern algorithms used to process string data, particularly those relevant to genomics. The course will cover the design and analysis of efficient algorithms for processing enormous collections of strings. Topics will include string search; inexact matching; string compression; string data structures such as suffix trees, suffix arrays, and searchable compressed indices; and the Borrows-Wheeler transform. Applications of these techniques in biology will be presented, including genome assembly, transcript assembly, whole-genome alignment, gene expression quantification, read mapping, and search of large sequence databases. No knowledge of biology is assumed, and the topics covered will be of use in other fields involving large collections of strings. Programming proficiency is required. Prerequisite: 15-251
02-717 Algorithms in Nature
Fall: 12 units
Computer systems and biological processes often rely on networks of interacting entities to reach joint decisions, coordinate and respond to inputs. There are many similarities in the goals and strategies of biological and computational systems which suggest that much can be learned from the other. These include the distributed nature of the networks (in biology molecules, cells, or organisms often operate without central control), the ability to successfully handle failures and attacks on a subset of the nodes, modularity and the ability to reuse certain components or sub-networks in multiple applications and the use of stochasticity in biology and randomized algorithms in computer science. In this course we will start by discussing classic biologically motivated algorithms including neural networks (inspired by the brain), genetic algorithms (sequence evolution), non-negative matrix factorization (signal processing in the brain), and search optimization (ant colony formation). We will then continue to discuss more recent bi-directional studies that have relied on biological processes to solve routing and synchronization problems, discover Maximal Independent Sets (MIS), and design robust and fault tolerant networks. In the second part of the class students will read and present new research in this area. Students will also work in groups on a final project in which they develop and test a new biologically inspired algorithm. See also: www.algorithmsinnature.org no prior biological knowledge required.

02-718 Computational Medicine
Spring: 12 units
Modern medical research increasingly relies on the analysis of large patient datasets to enhance our understanding of human diseases. This course will focus on the computational problems that arise from studies of human diseases and the translation of research to the bedside to improve human health. The topics to be covered include computational strategies for advancing personalized medicine, pharmacogenomics for predicting individual drug responses, metagenomics for learning the role of the microbiome in human health, mining electronic medical records to identify disease phenotypes, and case studies in complex human diseases such as cancer and asthma. We will discuss how machine learning methodologies such as regression, classification, clustering, semi-supervised learning, probabilistic modeling, and time-series modeling are being used to analyze a variety of datasets collected by clinicians. Class sessions will consist of lectures, discussions of papers from the literature, and guest presentations by clinicians and other domain experts. Grading will be based on homework assignments and a project. 02-250 is a suggested pre-requisite. Prerequisites: 10-401 or (10-601 and 10-701)

02-719 Genomics and Epigenetics of the Brain
Fall: 12 units
This course will provide an introduction to genomics, epigenetics, and their application to problems in neuroscience. The rapid advances in genomic technology are in the process of revolutionizing how we conduct molecular biology research. These new techniques have given us an appreciation for the role that epigenetics modifications of the genome play in gene regulation, development, and inheritance. In this course, we will cover the biological basis of genomics and epigenetics, the basic computational tools to analyze genomic data, and the application of those tools to neuroscience. Through programming assignments and reading primary literature, the material will also serve to demonstrate neuroscience, including the diversity of neural cell types, neural plasticity, the role that epigenetics plays in behavior, and how the brain is influenced by neurological and psychiatric disorders. Although the course focuses on neuroscience, the material is accessible and applicable to a wide range of topics in biology. 02-250 is a suggested pre-requisite. Prerequisites: (03-151 or 03-121) and 03-220 and (15-110 or 02-201 or 15-121)

02-721 Algorithms for Computational Structural Biology
Intermittent: 12 units
Some of the most interesting and difficult challenges in computational biology and bioinformatics arise from the determination, manipulation or exploitation of molecular structures. This course will survey these challenges and present a variety of computational methods for addressing them. Topics will include: molecular dynamics simulations, computer-aided drug design, and computer-aided protein design. The course is appropriate for both students with backgrounds in computer science and those in the life sciences.

02-725 Computational Methods for Proteomics and Metabolomics
Spring: 12 units
Proteomics and metabolomics are the large scale study of proteins and metabolites, respectively. In contrast to genomes, proteomes and metabolomes vary with time and the specific stress or conditions an organism is under. Applications of proteomics and metabolomics include determination of protein and metabolite functions (including in immunology and neurobiology) and discovery of biomarkers for disease. These applications require advanced computational methods to analyze experimental measurements, create models from them, and integrate with information from diverse sources. This course specifically covers computational mass spectrometry, structural proteomics, proteogenomics, metabolomics, genome mining and metagenomics. Prerequisites: 02-250 or 02-604. Prerequisites: 02-604 or 02-250

02-730 Cell and Systems Modeling
Fall: 12 units
This course will introduce students to the theory and practice of modeling biological systems from the molecular to the organism level with an emphasis on intracellular processes. Topics covered include kinetic and equilibrium descriptions of biological processes, systematic approaches to model building and parameter estimation, analysis of biochemical circuits modeled as differential equations, modeling the effects of noise using stochastic methods, modeling spatial effects, and modeling at higher levels of abstraction or scale using logical or agent-based approaches. A range of biological models and applications will be considered including gene regulatory networks, cell signaling, and cell cycle regulation. Weekly lab sessions will provide students hands-on experience with methods and models presented in class. Course requirements include regular class participation, bi-weekly homework assignments, a take-home exam, and a final project. Prerequisites: The course is designed for graduate and upper-level undergraduate students with a wide variety of backgrounds. The course is intended to be self-contained but students may need to do some additional work to gain fluency in core concepts. Students should have a basic knowledge of calculus, differential equations, and chemistry as well as some previous exposure to molecular biology and biochemistry. Experience with programming and numerical computation is useful but not mandatory. Laboratory exercises will use MATLAB as the primary modeling and computational tool augmented by additional software as needed. Prerequisites: (33-121 or 03-151 or 03-121) and (03-232 or 03-231) and 21-112 and 09-105

02-740 Bioimage Informatics
Spring: 12 units
The goals of this course are to provide students with the following: the ability to use mathematical techniques such as linear algebra. Fourier theory and sampling in more advanced signal processing settings; fundamentals of multiresolution and wavelet techniques; and in-depth coverage of some bioimaging applications such as compression and denoising. Upon successful completion of this course, the student will be able to explain the importance and use of signal representations in building more sophisticated signal processing tools, such as wavelets; think in basic time-frequency terms; describe how Fourier theory fits in a bigger picture of signal representations; use basic multirate building blocks, such as a two-channel filter bank; characterize the discrete wavelet transform and its variations; construct a time-frequency decomposition to fit a given signal; explain how these tools are used in various applications; and apply these concepts to solve a practical bioimaging problem through an independent project. Pre-requisite: 18-791, or permission of instructor. (Also known as 18-799) 02-250 is a suggested pre-requisite. Prerequisites: 18-791 or 18-799
02-750 Automation of Biological Research: Robotics and Machine Learning
Fall: 12 units
This course is for Ph.D students doing supervised research for academic credit.

Course Website: https://sites.google.com/site/automatonofbiologicalresearch/?pli=1

Prerequisites: 10-601 or 10-701

02-760 Laboratory Methods for Computational Biologists
Spring: 6 units
This course is designed for graduate and upper-level undergraduate students with a wide variety of backgrounds. The course is intended to be self-contained but students may need to do some additional work to gain fluency in core concepts. Students should have a basic knowledge of biology, statistics, and programming.

02-801 Computational Biology Internship
Fall and Summer: 3 units
This course is for students participating in an internship or co-op.

02-900 Ph.D. Thesis Research
All Semesters
This course is for Ph.D students doing supervised research for academic credit.

Computer Science Courses

15-050 Study Abroad
All Semesters
Students who are interested in studying abroad should first contact the Office of International Education. More information on Study Abroad is available on OIE's Study Abroad page and at the CS Undergraduate Office.

15-075 Computer Science Co-Op
Fall and Spring
This course is for Computer Science students who wish to have an internship experience as part of their curriculum. Students are required to write a one-page summary statement prior to registration that explains how their internship connects with their CS curriculum, specifically on how it uses material they have learned as well as prepares them for future courses. Near the end of the internship, students will be required to submit a reflection paper that describes the work they did in more detail, including lessons learned about the work experience and how they utilized their CS education to work effectively. International students should consult with the Office of International Education for appropriate paperwork and additional requirements before registration. Units earned count toward the total required units necessary for degree completion; students should speak with an academic advisor for details. This course may be taken at most 3 times for a total of 9 units maximum. Students normally register for this course for use during the summer semester.

15-090 Computer Science Practicum
Summer: 3 units
This course is for Computer Science students who wish to have an internship experience as part of their curriculum. Students are required to write a one-page summary statement prior to registration that explains how their internship connects with their CS curriculum, specifically on how it uses material they have learned as well as prepares them for future courses. Near the end of the internship, students will be required to submit a reflection paper that describes the work they did in more detail, including lessons learned about the work experience and how they utilized their CS education to work effectively. International students should consult with the Office of International Education for appropriate paperwork and additional requirements before registration. Units earned count toward the total required units necessary for degree completion; students should speak with an academic advisor for details. This course may be taken at most 3 times for a total of 9 units maximum. Students normally register for this course for use during the summer semester.

15-104 Introduction to Computing for Creative Practice
Fall: 10 units
[IDEAté portal course] An introduction to fundamental computing principles and programming techniques for creative cultural practices, with special consideration to applications in music, design and the visual arts. Intended for students with little to no prior programming experience, the course develops skills and understanding of text-based programming in a procedural style, including idioms of sequencing, selection, iteration, and recursion. Topics include data organization (arrays, files, trees), interfaces and abstraction (modular software design, using sensor data and software libraries), basic algorithms (searching and sorting), and computational principles (randomness, concurrency, complexity). Intended for students following an IDEAté concentration or minor who have not taken 15-112.

15-110 Principles of Computing
All Semesters: 10 units
A course in fundamental computing principles for students with minimal or no computing background. Programming constructs: sequencing, selection, iteration, and recursion. Data organization: arrays and lists. Use of abstraction in computing: data representation, computer organization, computer networks, functional decomposition, and application programming interfaces. Use of computational principles in problem-solving: divide and conquer, randomness, and concurrency. Classification of computational problems based on complexity, non-computable functions, and using heuristics to find reasonable solutions to complex problems. Social, ethical and legal issues associated with the development of new computational artifacts will also be discussed.

15-112 Fundamentals of Programming and Computer Science
All Semesters: 12 units
A technical introduction to the fundamentals of programming with an emphasis on producing clear, robust, and reasonably efficient code using top-down design, informal analysis, and effective testing and debugging. Starting from first principles, we will cover a large subset of the Python programming language, including its standard libraries and programming paradigms. We will also target numerous deployment scenarios, including standalone programs, shell scripts, and web-based applications. This course assumes no prior programming experience. Even so, it is a fast-paced and rigorous preparation for 15-122. Students seeking a more gentle introduction to computer science should consider first taking 15-110. NOTE: students must achieve a C or better in order to use this course to satisfy the pre-requisite for any subsequent Computer Science course.

15-121 Introduction to Data Structures
Fall: 10 units
A continuation of the process of program design and analysis for students with some prior programming experience (functions, loops, and arrays, not necessarily in Java). The course reinforces object-oriented programming techniques in Java and covers data aggregates, data structures (e.g., linked lists, stacks, queues, trees, and graphs), and an introduction to the analysis of algorithms that operate on those data structures.

Prerequisite: 15-112
15-122 Principles of Imperative Computation
Fall and Spring: 10 units
For students with a basic understanding of programming (variables, expressions, loops, arrays, functions). Teaches imperative programming and methods for ensuring the correctness of programs. Students will learn the process and concepts needed to go from high-level descriptions of algorithms to correct imperative implementations, with specific application to basic data structures and algorithms. Much of the course will be conducted in a subset of C amenable to verification, with a transition to full C near the end. This course prepares students for 15-213 and 15-210. NOTE: students must achieve a C or better in order to use this course to satisfy the pre-requisite for any subsequent Computer Science course.
Prerequisite: 15-112 Min. grade C
Course Website: http://www.cs.cmu.edu/~15122

15-128 Freshman Immigration Course
Fall: 1 unit
The Freshman Immigration Course is taken by first-semester Computer Science majors on the Pittsburgh campus. The course is designed to acquaint incoming majors with computer science at CMU. Talks range from historical perspectives in the field to descriptions of the cutting edge research being conducted in the School of Computer Science. Enrollment is limited to SCS Freshmen ONLY.

15-129 Freshman Immigration II
Fall: 3 units
This course is ONLY offered at Carnegie Mellon in Qatar. Students and instructors will solve different problems each week by searching the Web and other likely places for answers. The problems will be submitted by other faculty who will grade the quality of the answers. Students will learn strategies and techniques for finding information on the Web more efficiently; learn when to start with a search engine, a subject-oriented directory, or other tools; explore and practice using advanced search syntax for major search engines; experience specialized search engines for images, sound, multimedia, newsgroups, and discussion lists as well as subject-specific search engines; discover valuable resources to help keep you up-to-date in this fast-changing environment.

15-131 Great Practical Ideas for Computer Scientists
Fall: 2 units
THIS COURSE IS OPEN TO CS FRESHMAN ONLY. Throughout your education as a Computer Scientist at Carnegie Mellon, you will take courses on programming, theoretical ideas, logic, systems, etc. As you progress, you will be expected to pick up the so-called “tools of the trade.” This course is intended to help you learn what you need to know in a friendly, low-stress, high-support way. We will discuss UNIX, LaTeX, debugging and many other essential tools. Laptop required. (Laptops will be available for those without their own laptops.)

15-150 Principles of Functional Programming
Fall and Spring: 10 units
An introduction to programming based on a “functional” model of computation. The functional model is a natural generalization of algebra. It describes programs as formulas that describe the output of a computation in terms of its inputs—that is, as a function. But instead of being confined to real- or complex-valued functions, the functional model extends the algebraic view to a very rich class of data types, including not only aggregates built up from other types, but also functions themselves as values. This course is an introduction to programming that is focused on the central concepts of function and type. One major theme is the interplay between inductive types, which are built up incrementally; recursive functions, which compute over inductive types by decomposition; and proof by structural induction, which is used to prove the correctness and time complexity of a recursive function. Another major theme is the role of types in structuring large programs into separate modules, and the integration of imperative programming through the introduction of data types whose values may be altered during computation. NOTE: students must achieve a C or better in order to use this course to satisfy the pre-requisite for any subsequent Computer Science course.
Prerequisites: (21-127 Min. grade C or 15-151 Min. grade C) and 15-112 Min. grade C

15-151 Mathematical Foundations for Computer Science
Fall: 10 units
*CS majors only* This course is offered to incoming Computer Science freshmen and focuses on the fundamental concepts in Mathematics that are of particular interest to Computer Science such as logic, sets, induction, functions, and combinatorics. These topics are used as a context in which students learn to formalize arguments using the methods of mathematical proof. This course uses experimentation and collaboration as ways to gain better understanding of the material. Open to CS freshmen only. NOTE: students must achieve a C or better in order to use this course to satisfy the pre-requisite for any subsequent Computer Science course.

15-199 Special Topics: Discovering Logic
Intermittent: 3 units
This course is ONLY offered at Carnegie Mellon in Qatar. This course has the purpose of introducing first-year Computer Science students to elements of formal logic as well as to the historical context in which this discipline developed. As all subsequent courses in the CS curriculum rely on students having mastered basic logical notions and skills, it will test and enhance your preparation, thereby putting you in a better position to succeed in the program. It will also help you understand and appreciate how CS came about since Computer Science grew out of logic. The specific knowledge and skills you will learn in is course include: an enhanced ability to research topics, give presentations and write technical prose, some elementary logic, some historical depth into Computer Science, mathematics and logic itself. This course is open to Computer Science freshmen only.

15-210 Parallel and Sequential Data Structures and Algorithms
Fall and Spring: 12 units
Teaches students about how to design, analyze, and program algorithms and data structures. The course emphasizes parallel algorithms and analysis, and how sequential algorithms can be considered a special case. The course goes into more theoretical content on algorithm analysis than 15-122 and 15-150 while still including a significant programming component and covering a variety of practical applications such as problems in data analysis, graphics, text processing, and the computational sciences. NOTE: students must achieve a C or better in order to use this course to satisfy the pre-requisite for any subsequent Computer Science course.
Prerequisites: 15-150 Min. grade C and 15-122 Min. grade C

15-213 Introduction to Computer Systems
Fall and Spring: 12 units
This course provides a programmer’s view of how computer systems execute programs, store information, and communicate. It enables students to become more effective programmers, especially in dealing with issues of performance, portability and robustness. It also serves as a foundation for courses on compilers, networks, operating systems, and computer architecture, where a deeper understanding of systems-level issues is required. Topics covered include: machine-level code and its generation by optimizing compilers, performance evaluation and optimization, computer arithmetic, memory organization and management, networking technology and protocols, and supporting concurrent computation. NOTE FOR GRADUATE STUDENTS: This course is not open to graduate students beginning Spring 2015. Graduate students must register for 15-513 instead.
Prerequisite: 15-122 Min. grade C

15-214 Principles of Software Construction: Objects, Design, and Concurrency
Fall and Spring: 12 units
Software engineers today are less likely to design data structures and algorithms from scratch and more likely to build systems from library and framework components. In this course, students engage with concepts related to the construction of software systems at scale, building on their understanding of the basic building blocks of data structures, algorithms, program structures, and computer structures. The course covers technical topics in four areas: (1) concepts of design for complex systems, (2) object oriented programming, (3) static and dynamic analysis for programs, and (4) concurrent and distributed software. Student assignments involve engagement with complex software such as distributed massively multiplayer game systems and frameworks for graphical user interaction.
Prerequisites: (15-122 Min. grade C or 15-121 Min. grade C) and (21-127 Min. grade C or 15-151 Min. grade C)
15-237 Special Topic: Cross-Platform Mobile Web Apps
Intermittent: 12 units
An introduction to writing cross-platform mobile web apps. Using a tool chain based on HTML5, CSS3, JavaScript, and a variety of supporting frameworks, we will write apps that are effectively designed both for desktop and mobile browsers, and which can be converted into native apps for Android, iOS, and Windows Phone 7 devices. Additional topics will include designing user interfaces for mobile devices, accessing mobile device APIs (such as accelerometer, GPS, compass, or camera), and power management issues. While this course focuses on browser-side technologies, we will briefly explore JavaScript-based server-side technologies (though students should consider 15-437 for extensive treatment of server-side topics). Note that we will not be writing native apps in Objective-C for iOS nor in Java for Android, though we may include some brief exposure to these technologies near the end of the course.
Prerequisite: 15-112 Min. grade C

15-251 Great Ideas in Theoretical Computer Science
Fall and Spring: 12 units
This course is about how to use theoretical ideas to formulate and solve problems in computer science. It integrates mathematical material with general problem solving techniques and computer science applications. Examples are drawn from algorithms, complexity theory, game theory, probability theory, graph theory, automata theory, algebra, cryptography, and combinatorics. Assignments involve both mathematical proofs and programming. NOTE: students must achieve a C or better in order to use this course to satisfy the pre-requisite for any subsequent Computer Science course.
Prerequisites: (15-122 Min. grade C or 15-150 Min. grade C) and (15-151 Min. grade C or 21-127 Min. grade C)
Course Website: http://www.cs.cmu.edu/~15251

15-294 Special Topic: Rapid Prototyping Technologies
Intermittent: 5 units
This mini-course introduces students to rapid prototyping technologies with a focus on laser cutting and 3D printing. The course has three components: 1) A survey of rapid prototyping and additive manufacturing technologies, the maker and open source movements, and societal impacts of these technologies; 2) An introduction to the computer science behind these technologies: CAD tools, file formats, slicing algorithms; 3) Hands-on experience with SolidWorks, laser cutting, and 3D printing, culminating in student projects (i.e. artistic creations, functional objects, replicas of famous calculating machines, etc.). Please note that there will be a usage/materials fee for this course.
Prerequisites: 15-112 Min. grade C or 15-104 Min. grade C

15-295 Competition Programming and Problem Solving
Fall and Spring: 5 units
Each year, Carnegie Mellon fields two teams for participation in the ACM-ICPC Regional Programming Contest. During recent years, those teams have earned the right to represent Carnegie Mellon at the ACM-ICPC World Finals. This course is a vehicle for those who consistently and rigorously train in preparation for the contests to earn course credit for their effort and achievement. Preparation involves the study of algorithms, the practice of programming and debugging, the development of test sets, and the growth of team, communication, and problem solving skills. Neither the course grade nor the number of units earned are dependent on ranking in any contest. Students are not required to earn course credit to participate in practices or to compete in ACM-ICPC events.
Prerequisite: 15-122 Min. grade C

15-300 Research and Innovation in Computer Science
Fall: 9 units
This Fall course is the first part of a two-course sequence that is designed to help prepare students to invent the future state-of-the-art in the field of computer science. Course topics will include the following: an overview of important things to know about how research and innovation works in the field of computer science; a survey of the current cutting-edge of research in the field, both here at Carnegie Mellon and elsewhere; critical thinking skills when reading research publications that disagree with each other; strategies for coping with open-ended problems; and technical communication skills for computer scientists. Students will also match up with a faculty mentor for a potential Technology Innovation Project (to be performed in the Spring), put together a detailed plan of attack for that project, and start to get up to speed (including background reading, etc.). This course can be used to satisfy the Technical Communications requirement for the CS major.
Prerequisites: 15-213 Min. grade C and 15-210 Min. grade C and 76-101 Min. grade C and 15-251 Min. grade C

15-312 Foundations of Programming Languages
Spring: 12 units
This course discusses in depth many of the concepts underlying the design, definition, implementation, and use of modern programming languages. Formal approaches to defining the syntax and semantics are used to describe the fundamental concepts underlying programming languages. A variety of programming paradigms are covered such as imperative, functional, logic, and concurrent programming. In addition to the formal studies, experience with programming in the languages is used to illustrate how different design goals can lead to radically different languages and models of computation.
Prerequisites: 15-251 Min. grade C and 15-210 Min. grade C

15-313 Foundations of Software Engineering
Fall: 12 units
Students gain exposure to the fundamentals of modern software engineering. This includes both core CS technical knowledge and the means by which this knowledge can be applied in the practical engineering of complex software. Topics related to software artifacts include design models, patterns, coding, static and dynamic analysis, testing and inspection, measurement, and software architecture and frameworks. Topics related to software process include modeling, requirements engineering, process models and evaluation, team development, and supply chain issues including outsourcing and open source. This course has a strong technical focus, and will include both written and programming assignments. Students will get experience with modern software engineering tools.
Prerequisite: 15-214

15-314 Programming Language Semantics
Spring: 12 units
This course is designed for advanced undergraduates with interests in the mathematical and logical foundations of programming languages. The course introduces the foundational concepts and fundamental techniques of the most prominent and successful approaches to programming language semantics that have been developed. Broadly speaking, semantics is concerned with the provision of mathematical meanings to programs, at an appropriate level of abstraction, to allow formalization of program behavior and facilitate proofs of correctness. Our aim is to demonstrate the utility of a scientific approach to programming and languages. We focus on the most important and most general frameworks for semantic description: denotational and operational semantics. These frameworks are widely applicable and offer complementary approaches to language definition, with various advantages. We also discuss formal specifications, and logics of program correctness. We make extensive use of mathematical and structural induction, and computational induction. We use semantics to describe program behavior, guide the development of correct programs, specify and prove the correctness of a compiler, validate program logics, and derive laws of program equivalence. We discuss imperative and functional languages, sequential and parallel, high-level and low-level, as time permits.
Prerequisites: 15-150 Min. grade C and 15-251 Min. grade C

15-316 Software Foundations of Security and Privacy
Spring: 9 units
Security and privacy issues in computer systems continue to be a pervasive issue in technology and society. Understanding the security and privacy needs of software, and being able to rigorously demonstrate that those needs are met, is key to eliminating vulnerabilities that cause these issues. Students who take this course will learn the principles needed to make these assurances about software, and some of the key strategies used to make sure that they are correctly implemented in practice. Topics include: policy models and mechanisms for confidentiality, integrity, and availability; language-based techniques for detecting and preventing security threats, mechanisms for enforcing privacy guarantees, and the interaction between software and underlying systems that can give rise to practical security threats. Students will also gain experience applying many of these techniques to write code that is secure by construction.
Prerequisite: 15-213 Min. grade C

15-317 Constructive Logic
Fall: 9 units
This interdisciplinary junior-level course is designed to provide a thorough introduction to modern constructive logic, its roots in philosophy, its numerous applications in computer science, and its mathematical properties. Some of the topics to be covered are intuitionistic logic, inductive definitions, functional programming, type theory, realizability, connections between classical and constructive logic, decidable classes.
Prerequisite: 15-150 Min. grade C
Course Website: http://symbolaris.com/course/constlog16.html
15-319 Cloud Computing
Fall and Spring: 9 units
This course gives students an overview of Cloud Computing, which is the delivery of computing as a service over a network, whereby distributed resources are rented, rather than owned, by an end user as a utility. Students will study its enabling technologies, building blocks, and gain hands-on experience through projects utilizing public cloud infrastructures. Cloud computing services are widely adopted by many organizations across domains. The course will introduce the cloud and cover the topics of data centers, software stack, virtualization, software defined networks and storage, cloud storage, and programming models. We will start by discussing the clouds motivating factors, benefits, challenges, service models, SLAs and security. We will discuss several concepts behind data center design and management, which enable the economic and technological benefits of the cloud paradigm. Next, we will study how CPU, memory and I/O resources, network (SDN) and storage (SDS) are virtualized, and the key role of virtualization to enable the cloud. Subsequently, students will study cloud storage concepts like data distribution, durability, consistency and redundancy. We will discuss distributed file systems, NoSQL databases and object storage using HDF5, CephFS, HBASE, MongoDB, Cassandra, DynamoDB, S3, and Swift as case studies. Finally, students will study the MapReduce, Spark and GraphLab programming models. Students will work with Amazon Web Services and Microsoft Azure, to rent and provision compute resources and then program and deploy applications using these resources. Students will develop and evaluate scaling and load balancing solutions, work with cloud storage systems, and develop applications in several programming paradigms. 15619 students must complete an extra team project which entails designing and implementing a cost- and performance-sensitive web-service for querying big data.
Prerequisite: 15-213 Min. grade C

15-322 Introduction to Computer Music
Fall: 9 units
Computers are used to synthesize sound, process signals, and compose music. Personal computers have replaced studios full of sound recording and processing equipment, completing a revolution that began with recording and electronics. In this course, students will learn the fundamentals of digital audio, basic sound synthesis algorithms, and techniques for digital audio effects and processing. Students will apply their knowledge in programming assignments using a very high-level programming language for sound synthesis and composition. In a final project, students will demonstrate their mastery of tools and techniques through music composition or by the implementation of a significant sound-processing technique.
Prerequisite: 15-112 Min. grade C

15-323 Computer Music Systems and Information Processing
Spring: 9 units
This course presents concepts and techniques for representing and manipulating discrete music information, both in real time and off line. Representations of music as explicitly timed event sequences will be introduced, and students will learn how to build efficient run-time systems for event scheduling, tempo control, and interactive processing. The MIDI protocol is used to capture real-time performance information and to generate sound. The course will also cover non-real-time processing of music data, including Markov models, style recognition, computer accompaniment, query-by-humming, and algorithmic composition. This course is independent of, and complementary to 15-322, Introduction to Computer Music, which focuses on sound synthesis and signal processing.
Prerequisite: 15-122 Min. grade C

15-346 Special Topic: Perspectives in Computer Architecture
Intermittent: 6 units
This course is ONLY offered at Carnegie Mellon in Qatar. This course will provide various perspectives in the field of computer architecture by world renowned scientists. The course will bring together basic architecture principles and designs of uniprocessor and multicore computers. First, we will introduce the Von-Neuman architecture and the major hardware components of a modern computer. Students will learn how to describe an architecture design using a Hardware Description Language such as Verilog. Second, we will learn basic architectural techniques including instruction level parallelism, pipelining and cache memory. An overview of microprocessor architectures, specifically on how they differ from uniprocessor ones, the promises they offer, and the serious challenges they bring, will be provided. We will also discuss cache organization techniques in multicore. Students will gain insight into the dominating principles that dominated past processor architectures and how they will continue to change for future processor design targeting emerging technologies such as mobile and wearable computing. The concepts delivered in the lectures will be reinforced and extended through student presentations on multiple directions in computer architecture.
Prerequisite: 15-213

15-348 Embedded Systems
Spring: 9 units
This course is offered only at Carnegie Mellon's campus in Qatar. This course covers the broad range of foundational skills that apply across all embedded computer system application areas, from thermostats to self-driving vehicles. The emphasis is at the layer where hardware meets software. Topics include microcontroller hardware, assembly language, embedded C programming, analog I/O, timers, code optimization, interrupts, and concurrency. Real world engineering practices, constraints, and example applications are integrated throughout the course. Weekly hands-on hardware and software experiences with an industry-strength automotive embedded controller are coordinated with the lecture content to reinforce core skills.
Prerequisite: 15-122

15-349 Introduction to Computer and Network Security
Fall: 9 units
This course is ONLY offered at Carnegie Mellon in Qatar. This course is meant to offer Computer Science undergraduate students in their junior or senior year a broad overview of the field of computer security. Students will learn the basic concepts in computer security including software vulnerability analysis and defense, networking and wireless security, applied cryptography, as well as ethical, legal, social and economic facets of security. Students will also learn the fundamental methodology for how to design and analyze security critical systems.
Prerequisite: 15-122

15-351 Algorithms and Advanced Data Structures
Fall: 12 units
The goal of this course is to investigate the relationship between algebra and computation. The course is designed to expose students to algorithms used for symbolic computation, as well as to the concepts from modern algebra which are applied to the development of these algorithms. This course provides a hands-on introduction to many of the most important ideas used in symbolic mathematical computation, which involves solving systems of polynomial equations (via Groebner bases), analytic integration, and solving linear difference equations. Throughout the course the computer algebra system Mathematica will be used for computation.
Prerequisites: 15-251 Min. grade C or 21-228 Min. grade C
Course Website: http://www.cs.cmu.edu/~cdm/

15-355 Modern Computer Algebra
Spring: 9 units
This course is about the computational aspects of some of the standard concepts of discrete mathematics (relations, functions, logic, graphs, algebra, automata), with emphasis on efficient algorithms. We begin with a brief introduction to computability and computational complexity. Other topics include: iteration, orbits and fixed points, order and equivalence relations, propositional logic and satisfiability testing, finite fields and shift register sequences, finite state machines, and cellular automata. Computational support for some of the material is available in the form of a Mathematica package.
Prerequisites: 15-251 Min. grade C or 21-228 Min. grade C
Course Website: http://www.andrew.cmu.edu/course/15-355/
15-359 Probability and Computing
Intermittent: 12 units
Probability theory has become indispensable in computer science. In areas such as artificial intelligence and computer science theory, probabilistic methods and ideas based on randomization are central. In other areas such as networks and systems, probability is becoming an increasingly useful framework for handling uncertainty and modeling the patterns of data that occur in complex systems. This course gives an introduction to probability as it is used in computer science theory and practice, drawing on applications and current research developments as motivation and context. Topics include combinatorial probability and random graphs, heavy tail distributions, concentration inequalities, various randomized algorithms, sampling random variables and computer simulation, and Markov chains and their many applications, from Web search engines to models of network protocols. The course will assume familiarity with 3-D calculus and linear algebra.
Prerequisites: 15-251 Min. grade C and 15-259 and 21-241
Course Website: http://www.cs.cmu.edu/~harchol/15359/class.html

15-365 Experimental Animation
Spring: 12 units
[IdaeTe course] This class will explore animation from the student's perspective with a sense of investigation toward both form and content. Topics in the class will include non-linear narrative, visual music, puppet and non-traditional materials, manipulation of motion and performance capture data, immersive environments. Prerequisite: 15-213 Min. grade C

15-369 Special Topics: Perceptual Computing
Intermittent: 9 units
This course is ONLY offered at Carnegie Mellon in Qatar. What can today's computers see, hear, and feel? This project-based course is designed to provide students exposure to the state-of-the-art in machine perception and the algorithms behind them. Student groups will design a perceptual computing project around Intel's Creative Camera or Microsoft's Kinect. Students will learn to use tools in face detection and recognition, hand and finger tracking, and speech recognition, along with algorithms to make decisions based on these input modalities. Prerequisites: 15-122 and 21-241

15-381 Artificial Intelligence: Representation and Problem Solving
Fall: 9 units
This course is about the theory and practice of Artificial Intelligence. We will study modern techniques for computers to represent task-relevant information and make intelligent (i.e. satisficing or optimal) decisions towards the achievement of goals. The search and problem solving methods are applicable throughout a large range of industrial, civil, medical, financial, robotic, and information systems. We will investigate questions about AI systems such as: how to represent knowledge, how to effectively generate appropriate sequences of actions and how to search among alternatives to find optimal or near-optimal solutions. We will also explore how to deal with uncertainty in the world, how to learn from experience, and how to learn decision rules from data. We expect that by the end of the course students will have a thorough understanding of the algorithmic foundations of AI, how probability and AI are closely interrelated, and how automated agents learn. We also expect students to acquire a strong appreciation of the big-picture aspects of developing fully autonomous intelligent agents. Other lectures will introduce additional aspects of AI, including natural language processing, web-based search engines, industrial applications, autonomous robotics, and economic/game-theoretic decision making.
Prerequisite: 15-122 Min. grade C

15-383 Introduction to Text Processing
Fall: 6 units
Text processing is a mini-course about text basic techniques of processing human language in text format. The course has theoretical and hands-on components. In the theoretical component, the course will discuss challenges in processing human languages, and review the basics of statistics and probability theory and their application to language problems. In the hands-on part, students will learn about Python programming and use it to process large volumes of text using various techniques. The processing will range from simple steps such as tokenization and part-of-speech tagging to full-fledged applications such as statistical machine translation, search and document/topic classification. The course is suited for junior and senior students in CS and IS.
Prerequisites: 15-122 Min. grade C or 15-121 Min. grade C

15-385 Introduction to Computer Vision
Spring: 6 units
An introduction to the science and engineering of computer vision, i.e. the analysis of the patterns in visual images with the view to understanding the objects and processes in the world that generate them. Major topics include image formation and sensing, fourier analysis, edge and contour detection, inference of depth, shape and motion, classification, recognition, tracking, and active vision. The emphasis is on the learning of fundamental mathematical concepts and techniques and applying them to solve real vision problems. The discussion will also include comparison with human and animal vision from psychological and biological perspectives. Students will learn to think mathematically and develop skills in translating ideas and mathematical thoughts into programs to solve real vision problems.
Prerequisites: 15-122 Min. grade C and 21-241

15-386 Neural Computation
Spring: 9 units
Computational neuroscience is an interdisciplinary science that seeks to understand how the brain computes to achieve natural intelligence. It seeks to understand the computational principles and mechanisms of intelligent behaviors and mental abilities — such as perception, language, motor control, and learning — by building artificial systems and computational models with the same capabilities. This course explores how neurons encode and process information, adapt and learn, communicate, cooperate, and compute at the individual level as well as at the levels of networks and systems. It will introduce basic concepts in computational modeling, information theory, signal processing, system analysis, statistical and probabilistic inference. Concrete examples will be drawn from the visual system and the motor systems, and studied from computational, psychological and biological perspectives. Students will learn to perform computational experiments using Matlab and quantitative studies of neurons and neuronal networks.
Prerequisites: 15-112 Min. grade C and 21-122

15-387 Computational Perception
In this course, we will first cover the biological and psychological foundational knowledge of biological perceptual systems, and then apply computational thinking to investigate the principles and mechanisms underlying natural perception. The course will focus on vision this year, but will also touch upon other sensory modalities. You will learn how to reason scientifically and computationally about problems and issues in perception, how to extract the essential computational properties of those abstract ideas, and finally how to convert these into explicit mathematical models and computational algorithms. Topics include perceptual representation and inference, perceptual organization, perceptual constancy, object recognition, learning and scene analysis. Prerequisites: First year college calculus, some basic knowledge of linear algebra and probability and some programming experience are desirable.
Prerequisites: 21-241 and 21-122 and 15-112 Min. grade C

15-388 Practical Data Science
Intermittent: 9 units
Data science is the study and practice of how we can extract insight and knowledge from large amounts of data. This course provides a practical introduction to the “full stack” of data science analysis, including data collection and processing, data visualization and presentation, statistical model building using machine learning, and big data techniques for scaling these methods. Topics covered include: collecting and processing data using relational methods, time series approaches, graph and network models, free text analysis, and spatial geographic methods; analyzing the data using a variety of statistical and machine learning methods include linear and non-linear regression and classification, unsupervised learning and anomaly detection, plus advanced machine learning methods like kernel approaches, boosting, or deep learning; visualizing and presenting data, particularly focusing the case of high-dimensional data; and applying these methods to big data settings, where multiple machines and distributed computation are needed to fully leverage the data. Students will complete weekly programming homework that emphasize practical understanding of the methods described in the course. In addition, students will develop a tutorial on an advanced topic, and will complete a group project that applies these data science techniques to a practical application chosen by the team; these two longer assignments will be done in lieu of a midterm or final.
Prerequisite: 15-112 Min. grade C
Course Website: http://www.dataasciencecourse.org
15-390 Entrepreneurship for Computer Science
Fall: 9 units
This course is designed to develop skills related to entrepreneurship and innovation for non-business undergraduate and graduate students in the School of Computer Science. The course assumes no background courses in business and is appropriate for those who are interested in bringing innovations to market either through new companies or existing companies. The course provides an overview of entrepreneurship and innovation, develops an entrepreneurial frame of mind, and provides a framework for learning the rudiments of how to generate ideas. Students come up with or are presented with potential ideas and learn how to develop these ideas into opportunities, and to explore their potential for becoming viable businesses. They learn how to do market research, to develop go-to-market strategies, value propositions and to differentiate their products or services from potential competitors. The focus is on understanding and developing strategies for approaching the key elements of the entrepreneurial process...opportunity, resources and team. The course consists of a balance of lectures, case studies and encounters with entrepreneurs, investors and business professionals. The students are exposed to financial and intellectual property issues, and encounter a real world perspective on entrepreneurship, innovation and leadership. The output of the course is a mini-business plan or venture opportunity screening document that can be developed into a business plan in a subsequent course entitled New Venture Creation or through independent study.

15-392 Special Topic: Secure Programming
Spring: 9 units
This course provides a detailed explanation of common programming errors in C and C++ and describes how these errors can lead to software systems that are vulnerable to exploitation. The course concentrates on security issues intrinsic to the C and C++ programming languages and associated libraries. It does not emphasize security issues involving interactions with external systems such as databases and web servers, as these are rich topics on their own. Topics to be covered include the secure and insecure use of integers, arrays, strings, dynamic memory, formatted input/output functions, and file I/O.
Prerequisite: 15-213 Min. grade C
Course Website: https://www.securecoding.cert.org/confluence/display/sci/15392+Secure+Programming

15-400 Research Practicum in Computer Science
Spring: 12 units
This Spring course is the second part of a two-course sequence that is designed to help prepare students to invent the future state-of-the-art in the field of computer science. Building directly upon 15-300 (the prerequisite for this course), students will conduct a semester-long independent research project, under the guidance of both the course staff and a faculty project mentor. The course does not meet for lecture or recitations. Instead, the project, under the guidance of both the course staff and a faculty project mentor, begins to discuss your educational goals.

15-405 Engineering Distributed Systems
Spring: 9 units
This is a course for students with strong design and implementation skills who are likely to pursue careers as software architects and lead engineers. It may be taken by well-prepared undergraduates with excellent design and implementation skills in low-level systems programming. The course assumes a high level of proficiency in all aspects of operating system design and implementation. This course will help students prepare for leadership roles in creating and evolving the complex, large-scale computer systems that society will increasingly depend on in the future. The course will teach the organizing principles of such systems, identifying a core set of versatile techniques that are applicable across many system layers. Students will acquire the knowledge base, intellectual tools, hands-on skills and modes of thought needed to build well-engineered computer systems that withstand the test of time, growth in scale, and stresses of live use. Topics covered include: caching, prefetching, damage containment, scale reduction, hints, replication, hash-based techniques, and fragmentation reduction. A substantial project component is an integral part of the course. A high level of proficiency in systems programming is expected. If you do not have the 15-410 prerequisite you will need to get approval from the faculty.
Prerequisite: 15-410 Min. grade B

15-410 Operating System Design and Implementation
Fall and Spring: 15 units
Operating System Design and Implementation is a rigorous hands-on introduction to the principles and practice of operating systems. The core experience is writing a small Unix-inspired OS kernel, in C with some x86 assembly language, which runs on a PC hardware simulator (and on actual PC hardware if you wish). Work is done in two-person teams, and “team programming” skills (source control, modularity, documentation) are emphasized. The size and scope of the programming assignments typically result in students significantly developing their design, implementation, and debugging abilities. Core concepts include the process model, virtual memory, threads, synchronization, and deadlock; the course also surveys higher-level OS topics including file systems, interprocess communication, networking, and security. Students, especially graduate students, who have not satisfied the prerequisite at Carnegie Mellon are strongly cautioned - to enter the class you must be able to write a storage allocator in C, use a debugger, understand 2’s-complement arithmetic, and translate between C and x86 assembly language. The instructor may require you to complete a skills assessment exercise before the first week of the semester in order to remain registered in the class. Auditing this course is usually full, and we generally receive many more requests to audit than we can accept. If you wish to audit, please have your advisor contact us before the semester begins to discuss your educational goals.
Prerequisite: 15-213 Min. grade C

15-411 Compiler Design
Fall: 15 units
This course covers the design and implementation of compiler and run-time systems for high-level languages, and examines the interaction between language design, compiler design, and run-time organization. Topics covered include syntactic and lexical analysis, handling of user-defined types and type-checking, context analysis, code generation and optimization, and memory management and run-time organization.
Prerequisite: 15-213 Min. grade C
Course Website: http://symbolaris.com/course/compiler12.html

15-412 Operating System Practicum
Fall
The goal of this class is for students to acquire hands-on experience with operating-system code as it is developed and deployed in the real world. Groups of two to four students will select, build, install, and become familiar with an open-source operating system project; propose a significant extension or upgrade to that project; and develop a production-quality implementation meeting the coding standards of that project. Unless infeasible, the results will be submitted to the project for inclusion in the code base. Variations on this theme are possible at the discretion of the instructor. For example, it may be possible to work within the context of a non-operating-system software infrastructure project (window system, web server, or embedded network device kernel) or to extend a 15-410 student kernel. In some bi-situations students may work alone. Group membership and unit count (9 units versus 12) will be decided by the third week of the semester. Contributing to a real-world project will involve engaging in some mixture of messy, potentially open-ended activities such as: learning a revision control system; writing a short design document, creating and updating a simple project plan, participating in an informal code review, synthesizing scattered information about hardware and software, classifying and/or reading large amounts of code written by various people over a long period of time, etc.
Prerequisite: 15-410
15-413 Software Engineering Practicum
Spring: 12 units
This course is a project-based course in which students conduct a semester-long project for a real client in small teams. The project defines real-world needs for the client in their company. This is not a lecture-based course; after the first few weeks the course consists primarily of weekly team meetings with the course instructors, with teams making regular presentations on their software development process. Teams will give presentations and deliver documents on topics such as: risk management, project planning requirements architecture detailed design quality assurance final product presentations reflections on the experience. Evaluation will be based on the in-class presentations, process and project documentation, how well the teams follow software engineering (SE) practices, and the client's satisfaction with the product. Individual grades will be influenced by peer reviews, individual reflection documents, mentor impressions, and presentation performance. Students will leave the course with a firsthand understanding of the software engineering realities that drive SE practices, will have concrete experience with these practices, and will have engaged in active reflection on this experience. They will have teamwork, process, and product skills to support immediate employability in a software engineering organization, along with a deeper understanding that prepares them to evaluate the new processes and techniques they will encounter in the workplace.

15-414 Bug Catching: Automated Program Verification and Testing
Intermittent: 9 units
Many CS and ECE students will be developing software and hardware that must be ultra reliable and safe at one point in their careers. Such designs can be costly, even life threatening. There have already been a number of well publicized errors like the Intel Pentium floating point error and the Arian 5 crash. In this course we will study tools for finding and preventing logical errors. Three types of tools will be studied: automated theorem proving, state exploration techniques like model checking and tools based on static program analysis. Although students will learn the theoretical basis for such tools, the emphasis will be on actually using them on real examples. This course can be used to satisfy the Logic & Languages requirement for the Computer Science major.
Prerequisites: 15-122 Min. grade C and 15-251 Min. grade C

15-415 Database Applications
Fall: 12 units
This course covers the fundamental topics for Database Management Systems: Database System Architectural Principles (ACID properties; data abstraction; external, conceptual, and internal schemata; data independence; data definition and data manipulation languages), Data models (entity-relationship and relational data models; data structures, integrity constraints, and operations for each data model; relational query languages; SQL, algebra, calculus). Theory of database design (functional dependencies; normal forms; dependency preservation; information loss), Query Optimization (equivalence of expressions, algebraic manipulation; optimization of selections and joins), Storage Strategies (indices, B-trees, hashing), Query Processing (execution of sort, join, and aggregation operators), and Transaction Processing (recovery and concurrency control).
Prerequisites: 15-210 Min. grade C and 15-213 Min. grade C
Course Website: http://15415.courses.cs.cmu.edu/

15-417 HOT Compilation
Intermittent: 11 units
This course covers the implementation of compilers for higher-order, typed languages such as ML and Haskell, and gives an introduction to type-preserving compilation. Topics covered include type inference, elaboration, CPS conversion, closure conversion, garbage collection, phase splitting, and typed assembly language.
Prerequisites: 15-312 or 15-317

15-418 Parallel Computer Architecture and Programming
Spring: 12 units
The fundamental principles and engineering tradeoffs involved in designing modern parallel computers, as well as the programming techniques to effectively utilize these machines. Topics include naming shared data, synchronizing threads, and the latency and bandwidth associated with communication. Case studies on shared-memory, message-passing, data-parallel and dataflow machines will be used to illustrate these techniques and tradeoffs. Programming assignments will be performed on one or more commercial multiprocessors, and there will be a significant course project.
Prerequisite: 15-213 Min. grade C

15-421 Information Security and Privacy
Fall: 12 units
As layers upon layers of technology mediate our activities, issues of information security and privacy are becoming increasingly pervasive and complex. This course takes a multi-disciplinary perspective of information security and privacy, looking at technologies as well as business, legal, policy and usability issues. The objective is to prepare students to identify and address critical security and privacy issues involved in the design, development and deployment of robust computer and information systems. Examples used to introduce concepts covered in the class range from enterprise systems to mobile computing, the Internet of Things, social networking and digital currencies. Topics Covered: Information Security and Privacy, The big picture: A gentle introduction to cryptography, Certificates, PKI, Decentralized Trust Management; Authentication; Internet Security protocols; Risk management; Trusted Computing; Systems security; Web attacks; Cybercrime; Understanding the cyber security legal landscape; Information Privacy; Fundamental concepts & legal landscape; Information Security and Privacy: Emerging challenges (e.g. Cloud Security and Privacy, Mobile and IoT Security and Privacy, Social Networking Security and Privacy) Prerequisites: 76-101 and 15-112
Course Website: http://www.normsadeh.com/isp-content

15-423 Special Topic: Digital Signal Processing for Computer Science
Spring: 12 units
Digital signals comprise a large fraction of the data analyzed by computer scientists. Sound, e.g. speech and music, images, radar and many other signal types that were conventionally considered to be the domain of the Electrical engineer are now also in the domain of computer scientists, who must analyze them, make inferences, and develop machine learning techniques to analyze, classify and reconstruct such data. In this course we will cover the basics of Digital Signal Processing. We will concentrate on the basic mathematical formulations, rather than in-depth implementation details. We will cover the breadth of topics, beginning with the basics of signals and their representations, the theory of sampling, important transform representations, key processing techniques, and spectral estimation.
Prerequisites: (15-112 Min. grade C or 15-122 Min. grade C) and (15-359 or 21-325 or 36-217 or 36-225 or 36-625)

15-424 Foundations of Cyber-Physical Systems
Intermittent: 12 units
Cyber-physical systems (CPSs) combine cyber effects (computation and/or communication) with physical effects (motion or other physical processes). Designing algorithms to control CPSs, such as those in cars, aircraft and robots, is challenging due to their tight coupling with physical behavior. At the same time, it is vital that these algorithms be correct, since we rely on CPSs for safety-critical tasks like keeping aircraft from colliding. Students in this course will understand the core principles behind CPSs, develop models and controls, identify safety specifications and critical properties of CPSs, understand abstraction and system architectures, learn how to design by invariant, reason rigorously about CPS models, verify CPS models on the appropriate scale, understand the semantics of a CPS model and develop an intuition for operational effects. Students will write hybrid programs (HPs), which capture relevant dynamical aspects of CPSs in a simple programming language with a simple semantics, allowing the programmer to refer to real-valued variables representing real quantities and specify their dynamics as part of the HP.
Prerequisites: 15-122 Min. grade C and 21-122 Min. grade C and (15-241 Min. grade C or 18-202 Min. grade C or 15-251 Min. grade C)
Course Website: http://fipc.org/course/fcps17.html

15-437 Web Application Development
Fall and Spring: 12 units
This course will introduce concepts in programming web application servers. We will study the fundamental architectural elements of programming web sites that produce content dynamically. The primary technology introduced will be the Django framework for Python, but we will cover related topics as necessary so that students can build significant applications. Such topics include: HTTP, HTML, CSS, Javascript, XML, Design Patterns, Relational and Non-relational Databases, Object-Relation Mapping tools, Security, Web Services, Cloud Deployment, Internationalization, and Scalability and Performance issues. Students must have programming and software design experience equivalent to about a typical junior CS major— a sequence of three college CS courses or more. Python-specific experience is not necessary. Students must provide their own computer hardware for this course. Please see the Related URL above for more information.
Prerequisite: 15-214
15-440 Distributed Systems
Fall and Spring: 12 units
The goals of this course are twofold: First, for students to gain an understanding of the principles and techniques behind the design of distributed systems, such as locking, concurrency, current programming, naming, abstraction and modularity, imperfect communication and other types of failure, protection from accidental and malicious harm, optimism, and the use of instrumentation and monitoring and debugging tools in problem solving. As the creation and management of software systems is a fundamental goal of any undergraduate systems course, students will design, implement, and debug large programming projects. As a consequence, competency in both the C and Java programming languages is required.
Prerequisite: 15-213 Min. grade C

15-441 Computer Networks
Fall and Spring: 12 units
The emphasis in this course will be on the basic performance and engineering trade-offs in the design and implementation of computer networks. To make the issues more concrete, the class includes several multi-week projects requiring significant design and implementation. The goal is for students to learn not only what computer networks are and how they work today, but also why they are designed the way they are and how they are likely to evolve in the future. We will draw examples primarily from the Internet. Topics to be covered include: network architecture, routing, congestion/flow/error control, naming and addressing, peer-to-peer and the web, internetworking, and network security.
Prerequisite: 15-213 Min. grade C

15-445 Database Systems
Fall: 12 units
This course is on the design and implementation of database management systems. Topics include data models (relational, document, key/value), storage models (n-ary, decomposition), query languages (SQL, stored procedures), storage architectures (heaps, log-structured), indexing (order preserving trees, hash tables), transaction processing (ACID, concurrency control), recovery (logging, checkpoints), query processing (joins, sorting, aggregation, optimization), and parallel architectures (multi-core, distributed). Case studies on open-source and commercial database systems will be used to illustrate these techniques and trade-offs. The course is appropriate for students with strong systems programming skills.
Prerequisite: 15-213 Min. grade C
Course Website: http://15445.courses.cs.cmu.edu

15-449 Engineering Distributed Systems
Spring: 9 units
This is a course for students with strong design and implementation skills who are likely to pursue careers as software architects and lead engineers. It may be taken by well-prepared undergraduates with excellent design and implementation skills in low-level systems programming. The course assumes a high level of proficiency in all aspects of operating system design and implementation. This course will help students prepare for leadership roles in creating and evolving the complex, large-scale computer systems that society will increasingly depend on in the future. The course will teach the organizing principles of such systems, identifying a core set of versatile techniques that are applicable across many system layers. Students will acquire the knowledge base, intellectual tools, hands-on skills and modes of thought needed to build well-engineered computer systems that withstand the test of time, growth in scale, and stresses of live use. Topics covered include: caching, prefetching, damage containment, scale reduction, hints, replication, hash-based techniques, and fragmentation reduction. A substantial project component is an integral part of the course. A high level of proficiency in systems programming is expected. If you do not have the 15-410 prerequisite you will need to get approval from the faculty.
Prerequisite: 15-410 Min. grade B

15-451 Algorithm Design and Analysis
Fall and Spring: 12 units
This course is about the design and analysis of algorithms. We study specific algorithms for a variety of problems, as well as general design and analysis techniques. Specific topics include searching, sorting, algorithms for graph problems, efficient data structures, lower bounds and NP-completeness. A variety of other topics may be covered at the discretion of the instructor. These include parallel algorithms, randomized algorithms, geometric algorithms, low level techniques for efficient programming, cryptography, and cryptographic protocols.
Prerequisites: 15-210 Min. grade C and 15-251 Min. grade C and 21-241

15-453 Formal Languages, Automata, and Computability
Intermittent: 9 units
An introduction to the fundamental ideas and models underlying computing: finite automata, regular sets, pushdown automata, context-free grammars, Turing machines, undecidability, and complexity theory.
Prerequisites: 15-251 Min. grade C or 21-228 Min. grade C

15-455 Undergraduate Complexity Theory
Intermittent: 9 units
Complexity theory is the study of how much of a resource (such as time, space, parallelism, or randomness) is required to perform some of the computations that interest us the most. In a standard algorithms course, one concentrates on giving resource efficient methods to solve interesting problems. In this course, we concentrate on techniques that prove or suggest that there are no efficient methods to solve many important problems. We will develop the theory of various complexity classes, such as P, NP, co-NP, PH, #P, PSPACE, NC, AC, L, NL, UP, RP, BPP, IP, and PCP. We will study techniques to classify problems according to our available taxonomy. By developing a subtle pattern of reductions between classes we will suggest an (as yet unproven!) picture of how by using limited amounts of various resources, we limit our computational power.
Prerequisite: 15-251 Min. grade C

15-456 Computational Geometry
Intermittent: 9 units
How do you sort points in space? What does it even mean? This course takes the ideas of a traditional algorithms course, sorting, searching, segmenting, graphs, and optimization, and extends them to problems on geometric inputs. We will cover many classical geometrical constructions and novel algorithmic methods. Some of the topics to be covered are convex hulls, Delaunay triangulations, graph drawing, point location, geometric medians, polytopes, configuration spaces, linear programming, and others. This course is a natural extension to 15-451, for those who want to learn about algorithmic problems in higher dimensions.
Prerequisite: 15-451 Min. grade C

15-457 Special Topics in Theory: Advanced Algorithms
Intermittent: 12 units
Selected advanced topics in algorithms and computational theory. Topics vary from semester to semester.
Prerequisite: 15-451 Min. grade B

15-458 Discrete Differential Geometry
Fall: 12 units
Geometry plays a vital role in both engineering and scientific discovery, as well as in our everyday lives via emerging technologies like depth cameras and 3D printing. This course teaches students how to think about three-dimensional shape, both mathematically and computationally. Students will get a crash course in differential geometry, and will apply this knowledge to design and implement practical algorithms for 3D geometry processing. Basic geometric concepts (like curvature) are examined via complementary computational and mathematical points of view, with an emphasis on visual intuition and real-world applications. In homework, students will derive and implement core geometry processing algorithms; they will also explore a topic of their choice in a final class project. MS and PhD students will complete additional reading and coding exercises, and will perform a more comprehensive literature review for their final project. Topics include curves and surfaces, curvature, connections and parallel transport, exterior calculus, simplicial homology, conformal mapping, finite element methods, and numerical linear algebra; applications include approximation of curvature, curve and surface smoothing, surface parameterization, vector field design, and computation of geodesic distance.
Prerequisites: (15-122 Min. grade C or 15-112 Min. grade C) and 21-241 and 21-259
Course Website: http://geometry.cs.cmu.edu/ddg

15-462 Computer Graphics
Fall and Spring: 12 units
This course provides a comprehensive introduction to computer graphics modeling, animation, and rendering. Topics covered include basic image processing, geometric transformations, geometric modeling of curves and surfaces, animation, 3-D viewing, visibility algorithms, shading, and ray tracing.
Prerequisites: (15-213 Min. grade C and 21-240 and 21-259) or (21-241 and 15-213 Min. grade C and 21-259) or (18-213 Min. grade C and 18-202)
15-463 Computational Photography
Fall: 12 units
Computational Photography is an emerging new field created by the convergence of computer graphics, computer vision and photography. Its role is to overcome the limitations of the traditional camera by using computational techniques to produce a richer, more vivid, perhaps more perceptually meaningful representation of our visual world. The aim of this advanced undergraduate course is to study ways in which samples from the real world (images and video) can be used to generate compelling computer graphics imagery. We will learn how to acquire, represent, and render scenes from digitized photographs. Several popular image-based algorithms will be presented, with an emphasis on using these techniques to build practical systems. This hands-on emphasis will be reflected in the programming assignments, in which students will have the opportunity to acquire their own images of indoor and outdoor scenes and develop the image analysis and synthesis tools needed to render and view the scenes on the computer.
Prerequisites: 15-213 Min. grade C and (21-241 or 18-202)
Course Website: http://graphics.cs.cmu.edu/courses/15-463/

15-464 Technical Animation
Spring: 12 units
This course introduces techniques for computer animation such as keyframing, procedural methods, motion capture, and simulation. The course also includes a brief overview of story-boarding, scene composition, lighting and sound track generation. The second half of the course will explore current research topics in computer animation such as dynamic simulation of flexible and rigid objects, automatically generated control systems, and evolution of behaviors. The course should be appropriate for graduate students in all areas and for advanced undergraduates.
Prerequisite: 15-462 Min. grade C

15-465 Animation Art and Technology
Spring: 12 units
Animation Art and Technology is an interdisciplinary course cross-listed between Art and Computer Science. Faculty and teaching assistants from computer science and art teach the class as a team. It is a project-based course in which four to five interdisciplinary teams of students produce animations. Most of the animations have a substantive technical component and the students are challenged to consider innovation with content to be equal with the technical. The class includes basic tutorials for work in Maya leading toward more advanced applications and extensions of the software such as motion capture and algorithms for animating cloth, hair, particles, and grouping behaviors. The first class will meet in CFA room 303.
Prerequisites: 15-213 Min. grade C or 18-213 Min. grade C

15-466 Computer Game Programming
Spring: 12 units
The goal of this course is to acquaint students with the code required to turn ideas into games. This includes both runtime systems — e.g., AI, sound, physics, rendering, and networking — and the asset pipelines and creative tools that make it possible to author content that uses these systems. In the first part of the course, students will implement small games that focus on specific runtime systems, along with appropriate asset editors or exporters. In the second part, students will work in groups to build a larger, polished, open-ended game project. Students who have completed the course will have the skills required to extend — or build from scratch — a modern computer game. Students wishing to take this class should be familiar with the C++ language and a basic understanding of the OpenGL API. If you meet these requirements but have not taken 15-462 (the format prerequisite), please contact the instructor.
Prerequisite: 15-462

15-487 Introduction to Computer Security
Fall: 12 units
This course will introduce students to the fundamentals of computer security and applied cryptography. Topics include software security, networking and wireless security, and applied cryptography. Students will also learn the fundamental methodology for how to design and analyze security critical systems.
Prerequisite: 15-213

15-491 Special Topic: CMRoboBits: AI and Robots for Daily-Life Problems
Fall: 12 units
This course will be a project-based course in which we will look at AI and robotics artifacts and techniques to automate solutions to real-world problems, in particular related to life in cities. The course will start by collecting and brainstorming about real problems biased to ones that involve the physical space in addition to the cyber information space, such as traffic rush hour, noise in cities, 3D building inspection, service and data gathering. We will then formalize the chosen problems and analyze existing real data. The course will proceed by possibly enabling the students to prototype their projects beyond simulation, and using the CORAL lab robots, e.g., the robot, or NAO robots or drones. The course work will be a single large project, performed by groups of up to 3 students. The projects will be divided in three phases, due at the end of February, March, and the end of the course. Students are expected to have programming experience in C++ or python.
Prerequisite: 15-122 Min. grade C

15-492 Special Topic: Speech Processing
Fall: 12 units
Speech Processing offers a practical and theoretical understanding of how human speech can be processed by computers. It covers speech recognition, speech synthesis and spoken dialog systems. The course involves practicals where the student will build working speech recognition systems, build their own synthetic voice and build a complete telephone spoken dialog system. This work will be based on existing toolkits. Details of algorithms, techniques and limitations of state of the art speech systems will also be presented. This course is designed for students wishing understand how to process real data for real applications, applying statistical and machine learning techniques as well as working with limitations in the technology.
Prerequisite: 15-122 Min. grade C
Course Website: http://www.speech.cs.cmu.edu/15-492/

15-494 Cognitive Robotics: The Future of Robot Toys
Spring: 12 units
This course will explore the future of robot toys by analyzing and programming Anki Cozmo, a new robot with built-in artificial intelligence algorithms. Como is distinguished from earlier consumer robots by its reliance on vision as the primary sensing mode and its sophisticated use of A.I. Its capabilities include face and object recognition, map building, path planning, and object pushing and stacking. Although marketed as a pre-programmed children’s toy, Cozmo’s open source Python SDK allows anyone to develop new software for it, which means it can also be used for robotics education and research. The course will cover robot software architecture, human-robot interaction, perception, and planning algorithms for navigation and manipulation. Prior robotics experience is not required, just strong programming skills.
Prerequisite: 15-122 Min. grade C

15-591 Independent Study in Computer Science
Fall and Spring
The School of Computer Science offers Independent Study courses, which allow motivated students to work on projects under the supervision of a faculty advisor while receiving academic credit. Independent studies are usually one semester in duration and require prior approval from the faculty member and the School of Computer Science.

15-592 Independent Study in Computer Science
Fall
The School of Computer Science offers Independent Study courses, which allow motivated students to work on projects under the supervision of a faculty advisor while receiving academic credit. Independent studies are usually one semester in duration and require prior approval from the faculty member and the School of Computer Science.

15-593 Independent Study in Computer Science
Fall
The School of Computer Science offers Independent Study courses, which allow motivated students to work on projects under the supervision of a faculty advisor while receiving academic credit. Independent studies are usually one semester in duration and require prior approval from the faculty member and the School of Computer Science.

15-594 Independent Study in Computer Science
Fall
The School of Computer Science offers Independent Study courses, which allow motivated students to work on projects under the supervision of a faculty advisor while receiving academic credit. Independent studies are usually one semester in duration and require prior approval from the faculty member and the School of Computer Science.
15-599 SCS Honors Undergraduate Research Thesis
Fall and Spring
Available only to students registered in the CS Senior Research Thesis Program.

15-610 Engineering Distributed Systems
Spring: 12 units
This is a master's level course for students with strong design and implementation skills who are likely to pursue careers as software architects and lead engineers. It may be taken by well-prepared undergraduates with excellent design and implementation skills in low-level systems programming. The course assumes a high level of proficiency in all aspects of operating system design and implementation. This course will help students prepare for leadership roles in creating and evolving the complex, large-scale computer systems that society will increasingly depend on in the future. The course will teach the organizing principles of such systems, identifying a core set of versatile techniques that are applicable across many system layers. Students will acquire the knowledge base, intellectual tools, hands-on skills and modes of thought needed to build well-engineered computer systems that withstand the test of time, growth in scale, and stresses of live use. Topics covered include: caching, prefetching, damage containment, scale reduction, hints, replication, hash-based techniques, and fragmentation reduction. A substantial project component is an integral part of the course. A high level of proficiency in systems programming is expected.
Prerequisite: 15-410 Min. grade B
Course Website: http://www.cs.cmu.edu/~15-610

15-611 Compiler Design
Fall: 15 units
This course covers the design and implementation of compiler and run-time systems for high-level languages, and examines the interaction between language design, compiler design, and run-time organization. Topics covered include syntactic and lexical analysis, handling of user-defined types and type-checking, context analysis, code generation and optimization, and memory management and run-time organization.
Prerequisites: 15213 or 15312
Course Website: http://symbolaris.com/course/compiler12.html

15-617 HOT Compilation
Spring: 12 units
The course covers the implementation of compilers for higher-order, typed languages such as ML and Haskell, and gives an introduction to type-preserving compilation. Topics covered include type inference, elaboration, CPS conversion, closure conversion, garbage collection, phase splitting, and typed assembly language.

15-645 Database Systems
Fall: 12 units
This course is on the design and implementation of database management systems. Topics include data models (relational, document, key/value), storage models (n-ary, decomposition), query languages (SQL, stored procedures), storage architectures (heaps, log-structured), indexing (order preserving trees, hash tables), transaction processing (ACID, concurrency control), recovery (logging, checkpoints), query processing (joins, sorting, aggregation, optimization), and parallel architectures (multicore, distributed). Case studies on open-source and commercial database systems will be used to illustrate these techniques and trade-offs. The course is appropriate for students with strong systems programming skills.
Course Website: http://15445.courses.cs.cmu.edu

15-688 Practical Data Science
Fall: 12 units
Data science is the study and practice of how we can extract insight and knowledge from large amounts of data. This course provides a practical introduction to the "full stack" of data science analysis, including data collection and processing, data visualization and presentation, statistical model building using machine learning, and big data techniques for scaling these methods. Topics covered include: collecting and processing data using relational methods, time series approaches, graph and network models, free text analysis, and spatial geographic methods; analyzing the data using a variety of statistical and machine learning methods include linear and non-linear regression and classification, unsupervised learning and anomaly detection; plus advanced machine learning methods like kernel approaches, boosting, or deep learning; visualizing and presenting data, particularly focusing the case of high-dimensional data; and applying these methods to big data settings, where multiple machines and distributed computation are needed to fully leverage the data. Students will complete weekly programming homework that emphasize practical understanding of the methods described in the course. In addition, students will develop a tutorial on an advanced topic, and will complete a group project that applies these data science techniques to a practical application chosen by the team; these two longer assignments will be done in lieu of a midterm or final.
Course Website: http://www.data sciencecourse.org

15-705 Engineering Distributed Systems
Spring: 12 units
This course is for students with strong design and implementation skills who are likely to pursue careers as software architects and lead engineers. It may be taken by well-prepared undergraduates with excellent design and implementation skills in low-level systems programming. The course assumes a high level of proficiency in all aspects of operating system design and implementation. This course will help students prepare for leadership roles in creating and evolving the complex, large-scale computer systems that society will increasingly depend on in the future. The course will teach the organizing principles of such systems, identifying a core set of versatile techniques that are applicable across many system layers. Students will acquire the knowledge base, intellectual tools, hands-on skills and modes of thought needed to build well-engineered computer systems that withstand the test of time, growth in scale, and stresses of live use. Topics covered include: caching, prefetching, damage containment, scale reduction, hints, replication, hash-based techniques, and fragmentation reduction. A substantial project component is an integral part of the course. A high level of proficiency in systems programming is expected. Please refer to http://www.cs.cmu.edu/~csd-grad/courseschedules14.html this link for the most recent schedule updates.

15-719 Advanced Cloud Computing
Spring: 12 units
Computing in the cloud has emerged as a leading paradigm for cost-effective, scalable, well-managed computing. A Users pay for services provided in a broadly shared, power efficient datacenter, enabling dynamic computing needs to be met without paying for more than is needed. A Actual machines may be virtualized into machine-like services, or more abstract programming platforms, or application-specific services, with the cloud computing infrastructure managing sharing, scheduling, reliability, availability, elasticity, privacy, provisioning and geographic replication. Please refer to https://www.cs.cmu.edu/~csd-grad/courseschedules14.html for the most recent schedule updates.
Prerequisites: 15-513 Min. grade B or 15-213 Min. grade B
Course Website: http://www.cs.cmu.edu/~15719/
15-749 Engineering Distributed Systems
Fall: 12 units
Computing has changed beyond recognition in half a century, from the room-filling mainframes of the 1960s to today’s smartphones and wearable devices. Networks have also changed dramatically: from the 300-baud dialup modems of the early networking era to gigabit LANs, Wi-Fi and 4G today. Who knows what changes are in store for us over the next half century? Astonishingly, in spite of this tremendous change in hardware technology over time, a small core set of techniques for building distributed systems has emerged and remained surprisingly stable and applicable across many system layers. Many flavors of these techniques exist, and they continuously evolve over time to reflect changing trade-offs in the design space. What are these core techniques, and how can we leverage them in creating distributed systems today and in the future? That is the central question addressed by this course. Students will acquire the knowledge base, intellectual tools, hands-on skills and modes of thought needed to build well-engineered distributed systems that withstand the test of time, growth in scale, and stresses of live use. Strong design and implementation skills are expected of all students. The course assumes a high level of proficiency in all aspects of operating system design and implementation. A substantial project component is an integral part of the course. Please refer to http://www.cs.cmu.edu/~csd-grad/courseschedules14.html this link for the most recent schedule updates.
Course Website: http://www.cs.cmu.edu/~15-749/

15-751 A Theorist’s Toolkit
Intermittent: 12 units
Course description: This course will take a random walk through various mathematical topics that come in handy for theoretical computer science. It is intended mainly for undergraduate students (very strong undergraduates) who want to do theory research. The idea for the course comes from other courses by Arora (2002, 2007), Håstad (2004/05), Kelner (2007, 2009), and Tulsiani (2013). Prerequisites Students should have a solid undergraduate background in math (e.g., elementary combinatorics, graph theory, discrete probability, basic algebra/calculus) and theoretical computer science (running time analysis, big-O/Omega/Theta, P and NP, basic fundamental algorithms). Mathematical maturity is a must. Classes will begin week of September 5th. Math is a tool for science, and theoretical computer science (TCS) is science in math. The course will cover topics ranging from the very basic (e.g., circuits, counting, basic algebra) to the very advanced (e.g., linear programming, graph properties) as well as scheduling algorithms (e.g., work stealing) for mapping parallel computation to hardware for optimal efficiency and performance. There will be one midterm, several assignments (involving programming), and one project; the project will account for 50% of the grade. Prerequisites Students must have a solid undergraduate background in math (e.g., elementary combinatorics, graph theory, discrete probability, basic algebra/calculus) and theoretical computer science (running time analysis, big-O/Omega/Theta, P and NP, basic fundamental algorithms). Mathematical maturity is a must. https://www.cs.cmu.edu/~csd-grad/courseschedulef16.html CLASSES WILL BEGIN WEEK OF SEPTEMBER 5th.
Course Website: http://www.cs.cmu.edu/~odonnell/toolkit16/

15-769 Special Topics in Graphics:
Intermittent: 12 units
Please refer to http://www.cs.cmu.edu/~csd-grad/courseschedulef17.html for the most recent schedule updates. Class will not begin until the week of September 4th.
Course Website: http://graphics.cs.cmu.edu/courses/15869/fall2014/

15-883 Computational Models of Neural Systems
Intermittent: 12 units
This course is an in-depth study of information processing in real neural systems from a computer science perspective. We will examine several brain areas, such as the hippocampus and cerebellum, where processing is sufficiently well understood that it can be discussed in terms of specific representations and algorithms. We will focus primarily on computer models of these systems, after establishing the necessary anatomical, physiological, and psychophysical context. There will be some neuroscience tutorial lectures for those with no prior background in this area. Please refer to http://www.cs.cmu.edu/~csd-grad/courseschedulef17.html for the most recent schedule updates.
Course Website: http://www.cs.cmu.edu/afs/cs/academic/class/15883-f13/

15-897 Advanced Topics in Parallel Computing:
Spring: 12 units
Section A: In this special topics course on parallel computing, the students will learn about the foundations of parallel computing and complete a research project. The course will cover both theory and practice. The topics covered will include basic parallel algorithms for sequences (e.g., mapping, reductions), trees (e.g., tree contraction), and graphs (graph search, connectivity, graph contraction) as well as scheduling algorithms (e.g., work stealing) for mapping parallel computation to hardware for optimal efficiency and performance. There will be one midterm, several assignments (involving programming), and one project; the project will account for 50% of the grade. ** Project Ideas ** The project can be theoretical or practical. The ideal project would be one that adds parallelism to the student's current research. For example, if the student does research in a field such as algorithms/graphs/robotics and he or she can design/analyze/implement a parallel algorithm (possibly for an open problem); if the student does research on a field such as programming languages or databases, she or he can design/implement language/query support for parallelism; if the student does research on machine learning, he or she can develop a framework for performing approximate inference in parallel. Projects can be done individually or in groups and will lead to a small paper. We envision the students in the class to review other project papers (as in a program committee) and provide feedback to the authors. Please refer to https://www.cs.cmu.edu/~csd-grad/courseschedules17.html for the most recent schedule updates.

Human-Computer Interaction Courses
05-291 Learning Media Design
Fall: 12 units
[IdEAte collaborative course] Learning is a complex human phenomenon with cognitive, social and personal dimensions that need to be accounted for in the design of technology enhanced learning experiences. In this studio course students will apply learning science concepts to critique existing forms of learning media, establish a set of design precedents to guide project work and produce a series of design concepts that support learning interactions in a real-world context. Collaborating in small interdisciplinary teams, students will partner with a local informal learning organization (e.g. museum, after school program provider, maker space) to conduct learning design research studies, synthesize findings, establish learning goals and iteratively prototype and assess design concepts. As final deliverables, students will present their design research findings, design concepts, and prototypes to stakeholders, and draft a media-rich proposal for their learning media concept to pitch to a local funder. Please note that there may be usage/materials fees associated with this course. Please note that there may be usage/materials fees associated with this course. Prerequisite: 05-292
Project Ideas: 05-292

05-292 Learning Media Methods
Spring: 6 units
[IdEAte course] Learning Media Methods brings together students from across the disciplines to consider the design of mediated learning experiences though a project-based inquiry course. Students will be introduced to a range of design research methods and associated frameworks that explore the cognitive, social and affective dimensions of learning in everyday contexts through readings, invited lectures, in-class activities and assignments. Students will conduct a series of short design research studies to define learning goals and iteratively prototype and assess design concepts. As final deliverables, we will look at how to position technology and question its role in the setting to engage and foster positive learning interactions. The course will culminate in a media-rich presentation of design concepts to a stakeholder audience, and include an evaluation plan describing how learning outcomes for the project would be assessed.

05-320 Social Web
Intermittent: 12 units
With the growth of online environments like MySpace, Second Life, World of Warcraft, Wikipedia, blogs, online support groups, and open source development communities, the web is no longer just about information. This course, jointly taught by a computer scientist and a behavioral scientist, will examine a sampling of the social, technical and business challenges social web sites must solve to be successful, teach students how to use high-level tools to analyze, design or build online communities, and help them understand the social impact of spending at least part of their lives online. This class is open to advanced undergraduates and graduate students with either technical or non-technical backgrounds. Course work will include lectures and class discussion, homework, class presentations, and a group research or design project.
05-341 Organizational Communication
All Semesters: 9 units
Most of management is communication. You communicate to get information that will be the basis of decisions, coordinate activity, to provide a vision for the people who work for and with you, and to sell yourself and your work. The goal of this course is to identify communication challenges within work groups and organizations and ways to overcome them. To do this requires that we know how communication normally works, what parts are difficult, and how to fix it when it goes wrong. The focus of this course is on providing you with a broad understanding of the way communication operates within dyads, work groups, and organizations. The intent is to give you theoretical and empirical underpinnings for the communication you will undoubtedly participate in when you move to a work environment, and strategies for improving communication within your groups. Because technology is changing communication patterns and outcomes both in organizations and more broadly in society, the course examines these technological changes. Readings come primarily from the empirical research literature.
Course Website: http://www.hcii.cmu.edu/courses/organizational-communication

05-391 Designing Human Centered Software
All Semesters: 12 units
Why are things so hard to use these days? Why doesn't this thing I just bought work? Why is this web site so hard to use? These are frustrations that we have all faced from systems not designed with people in mind. The question this course will focus on is: how can we design human-centered systems that people find useful and usable? This course is an introduction to designing, prototyping, and evaluating user interfaces. If you take only one course in Human-Computer Interaction, this is the course for you. This class is open to all undergrads and grad students, with either technical or non-technical backgrounds. We will cover theory as well as practical application of ideas from Human-Computer Interaction. Course work includes lectures, class discussion, homework, class presentations, and group project.
Prerequisites: 15-110 or 15-104 or 15-112
Course Website: http://www.hcii.cmu.edu/courses/designing-human-centered-software

05-392 Interaction Design Overview
Fall: 9 units
This studio course offers a broad overview of communication and interaction design. Students will learn design methodologies such as brainstorming, sketching, storyboarding, wire framing, and prototyping. Students learn to take a human-centered design approach to their work. Assignments include short in-class exercises as well as individual and team-based projects. Students take part in studio critiques, engaging in critical discussions about the strengths and weaknesses of their own work and the work of others. No coding is required.

05-395 Applications of Cognitive Science
Spring: 9 units
The goal of this course is to examine cases where basic research on cognitive science, including cognitive neuroscience, has made its way into application, in order to understand how science gets applied more generally. The course focuses on applications that are sufficiently advanced as to have made an impact outside of the research field per se; for example, as a product, a change in practice, or a legal statute. Examples are virtual reality (in vision, hearing, and touch), cognitive tutors, phonologically based reading programs, latent semantic analysis applications to writing assessment, and measures of consumers' implicit attitudes. The course will use a case-study approach that considers a set of applications in detail, while building a general understanding of what it means to move research into the applied setting. The questions to be considered include: What makes a body of theoretically based research applicable? What is the pathway from laboratory to practice? What are the barriers - economic, legal, entrenched belief or practice? The format will emphasize analysis and discussion by students. They should bring to the course an interest in applications, as one of the primary goals is to determine the extent of the impact of cognitive science and to develop a sense of what is possible.
Course Website: http://www.hcii.cmu.edu/courses/applications-cognitive-science

05-410 User-Centered Research and Evaluation
Fall: 12 units
This course provides an overview and introduction to the field of human-computer interaction (HCI). It introduces students to tools, techniques, and sources of information about HCI and provides a systematic approach to design. The course increases awareness of good and bad design through observation of existing technology, and teaches the basic skills of task analysis, and analytic and empirical evaluation methods. This is a companion course to courses in visual design (15-422) and software implementation (05-430, 05-431). When registering for this course, undergraduate students are automatically placed the wait list. Students will then be moved into the class, based on if they are in the BHCI second major and year in school (e.g., seniors, juniors, etc). This course is NOT open to students outside the HCI major. When registering for this course, undergraduate students are automatically placed the wait list. Students will then be moved into the class, based on if they are in the BHCI second major and year in school.

05-413 Human Factors
Fall: 9 units
This course uses theory and research from human factors, cognitive science, and social science to understand and design the interactions of humans with the built world, tools, and technology. The course emphasizes current work in applied domains such as automotive design, house construction, medical human factors, and design of information devices. The course also will emphasize not only individual human factors (e.g., visual response, anthropometry) but also the organizational arrangements that can amplify or correct human factors problems. Through reading, discussion, and projects, you will learn about human perceptual, cognitive, and physical processes that affect how people interact with, and use, technology and tools. You will learn why we have so many automobile accidents, voting irregularities, and injuries from prescription medication. You will learn some tried and true solutions for human factors problems, and some of the many problems in human factors that remain. You will also have gained experience in research in this field.

05-417 Computer-mediated Communication
Spring: 6 units
This course examines fundamental aspects of interpersonal communication and considers how different types of computer-mediated communications (CMC) technologies affect communication processes. Among the topics we will consider are: conversational structure and CMC, tools to support nonverbal and paralinguistic aspects of communication such as gesture and eye gaze, and social and cultural dimensions of CMC. Students will be expected to post to weekly discussion lists, to write a paper on a specific aspect of CMC, and to present a talk on their final project to the class. The course should be appropriate for graduate students in all areas and for advanced undergraduates.

05-418 Design Educational Games
Spring: 12 units
The potential of digital games to improve education is enormous. However, it is a significant challenge to create a game that is both fun and educational. In this course, students will learn to meet this challenge by combining processes and principles from game design and instructional design. Students will also learn to evaluate their games for fun, learning, and the integration of the two. They will be guided by the EDGE framework for the analysis and design educational games. The course will involve a significant hands-on portion, in which students learn a design process to create educational games / digital or non-digital. They will also read about existing educational games and discuss game design, instructional design, learning and transfer, and the educational effectiveness of digital games. They will analyze an educational game and present their analysis to the class.
Course Website: http://www.hcii.cmu.edu/courses/design-educational-games
05-430 Programming Usable Interfaces
Spring: 15 units
This course is combines lecture, and an intensive programming lab and design studio. It is for those who want to express their creative ideas in working prototypes. It will cover the importance of human-computer interaction/interface design, iterative design, input/output techniques, how to design and evaluate interfaces, and research topics that will impact user interfaces in the future. In lab, you will learn how to design and program effective graphical user interfaces, and how to perform user tests. We will cover a number of prototyping tools and require prototypes to be constructed in each, ranging from animated mock-ups to fully functional programs. Assignments will require implementing UlS, testing that interface with users, and then modifying the interface based on findings. Some class sessions will feature design reviews of student work. This course is for HCII Masters students and HCI dual majors with a minimal programming background. Students will often not be professional programmers, but will need to interact with programmers. RECITATION SELECTION: Students taking this course can sign up for either Prototyping Lab recitation. PREREQUISITES: Proficiency in a programming language, program structure, algorithm analysis, and data abstraction. Normally met through an introductory programming course using C, C++, Pascal or Java, such as 15100, 15112, 15127 or equivalent. Students entering this course should be able to independently write a 300-line program in 48 hours. Prerequisites: 15-110 or 15-112 or 15-127 or 15-100 or 15-104

05-431 Software Structures for User Interfaces
Fall: 15 units
SSUI (15-credit, combined lecture and lab) This course considers the basic and detailed concepts that go into building software to implement user interfaces. It considers factors of input, output, application interface, and related infrastructure as well as the typical patterns used to implement them. It will also consider how these components are organized and managed within a well-structured object oriented paradigm. After considering these fundamental concepts in the first portion of the class, the later part will consider advanced topics related to emerging future concepts in user interface design. The course includes an intensive programming lab, either on the topic of mobile or web interfaces. This course is intended for HCII Master, BHCI dual majors and others who wish to understand the structures needed for professional development of interactive systems, and has a strong programming background. PREREQUISITES: Comfort in programming and related concepts equivalent to an undergraduate CS degree. Should be proficient in programming, and comfortable with abstract concepts relating to program structure, algorithm analysis, and data abstraction. WAI LIST LOGISTICS: Note that ALL students who register for this class will initially be placed on a waitlist. Your position on the waitlist is not an indication of whether you will be accepted into the class. Contacting the instructor will not move you off the waitlist. Priority for getting off the waitlist are MHCI students, BHCI students (more senior students first), and then others.

05-432 Personalized Online Learning
Fall: 12 units
Online learning has become widespread (e.g., MOOCs, online and blended courses, and Khan Academy) and many claim it will revolutionize higher education and K-12. How can we make sure online learning is maximally effective? Learners differ along many dimensions and they change over time. Therefore, adapting learning technologies to adapt to learners can provide individualized learning experiences. This course covers a number of proven personalization techniques used in advanced learning technologies. One of the techniques is the use of cognitive modeling to personalize practice of complex cognitive skills in intelligent tutoring systems. This approach, developed at CMU, may well be the most significant application of cognitive science in education and is commercially successful. We will also survey newer techniques, such as personalizing based on student meta-cognition, affect, and motivation. Finally, we will look at personalization approaches that are widely believed to be effective but have not proven to be so. The course involves readings and discussion of different ways of personalizing instruction, with an emphasis on cognitive modeling approaches. Students will learn to use the Cognitive Tutor Authoring Tools (CTAT, http://ctat.pact.cs.cmu.edu) to implement tutor prototypes that rely on computer-executable models of human problem solving to personalize instruction. The course is meant for graduate or advanced undergraduate students in Human-Computer Interaction, Psychology, Computer Science, Design, or related fields, who are interested in educational applications. Students should either have some programming skills or experience in the cognitive psychology of human problem solving, or experience with instructordesign.

05-433 Programming Usable Interfaces OR Software Structures for Usable Interfaces
Fall: 6 units
Section A: Programming Usable Interfaces Section B: Software Structures for Usable Interfaces This is a lecture-only course (see 05-430/05-630 or 05-431/631 for the lecture + lab version of these courses) that is intended for those who want to learn how to design and evaluate user interfaces. We will cover the importance of human-computer interaction and interface design, the iterative design cycle used in HCI, an overview of input and output techniques, how to design and evaluate interaction techniques, and end with a discussion of hot topics in research that will impact user interfaces in the future. This course is only intended for HCII Masters students or HCI undergraduate majors who have already taken an associated User Interface lab, or non-MHCI/BHCI students interested in the design of user interfaces. PREREQUISITES: There are no prerequisites for this lecture-only course. WAI LIST LOGISTICS: Note that ALL students who register for this class will initially be placed on a waitlist. Your position on the waitlist is not an indication of whether you will be accepted into the class. Contacting the instructor will not move you off the waitlist. Priority for getting off the waitlist are MHCI students, BHCI students (more senior students first), and then others.

05-434 Machine Learning in Practice
Fall and Spring: 12 units
Machine Learning is concerned with computer programs that enable the behavior of a computer to be learned from examples or experience rather than dictated through rules written by hand. It has practical value in many application areas of computer science such as on-line communities and digital libraries. This class is meant to teach the practical side of machine learning for applications, such as mining newsgroup data or building adaptive user interfaces. The emphasis will be on learning the process of applying machine learning effectively to a variety of problems rather than emphasizing an understanding of the theory behind what makes machine learning work. This course does not assume any prior exposure to machine learning theory or practice. In the first 2/3 of the course, we will cover a wide range of learning algorithms that can be applied to a variety of problems. In particular, we will cover topics such as decision trees, rule based classification, support vector machines, Bayesian networks, and clustering. In the final third of the class, we will go into more depth on one application area, namely the application of machine learning to problems involving text processing, such as information retrieval or text categorization. 05-834 is the HCI graduate section. If you are an LTI student, please sign up for the LTI graduate course number (11-663) ONLY to count properly towards your degree requirements. 05-434 is the HCI undergraduate section. If you are an LTI student, please sign up for the LTI undergraduate course number (11-344) ONLY to count properly towards your degree requirements.

Course Website: http://www.hcii.cmu.edu/courses/applied-machine-learning

05-435 Advanced Fabrication Techniques for HCI
Fall: 12 units
This course will consider how new fabrication techniques such as 3D printing, laser cutting, CNC machining and related computer controlled technologies can be applied to problems in Human-Computer Interaction. Each offering will concentrate on a particular application domain for its projects. This year the course will consider assistive technology. This course will be very hands-on and skills-oriented, with the goal of teaching students the skills necessary to apply these technologies to HCI problems such as rapid prototyping of new device concepts. To this end? Every student in this course will build and take home a 3D printer. (There will be $400-$500 cost associated with this course to make that possible. Details on this are still to be determined.)

05-439 The Big Data Pipeline: Collecting and Using Big Data for Interactive Systems
Spring: 12 units
This course covers techniques and technologies for creating data driven interfaces. You will learn about the entire data pipeline from sensing to cleaning data to different forms of analysis and computation.

Course Website: http://data.cmu.edu
05-440 Interaction Techniques
Interrnet: 12 units
This course will provide a comprehensive study of the many ways to interact with computers and computerized devices. An “interaction technique” starts when the user does something that causes an electronic device to respond, and includes the direct feedback from the device to the user. Examples include physical buttons and switches, on-screen menus and scroll bars operated by a mouse, touch screen widgets and gestures such as flick-to-scroll, text entry on computers or touch screens, consumer electronic controls such as remote controls, game controllers, and adaptations of all of these for people with disabilities. We will start with a history of the invention and development of these techniques, discuss the various options used today, and continue on to the future with the latest research on interaction techniques presented at conferences such as ACM CHI and UIST. Guest lectures from inventors of interaction techniques are planned. Students will have a choice for final projects that can focus on historical or novel interaction techniques. Fall (will be to creating a novel technique, perform a user study of it, and write a paper about the result, which may be suitable for conference submission. Another option will be to investigate and write a paper or make a video about the history and various past designs for widely used interaction techniques, possibly including an interview with the inventor(s).
Course Website: http://www.cs.cmu.edu/~bam/uicourse/05440inter/

05-452 Service Design
Fall: 12 units
In this course, we will collectively define and study services and product service systems, and learn the basics of designing them. We will do this through lectures, studio projects, and verbal and written exposition. Classwork will be done individually and in teams.

05-499 Special Topics in HCI
Fall and Spring: 12 units
Special Topics in HCI is an opportunity for students interested in HCI to gain a deeper understanding of a specific area in this field. Each class is designed to cover an emerging research area within HCI, from designing large-scale peer learning systems to designing games around audience agency. All sections will help students: (1) build a more comprehensive understanding of an area of study within HCI, (2) work closely with faculty and peers to create mini-projects or team assignments that help students master the course material, (3) explore evidence-based research methods and techniques in HCI. Sections will vary in topic and often change from semester to semester. Because of this, students can take multiple sections, as they are individual classes. Undergraduate sections are listed as 499 and graduate sections are listed as 699. For descriptions of specific sections for this academic year, visit the "Courses" section on the Human-Computer Interaction Institute website: http://hcii.cmu.edu/academics/courses
Course Website: http://www.hcii.cmu.edu/academics/courses

05-540 Rapid Prototyping of Computer Systems
Spring: 12 units
This is a project-oriented course, which will deal with all four aspects of project development: the application, the artifact, the computer-aided design environment, and the physical prototyping facilities. The class consists of students from different disciplines who must synthesize and implement a system in a short period of time. Upon completion of this course the student will be able to: generate systems specifications from a perceived need; place functionality between hardware and software; produce interface specifications for a system composed of numerous subsystems; use computer-aided development tools; fabricate, integrate, and debug a hardware/software system; and evaluate the system in the context of an end user application. The class consists of students from different disciplines who must synthesize and implement a system in a short period of time.
Course Website: http://www.hcii.cmu.edu/courses/rapid-prototyping-computer-systems

05-571 Undergraduate Project in HCI
Spring: 12 units
Experiential learning is a key component of the MHCI program. Through a substantial team project, students apply classroom knowledge in analysis and evaluation, implementation, design, and develop skills working in multidisciplinary teams. Students team with Carnegie Mellon University-based clients or external clients to iteratively design, build and test a software application which people directly use.
Prerequisites: 05-431 Min. grade B or 05-610 Min. grade B or 05-630 Min. grade B or 05-631 Min. grade B or 05-410 Min. grade B or 05-430 Min. grade B
Course Website: http://www.hcii.cmu.edu/courses/undergraduate-project-hci

05-589 Independent Study in HCI-UG
All Semesters
In collaboration with and with the permission of the professor, undergraduate students may engage in independent project work on any number of research topics that are sponsored by faculty. Students must complete an Independent Study Proposal, negotiate the number of units to be earned, complete a contract, and present a tangible deliverable. The Undergraduate Program Advisor's signature is required for HCI undergraduate-level Independent Study courses.

05-600 HCI Pro Seminar
Fall: 6 units
Students will attend weekly HCI Pro Seminar Series of talks given by national leaders in the field of Human-Computer Interaction, attend communication workshops and conflict management workshops.
Course Website: http://www.hcii.cmu.edu

05-610 User-Centered Research and Evaluation
Fall: 12 units
This course provides overview and introduction to the field of human-computer interaction (HCI). It introduces students to tools, techniques, and sources of information about HCI and provides a systematic approach to design. The course increases awareness of good and bad design through observation of existing technology, and teaches the basic skills of task analysis, and analytic and empirical evaluation methods. This is a companion course to HCI Pro Seminar in visual design (05-650) and software implementation (05-630, 05-631). This course is NOT open to students outside of the HCI major.
Course Website: http://www.hcii.cmu.edu

05-650 Interaction Design Studio II
Spring: 12 units
This course follows Interaction Design Fundamentals (05-651). Students are expected to apply what they have learned about design thinking and methodologies as a starting point for all assignments. Students will work in teams to perform guerrilla research, gather and synthesize data, and consider the needs of multiple stakeholders in their design of mobile services and other intelligent systems. Design concepts go beyond user interfaces to include sensors, controls, and ubiquitous computing. Emphasis is placed on the quality of the students ideas and their ability to give form to their design concepts. By completing and presenting their work, students will gain skills related to professional UX design practice.
Prerequisites: 05-651 or 51-761 or 51-262 or 51-261
Course Website: http://www.hcii.cmu.edu/courses/interaction-design-studio

05-651 Interaction Design Studio I
Fall: 12 units
This studio course introduces students to design thinking and the basic practices of interaction design. We follow a human-centered design process that includes research, concept generation, prototyping, and refinement. Students must work effectively as individuals and in small teams to design mobile information systems and other interactive experiences. Assignments approach design on three levels: specific user interactions, contexts of use, and larger systems. Students will become familiar with design methodologies such as sketching, storyboarding, wire framing, prototyping, etc. No coding is required. This course serves as a prerequisite for Interaction Design Studio (05-650).

05-823 E-Learning Design Principles and Methods
Fall: 12 units
This course is about e-learning design principles, the evidence and theory behind them, and how to apply these principles to develop effective educational technologies. It is organized around the book "e-Learning and the Science of Instruction: Proven Guidelines for Consumers and Designers of Multimedia Learning" by Clark & Mayer with further readings drawn from cognitive science, educational psychology, and human-computer interaction. You will learn design principles 1) for combining words, audio, and graphics in multimedia instruction, 2) for combining examples, explanations, practice and feedback in online support for learning by doing, and 3) for balancing learner versus system control and supporting student metacognition. You will read about the experiments that support these design principles, see examples of how to design such experiments, and practice applying the principles in educational technology development.
Course Website: http://www.learnlab.org/research/wiki/index.php/E-learning_Design_Principles_2013#Course_Details

05-840 Tools for Online Learning
Fall: 12 units
This course is for students in the METALS program.
Institute for Software Research Courses

08-200 Ethics and Policy Issues in Computing
Spring: 9 units
Should autonomous robots make life and death decisions on their own? Should we allow them to select a target and launch weapons? To diagnose injuries and perform surgery when human doctors are not around? Who should be permitted to observe you, find out who your friends are, what you do and say with them, what you buy, and where you go? Do social media and personalized search restrict our intellectual horizons? Do we live in polarizing information bubbles, just hearing echoes of what we already know and believe? As computing technology becomes ever more pervasive and sophisticated, we are presented with an escalating barrage of decisions about who, how, when, and for what purposes technology should be used. This course will provide an intellectual framework for discussing these pressing issues of our time, as we shape the technologies that in turn shape us. We will seek insight through reading, discussion, guest lectures, and debates. Students will also undertake an analysis of a relevant issue of their choice, developing their own position, and acquiring the research skills needed to lend depth to their thinking. The course will enhance students’ ability to think clearly about contentious technology choices, formulate smart positions, and support their views with winning arguments.

08-202 Computational Approaches to Social Complexity
Spring: 9 units
Everything is connected! From small groups to economic markets to global societies - interactions among people, organizations, technology, and policies lead to complex systems. These connected systems cannot be described with simple equations but rather with networks, non-linearities and tipping points. This course covers a set of computational approaches to analyzing and understanding the dynamics of the social world. Techniques include data capture, social network analysis, advanced information visualization, and agent based simulation. Students will engage in hands-on projects learning these techniques to explain and predict social changes arising from the complexity of everyday life.

08-302 Introduction to Network Science
Fall: 9 units
Everything is connected - from your friends, to everyone at CMU, to economic markets, disease outbreaks, and global societies. Relationships and flows of information among people and organizations form complex systems that are the fundamental structures governing our world, yet defy easy understanding. To analyze these interconnected systems we must turn to network science. This course covers the mathematical and graph theoretical foundations of network science, as well as theories and algorithms for analyzing and visualizing structures and dynamics of networked systems. Topics covered in this class include: centrality metrics, community detection, diffusion processes, scale-free and small-world networks, social media analytics, and network visualization. A special emphasis is on algorithmic challenges and solutions in the context of big data networks. Students will engage in hands-on projects learning network science, solidifying their mathematical and graph theory knowledge and learning how to apply it to a range of real world problems. This course is open to students in all majors who have earned a C or better in 15-151 or 21-127.
Prerequisites: 21-127 Min. grade C or 15-151 Min. grade C

08-303 Special Topic: Cryptocurrencies, Blockchains and Applications
Spring
Cryptocurrencies such as Bitcoin have gained large popularity in recent years, in no small part due to the fantastic potential applications they could facilitate. This course will first provide an overview of the technological mechanisms behind cryptocurrencies and distributed consensus and distributed ledgers (“blockchains”), introducing along the way the necessary cryptographic tools. It will then focus on more advanced blockchain applications, such as “smart contracts,” that is, contracts written as code. Finally, the course will also introduce some of the legal and policy questions surrounding cryptocurrencies. Prerequisites: 15-213 or equivalent strongly recommended

08-340 Green Computing
Spring: 9 units
Energy is a key societal resource. However, our energy usage is rising at an alarming rate and therefore it has become critical to manage its consumption more efficiently for long term sustainability. This course introduces students to the exciting area of “Green Computing”, and is organizationally divided into two tracks. The first track is “Energy-Efficient Computing”, which considers the state of the art techniques for improving the energy efficiency of mobile devices, laptop, and desktop class computers and finally to data centers. We will cover energy efficiency across the hardware/software stack, starting from the individual components like processors and radio interfaces to system level architectures and optimizations. The second track is “Applying Computing towards Sustainability”, covering topics that leverage computing to reduce the energy footprint of our society. In particular, we will focus on Smart Buildings and the Smart Grid, covering topics such as sensing, modeling and controlling the energy usage of buildings, new operating systems or software stacks for the smart infrastructure, as well as the privacy and security issues with the new “Internet of Things”. The goal of this course is to help students acquire some of the knowledge and the skills needed to do research in this space of “Green Computing”. Although the course is listed within SCS, it should be of interest to students in several departments, including ECE, MechE, CEE, EPP and Architecture.
Course Website: http://www.synergylabs.org/yuvraj/courses/08-840/

08-421 Building user-focused sensing systems
Spring: 12 units
These days we are surrounded by sensing and computation. Smart devices, such as smartphones, smartwatches, are packed with sensors. While they are already very useful devices, we have only started to scratch the surface here. The aim of this class will be to introduce the students to building and understanding smart sensing devices. The course will include discussion into contribution of various fields, including human-computer interaction, embedded computing, computer vision, distributed systems, machine learning, signal processing, security, and privacy. We will discuss how these various disciplines are coming together to form an end-to-end system that generates useful and user-actionable data. We will take a hands-on approach towards building and evaluating these systems. The students will gain practical experience in developing sensing systems in different application domains, such as activity recognition, health sensing, gestural interaction, etc. You will learn about embedded systems and understand the advantages and limitations of different platforms. You will learn about sensors and how to interface them with the real world to be able to get useful and actionable data. You will learn how to build a network of sensors that can communicate with each other. You will also learn about storing the sensor data for visualization, analysis and presentation both locally and to the cloud. The course will be a combination of lectures, tutorials, class discussions, and demonstrations. Students will be evaluated based on 5 mini-projects/assignments, class participation, weekly reading summaries, and a final project. All hardware resources will be provided to the students and they will be given an option to take their final prototypes with them for the cost of the hardware components. The only requirement for the class is that the students should have reasonable programming experience and an interest in tinkering.

08-530 Pervasive and Ubiquitous Computing
Fall: 12 units
The aim of the class will be to explore the area of Ubiquitous Computing (ubicomp) and allow students to work on a variety of small technology projects. Students will be exposed to the basics of building ubicomp systems, emerging new research topics, and advanced prototyping techniques. This course will focus more on class discussions and hands on demonstrations, while formal lectures will be conducted only as needed. Students will be evaluated on their class participation, reading summaries, and mini projects.

08-532 Law of Computer Technology
Fall: 9 units
The Law of Computer Technology A survey of how legislatures and courts cope with rapidly advancing computer technologies and how scientific information is presented to, and evaluated by, civil authorities. The course is also an introduction to the legal process generally and the interaction between the legal system and technology organizations. Topics include: patents, copyrights in a networked world, law of the Internet, free speech, data security, technology regulation, international law, and transborder crime. Open to juniors, seniors and graduate students in any school. Open to sophomores by permission of the instructor. Prerequisites: none.
08-533 Privacy Policy, Technology and Law
Fall and Spring: 9 units
This course focuses on policy issues related to privacy from the perspectives of governments, organizations, and individuals. We will begin with a historical and philosophical study of privacy and then explore recent public policy issues. We will examine the privacy protections provided by laws and regulations, as well as the ways technology can be used to protect privacy. We will emphasize technology-related privacy concerns and mitigation, for example: social networks, smartphones, behavioral advertising (and tools to prevent targeted advertising and tracking), anonymous communication systems, big data, and drones. This is part of a series of courses offered as part of the MSIT-Privacy Engineering masters program. These courses may be taken in any order or simultaneously. Foundations of Privacy (Fall semester) offers more in-depth coverage of technologies and algorithms used to reason about and protect privacy. Engineering Privacy in Software (Spring semester) focuses on the methods and tools needed to design systems for privacy. This course is intended primarily for graduate students and advanced undergraduate students with some technical background. Programming skills are not required. 8-733, 19-608, and 95-818 are 12- unit courses for PhD students. Students enrolled under these course numbers will have extra assignments and will be expected to do a project suitable for publication. 8-533 is a 9-unit course for undergraduate students. Masters students may register for any of the course numbers permitted by their program. This course will include a lot of reading, writing, and class discussion. Students will be able to tailor their assignments to their skills and interests. However, all students will be expected to do some writing and some technical work.
Course Website: http://cups.cs.cmu.edu/courses/privpolawtech.html

08-540 Green Computing
Spring: 9 units
Energy is a key societal resource. However, our energy usage is rising at an alarming rate and therefore it has become critical to manage its consumption more efficiently for long term sustainability. This course introduces students to the exciting area of “Green Computing”, and is organizationally divided into two tracks. The first track is “Energy-Efficient Computing”, which considers the state of the art techniques for improving the energy efficiency of mobile devices, to laptop and desktop class computers and finally to data centers. We will cover energy efficiency across the hardware/software stack, starting from the individual components like processors and radio interfaces to system level architectures and optimizations. The second track is “Applying Computing towards Sustainability”, covering topics that leverage computing to reduce the energy footprint of our society. In particular, we will focus on Smart Buildings and the Smart Grid, covering topics such as sensing, modeling and controlling the energy usage of buildings, new operating systems or software stacks for the smart infrastructure, as well as the privacy and security issues with the new “internet of things”. The goal of this course is to help students acquire some of the knowledge and the skills needed to do research in this space of “Green Computing”. Although the course is listed within SCS, it should be of interest to students in several departments, including ECE, MechE, CEE, EPP and Architecture.
Course Website: http://www.synergylabs.org/yuvraj/courses/08-840/

08-541 Hardware and Software Systems for Smart Homes and Buildings
Spring: 9 units
Smart automation in the home and buildings hold tremendous promise to enhance the quality of our lives and improving energy efficiency. In this course we will take a hand on approach towards building hardware and software systems within the context of these emerging smart homes and smart building domains. You will learn about embedded systems and how to choose the appropriate one for different smart home projects, including understanding the advantages and limitations of such platforms. You will learn about sensors and how to interface them with the real world to be able to get useful and actionable data. You will learn about designing systems from the ground up, including schematic and PCB layouts. You will learn about different radio technologies, so that you can build a network of sensors that can communicate with each other. You will also learn about storing the sensor data for visualization, analysis and presentation both locally and to the cloud. The course will comprise of hands on exercises, which build upon each other as the course progresses through the semester. We will likely use an embedded platform such as the Raspberry Pi or the BeagleBone Black (cost between $75 and $100) and each student will be expected to buy one of these platforms to use throughout the semester. From time to time we will also discuss latest research papers and case studies on topics related to smart home and buildings, with the focus towards deployed systems. The course grade will be computed based on the homework and mini-project assignments, class presentations and a semester long course project. It is expected that students will have some programming background, are excited to take on hands on projects, and want to learn how to build end to end systems for smart homes. As a side effect, perhaps some of your ideas will lead to the next NEST (sold for $52.26 in 2014!).

08-671 Java for Application Programmers
Fall and Spring: 6 units
This course provides an intensive exploration of computer programming in the Java Language for Masters students who have had some prior, but perhaps limited, programming experience in Java or in some other programming language. The course starts with a review of the fundamental topics of programming in Java (data types, operators, control structures) along with discussions of object oriented programming (classes, instances, and class hierarchies). After covering the basics we move on using Java to cover simple graphical user interfaces (using Swing), file I/O, concurrency (i.e., threads), network I/O (HTTP), simple data structures (lists and maps). Students are required to have a reasonably modern laptop computer on which they will need to install the Java software used for this course.

08-672 J2EE Web Application Development
Fall: 6 units
This course will introduce concepts in programming web application servers. We will study the fundamental architectural elements of programming web sites that produce content dynamically. We will be demonstrating the course using Java Servlets and Java Server Pages. We will also cover the related topics as necessary so that students may build significant applications. Such topics are expected to include: HTML, CSS, Fundamental Relational and Non-Relational Databases, Object-Relation Mapping Tools, Security Issues, AJAX, and Cloud Deployment. Students are required to be familiar with Java Programming before taking this course. Those who are not are encouraged to take 08-671 in mini 1 before taking this course. Students are required to have a reasonably modern laptop computer on which they need to install the Java software used for this course.

08-725 Social Software Engineering
Fall: 6 units
Today, socialness appears to be of great significance and under constant scrutiny in the domain of software, where it is both fostered and leveraged. As it relates to various aspects of software and how software is build - from people, products and processes, to technologies, tools and trade and even more - understanding social and human aspects of software engineering is essential. This course will pull together relevant literature from multiple domains to better understand the various social and human aspects of software engineering. To that end, we will explore both: the social aspects of software engineering and the engineering aspects of social software. The course will work from an initial set of topics and readings, but it is expected that both will evolve as the course progresses. The topics could span several dimensions of software engineering such as (but not limited to) the following: * Social software and social-ness of software * SDLC phases (requirements, design, development, testing & further) * Processes (personal, team, management etc) * Methods & model development & use of software) * Measurement (frameworks, models and metrics) * Environment (development and operation/use, tools / technologies / contexts) * Creation and dissemination of knowledge * Innovation * Role of users (involvement and solving expertise) * Standards, regulation and governance * Case studies
08-733 Privacy Policy, Technology and Law
Fall: 12 units
This course focuses on policy issues related to privacy from the perspectives of governments, organizations, and individuals. We will begin with a historical and philosophical study of privacy and then explore recent public policy issues. We will examine the privacy protections provided by laws and regulations, as well as the way technology can be used to protect privacy. We will emphasize technology-related privacy concerns and mitigation, for example: social networks, smartphones, behavioral advertising (and tools to prevent targeted advertising and tracking), anonymous communication systems, big data, and drones. This is part of a series of courses offered as part of the MSIT-Privacy Engineering masters program. These courses may be taken in any order or simultaneously. Foundations of Privacy (Fall semester) offers more in-depth coverage of technologies and algorithms used to reason about and protect privacy. Engineering Privacy in Software (Spring semester) focuses on the methods and tools needed to design systems for privacy. This course is intended primarily for graduate students and advanced undergraduate students with some technical background. Programming skills are not required. 8-733, 19-608, and 95-818 are 12-unit courses for PhD students. Students enrolled under these course numbers will have extra assignments and will be expected to do a project suitable for publication. 8-533 is a 9-unit course for undergraduate students. Masters students may register for any of the course numbers permitted by their program. This course will include a lot of reading, writing, and class discussion. Students will be able to tailor their assignments to their skills and interests. However, all students will be expected to do some writing and some technical work.

Course Website: http://cups.cs.cmu.edu/courses/privpolawtech.html

08-781 Mobile and IoT Computing Services
Spring: 9 units
With over 6 billion mobile phone users worldwide, including well over a billion smart phone users, new wireless and pervasive computing applications and services are changing the way enterprises interact with customers and employees. The explosion in smart phone ownership along with the deployment of 4G networks is leading to a slew of new mobile apps and services. They range from mobile commerce services to wireless enterprise apps and mobile social networking apps, all the way to more futuristic pervasive computing scenarios where phones interact with a number of other everyday objects (e.g. smart homes, smart cars, smart glasses, health/fitness sensors). These apps and services are emerging as part of an increasingly rich ecosystem where context awareness and intelligent predictive technologies are used to offer increasingly personalized experiences to users. This same ecosystem has emerged as the engine behind increasingly targeted marketing and advertising scenarios that also raise challenging privacy issues. The course objective is to introduce participants to the technologies, services and business models associated with Mobile and Pervasive Commerce. It also provides an overview of future trends and ongoing research. You will learn to evaluate critical design tradeoffs associated with different mobile technologies, architectures, interfaces and business models and how they impact the usability, security, privacy and commercial viability of mobile and pervasive computing services and apps. Topics include Mobile Communications, Mobile OS, Mobile Web technologies including app development, Mobile Security, Mobile Payments, Mobile Web Apps and Services (e.g. Mobile Entertainment, Mobile Banking, Mobile, Mobile Social Networking, Mobile Health, etc.), Location-Based Services, RFID, Mobile Enterprise Apps, Pervasive Computing Applications, Context awareness, intelligent assistant technologies, and privacy.

Course Website: http://www.normsadeh.com/ms-course-overview

08-801 Dynamic Network Analysis
Spring: 12 units
Who knows who? Who knows what? Who communicates with whom? Who is influential? How do ideas, diseases, and technologies propagate through groups? How do social media, social, knowledge, and technology networks differ? How do these networks evolve? How do network constrain and enable behavior? How can a network be compromised or made resilient? Such questions can be addressed using Network Science. Network Science, a.k.a. social network analysis and link analysis, is a fast-growing interdisciplinary field aimed at understanding simple & high dimensional networks, from both a static and a dynamic perspective. Across an unlimited application space, graph theoretic, statistical, & simulation methodologies are used. An interdisciplinary perspective on network science is provided, with an emphasis on high-dimensional dynamic data. The fundamentals of network science, methods, theories, metrics & confidence estimation, constraints on data collection & bias, and key research findings & challenges are examined. Illustrative networks discussed include social media based (e.g., twitter), disaster response, organizational, semantic, political elite, crises, terror, & P2P networks. Critical procedures covered include: basic centralities and metrics, group and community detection, link inference, network change detection, comparative analytics, and big data techniques. Applications from business, science, art, medicine, forensics, social media & numerous other areas are explored. Key issues addressed: Conceptualization, measurement, comparison & evaluation of networks. Identification of influential nodes and hidden groups. Network emergence, evolution, change & destabilization. Graduate course taught every other year. Prerequisite: Undergraduate-level statistics course or instructor permission. Linear algebra is recommended. Students are encouraged to bring & use their own data, or to use provided data.

08-805 Engineering Privacy in Software
Spring: 12 units
This section is for Ph.D. students; all other students should enroll in 08-605. Privacy harms that involve personal data can often be traced back to software design failures, which can be prevented through sound engineering practices. In this course, students will learn how to identify privacy threats due to surveillance activities that enhance modern information systems, including location tracking, behavioral profiling, recommender systems, and social networking. Students will learn to analyze systems to identify the core operating principles and technical means that introduce privacy threats, and they will learn to evaluate and mitigate privacy risks to individuals by investigating system design alternatives. Strategies to mitigating privacy risk will be based on emerging standards and reliable privacy preference data. Students will have the opportunity to study web-, mobile- and cyber-physical systems across a range of domains, including advertising, healthcare, law enforcement and social networking. In addition, students will know how, and when, to interface with relevant stakeholders, including legal, marketing and other developers in order to align software design with privacy policy and law.

08-810 Computational Modeling of Complex Socio-Technical Systems
Spring: 12 units
Social and cultural systems are complex. Whether considering world transforming events such as the Arab Spring or the impact of health care reforms, the interactions among people, technology, and organizations can generate unanticipated outcomes. Computer simulation is a critical methodology for explaining and predicting these events. In this course, the basics of simulation modeling, design, testing and validation are covered. Different simulation approaches are contrasted such as agent-based modeling and system dynamics.
11-291 Applied Computational Intelligence Lab  
Intermittent: 9 units  
What would an "intelligent" picture on the wall do? What if it could see and hear you? What should it say if it could talk? What if your pantry, wardrobe or medicine cabinet could sense, think and act? What should they do and say? What should your cell phone be saying to you? These are not whimsical or theoretical questions...they inevitably arise as ordinary everyday objects around us acquire the ability to sense changes in their environment, think about their implications, and act in pursuit of their goals. These objects are connected to the web and become conduits for services, erasing the distinction between products and services. The ability to invent and build smart products/services is becoming a key skill in the new technology-driven services economy. The focus of the course will be on building "ordinary" objects that can sense, think and act in the real world and on exploring the implications of these capabilities. Students will select their own project and by the end of the semester will create a working prototype that will be exhibited in a public place. Prizes will be offered for the most creative projects. In the course of their projects, students will learn how to use state-of-the-art tools for: Object detection using video cameras, microphones and other sensors Movement and gesture detection Speech recognition and generation Reasoning and planning. While the course organizers have many ideas for specific projects, students will be encouraged to design their own projects. Students are expected to work in small groups on their own time and receive faculty advice as needed. There will be weekly meetings of the whole class.  
Prerequisites: 21-127 Min. grade C and 15-122 Min. grade C  
11-344 Machine Learning in Practice  
Fall: 12 units  
Machine Learning is concerned with computer programs that enable the behavior of a computer to be learned from examples or experience rather than dictated through rules written by hand. It has practical value in many application areas of computer science such as on-line communities and digital libraries. This class is meant to teach the practical side of machine learning for applications, such as mining newspaper data or building adaptive user interfaces. The emphasis will be on learning the process of applying machine learning effectively to a variety of problems rather than emphasizing an understanding of the theory behind what makes machine learning work. This course does not assume any prior exposure to machine learning theory or practice. In the first 2/3 of the course, we will cover a wide range of learning algorithms that can be applied to a variety of problems. In particular, we will cover topics such as decision trees, rule based classification, support vector machines, Bayesian networks, and clustering. In the final third of the class, we will go into more depth on one application area, namely the application of machine learning to problems involving text processing, such as information retrieval or text categorization.  
11-364 An Introduction to Knowledge-Based Deep Learning and Socratic Coaches  
Spring: 12 units  
The subject of this course will be deep learning, one of the most dynamic and exciting emerging areas of computer science. Deep learning deals with and is conquering the problems resulting from the enormous quantity of data that now surrounds us. Furthermore, the course will explore knowledge-based deep learning, a new methodology invented by the instructor that offers many potential advantages over conventional deep learning. This is a learn-by-doing, team-project based course, which will be divided into four phases. In phase one, each student will read and present a number of papers describing state-of-the-art deep learning systems and successful applications. In phase two, each team will implement the system described in one of the papers. In phase three, each team will scale that implementation to one of the large benchmark datasets. In phase four, each team will do a special research project implementing a knowledge-based deep learning system based on pending patent applications of Professor Baker. As a potential follow-on for successful projects, students may participate in a summer course on entrepreneurial applications of deep learning or work as interns in a startup based on the knowledge-based deep learning projects. Prerequisite: Strong quantitative aptitude, programming skill, ability to quickly absorb new ideas, teamwork skills.  
11-411 Natural Language Processing  
Intermittent: 12 units  
This course is about a variety of ways to represent human languages (like English and Chinese) as computational systems, and how to exploit those representations to write programs that do neat stuff with text and speech data, like translation, summarization, extracting information, question answering, natural interfaces to databases, and conversational agents. This field is called Natural Language Processing or Computational Linguistics, and it is extremely multidisciplinary. This course will therefore include some ideas central to Machine Learning and to Linguistics. We'll cover computational treatments of words, sounds, sentences, meanings, and conversations. We'll see how probabilities and real-world text data can help. We'll see how different levels interact in state-of-the-art approaches to applications like translation and information extraction. From a software engineering perspective, there will be an emphasis on rapid prototyping, a useful skill in many other areas of Computer Science.  
Prerequisites: 15-122  
11-421 Grammars and Lexicons  
Fall: 12 units  
Grammars and Lexicons is an introductory graduate course on linguistic data analysis and theory, focusing on methodologies that are suitable for computational implementations. The course covers major syntactic and morphological phenomena in a variety of languages. The emphasis will be on examining both the diversity of linguistic structures and the constraints on variation across languages. Students will be expected to develop and defend analyses of data, capturing linguistic generalizations and making correct predictions within and across languages. The goal is for students to become familiar with the range of phenomena that occur in human languages so that they can generalize the insights into the design of computational systems. The theoretical framework for syntactic and lexical analysis will be Lexical Functional Grammar. Grades will be based on problem sets and take-home exams.  
Spring: 12 units  
TBA  
11-441 Machine Learning for Text Mining  
Fall and Spring: 9 units  
This course provides a comprehensive introduction to the theory and implementation of algorithms for organizing and searching large text collections. The first half of the course studies text search engines for enterprise and Web environments; the open-source Indri search engine is used as a working example. The second half studies text mining techniques such as clustering, categorization, and information extraction. Programming assignments give hands-on experience with document ranking algorithms, categorizing documents into browsing hierarchies, and related topics.  
11-442 Search Engines  
Fall: 12 units  
This course studies the theory, design, and implementation of text-based search engines. The core components include statistical characteristics of text, representation of information needs and documents, several important retrieval models, and experimental evaluation. The course also covers common elements of commercial search engines, for example, integration of diverse search engines into a single search service (“federated search”, “vertical search”), personalized search results, diverse search results, and sponsored search. The software architecture components include design and implementation of large-scale, distributed search engines.  
Course Website: http://boston.lti.cs.cmu.edu/classes/11-642/  
11-492 Speech Processing  
Fall: 12 units  
Speech Processing offers a practical and theoretical understanding of how human speech can be processed by computers. It covers speech recognition, speech synthesis and spoken dialog systems. The course involves practicals where the student will build working speech recognition systems, build their own synthetic voice and build a complete telephone spoken dialog system. This work will be based on existing toolkits. Details of algorithms, techniques and limitations of state of the art speech systems will also be presented. This course is designed for students wishing understand how to process real data for real applications, applying statistical and machine learning techniques as well as working with limitations in the technology.  
Prerequisite: 15-211 Min. grade B
11-610 MCDS Practicum Internship
Fall
The MCDS Practicum course is used for recording CDS students summer internships for the MCDS Program. Section A is used for 7-month internship opportunities Section B is used for Returning Fall Analytic students who DO NOT attain a 7-mo internships. Section R is used to record MCDS students Internship Requirements

11-661 Language and Statistics
Fall: 12 units
Language technologies (search, text mining, information retrieval, speech recognition, machine translation, question answering, biological sequence analysis...) are at the forefront of this century’s information revolution. In addition to their use of machine learning, these technologies rely centrally on classic statistical estimation techniques. Yet most CS and engineering undergraduate programs do not prepare students in this area beyond an introductory prob&stats course. This course is designed to plug this hole. The goal of “Language and Statistics” is to ground the data-driven techniques used in language technologies in sound statistical methodology. We start by familiarizing students with key language technology problems in both an information theoretic framework (the source-channel paradigm) and a Bayesian framework (the Bayes classifier). We then discuss the statistical properties of words, sentences, documents and whole languages, and the computational formalisms used to represent language. These discussions naturally lead to specific concepts in statistical estimation. Topics include: Zipf's distribution and type-token curves; point estimators, Maximum Likelihood estimation, bias and variance, sparseness, smoothing and clustering; interpolation, shrinkage, and backoff; entropy, cross entropy and mutual information; decision tree models applied to language; latent variable models and the EM algorithm; hidden Markov models; exponential models and maximum entropy; semantic modeling and dimensionality reduction; probabilistic context-free grammars and syntactic language models. The course is designed for LTI & SCS graduate students, but others are welcome. CS UG upperclassmen who’ve taken it have done well, though they found it challenging. The 11-661 version does not require the course project. Prerequisites: Strong quantitative aptitude. Comfort with basic UG-level probability. Some programming skill.

Course Website: http://www.cs.cmu.edu/~roni/11661/

11-696 MIIS Capstone Planning Seminar
Spring: 6 units
The MIIS Capstone Planning Seminar prepares students to complete the MIIS Capstone Project in the following semester. Students are organized into teams that will work together to complete the capstone project. They define project goals, requirements, success metrics, and deliverables; and they identify and acquire data, software, and other resources required for successful completion of the project. The planning seminar must be completed in the semester prior to taking the capstone project.

11-711 Algorithms for NLP
All Semesters: 12 units
Algorithms for NLP is an introductory graduate-level course on the computational properties of natural languages and the fundamental algorithms for processing natural language. The course will provide an in-depth presentation of the major algorithms used in NLP, including Lexical, Morphological, Syntactic and Semantic analysis, with the primary focus on parsing algorithms and their analysis.

11-714 Graduate Seminar on Dialog Processing
All Semesters: 6 units
Dialog systems and processes are becoming an increasingly vital area of interest both in research and in practical applications. The purpose of this course will be to examine, in a structured way, the literature in this area as well as learn about ongoing work. The course will cover traditional approaches to the problem, as exemplified by the work of Grosz and Sidner, as well as more recent work in dialog, discourse and evaluation, including statistical approaches to problems in the field. We will select several papers on a particular topic to read each week. While everyone will do all readings, a presenter will be assigned to overview the paper and lead the discussion. On occasion, a researcher may be invited to present their own work in detail and discuss it with the group. A student or researcher taking part in the seminar will come away with a solid knowledge of classic work on dialog, as well as familiarity with ongoing trends.

11-712 Grammars and Lexicons
All Semesters: 12 units
Grammars and Lexicons is an introductory graduate course on linguistic data analysis and theory, focusing on methodologies that are suitable for computational implementations. The course covers major syntactic and morphological phenomena in a variety of languages. The emphasis will be on examining both the diversity of linguistic structures and the constraints on variation across languages. Students will be expected to develop and defend analyses of data, capturing linguistic generalizations and making correct predictions within and across languages. The goal is for students to become familiar with the range of phenomena that occur in human languages so that they can generalize the insights into the design of computational systems. The theoretical framework for this course is an extended version of the typed lambda calculus. Lexical and syntactical analysis will be Lexical Functional Grammar. Grades will be based on problem sets and take-home exams.

11-722 Grammar Formalisms
Intermittent: 12 units
The goal of this course is to familiarize students with grammar formalisms that are commonly used for research in computational linguistics, language technologies, and linguistics. We hope to have students from a variety disciplines (linguistics, computer science, psychology, modern languages, philosophy) in order to cover a broad perspective in class discussions. Comparison of formalisms will lead to a deeper understanding of human language and natural language processing algorithms. The formalisms will include: Head Driven Phrase Structure Grammar, Lexical Functional Grammar, Tree Adjoining Grammar and Categorial Grammar. If time permits, we will cover Penn Treebank, dependency grammar, and Construction Grammar. We will cover the treatment of basic syntactic and semantic phenomena in each formalism, and will also discuss algorithms for parsing and generating sentences for each formalism. If time permits, we may discuss formal language theory and generative capacity. The course is taught jointly by the following faculty of the Language Technologies Institute: Alan Black Alon Lavie Lori Levin (main coordinator)

11-731 Machine Translation and Sequence-to-Sequence Models
Spring: 12 units
Instructors: Graham Neubig. Prerequisites: This course has no official pre-requisites, although 11-711 “Algorithms for NLP” or 10-701 “Machine Learning” would be helpful. Course Description: Machine Translation and Sequence-to-Sequence Models is an introductory graduate-level course surveying the primary approaches and methods for developing systems to translate between human languages, or other sequential data. The main objective of the course is to obtain basic understanding and implementation skills for modern methods for MT and sequence transduction, including how to design models, how to learn the model parameters, how to search for the best output, and how to create training data. The course will focus on machine translation, but also briefly cover tasks such as dialog response generation, image caption generation, and others.

11-741 Machine Learning for Text Mining
Fall and Spring: 12 units
This course studies the theory, design, and implementation of text-based information systems. The Information Retrieval core components of the course include statistical characteristics of text, representation of information needs and documents, several important retrieval models (Boolean, vector space, probabilistic, inference net, language modeling), clustering algorithms, automatic text categorization, and experimental evaluation. The software architecture components include design and implementation of high-capacity text retrieval and text filtering systems. A variety of current research topics are also covered, including cross-lingual retrieval, document summarization, machine learning, topic detection and tracking, and multi-media retrieval. Prerequisites: Programming and data-structures at the level of 15-212 or higher. Algorithms comparable to the undergraduate CS algorithms course (15-451) or higher. Basic linear algebra (21-241 or 21-341). Basic statistics (36-202) or higher.

11-751 Speech Recognition and Understanding
All Semesters: 12 units
The technology to allow humans to communicate by speech with machines or by which machines can understand when humans communicate with each other is rapidly maturing. This course provides an introduction to the theoretical tools as well as the experimental practice that has made the field what it is today. We will cover theoretical foundations, essential algorithms, major approaches, experimental strategies and current state-of-the-art systems and will introduce the participants to ongoing research, representation, algorithms and interface design. This course is suitable for graduate students with some background in computer science and electrical engineering, as well as for advanced undergraduates. Prerequisites: Sound mathematical background, knowledge of basic statistics, good programming skills. No prior experience with speech recognition is necessary. This course is primarily for graduate students in LTI, CS, Robotics, ECE, Psychology, or Computational Linguistics. Others by prior permission of instructor.
11-752 Speech II: Phonetics, Prosody, Perception and Synthesis
Spring: 12 units
The goal of the course is to give the student basic knowledge from several fields that is necessary in order to pursue research in automatic speech processing. The course will begin with a study of the acoustic content of the speech signal. The students will use the spectrographic display to examine the signal and discover its variable properties. Phones in increasingly larger contexts will be studied with the goal of understanding coarticulation. Phonological rules will be studied as a contextual aid in understanding the spectrographic display. The spectrogram will then serve as a first introduction to the basic elements of prosody. Other displays will then be used to study the three parts of prosody: amplitude, duration, and pitch. Based on these three elements, the student will then examine how the three interact in careful and spontaneous speech. Next, the students will explore perception. Topics covered will be: physical aspects of perception, psychological aspects of perception, testing perception processes, practical applications of knowledge about perception. The second part of this course will cover all aspects of speech synthesis. Students need only have a basic knowledge of speech and language processing. Some degree of programming and statistical modelling will be beneficial, but not required. Taught every other year

11-761 Language and Statistics
Fall: 12 units
Language technologies (search, text mining, information retrieval, speech recognition, machine translation, question answering, biological sequence analysis,...) are at the forefront of this century's information revolution. In addition to their use in machine learning, these technologies rely centrally on classic statistical estimation techniques. Yet most CS and engineering undergraduate programs do not prepare students in this area beyond an introductory prob&stats course. This course is designed to plug this hole. The goal of "Language and Statistics" is to ground the data-driven techniques used in language technologies in sound statistical methodology. We start by formulating various language technology problems in both an information theoretic framework (the source-channel paradigm) and a Bayesian framework (the Bayes classifier). We then discuss the statistical properties of words, sentences, documents and whole languages, and the computational formalisms used to represent language. These discussions naturally lead to specific concepts in statistical estimation. Topics include: Zipf's distribution and type-token curves; point estimators, Maximum Likelihood estimation, bias and variance, sparseness, smoothing and clustering; interpolation, shrinkage, and backoff; entropy, cross entropy and mutual information; decision tree models applied to language; latent variable models and the EM algorithm; hidden Markov models; exponential models and maximum entropy; semantic modeling and dimensionality reduction; probabilistic context-free grammars and syntactic language models. The course is designed for LTI & SCS graduate students, but others are welcome. CS UG upperclassmen who've taken it have done well, though they found it challenging. The 11-661 version does not require the course project. Prerequisites: Strong quantitative aptitude. Comfort with basic UG-level probability. Some programming skill. Course Website: http://www.cs.cmu.edu/~roni/11761/

11-762 Language and Statistics II
Fall: 12 units
This course will cover modern empirical methods in natural language processing. It is designed for language technologies students who want to understand statistical methodology in the language domain, and for machine learning students who want to know about current problems and solutions in text processing. Students will, upon completion, understand how statistical modeling and learning can be applied to text, be able to develop and apply new statistical models for problems in their own research, and be able to critically read papers from the major related conferences (EMNLP and ACL). A recurring theme will be the tradeoffs between computational cost, mathematical elegance, and applicability to real problems. The course will be organized around methods, with concrete tasks introduced throughout. The course is designed for SCS graduate students. Prerequisite: Language and Statistics (11-761) or permission of the instructor. Recommended: Algorithms for Natural Language Processing (11-711), Machine Learning (15-681, 15-781, or 11-746). Prerequisite: 11-761

11-792 Intelligent Information Systems Project
Spring: 12 units
The Software Engineering for IS sequence combines classroom material and assignments in the fundamentals of software engineering (11-791) with a self-paced, faculty-supervised directed project (11-792). The two courses cover all elements of project design, implementation, evaluation, and documentation. Students may elect to take only 11-791; however, if both parts are taken, they should be taken in proper sequence. Prerequisite: 11-791. The course is required for VLS students. Prerequisites: 15-393 or 11-791

11-927 MIIS Capstone Project
Fall: 36 units
The capstone project course is a group-oriented demonstration of student skill in one or more areas covered by the degree. Typically the result of the capstone project is a major software application. The capstone project course consists of two components. The classroom component guides students in project planning, team management, development of requirements and design specifications, and software tools for managing group-oriented projects. The lab component provides project-specific technical guidance and expertise, for example in the development of a question answering system, dialog, or sentiment analysis application. Thus, each project receives two types of supervision, often from two separate members of the faculty.

Machine Learning Courses

10-401 Introduction to Machine Learning (Undergrad)
Fall and Spring: 12 units
Machine learning is a subfield of computer science with the goal of exploring, studying, and developing learning systems, methods, and algorithms that can improve their performance with learning from data. This course is designed to give undergraduate students a one-semester-long introduction to the main principles, algorithms, and applications of machine learning. Topics. The topics of this course will be in part parallel with those covered in the graduate machine learning courses (10-715, 10-701, 10-601), but with a greater emphasis on applications and case studies in machine learning. After completing the course, students will be able to: *select and apply an appropriate supervised learning algorithm for classification problems (e.g., naive Bayes, perceptron, support vector machine, logist regression). *select and apply an appropriate supervised learning algorithm for regression problems (e.g., linear regression, ridge regression). *recognize different types of unsupervised learning problems, and select and apply appropriate algorithms (e.g., clustering, linear and nonlinear dimensionality reduction). Work with probabilities (Bayes rule, conditioning, expectations, independence), linear algebra (vector and matrix operations, eigenvectors, SVD), and calculus (gradients, Jacobians) to derive machine learning methods such as linear regression, naive Bayes, and principal components analysis. *understand machine learning principles such as model selection, overfitting, and underfitting, and techniques such as cross-validation and regularization. *implement machine learning algorithms such as logistic regression via stochastic gradient descent, linear regression (using a linear algebra toolbox), perceptron, or k-means clustering. *run appropriate supervised and unsupervised learning algorithms on real and synthetic data sets and interpret the results. Prerequisites: 15-122 Min. grade C and (21-128 Min. grade C or 21-127 Min. grade C or 15-151 Min. grade C) and (36-217 Min. grade C or 21-325 Min. grade C or 15-359 Min. grade C or 36-225 Min. grade C)

10-403 Deep Reinforcement Learning & Control
Spring: 12 units
TBD

10-600 Mathematical background for Machine Learning
Fall and Spring: 6 units
This course was superseded by the two mini-courses, 10-606 and 10-607.
10-601 Introduction to Machine Learning (Master's)

Fall and Spring: 12 units

Machine Learning (ML) develops computer programs that automatically improve their performance through experience. This includes learning many types of tasks based on many types of experience, e.g., spotting high-risk medical patients, recognizing speech, classifying text documents, detecting credit card fraud, or driving autonomous vehicles. 10-601 covers all or most of: concept learning, decision trees, neural networks, linear learning, active learning, estimation & the bias-variance tradeoff, hypothesis testing, Bayesian learning, the MLD principle, the Gibbs classifier, Naive Bayes, Bayes Nets & Graphical Models, the EM algorithm, Hidden Markov Models, K-Nearest-Neighbors and nonparametric learning, reinforcement learning, bagging, boosting and discriminative training. Grading will be based on weekly or biweekly assignments (written and/or programming), a midterm, a final exam, and possibly a project (details may vary depending on the section). 10-601 is recommended for CS Seniors & Juniors, quantitative Masters students, & non-MLD PhD students. Prerequisites (strictly enforced): strong quantitative aptitude, college prob&stats course, and programming proficiency. For learning to apply ML practically & effectively, without the above prerequisites, consider 11.344/05834 instead. If you are unsure whether you have sufficient mathematical background to do well in this course, you should consider taking the minis 10-606/10-607 Mathematical Background for Machine Learning. You can evaluate your ability to take the course via a self-assessment exam at: https://eno-apj.apps.spspot.com/view.html?9aizfnFuYS1hcHByGQsSDFF1ZXN0aW9uYXlGZwBIaGlcg0-XCgw

ML course comparison: https://goo.gl/mrnR2EL

Prerequisites: 15-122 Min. grade C and (15-151 Min. grade C or 21-127 Min. grade C) and (36-217 Min. grade C or 21-325 Min. grade C or 15-359 Min. grade C or 36-225 Min. grade C)

10-605 Machine Learning with Large Datasets

Spring: 12 units

Large datasets are difficult to work with for several reasons. They are difficult to visualize, and it is difficult to understand what sort of errors and biases are present in them. They are computationally expensive to process, and often the cost of learning is hard to predict - for instance, and algorithm that runs quickly in a dataset that fits in memory may be exorbitantly expensive when the dataset is too large for memory. Large datasets may also display qualitatively different behavior in terms of which learning methods produce the most accurate predictions. This course is intended to provide a student practical knowledge of, and experience with, the issues involving large datasets. Among the issues considered are: scalable learning techniques, such as streaming machine learning techniques; parallel infrastructures such as map-reduce; practical techniques for reducing the memory requirements for learning methods, such as feature hashing and Bloom filters; and techniques for analysis of programs in terms of memory, disk usage, and (for parallel methods) communication complexity. The class will include programming assignments, and a one-month short project chosen by the student. The project will be designed to compare the scalability of variant learning algorithms on datasets. An introductory course in machine learning, like 10-601 or 10-701, is a prerequisite or a co-requisite. If you are interested in taking this course and 10-601 concurrently please tell the instructor. The course will include several substantial programming assignments, so an additional prerequisite is 15-211, or 15-214, or comparable familiarity with java and good programming skills.

Prerequisites: 15-210 or 15-214

Course Website: http://goo.gl/W2kPqO

10-607 Mathematical Background for Machine Learning II

Fall and Spring: 6 units

This course provides a place for students to practice the necessary mathematical background for further study in machine learning — particularly for taking 10-601 and 10-701. Topics covered include probability, linear algebra (inner product spaces, linear operators), multivariate differential calculus, optimization, and likelihood functions. The course assumes some background in each of the above, which would be impossible in a course of this length. Some coding will be required: the course will provide practice with translating the above mathematical concepts into concrete programs. The Mathematical Background for Machine Learning sequence is split into two minis, with 10-606 being a prerequisite for 10-607.

Prerequisite: 10-606

10-701 Introduction to Machine Learning (PhD)

Fall and Spring: 12 units

Machine learning studies the question “How can we build computer programs that automatically improve their performance through experience?” This includes learning to perform many types of tasks based on many types of experience. For example, it includes robots learning to better navigate based on experience gained by roaming their environments, medical decision aids that learn to predict which therapies work best for which diseases based on data mining of historical health records, and speech recognition systems that learn to better understand your speech based on experience listening to you. This course is designed to give PhD students a thorough grounding in the methods, mathematics and algorithms needed to do research and applications in machine learning. Students entering the class with a pre-existing working knowledge of probability, statistics and algorithms will be at an advantage, but the class has been designed so that anyone with a strong numerate background can catch up and fully participate. You can evaluate your ability to take the course via a self-assessment exam that will be made available to you after you register. If you are interested in this topic, but are not a PhD student, or are a PhD student not specializing in machine learning, you might consider taking the master’s level course on Machine Learning, 10-601. This class may be appropriate for MS and undergrad students who are interested in the theory and algorithms behind ML. If you are unsure whether you have sufficient mathematical background to do well in this course, you should consider taking the minis 10-606/10-607 Mathematical Background for Machine Learning. You can evaluate your ability to take the course via a self-assessment exam at: https://goo.gl/mrnR2EL

Prerequisites: 15-122 Min. grade C and (15-151 Min. grade C or 21-127 Min. grade C) and (36-225 Min. grade C or 21-325 Min. grade C or 15-359 Min. grade C)

10-703 Deep Reinforcement Learning & Control

Spring: 12 units

This course will cover latest advances in Reinforcement Learning and Control, such as, deep Q learning, actor-critic methods, learning and planning, concurrent trajectory optimization and policy learning, inverse reinforcement learning, hierarchical reinforcement learning methods, forward predictive models, deep model predictive control, exploration strategies, adaptive control, applications to deep robotic learning. By the end of the course you should be able to: 1) code up a suitable reinforcement learning method in simulation or on robotic platform for a task 2) identify what are easy and hard problems in RL and learning for robotics The course will have a final project which will involve design of a reinforcement learning method in simulation or robotic platform. The homeworks will be in OpenAI gym. Pre-requisites: Students should have a basic background in algorithms, linear algebra, machine Learning, deep learning.
10-707 Topics in Deep Learning
Fall: 12 units
Building intelligent machines that are capable of extracting meaningful representations from high-dimensional data lies at the core of solving many AI related tasks. In the past few years, researchers across many different communities, from applied statistics to engineering, computer science and neuroscience, have developed deep (hierarchical) models — models that are composed of several layers of nonlinear processing. An important property of these models is that they can learn useful representations by re-using and combining intermediate concepts, allowing these models to be successfully applied in a wide variety of domains, including visual object recognition, information retrieval, natural language processing, and speech perception. This is an advanced graduate course, designed for Master’s and Ph.D. level students, and will assume a reasonable degree of mathematical maturity. The goal of this course is to introduce students to the recent and exciting developments of various deep learning methods. Some topics to be covered include: restricted Boltzmann machines (RBMs) and their multi-layer extensions Deep Belief Networks and Deep Boltzmann machines; sparse coding, autoencoders, variational autoencoders, convolutional neural networks, recurrent neural networks, generative adversarial networks, and attention-based models with applications in vision, NLP, and multimodal learning. We will also address mathematical issues, focusing on efficient large-scale optimization methods for inference and learning, as well as training density models with intractable partition functions. Prerequisite: ML: 10-701 or 10-715, and strong programming skills. Prerequisites: 10-701 Min. grade C or 10-401 Min. grade C or 10-715 Min. grade C or 10-601 Min. grade C.

10-715 Advanced Introduction to Machine Learning
Fall: 12 units
The rapid improvement of sensory techniques and processor speed, and the availability of inexpensive massive digital storage, have led to a growing demand for systems that can automatically comprehend and mine massive and complex data from diverse sources. Machine Learning is becoming the primary mechanism by which information is extracted from Big Data, and a primary pillar that Artificial Intelligence is built upon. This course is designed for Ph.D. students whose primary field of study is machine learning, or who intend to make machine learning methodological research a main focus of their thesis. It will give students a thorough grounding in the algorithms, mathematics, theories, and insights needed to do in-depth research and applications in machine learning. The topics of this course will in part parallel those covered in the general graduate machine learning course (10-701), but with a greater emphasis on depth in theory and algorithms. The course will also include additional advanced topics such as privacy in machine learning, interactive learning, reinforcement learning, online learning, Bayesian nonparametrics, and additional material on graphical models. Students entering the class are expected to have a pre-existing strong working knowledge of algorithms, linear algebra, probability, and statistics. If you are interested in this topic, but do not have the required background or are not planning to work on a PhD thesis with machine learning as the main focus, you might consider the general graduate Machine Learning course (10-701) or the Masters-level Machine Learning course (10-601). ML course comparison: https://goo.gl/mmR2EL
Course Website: https://sites.google.com/site/107152016/

10-806 Foundations of Machine Learning and Data Science
Fall: 12 units
This course will cover fundamental topics in Machine Learning and Data Science, including powerful algorithms with provable guarantees for making sense of and generalizing from large amounts of data. The course will start by providing a basic arsenal of useful statistical and computational tools, including generalization guarantees, core algorithmic methods, and fundamental analysis models. We will examine questions such as: Under what conditions can we hope to meaningfully generalize from limited data? How can we best combine different kinds of information such as labeled and unlabeled data, leverage multiple related learning tasks, or leverage multiple types of features? What can we prove about methods for summarizing and making sense of massive datasets, especially under limited memory? We will also examine other important constraints and resources in data science including privacy, communication, and taking advantage of limited interaction. In addressing these and related questions we will make connections to statistics, algorithms, linear algebra, complexity theory, information theory, optimization, game theory, and empirical machine learning research. Topics to be covered will include: - Fundamental measures of complexity for generalization, including VC-dimension and Rademacher complexity. - Core algorithmic tools including boosting, regularization, and online optimization with connections to game theory. - Spectral methods, streaming algorithms and other approaches for handling massive data. - Foundations and algorithms for addressing important constraints or externalities such as privacy, limited memory, and communication constraints. - Foundations for modern learning paradigms including semi-supervised learning, never-ending learning, interactive learning, and deep learning.
Course Website: http://www.cs.cmu.edu/~ninamf/courses/806/10-806-index.html

10-807 Topics in Deep Learning
Fall: 12 units
Building intelligent machines that are capable of extracting meaningful representations from high-dimensional data lies at the core of solving many AI related tasks. In the past few years, researchers across many different communities, from applied statistics to engineering, computer science and neuroscience, have developed deep (hierarchical) models — models that are composed of several layers of nonlinear processing. An important property of these models is that they can learn useful representations by re-using and combining intermediate concepts, allowing these models to be successfully applied in a wide variety of domains, including visual object recognition, information retrieval, natural language processing, and speech perception. This is an advanced graduate course, designed for Master’s and Ph.D. level students, and will assume a reasonable degree of mathematical maturity. The goal of this course is to introduce students to the recent and exciting developments of various deep learning methods. Some topics to be covered include: restricted Boltzmann machines (RBMs) and their multi-layer extensions Deep Belief Networks and Deep Boltzmann machines; sparse coding, autoencoders, variational autoencoders, convolutional neural networks, recurrent neural networks, generative adversarial networks, and attention-based models with applications in vision, NLP, and multimodal learning. We will also address mathematical issues, focusing on efficient large-scale optimization methods for inference and learning, as well as training density models with intractable partition functions. Prerequisite: ML: 10-701 or 10-715, and strong programming skills. Prerequisites: 10-715 Min. grade C or 10-701 Min. grade C.

10-822 Presentation Skills
Fall and Spring: 6 units
This course provides a forum for students to learn and refine public speaking and technical reading skills. The course will include brief workshops embedded throughout the semester to cover such things as effective structure of presentations and papers, how to give a short talk (think NIPS spotlights), “elevator” talks, structure of a research paper, conference presentations, proposal writing (think thesis and beyond), slide crafting, posters, critical evaluation, and public communications for research. Students will be expected to prepare and present a number of practice talks throughout the semester.
Robotics Courses

16-161 ROB Freshman Seminar: Artificial Intelligence and Humanity
Fall and Spring: 9 units
In 1965 British mathematician I. J. Good wrote, "An ultraintelligent machine could design even better machines; there would then unquestionably be an intelligence explosion, and the intelligence of man would be left far behind. As we enter an age where companies like Uber are testing driverless cars in Pittsburgh and innovative interfaces like IBM’s Watson can play Jeopardy and learn techniques for medical diagnoses, how are we to negotiate an intelligence explosion that for many individuals might threaten the very notions of what it means to be human? The future of human-to-machine relationships will likely define our historical epoch and yet, many young technologists and humanists underestimate the downstream impact of technological innovations on human society. Presently, we have little choice but to attend to this rapidly anxiety-ridden question. This seminar will attend to the challenge of present existential questions on what it means to be human (read not machine) in the context of a rapidly advancing technological age. We will consider human narratives throughout history that exam how government and individual citizens defined humanity in the context of slavery and colonialism as a framework for exploring and projecting what it means to be human in the age of rapidly advancing intelligent machines. We will trace the technological advancements of the recent five decades and identify historical precedents and speculative narratives that help us to consider issues like labor, economic disparity, negotiations of power, privacy, dignity and ethical responsibility within the context of human relations with advancing technological tools that are now coined, artificial intelligence.

16-223 Introduction to Physical Computing
Fall: 10 units
Physical computing refers to the design and construction of physical systems that use a mix of software and hardware to sense and respond to the surrounding world. Such systems blend digital and physical processes into toys and gadgets, kinetic sculpture, functional sensing and assessment tools, mobile instruments, interactive wearables, and more. This is a project-based course that deals with all aspects of conceiving, designing and developing projects with physical computing: the application, the artifact, the computer-aided design environment, and the physical prototyping facilities. The course is organized around a series of practical hands-on exercises which introduce the fundamentals of circuits, embedded programming, sensor signal processing, simple mechanisms, actuation, and time-based behavior. The key objective is gaining an intuitive understanding of how information and energy move between the physical, electronic, and computational domains to create a desired behavior. The exercises provide building blocks for collaborative projects which utilize the essential skills and challenge students to not only consider how to make things, but also for whom we design, and why the making is worthwhile. This course is an IDEaTE Portal Course for entry into either of the IDEaTE Intelligent Environments or Physical Computing programs. CFA/DC/TPR students can enroll under 16-223: CTF/ICS/SCS students can enroll in the 60-223 version of this course. Please note that there will be lab usage and materials fees associated with this course.

16-264 Humanoids
Spring: 12 units
This course surveys perception, cognition, and movement in humans, humanoid robots, and humanoid graphical characters. Application areas include more human-like robots, video game characters, and interactive movie characters.
Course Website: http://www.cs.cmu.edu/~cga/humanoids-grad/

16-299 Introduction to Feedback Control Systems
Spring: 12 units
This course is designed as a first course in feedback control systems for computer science majors. Course topics include classical linear control theory (differential equations, Laplace transforms, feedback control), linear state-space methods (controllability/observability, pole placement, LQR), nonlinear systems theory, and an introduction to control using computer learning techniques. Priorities will be given to computer science majors with robotics minor.
Prerequisites: 21-122 and 15-122

16-311 Introduction to Robotics
Spring: 12 units
This course presents an overview of robotics in practice and research with topics including vision, motion planning, mobile mechanisms, kinematics, inverse kinematics, and sensors. In course projects, students construct robots which are driven by a microcontroller, with each project reinforcing the basic principles developed in lectures. Students nominally work in teams of three: an electrical engineer, a mechanical engineer, and a computer scientist. This course will also expose students to some of the contemporary happenings in robotics, which includes current robot lab research, applications, robot contests and robots in the news.
Prerequisites: 18-202 Min. grade C or 22-140 Min. grade C or 24-311 Min. grade C or 21-260 Min. grade C or 21-241 Min. grade C
Course Website: http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/16311/www/current/

16-350 Planning Techniques for Robotics
Spring: 12 units
Planning is one of the core components that enable robots to be autonomous. Robot planning is responsible for deciding in real-time what should the robot do next, how to do it, where should the robot move next and how to move there. This class focuses on autonomous planning techniques in robotics and examines their use in ground and aerial robots, humanoids, mobile manipulation platforms and multi-robot systems. The students learn the theory of these methods and also implement them in a series of programming-based projects. To take the class students should have taken an intro to Robotics class and have a good knowledge of programming and data structures.
Course Website: http://www.cs.cmu.edu/~maxim/classes/robotplanning/

16-362 Mobile Robot Programming Laboratory
Fall: 12 units
This course is a comprehensive hands-on introduction to the concepts and basic algorithms needed to make a mobile robot function reliably and effectively. We will work in small groups with small robots that are controlled over wireless from your laptop computers. The robots are Neato household vacuum robots that have been converted to mini forklifts that can move pallets from place to place just like commercial automated guided vehicles do today. The robots are programmed in the modern MATLAB programming environment. It is a pretty easy language to learn, and a very powerful one for prototyping robotics algorithms. You will get a lot of experience in this course in addition to some theory. Lectures are focused on the content of the next lab. There is a lab every week and they build on each other so that complete robot software system results. The course will culminate with a class-wide robot competition that tests the performance of all of your code implemented in the semester. In order to succeed in the course, students must have a 2nd year science/engineering level background in mathematics (matrices, vectors, coordinate systems) and have already mastered at least one procedural programming language like C or Java. When the course is over, you will have written a single software system that has been incrementally extended in functionality and regularly debugged throughout the semester.
Course Website: http://www.frc.ri.cmu.edu/~alonzo/teaching/16x62/16x62.html

16-371 Personalized Responsive Environments
Spring: 9 units
[IDEaTE collaborative course]. Environmental factors have a significant impact on mood and productivity. Creating responsive environments necessitates the design of surroundings that are able to metamorphose in order to optimize user strengths and available resources and evolve in stride with user needs. This course will investigate the development of spaces that adapt to user preferences, moods, and task specific demands. Both the design and engineering of such personalized environments will be explored. Central course concepts will include, understanding the user, integrating various modalities (e.g., light, heat, sound) to support the changing needs of task and user, and the creation of adaptive environments that learn user preferences over time. Please note that there may be usage/materials fees associated with this course.
Prerequisites: 60-223 Min. grade C or 62-150 Min. grade C or 15-104 Min. grade C or 18-090 Min. grade C
16-374 IdeATe: Art of Robotic Special Effects
Spring: 12 units

Inspired by the early “trick” films of George Melies, this project-oriented course brings together robotics and film production technique to infuse cinema with the wonder of live magic. Students will learn the basics of film production using animatronics, camera motion control, and compositing. The projects apply these techniques to create innovative physical effects for short films, all the way from concept to post-production. The course emphasizes real-time practical effects to explore the immediacy and interactivity of improvisation and rehearsal. The robotics topics include animatronic rapid prototyping and programming human-robot collaborative performance. The course includes a brief overview of the history of special effects and robotics to set the work in context.

Course Website: https://courses.ideate.cmu.edu/16-374

16-375 Robotics for Creative Practice
Fall: 10 units
[IdeATe collaborative course]. This project-oriented course brings art and engineering together into making machines which are surprisingly animate. Students will iterate their concepts through several small projects focused on using embodied behavior as a creative medium for storytelling, performance, and human interaction. Students will learn skills for designing, constructing and programming simple robot systems, then exploring their results through exhibition and performance. Technical topics include systems thinking, dynamic physical and computational behavior, autonomy, embedded programming, and fabrication and deployment. Discussion topics include both contemporary kinetic sculpture and robotics research. Please note that there may be usage/materials fees associated with this course. Prerequisites: 15-104 or 60-223 or 62-150 or 16-223

16-384 Robot Kinematics and Dynamics
Fall: 12 units
Foundations and principles of robotic kinematics. Topics include transformations, forward kinematics, inverse kinematics, differential kinematics (Jacobians), manipulability, and basic equations of motion. Course also include programming on robot arms. Prerequisites: 15-122 Min. grade C or 16-311 or 24-131 or 18-202

16-385 Computer Vision
Spring: 9 units
Basic concepts in machine vision, including sensing and perception, 2D image analysis, pattern classification, physics-based vision, stereo and motion, and object recognition. Prerequisites: (18-202 Min. grade C and 15-122 Min. grade C) or (15-122 Min. grade C and 21-241 Min. grade C and 21-259 Min. grade C)

16-389 Art, Conflict and Technology in Northern Ireland
Spring: 12 units
Art, Conflict and Technology in Northern Ireland is a 12-unit course cross-listed between the School of Art, the Department of English, and the Robotics Institute. If you have ever considered how artists explore societal strife through their writing or visual arts practice, if you are interested in the social and political influences of evolving technology, or if you are a practicing artist who uses advancing technology as a tool for individual expression, this integrative course is for you. Throughout the semester we will examine the practice of a range of visual artists that include Rita Duffy, John Kindness and Willie Doherty and writers and dramatists like Dermot Healy, Patrick McCabe, and Christina Reid. Students will learn how to use CREATE Lab’s Gigapan and Hear Me systems as platforms for exploring the content presented in the class for the development of final projects. We will travel to Belfast for spring break 2015, to meet a variety of writers and artists whose work we will study, and stakeholders in the reconciliation efforts throughout the region. In addition to weekly lectures on Thursdays throughout the term, students will have a six-week lab on Tuesdays. Lab sessions begin in the second week of classes (January 20).

16-421 Vision Sensors
Spring: 12 units
This course covers the fundamentals of vision cameras and other sensors - how they function, how they are built, and how to use them effectively. The course presents a journey through the fascinating five hundred year history of “camera-making” from the early 1500’s “camera obscura” through the advent of film and lenses, to today’s mirror-based and solid state devices (CCD, CMOS). The course includes a significant hands-on component where students learn to use the sensors and understand, model and deal with the uncertainty (noise) in their measurements. While the first half of the course deals with conventional “single viewpoint” or “perspective” cameras, the second half of the course covers much more recent “multi-viewpoint” or “multi-perspective” cameras that includes a host of lenses and mirrors. Prerequisites: 21-241 and 21-111

Course Website: http://www.cs.cmu.edu/~ilIM/courses/vision-sensors/

16-423 Designing Computer Vision Apps
Fall: 12 units
Computer vision is a discipline that aims to extract information from images and videos. Nearly every smart device on the planet has a camera, and people are increasingly interested in how to develop apps that use computer vision to perform an ever expanding list of things including: 3D mapping, photo/image search, people/object tracking, augmented reality etc. This course is intended for students who are not familiar with computer vision, but want to come up to speed rapidly with the latest in environments, software tools and best practices for developing computer vision apps. No prior knowledge of computer vision or machine learning is required although a strong programming background is a must (at a minimum good knowledge of C/C++). The topics will include using conventional computer vision software tools (OpenCV, MATLAB toolboxes, VLFeat, CAFFE), and development on iOS devices using mobile vision libraries such as GPUImage and fast math libraries like Armadillo and Eigen. For consistency, all app development will be in iOS and it is expected that all students participating in the class have access to an Intel-based MAC running OS X Mavericks or later. Although the coursework will be focussed on a single operating system, the knowledge gained from this class is intended to generalize to other mobile platforms such as Android etc. Prerequisites: (21-240 and 15-213) or (21-241 and 15-213) or (18-213 and 18-202)

Course Website: http://16423.courses.cs.cmu.edu

16-441 Advanced CP/SIS: Urban Intervention
Fall and Spring: 12 units
This course introduces students to theories, practices, and communities for critical investigation of urban spaces and play within them. The course unfolds along two parallel trajectories: research (literature review, lectures, readings, demonstrations) and design (three iterated individualized projects and a fourth larger scale final project). The first half of the course will introduce students to a wide range of theories and techniques within urban intervention that draw from fluxus, the situationist international, activism and hacktivism, as well as public policy, philosophy, psychology and economics. Students will study theoretical and practical frameworks for artistic intervention into public urban spaces, while concurrently researching actual sites and communities within Pittsburgh for experimentation. Students are required to conceptualized projects on larger (urban) scales, and find ways to implement their projects safely and legally by pursuing the necessary administrative, social, technical, financial steps required to create meaningful interventions in public spaces. This class will specifically explore three media for urban intervention: Sound Outdoor video projection Robotics, Autonomy and Mobility in the way of remote control vehicles (e.g. cars, quad-copters, etc.). For each theme, students are required to produce one project that is iterated twice or more. The undergraduate (60441) and graduate (60741) sections of the course meet concurrently and follow the same syllabus and assignments. In addition to the coursework documented in the syllabus, Graduate level students are expected to write a research paper suitable for submission to a notable relevant academic conference. This process includes a rough draft, revisions and a completed and formatted paper ready for submission.
16-450 Robotics Systems Engineering
Fall: 12 units
Systems engineering examines methods of specifying, designing, analyzing and testing complex systems. In this course, principles and processes of systems engineering are introduced and applied to the development of robotic devices. The focus is on robotic system engineered to perform complex behavior. Such systems embed computing elements, integrate sensors and actuators, operate in a reliable and robust fashion, and demand rigorous engineering from conception through production. The course is organized as a progression through the systems engineering process of conceptualization, specification, design, and prototyping with consideration of verification and validation. Students completing this course will engineer a robotic system through its compete design and initial prototype. The project concept and teams can continue into the Spring-semester (16-474 Robotics Capstone) for system refinement, testing and demonstration. Prerequisites: 16-311 Min. grade B and (18-370 Min. grade B or 16-299 Min. grade B or 24-451 Min. grade B)

16-455 Human-Machine Virtuosity
Spring: 12 units
[IdAte course] Human dexterous skill embodies a wealth of physical understanding which complements computer-based design and machine fabrication. This project-oriented course explores the duality between hand and machine through the practical development of innovative design and fabrication systems. These systems fluidly combine the expressivity and intuition of physical tools with the scalability and precision of the digital realm. Students will develop novel hybrid design and production workflows combining analog and digital processes to support the design and fabrication of their chosen projects. Specific skills covered include 3D scanning, 3D modeling (CAD), 3D printing (additive manufacturing), computer based sensing, and human-robot interaction design. Areas of interest include architecture, art, and product design. Prerequisites: 16-223 Min. grade C or 60-223 Min. grade C

Course Website: https://courses.idate.cmu.edu/16-455

16-456 Reality Computing Studio
Spring: 12 units
[IdAte collaborative course] Reality computing encompasses a constellation of technologies focused around capturing reality (laser scanning, photogrammetry), working with spatial data (CAD, physical modeling, simulation), and using data to interact with and influence the physical world (augmented / virtual reality, projector systems, 3d printing, robotics). Taught in collaboration with the school of architecture, this studio asks students to apply these technologies to real world problems such as residential design, sustainability, and infrastructure monitoring.

Course Website: http://ideate.cmu.edu/about-ideate/departments/robotics-institute/reality-computing/

16-457 Reality Computing II
Spring: 12 units
[IdAte collaborative course] Reality computing encompasses a constellation of technologies focused around capturing reality (laser scanning, photogrammetry), working with spatial data (CAD, physical modeling, simulation), and using data to interact with and influence the physical world (augmented / virtual reality, projector systems, 3d printing, robotics). This iteration of the reality computing course will focus on “design realization”: the translation from digital design to fully realized tangible artifact. Collaborating with the UDBS design studio, and within the context of a full-scale residential prototype, students will investigate how reality computing technologies can be used to accelerate and advance the process of design realization by using reality computing to understand existing homes, map design data into the real world, and highlight conflicts between design and reality. Topics of special focus within the course are residential design (John Folan) and augmented reality and robotics (Pyry Matikainen).

Course Website: http://ideate.cmu.edu/about-ideate/departments/robotics-institute/reality-computing/

16-461 Experimental Capture
Fall: 9 units
Performance capture is used in applications as varied as special effects in movies, animation, sports training, physical rehabilitation, and human-robot/human-computer interaction. This course will survey state-of-the-art techniques and emerging ideas, in the industry and in academia, to capture, model, and render human performances. The course will be a mix between lectures and discussion of recent progress in human motion capture and analysis. The course evaluation will be project-based, in which students will capture their own body and face motion, and build projects around the data they collect individually and as a group. We will cover: 1. Capture Techniques: We will describe and use various systems including motion capture, video-based capture, depth sensors, scanners, and eye-gaze trackers; 2. Modeling and Representation: We will cover classic and contemporary representations of face and body pose and motion, including statistical and physics-based techniques; 3. Rendering Applications: As new-rendering paradigms emerge, new applications continue to develop. We will study recent progress in animation, synthesis, classification, and rehabilitation on new forms of displays. Please note that there may be usage/materials fees associated with this course.

Prerequisites: 15-365 or 60-422

16-465 Game Engine Programming
Spring: 10 units
This course is designed to help students understand, modify, and develop game engines. Game engines consist of reusable runtime and asset pipeline code. They provide game-relevant abstractions of low-level system services and libraries, making it easier to write bug-free games that work across multiple platforms. Game engines also handle artistic content, providing or integrating with authoring tools to ease the process of creating high-fidelity games. In this course, we will discuss the problems game engines attempt to solve, examine how current state-of-the-art engines address these problems, and create our own engines based on what we learn. We will cover both the content authoring and runtime aspects of engines. Coursework will consist of frequent, tightly-scoped programming and system design assignments; expeditions through game engine source code; and two group projects — one in an engine created from scratch, and one that requires modification of an existing engine. Prerequisites: Students will be expected to be fluent in at least one programming language. We will be working with C++, javascript, and a smattering of Python. We will be using git for version control and code sharing. The assignments in the course will be designed to be completed on an OSX or Linux workstation (e.g. the DeMeTe “virtual cluster”). Working with Windows will be possible, but might require extra effort. We will be building a 3D model pipeline around Blender, but no prior knowledge of the tool will be assumed.

Prerequisites: 15-112 Min. grade C or 62-150 Min. grade C or 15-104 Min. grade C or 15-213 Min. grade C

16-467 Human Robot Interaction
Spring: 12 units
The field of human-robot interaction (HRI) is fast becoming a significant area of research in robotics. The basic objective is to create and investigate interfaces that enable natural and effective modes of interaction with robotic technologies. HRI is highly interdisciplinary, bringing together methodologies and techniques from robotics, artificial intelligence, human-computer interaction, human factors, interaction design, psychology, anthropology, education, drama, and other fields. This course is primarily lecture-based, with in-class participatory mini-projects, group homework assignments, and a group term project that will enable students to put theory to practice using state-of-the-art interactive robots. The topics covered will include man-machine coupling, underlying robotic technologies, as they relate to human-robot interaction, interaction methodologies and techniques, the singularity, and will include significant discussion of application domains that feature HRI. This course has no prerequisites, but some basic familiarity with robots is recommended (programming knowledge is not necessary, but is useful for the term project).

Course Website: http://www.cs.cmu.edu/~reids/16-467/
16-474 Robotics Capstone
Spring: 12 units
In this course students refine the design, build, integrate, test, and demonstrate the performance of the robot they designed in the prerequisite Systems Engineering Course (16-450). The students are expected to continue to apply the process and methods of Systems Engineering to track requirements, evaluate alternatives, refine the cyberphysical architectures, plan and devise tests, verify the design, and validate system performance. In addition, the students learn and apply Project Management techniques to manage the technical scope, schedule, budget, and risks of their project. The course consists of lectures, class meetings, reviews, and a final demonstration. Lectures cover core topics in Project Management and special topics in Systems Engineering. During class meetings the students and instructor review progress on the project and discuss technical and project-execution challenges. There are three major reviews approximately at the end of each of the first three months of the semester. For each review, the students give a presentation and submit an updated version of the System Design and Development Document. The course culminates in a System Performance Validation Demonstration at the end of the semester. In addition to that the students hold a special demonstration of their robotic system for the broader Robotics community. Prerequisite: 16-450 Min. grade C

16-597 Undergraduate Reading and Research
Fall and Spring
Missing Course Description - please contact the teaching department.

16-621 MSCV Project I
Fall and Spring: 12 units
TBD

16-622 MSCV Capstone
Fall and Spring: 12 units
TBA

16-623 Advanced Computer Vision Apps.
Fall: 12 units
Computer vision is a discipline that attempts to extract information from images and videos. Nearly every smart device on the planet has a camera, and people are increasingly interested in how to develop apps that use computer vision to perform an ever expanding list of things including: 3D mapping, photo/image search, people/object tracking, augmented reality etc. This course is intended for graduate students who are familiar with computer vision, and are keen to learn more about the applying state of the art vision methods on smart devices and embedded systems. A strong programming background is a must (at a minimum good knowledge of C/C++), topics will include using conventional computer vision software tools (OpenCV, MATLAB toolboxes, VLFeat, CAFFE, Torch 7), and development on iOS devices using mobile vision libraries such as GPUImage, Metal and fast math libraries like Armadillo and Eigen. For consistency, all app development will be in iOS and it is expected that all students participating in the class have access to an Intel-based MAC running OS X Mavericks or later. Although the coursework will be focused on a single operating system, the knowledge gained from this class will easily generalize to other mobile platforms such as Android etc. Prerequisites: 16-385 or 16-720
Course Website: http://16623.courses.cs.cmu.edu

16-627 MSCV Seminar
Fall
(Only open to MSCV students.) MSCV students will be required to participate in this one-semester seminar course which will prepare them for the MSCV project starting in the Spring semester. The first part of this course will cover talks by computer vision and related faculty about the ongoing research, development programs related to Computer Vision at CMU. The second part of this course will include student/faculty tutorial on topics such as OpenCV, Dataset Creation, Mechanical Turk etc. The goal of this series is to get students acquainted with practical knowledge for a successful project. In the last month of the course, each lecture will cover up to four possible MSCV projects pitched by faculty or industrial sponsors. At the end of the course students will turn in their choices, and a faculty committee will assign them the final projects.

16-720 Computer Vision
Fall and Spring: 12 units
This course introduces the fundamental techniques used in computer vision, that is, the analysis of patterns in visual images to reconstruct and understand the objects and scenes that generated them. Topics covered include image formation and representation, camera geometry, and calibration, computational imaging, multi-view geometry, stereo, 3D reconstruction from images, motion analysis, physics-based vision, image segmentation and object recognition. The material is based on graduate-level texts augmented with research papers, as appropriate. Evaluation is based on homeworks and a final project. The homeworks involve considerable Matlab programming exercises. Texts recommended but not required: Title: “Computer Vision Algorithms and Applications” Author: Richard Szeliski Series: Texts in Computer Science Publisher: Springer ISBN: 978-1-84882-934-3 Title: “Computer Vision: A Modern Approach” Authors: David Forsyth and Jean Ponce Publisher: Prentice Hall ISBN: 0-13-085198-1 Course Website: http://www.andrew.cmu.edu/course/16-720/

16-725 Medical Image Analysis
Spring: 12 units
Students will gain theoretical and practical skills in medical image analysis, including skills relevant to general image analysis. The fundamentals of computational medical image analysis will be explored, leading to current research in applying geometry and statistics to segmentation, registration, visualization, and image understanding. Student will develop practical experience through projects using the National Library of Medicine Insight Toolkit (ITK), a popular open-source software library developed by a consortium of institutions including Carnegie Mellon University and the University of Pittsburgh. In addition to image analysis, the course will include interaction with clinicians at UPMC. It is possible that a few class lectures may be videoed for public distribution. Prerequisites: Knowledge of vector calculus, basic probability, and either C++ or python. Required textbook, “Machine Vision”, ISBN: 052116981X; Optional textbook, “Insight to Images”, ISBN: 9781568812175.
Course Website: http://www.cs.cmu.edu/~galeotti/methods_course

16-735 Robotic Motion Planning
Intermittent: 12 units
The robot motion field and its applications have become incredibly broad and theoretically deep at the same time. The goal of the course is to provide an up-to-date foundation in the motion planning field, make the fundamentals of motion planning accessible to the novice and relate low-level implementation to high-level algorithmic concepts. We cover basic path planning algorithms using potential functions, roadmaps and cellular decompositions. We also look at the recent advances in sensor-based implementation and probabilistic techniques, including sample-based roadmaps, rapidly exploring random trees, Kalman filtering, and Bayesian estimation.

16-741 Mechanics of Manipulation
Fall: 12 units
Kinematics, statics, and dynamics of robotic manipulator’s interaction with a task, focusing on intelligent use of kinematic constraint, gravity, and frictional forces. Automatic planning based on mechanics. Application examples drawn from manufacturing and other domains.
Course Website: http://www.cs.cmu.edu/afs/cs/academic/class/16741-s07/www/index.html

16-742 Geometry of Locomotion
Fall: 12 units
This course introduces geometric methods for the analysis of locomoting systems. Focusing on the kinematics of locomoting systems, the course covers topics from differential geometry, geometric mechanics, and motion planning. Specific topics include configuration spaces, manifolds, groups, Lie groups, representations of velocity, holonomic and nonholonomic constraints, constraint curvature, response to cyclic inputs and distance metrics. The primary goal of this class is to develop an intuitive understanding of these concepts and how they are used in locomoting systems, rather than working through a set of formal proofs and derivations. We do, however, incorporate enough mathematical formalism for this class to serve as a starting point for further investigation into this topic area. We also call upon biological data, when available, and relate to the mathematical formalisms in the class.
16-761 Mobile Robots
Spring: 12 units
This course covers all aspects of mobile robot systems design and programming from both a theoretical and a practical perspective. The basic subsystems of control, localization, mapping, perception, and planning are presented. For each, the discussion will include relevant methods from applied mathematics. Aspects of physics necessary in the construction of models of system and environmental behavior, and core algorithms which have proven to be valuable in a wide range of circumstances.
Course Website: http://www.frc.ri.cmu.edu/~alonzo/teaching/16-761/16-761.html

16-778 Mechatronic Design
Spring: 12 units
Mechatronics is the synergistic integration of mechanism, electronics, and computer control to achieve a functional system. This course is a semester-long multidisciplinary capstone hardware project design experience in which small (typically four-person) teams of electrical and computer engineering, mechanical engineering and robotics students deliver an end-of-course demonstration of a final integrated system capable of performing a mechatronic task. Throughout the semester, the students design, configure, implement, test and evaluate in the laboratory devices and subsystems culminating in the final integrated mechatronic system. Lectures will complement the laboratory experience with comparative surveys, operational principles, and integrated design issues associated with the spectrum of mechanism, microcontroller, electronic, sensor, and control components.
Course Website: http://www.ece.cmu.edu/courses/items/18578.html

16-782 Planning and Decision-making in Robotics
Fall: 12 units
Planning and Decision-making are critical components of autonomy in robotic systems. These components are responsible for making decisions that range from path planning and motion planning to coverage and task planning to taking actions that help robots understand the world around them better. This course studies underlying algorithmic techniques used for planning and decision-making in robotics and examines case studies in ground and aerial robots, humanoids, mobile manipulation platforms and multi-robot systems. The students will learn the algorithms and implement them in a series of programming-based projects.

16-833 Robot Localization and Mapping
Spring: 12 units
Robot localization and mapping are fundamental capabilities for mobile robots operating in the real world. Even more challenging than these individual problems is their combination: simultaneous localization and mapping (SLAM). Robust solutions are needed that can handle the uncertainty inherent in sensor measurements, while providing localization and map estimates in real-time. We will explore suitable efficient probabilistic inference algorithms at the intersection of linear algebra and probabilistic graphical models. We will also explore state-of-the-art systems.
Course Website: http://frc.ri.cmu.edu/~kaess/teaching/16-833(Spring2017)/

Software Engineering Courses

17-355 Program Analysis
Spring: 12 units
This course covers both foundations and practical aspects of the automated analysis of programs, which is becoming increasingly critical to find software errors and assure program correctness. The theory of abstract interpretation captures the essence of a broad range of program analyses and supports reasoning about their correctness. Building on this foundation, the course will describe program representations, data flow analysis, alias analysis, interprocedural analysis, dynamic analysis, and symbolic execution. Through assignments and projects, students will design and implement practical analysis tools that find bugs and verify properties of software.
Prerequisites: 15-251 and (15-150 or 15-214)
Prerequisites: 15-251 Min. grade C and (15-214 Min. grade C or 15-150 Min. grade C)

17-413 Software Engineering Reflection
Fall: 6 units
This course is an opportunity to reflect on a software engineering experience you have had in industry. It is structured as a writers workshop, in which you will work with the instructor and other students to identify and flesh out a software engineering theme that is illustrated by your industry experience. You will prepare a 10-page report on this theme, comparable to a practitioner’s report at a conference like ICSE or OOPSLA, and a 30-minute presentation to match. This course fulfills a requirement of the Software Engineering Minor program, but students in other programs may take the course if they meet the prerequisite industry experience and if space is available.

17-609 Global Software Development
Intermittent: 9 units
Software development is increasingly a globally-distributed undertaking. The search for talent across national boundaries and the integration of groups thrown together by mergers and acquisitions are but two of the many forces conspiring to fundamentally change the organizational context of software development. The skills that allow developers and managers to thrive in this milieu are among the most important in today’s development organizations. Distributed software development organizations are also receiving attention from researchers interested in communication, collaboration, and coordination over distances. Creating trust, awareness, shared understanding, and many other essentials of teamwork typically relies on face to face interaction. Creating effective technology-mediated mechanisms to support distributed teams requires a deep understanding of how individuals come together to form teams and organizations. This course covers a set of topics that are essential to both professionals who will become participants and leaders in globally-distributed projects, as well as researchers interested in studying virtual teams, distributed organizations, and global software development. Topics covered in this course will include: * Virtual teams, distributed organizations * Architectures and coordination * Distributed development environments * Lessons from open source * Open source ecologies * Challenges of culture * The outsourcing relationship * Facilitating trust, cooperation, social capital * Social networks and knowledge networks * Communication and awareness * Assessing coordination risk

17-615 Software Process Definition
Intermittent: 9 units
A software process definition is the cornerstone of implementing and improving a software process. The objective of this course is to prepare students to understand how processes work in the context of an operational, day-to-day engineering company, and most importantly how they can, as an individual within an engineering environment, change a process for the betterment of all. Although the focus is on software process, this course will be useful to all students who will be executing, improving, or defining most any type of process. An incremental methodology and modular approach to software process definition is used and covers: * guidelines for early success and building a sound foundation * organizing the process definition as it develops * approaches to avoid unnecessarily elaborate or formal notations * developing the process using organizational goals and constraints * using the environmental context that the process resides within and builds upon * although the focus is on software process, this course will be useful to all students who will be executing, improving, or defining most any type of process. Requirement: This course is intended for individuals who have operational software engineering experience or a comprehensive undergraduate coursework in software engineering.
17-619 Introduction to Real-Time Software and Systems
Intermittent: 12 units
Introduction to Real-Time Software and Systems presents an overview of time as it relates engineering complex systems. Any system that responds at the pace of relevant events has real-time concerns whether the timescale is short, like the flight controls for an aircraft, or longer, like the flight reservation system for an airline. Fundamental concepts, terminology, and issues of real-time systems are introduced in this course. The focus is on software solutions to real-time problems—solutions that must be both correct and timely. Software development is examined with emphasis on real-time issues during each phase of the software lifecycle. Real-time requirements analysis, architecting real-time systems, designing and modeling system timing, and implementation and testing strategies are studied. Modeling techniques using UML 2.0 are applied. Particular emphasis is placed on real-time scheduling to achieve desired timing, reliability, and robustness. Languages and operating systems for real-time computing, and real-time problems in concurrent and distributed systems are explored. This course provides a comprehensive view of real-time systems with theory, techniques and methods for the practitioner. After successfully completing this course, the student will be able to identify constraints and understand real-time issues in system development, and propose approaches to typical real-time problems. The aim of this course is to motivate and prepare students to pursue more in-depth study of specific problems in real-time computing and systems development. REQUIREMENT: Proficiency with a high-level programming language such as C or Ada and basic concepts of computing systems. Familiarity with software engineering concepts and system development lifecycle.

17-628 The Modern CIO
Fall: 6 units
In today's competitive marketplace, technology can deliver the decisive edge as a company interfaces with its existing customers, engages prospective clients, and communicates across internal, local, national and global channels. Chief Information Officers of successful companies must possess skills that reach beyond the technical realm. Research indicates that competence in the areas of business acumen, communication, financial knowledge, project and board management far surpasses the need for technical competence. This course focuses on the non-technical aspects of 'the corner office'. If the CTO is to be a true strategic business partner with others in the C-suite (CEO, COO, CFO), it is essential that he or she has a grasp not only on technology, but also on the company's core philosophy, business strategy and drivers. This course will capitalize on the technical knowledge acquired in the Master of Software Engineering Professional Programs, and using it to target high-level job opportunities.

17-635 Software Measurement
Fall: 9 units
Sections D and PP are NOT available for on-campus students. The purpose of this course is to introduce students to software measurement — from need identification through analysis and feedback and into the process. Much of the course material demonstrates concepts of software measurement that are used by managers and practitioners in industry today. The course is taught within the framework of the software engineering process. Required text: Selected Readings handed out in class Requirement: This course is intended for individuals who have industrial software engineering experience with a large project, or a comprehensive undergraduate course in software engineering.

17-643 Hardware for Software Engineers
Intermittent
The goal of this course is to provide an understanding of electronics beyond the average computer organization course. Its purpose is to enable software engineers to be more effective in domains where software and hardware are closely coupled. Examples of such domains include robotics, avionics, automotive, process control, and many others. Students successfully completing this course will be better prepared to communicate with hardware-oriented engineers, understand the hardware environment, and the subtle regions where software and hardware meet. Requirement: Students need not have a hardware background, but they should have a solid computer science background including languages, data structures, discrete math, operating systems, and computer organization. It is highly desirable that students have projects experience preferably real-world experience, although project course work and/or internships will suffice. Undergraduates need instructor approval (lattanz@cs.cmu.edu).

17-648 Engineering Data Intensive Scalable Systems
Intermittent: 12 units
Internet services companies such as Google, Yahoo!, Amazon, and Facebook have pioneered systems that have achieved unprecedented scale while still providing high level availability and a high cost-performance. These systems differ from mainstream high performance systems in fundamental ways. They are data intensive rather than compute intensive as we see with mainstream super computers spending the bulk of their time performing data I/O and manipulation rather than computation. They need to inherently support scalability, typically having high reliability and availability demands as well. Given that they often operate in the commercial space the cost-performance of these systems needs to be such that the organizations dependent on such systems can turn a profit. Designing and building these systems require a specialized set of skills. This course will cover the set of topics needed in order to design and build data intensive scalable systems. In this domain engineers not only need to know how to architect systems that are inherently scalable, but to do so in a way that also supports high availability, reliability, and performance. Given the large distributed nature of these systems basic distributed systems concepts such as consistency and time and synchronization are also important. These systems largely operate around the clock, placing an emphasis on operational concerns. This course will introduce students to these concerns with the intent that they understand the extent to which things like deploying, monitoring, and upgrading impact the design. The course will be a hands-on project oriented course. The basic concepts will be given during the lectures and applied in the project. The students will gain exposure to the core concepts needed to design and build such systems as well as current technologies in this space. Class size will be limited.

17-651 Models of Software Systems
Fall: 12 units
Scientific foundations for software engineering depend on the use of precise, abstract models for describing and reasoning about properties of software systems. This course considers a variety of standard models for representing sequential and concurrent systems, such as state machines, algebras, and traces. It shows how different logics can be used to specify properties of systems, such as functional correctness, deadlock freedom, and internal consistency. Concepts such as compositionality, abstraction, invariants, non-determinism, and inductive definitions are recurrent themes throughout the course. After completing this course, students will: 1. Understand the strengths and weaknesses of certain models and logics including state machines, algebraic and process models, and temporal logic; 2. Be able to select and describe appropriate abstract formal models for certain classes of systems, describe abstraction relations between different levels of description, and reason about the correctness of refinements; 3. Be able to prove elementary properties about systems described by the models introduced in the course; and 4. Understand some of the strengths and weaknesses of formal automated reasoning tools. Prerequisites: Undergraduate discrete math including first-order logic, sets, functions, relations, and simple proof techniques such as induction. Sections D, PP and G are NOT available for on-campus students. Admission to the class is by approval from the instructor: If you are not MSE/MSIT-SE/MITs, send email to garlan@cs.cmu.edu for permission to enroll. The email should briefly describe your background, whether you have taken an undergraduate discrete math course, and why you would like to take the course. The course must be taken for a letter grade (not pass/fail). This is a graduate level course.


17-652 Methods: Deciding What to Design
Fall: 12 units
Sections D and PP are NOT available for on-campus students. Practical development of software requires an understanding of successful methods for bridging the gap between a problem to be solved and a working software system. This course will teach a variety of ways of understanding the problem to be solved by the system you are developing and of framing an appropriate solution to that problem. After completing this course, you will be able to: identify different classes of problems and their structures; analyze technical, organizational, usability, business, and marketing constraints on solutions; apply engineering approaches to frame solutions; understand how your understanding of the problem should be reflected in the software design. PREREQUISITE: minimum of 3 months hands-on software development experience in industry. Students not accepted into the MSE program must present a current resume showing relevant industrial experience to department coordinator. This course is offered to only MSE/MITS and MSIT-SE students for fall semester. This course is for graduate students only. This course is for letter grade only (no pass/fail grades). To register for Methods course, you will need a requirement of at least one large software system, either through industrial experience or a comprehensive undergraduate course in software engineering, compilers, operating systems, or the like. This course is for letter grade only. Instructor wants each student who wants to take his methods course to state what experience they have had. This is a graduate course. Only undergrad SE minors may take this course.

Course Website: http://spoke.compose.cs.cmu.edu/methods-fall-05/res/bib.htm

17-653 Managing Software Development
Fall: 12 units
Sections D, F, PP, and G are NOT available for on-campus students. Large scale software development requires the ability to manage resources - both human and computer-based - through control of the development process. This course provides the knowledge and skills necessary to lead a project team, understand the relationship of software development to overall product engineering, estimate time and costs, and understand the software process. After completing this course, students will: be able to write a software project management plan, addressing issues of risk analysis, schedule, costs, team organization, resources, and technical approach be able to define the key process areas of the Capability Maturity Model and the technology and practices associated with each and a variety of software development life cycle models and explain the strengths, weaknesses, and applicability of each be able to understand the relationship between software products and overall systems (if embedded), or the role of the product in the organizational product line understand the legal issues involved in liability, warranty, patentability, and copyright be able to understand the purpose for generating specific instances of an architecture, and case studies of actual system architectures. The course teaches the skills and background students need to evaluate the architectures of existing systems and to design new systems in principled ways using well-founded architectural paradigms. After completing this course, students will be able to: describe an architecture accurately recognize major architectural styles in existing software systems generate new architectural alternatives for a problem and choose among them construct a medium-sized software system that satisfies an architectural specification use existing definitions and development tools to expedite such tasks understand the formal definition of a number of architectures and be able to reason about the properties of those architectures use domain knowledge to specialize an architecture for a particular family of applications REQUIREMENT: Experience with at least one large software system, either through industrial software development experience or an undergraduate course in software engineering, compilers, operating systems, or the like. This course is for letter grade only. Instructor wants each student who wants to take his architectural courses to have worked on a significant software system in your courses or in industry. Write to (garlan@cs.cmu.edu) for admission into the course stating the experience you have had. This is a graduate course. Only undergrad SE minors may take this course.

17-654 Analysis of Software Artifacts
Spring: 12 units
Analysis is the systematic examination of an artifact to determine its properties. This course will focus on analysis of software artifacts - primarily code, but also including design artifacts, architecture, and tests. Students will focus on functional properties, but also consider non-functional properties like performance and security. In order to illustrate core analysis concepts in some depth, the course will center on static program analysis; however, the course will also include a broad selection of techniques such as testing, model checking, theorem proving, dynamic analysis, and type systems. The course will balance theoretical discussions with lab exercises in which students will apply the ideas they are learning to real artifacts. After completing this course, students will: be able to apply various techniques to analyze artifacts and write about their findings be able to describe the properties of an artifact and their implications for its use

Course Website: http://www-2.cs.cmu.edu/~aldrich/courses/654/

17-664 Enterprise Application Integration
Intermittent: 12 units
Modern business enterprises are supported by hundreds of disparate applications that work together to achieve a common goal. These applications are typically large three-tier (or n-tier) application silos developed to support a particular facet of business. Unfortunately, in most cases these systems were never designed to have their services reused transparently across an organization. The result is often a lack of flexibility for large-scale reuse, lack of reliability when integrated, inappropriate security models for organization-level integration, and others. Enterprise Application Integration is a graduate-level course on how to design and deploy large-scale systems for supporting the critical backbone of an organization. Although a particular emphasis is put on Service Oriented Architecture (SOA) and Enterprise Service Bus (ESB), a broad set of topics is covered. These include security for large-enterprise systems, reliability for distributed long-running transactions, standards for intra- and extra-organization system integration, deployment and fault-tolerance of systems, and others. The aim of the course is to prepare future software architects to pragmatically deal with large-scale systems, so as to understand the trade-offs and implications of supporting the goals of an organization. PP section for Portugal students only.
17-690 Seminar in Software Process

Intermittent

The Seminar in Software Process course is a self-study and discussion course. Discussions center on how to implement effective and efficient software processes. The focus of the seminar is on systematic approaches to building software better, faster, and cheaper. A variety of process improvement and quality management strategies are discussed, including Total Quality Management, Baldrige Award, ISO 9001, ISO 15504 (SPICE), and others, but the course is primarily structured around the staged approach to improvement from the Capability Maturity Model (CMM).

Specific topics that are covered include software project management, configuration management, quality assurance, organizational learning, process definition, training, peer reviews, team building, change management, measurement, and statistical process control. These topics are addressed from a process management perspective. Required Textbook: M.B. Chrissis, M.D. Konrad, and S. Shrum, "CMMI: Guidelines for Process Integration and Product Improvement, Second Edition," Addison-Wesley, 2006, or you can use the online SEI technical reports, which are free. Note that the third edition of this book is planned for March 2011. This is a graduate course. Only undergrad SE minors may take this course.