CS 100A. Problem-solving Lab for CS106A. 1 Unit.
Additional problem solving practice for the introductory CS course CS 106A. Sections are designed to allow students to acquire a deeper understanding of CS and its applications, work collaboratively, and develop a mastery of the material. Limited enrollment, permission of instructor required. Concurrent enrollment in CS 106A required.

CS 100B. Problem-solving Lab for CS106B. 1 Unit.
Additional problem solving practice for the introductory CS course CS106B. Sections are designed to allow students to acquire a deeper understanding of CS and its applications, work collaboratively, and develop a mastery of the material. Limited enrollment, permission of instructor required. Concurrent enrollment in CS 106B required.

CS 101. Introduction to Computing Principles. 3-5 Units.
Introduces the essential ideas of computing: data representation, algorithms, programming “code”, computer hardware, networking, security, and social issues. Students learn how computers work and what they can do through hands-on exercises. In particular, students will see the capabilities and weaknesses of computer systems so they are not mysterious or intimidating. Course features many small programming exercises, although no prior programming experience is assumed or required. CS101 is not a complete programming course such as CS106A. CS101 is effectively an alternative to CS105. A laptop computer is recommended for the in-class exercises.

CS 102. Working with Data - Tools and Techniques. 3-4 Units.
Aimed at non-CS undergraduate and graduate students who want to learn a variety of tools and techniques for working with data. Many of the world’s biggest discoveries and decisions in science, technology, business, medicine, politics, and society as a whole, are now being made on the basis of analyzing data sets. This course provides a broad and practical introduction to working with data: data analysis techniques including databases, data mining, machine learning, and data visualization; data analysis tools including spreadsheets, Tableau, relational databases and SQL, Python, and R; introduction to network analysis and unstructured data. Tools and techniques are hands-on but at a cursory level, providing a basis for future exploration and application. Prerequisites: comfort with basic logic and mathematical concepts, along with high school AP computer science, CS106A, or other equivalent programming experience.

CS 103. Mathematical Foundations of Computing. 3-5 Units.
What are the theoretical limits of computing power? What problems can be solved with computers? Which ones cannot? And how can we reason about the answers to these questions with mathematical certainty? This course explores the answers to these questions and serves as an introduction to discrete mathematics, computability theory, and complexity theory. At the completion of the course, students will feel comfortable writing mathematical proofs, reasoning about discrete structures, reading and writing statements in first-order logic, and working with mathematical models of computing devices. Throughout the course, students will gain exposure to some of the most exciting mathematical and philosophical ideas of the late nineteenth and twentieth centuries. Specific topics covered include formal mathematical proofwriting, propositional and first-order logic, set theory, binary relations, functions (injections, surjections, and bijections), cardinality, basic graph theory, the pigeonhole principle, mathematical induction, finite automata, regular expressions, the Myhill-Nerode theorem, context-free grammars, Turing machines, decidable and recognizable languages, self-reference and undecidability, verifiers, and the P versus NP question. Students with significant proofwriting experience are encouraged to instead take CS164. Students interested in extra practice and support with the course are encouraged to concurrently enroll in CS103A. Prerequisite: CS106B or equivalent. CS106B may be taken concurrently with CS103.

CS 103A. Mathematical Problem-solving Strategies. 1 Unit.
Problem solving strategies and techniques in discrete mathematics and computer science. Additional problem solving practice for CS103. In-class participation required. Prerequisite: consent of instructor. Co-requisite: CS103.

CS 105. Introduction to Computers. 3-5 Units.
For non-technical majors. What computers are and how they work. Practical experience in programming. Construction of computer programs and basic design techniques. A survey of Internet technology and the basics of computer hardware. Students in technical fields and students looking to acquire programming skills should take 106A or 106X. Students with prior computer science experience at the level of 106 or above require consent of instructor. Prerequisite: minimal math skills.

CS 106A. Programming Methodology. 3-5 Units.
Introduction to the engineering of computer applications emphasizing modern software engineering principles: program design, decomposition, encapsulation, abstraction, and testing. Emphasis is on good programming style and the built-in facilities of respective languages. Uses the Python programming language. No prior programming experience required.

CS 106AX. Programming Methodologies in JavaScript and Python. 3-5 Units.
Introduction to the engineering of computer applications emphasizing modern software engineering principles: object-oriented design, decomposition, encapsulation, abstraction, and testing. This course targets an audience with prior programming experience, and that prior experience is leveraged so material can be covered in greater depth. Same as: Accelerated

CS 106B. Programming Abstractions. 3-5 Units.
Abstraction and its relation to programming. Software engineering principles of data abstraction and modularity. Object-oriented programming, fundamental data structures (such as stacks, queues, sets) and data-directed design. Recursion and recursive data structures (linked lists, trees, graphs). Introduction to time and space complexity analysis. Uses the programming language C++ covering its basic facilities. Prerequisite: 106A or equivalent.

CS 106E. Exploration of Computing. 3-4 Units.
A follow up class to CS106A for non-majors which will both provide practical web programming skills and cover essential computing topics including computer security and privacy. Additional topics will include digital representation of images and music, an exploration of how the Internet works, and a look at the internals of the computer. Students taking the course for 4 units will be required to carry out supplementary programming assignments in addition to the course’s regular assignments. Prerequisite: 106A or equivalent.

CS 106L. Standard C++ Programming Laboratory. 1 Unit.
Supplemental lab to 106B and 106X. Additional features of standard C++ programming practice. Possible topics include advanced C++ language features, standard libraries, STL containers and algorithms, object memory management, operator overloading, and inheritance. Prerequisite: consent of instructor. Corequisite: 106B or 106X.

CS 106S. Coding for Social Good. 1 Unit.
Survey course on applications of fundamental computer science concepts from CS106B/X to problems in the social good space (such as health, government, education, and environment). Each week consists of in-class activities designed by student groups, local tech companies, and nonprofits. Introduces students to JavaScript and the basics of web development. Some of the topics we will cover include mental health chatbots, tumor classification with basic machine learning, sentiment analysis of tweets on refugees and storytelling through virtual reality. Pre/Corequisite: CS106B or CS106X.
CS 106X. Programming Abstractions. 3-5 Units.
Intensive version of 106B for students with a strong programming background interested in a rigorous treatment of the topics at an accelerated pace. Significant amount of additional advanced material and substantially more challenging projects. Some projects may relate to CS department research. Prerequisite: excellence in 106A or equivalent, or consent of instructor.
Same as: Accelerated

CS 107. Computer Organization and Systems. 3-5 Units.
Introduction to the fundamental concepts of computer systems. Explores how computer systems execute programs and manipulate data, working from the C programming language down to the microprocessor. Topics covered include: the C programming language, data representation, machine-level code, computer arithmetic, elements of code compilation, memory organization and management, and performance evaluation and optimization. Prerequisites: 106B or X, or consent of instructor.

CS 107A. Problem-solving Lab for CS107. 1 Unit.
Additional problem solving practice for the introductory CS course CS107. Sections are designed to allow students to acquire a deeper understanding of CS and its applications, work collaboratively, and develop a mastery of the material. Limited enrollment, permission of instructor required. Concurrent enrollment in CS 107 required.

CS 107E. Computer Systems from the Ground Up. 3-5 Units.
Introduction to the fundamental concepts of computer systems through bare metal programming on the Raspberry Pi. Explores how five concepts come together in computer systems: hardware, architecture, assembly code, the C language, and software development tools. Students do all programming with a Raspberry Pi kit and several add-ons (LEDs, buttons). Topics covered include: the C programming language, data representation, machine-level code, computer arithmetic, compilation, memory organization and management, debugging, hardware, and I/O. Prerequisite: 106B or X, and consent of instructor. There is a $75 required course fee.

CS 108. Object-Oriented Systems Design. 3-4 Units.
Software design and construction in the context of large OOP libraries. Taught in Java. Topics: OOP design, design patterns, testing, graphical user interface (GUI) OOP libraries, software engineering strategies, approaches to programming in teams. Prerequisite: 107.

CS 109. Introduction to Probability for Computer Scientists. 3-5 Units.
Topics include: counting and combinatorics, random variables, conditional probability, independence, distributions, expectation, point estimation, and limit theorems. Applications of probability in computer science including machine learning and the use of probability in the analysis of algorithms. Prerequisites: 103, 106B or X, multivariate calculus at the level of MATH 51 or MATH 101 or equivalent.

CS 110. Principles of Computer Systems. 3-5 Units.
Principles and practice of engineering of computer software and hardware systems. Topics include: techniques for controlling complexity; strong modularity using client-server design, virtual memory, and threads; networks; atomicity and coordination of parallel activities. Prerequisite: 107.

CS 111S. How to Make VR: Introduction to Virtual Reality Design and Development. 2 Units.
In this hands-on, experiential course, students will design and develop virtual reality applications. You'll learn how to use the Unity game engine, the most popular platform for creating immersive applications. The class will teach the design best-practices and the creation pipeline for VR applications. Students will work in groups to present a final project in building an application for the Oculus Go headset. Enrollment is limited and by application only. See https://cs111s.stanford.edu for more information. Prerequisite: CS 106A or equivalent.

CS 124. From Languages to Information. 3-4 Units.
Extracting meaning, information, and structure from human language text, speech, web pages, social networks. Introducing methods (string algorithms, edit distance, language modeling, machine learning classifiers, neural embeddings, inverted indices, collaborative filtering, PageRank), applications (chatbots, sentiment analysis, information retrieval, question answering, text classification, social networks, recommender systems), and ethical issues in both. Prerequisites: CS103, CS107, CS109.
Same as: LINGUIST 180, LINGUIST 280

CS 129. Applied Machine Learning. 3-4 Units.
(Previously numbered CS 229A.) You will learn to implement and apply machine learning algorithms. This course emphasizes practical skills, and focuses on giving you skills to make these algorithms work. You will learn about commonly used learning techniques including supervised learning algorithms (logistic regression, linear regression, SVM, neural networks/deep learning), unsupervised learning algorithms (k-means), as well as learn about specific applications such as anomaly detection and building recommender systems. This class is taught in the flipped-classroom format. You will watch videos and complete in-depth programming assignments and online quizzes at home, then come to class for discussion sections. This class will culminate in an open-ended final project, which the teaching team will help you on. Prerequisites: Programming at the level of CS106B or 106X, and basic linear algebra such as Math 51.

CS 131. Computer Vision: Foundations and Applications. 3-4 Units.
Robots that can navigate space and perform duties, search engines that can index billions of images and videos, algorithms that can diagnose medical images for diseases, or smart cars that can see and drive safely. Lying in the heart of these modern AI applications are computer vision technologies that can perceive, understand and reconstruct the complex visual world. This course is designed for students who are interested in learning about the fundamental principles and important applications of computer vision. Course will introduce a number of fundamental concepts in computer vision and expose students to a number of real-world applications, plus guide students through a series of projects such that they will get to implement cutting-edge computer vision algorithms. Prerequisites: Students should be familiar with Python (i.e. have programmed in Python before) and Linux; plus Calculus & Linear Algebra.

CS 140. Operating Systems and Systems Programming. 3-4 Units.
Operating systems design and implementation. Basic structure; synchronization and communication mechanisms; implementation of processes, process management, scheduling, and protection; memory organization and management, including virtual memory; I/O device management, secondary storage, and file systems. Prerequisite: CS 110.

CS 140E. Operating systems design and implementation. 3-4 Units.
Students will implement a simple, clean operating system (virtual memory, processes, file system) in the C programming language, on a raspberry pi computer and use the result to run a variety of devices and implement a final project. All hardware is supplied by the instructor, and no previous experience with operating systems, raspberry pi, or embedded programming is required.
CS 141. Introduction to Computer Sound. 3 Units.
Core mathematics and methods for computer sound with applications to computer science. Background on digital signal processing; time- and frequency-domain methods. Project-focussed exploration of computer sound areas: fundamentals of sound analysis & synthesis, robotics and learning (sound features, filterbanks & deep learning, perception, localization, tracking, manipulation); speech (recognition, synthesis), virtual and augmented reality (3D auralization, HRTFs, reverberation), computational acoustics (wave simulation, physics-based modeling, animation sound), computer music (music synthesis, instrument modeling, audio effects, historical aspects), games (game audio, music and sound design, middleware), hardware acceleration (architectures, codecs, synthesizers). Prerequisite: CS 106A or equivalent programming experience.

CS 142. Web Applications. 3 Units.
Concepts and techniques used in constructing interactive web applications. Browser-side web facilities such as HTML, cascading styleheets, the document object model, and JavaScript frameworks and Server-side technologies such as server-side Java/Scripts, sessions, and object-oriented databases. Issues in web security and application scalability. New models of web application deployment. Prerequisites: CS 107 and CS 108.

CS 143. Compilers. 3-4 Units.
Principles and practices for design and implementation of compilers and interpreters. Topics: lexical analysis; parsing theory; symbol tables; type systems; scope; semantic analysis; intermediate representations; runtime environments; code generation; and basic program analysis and optimization. Students construct a compiler for a simple object-oriented language during course programming projects. Prerequisites: 103 or 103B, and 107.

CS 144. Introduction to Computer Networking. 3-4 Units.

CS 145. Data Management and Data Systems. 3-4 Units.
Introduction to the use, design, and implementation of database and data-intensive systems, including data models; schema design; data storage; query processing, query optimization, and cost estimation; concurrency control, transactions, and failure recovery; distributed and parallel execution; semi-structured databases; and data system support for advanced analytics and machine learning. Prerequisites: 103 and 107 (or equivalent).

CS 146. Introduction to Game Design and Development. 3-4 Units.
This project-based course provides a survey on designing and engineering video games. Through creating their own games each week, students explore topics including 2D/3D Art, Audio, User Interface design, Production, Narrative Design, Marketing, and Publishing. Speakers from the games industry will provide insights and context during a weekly seminar. Classroom meetings will be used to foster student project discussions, and deepen understanding of material. The course culminates with students forming project teams to create a final video game. Assignments will be completed within the Unity game development engine; prior Unity experience is welcomed but not required. Given class size limitations, an online survey will be used to achieve a diverse class composition. Prerequisite: CS 106 (B or X).

CS 147. Introduction to Human-Computer Interaction Design. 3-5 Units.
Introduces fundamental methods and principles for designing, implementing, and evaluating user interfaces. Topics: user-centered design, rapid prototyping, experimentation, direct manipulation, cognitive principles, visual design, social software, software tools. Learn by doing: work with a team on a quarter-long design project, supported by lectures, readings, and studios. Prerequisite: 106B or X or equivalent programming experience. Recommended that CS Majors have also taken one of 142, 193P or 193A.

CS 148. Introduction to Computer Graphics and Imaging. 3-4 Units.
Introductory prerequisite course in the computer graphics sequence introducing students to the technical concepts behind creating synthetic computer generated images. Begins with OpenGL/scanline rendering including discussions of underlying mathematical concepts including triangles, normals, interpolation, texture/bump mapping, etc. Importantly, the course will discuss handling light/color for image formats, computer displays, printers, etc., as well as how light interacts with the environment, constructing engineering models such as the BRDF, and various simplifications into more basic lighting and shading models. Ray tracing is introduced and compared to real world cameras to illustrate various concepts, and both anti-aliasing and acceleration structures are discussed. The final class mini-project consists of building out a ray tracer to create visually compelling images. Starter codes and code bits will be provided to aid in development, but this class focuses on what you can do with the code as opposed to what the code looks like. Therefore grading is weighted toward in person "demos" of the code in action - creativity and the production of impressive visual imagery are highly encouraged/rewarded. Prerequisites: CS 107, MATH 51.

CS 149. Parallel Computing. 3-4 Units.
This course is an introduction to parallelism and parallel programming. Most new computer architectures are parallel; programming these machines requires knowledge of the basic issues of and techniques for writing parallel software. Topics: varieties of parallelism in current hardware (e.g., fast networks, multicore, accelerators such as GPUs, vector instruction sets), importance of locality, implicit vs. explicit parallelism, shared vs. non-shared memory, synchronization mechanisms (locking, atomicity, transactions, barriers), and parallel programming models (threads, data parallel/streaming, MapReduce, Apache Spark, SPMD, message passing, SIMT, transactions, and nested parallelism). Significant parallel programming assignments will be given as homework. The course is open to students who have completed the introductory CS course sequence through 110.

CS 151. Logic Programming. 3 Units.
Logic Programming is a style of programming based on symbolic logic. In writing a logic program, the programmer describes the application area of the program (as a set of logical sentences) without reference to the internal data structures or operations of the system executing the program. In this regard, a logic program is more of a specification than an implementation; and logic programs are often called runnable specifications. This course introduces basic logic programming theory, current technology, and examples of common applications, notably deductive databases, logical spreadsheets, enterprise management, computational law, and game playing. Work in the course takes the form of readings and exercises, weekly programming assignments, and a term-long project. Prerequisite: CS 106B or equivalent.
CS 152. Trust and Safety Engineering. 3 Units.
An introduction to the ways consumer internet services are abused to cause real human harm and the potential operational, product and engineering responses. Students will learn about spam, fraud, account takeovers, the use of social media by terrorists, misinformation, child exploitation, harassment, bullying and self-harm. This will include studying both the technical and sociological roots of these harms and the various online providers have responded. Our goal is to provide students with an understanding of how the technologies they may build have been abused in the past and how they might spot future abuses earlier. The class is taught by a long-time practitioner and supplemented by guest lecturers from tech companies and non-profits. Fulfills the Technology in Society requirement. Prerequisite: CS106B or equivalent for grad students. Content note: This class will cover real-world harmful behavior and expose students to potentially upsetting material.

CS 154. Introduction to Automata and Complexity Theory. 3-4 Units.
This course provides a mathematical introduction to the following questions: What is computation? Given a computational model, what problems can we hope to solve in principle with this model? Besides those solvable in principle, what problems can we hope to efficiently solve? In many cases we can give completely rigorous answers; in other cases, these questions have become major open problems in computer science and mathematics. By the end of this course, students will be able to classify computational problems in terms of their computational complexity (Is the problem regular? Non-regular? Decidable? Recognizable? Neither? Solvable in P? NP-complete? PSPACE-complete?, etc.). Students will gain a deeper appreciation for some of the fundamental issues in computing that are independent of trends of technology, such as the Church-Turing Thesis and the P versus NP problem. Prerequisites: CS 103 or 103B.

CS 155. Computer and Network Security. 3 Units.
For seniors and first-year graduate students. Principles of computer systems security. Attack techniques and how to defend against them. Topics include: network attacks and defenses, operating system security, application security (web, apps, databases), malware, privacy, and security for mobile devices. Course projects focus on building reliable code. Prerequisite: 110. Recommended: basic Unix.

CS 157. Computational Logic. 3 Units.
Rigorous introduction to Symbolic Logic from a computational perspective. Encoding information in the form of logical sentences. Reasoning with information in this form. Overview of logic technology and its applications - in mathematics, science, engineering, business, law, and so forth. Topics include the syntax and semantics of Propositional Logic, Relational Logic, and Herbrand Logic, validity, contingency, unsatisfiability, logical equivalence, entailment, consistency, natural deduction (Fitch), mathematical induction, resolution, compactness, soundness, completeness.

CS 161. Design and Analysis of Algorithms. 3-5 Units.

CS 166. Data Structures. 3-4 Units.
This course is designed as a deep dive into the design, analysis, implementation, and theory of data structures. Over the course of the quarter, we'll explore fundamental techniques in data structure design (somesorts, amortization, randomization, word-level parallelism, etc.). In doing so, we'll see a number of classic data structures like Fibonacci heaps and suffix trees as well as more modern data structures like count-min sketches and range minimum queries. By the time we've finished, we'll have seen some truly beautiful strategies for solving problems efficiently. Prerequisites: CS107 and CS161.

CS 168. The Modern Algorithmic Toolbox. 3-4 Units.
This course will provide a rigorous and hands-on introduction to the central ideas and algorithms that constitute the core of the modern algorithms toolkit. Emphasis will be on understanding the high-level theoretical intuitions and principles underlying the algorithms we discuss, as well as developing a concrete understanding of when and how to implement and apply the algorithms. The course will be structured as a sequence of one-week investigations; each week will introduce one algorithmic idea, and discuss the motivation, theoretical underpinning, and practical applications of that algorithmic idea. Each topic will be accompanied by a mini-project in which students will be guided through a practical application of the ideas of the week. Topics include hashing, dimension reduction and LSH, boosting, linear programming, gradient descent, sampling and estimation, and an introduction to spectral techniques. Prerequisites: CS107 and CS161, or permission from the instructor.

CS 170. Stanford Laptop Orchestra: Composition, Coding, and Performance. 1-5 Unit.
Classroom instantiation of the Stanford Laptop Orchestra (SLOrk) which includes public performances. An ensemble of more than 20 humans, laptops, controllers, and special speaker arrays designed to provide each computer-mediated instrument with its sonic identity and presence. Topics and activities include issues of composing for laptop orchestras, instrument design, sound synthesis, programming, and live performance. May be repeated four times for credit. Space is limited; see https://cs170.stanford.edu/courses/128 for information about the application and enrollment process. May be repeat for credit. Same as: MUSIC 128

CS 181. Computers, Ethics, and Public Policy. 4 Units.
Ethical and social issues related to the development and use of computer technology. Ethical theory, and social, political, and legal considerations. Scenarios in problem areas: privacy, reliability and risks of complex systems, and responsibility of professionals for applications and consequences of their work. Prerequisite: CS106A. To take this course, students need permission of instructor and may need to complete an assignment due at the first day of class. Please see https://cs181.stanford.edu for more information.

CS 181W. Computers, Ethics, and Public Policy. 4 Units.
Writing-intensive version of CS181. Satisfies the WIM requirement for Computer Science, Engineering Physics,STS, and Math/Comp Sci undergraduates. To take this course, students need permission of instructor and may need to complete an assignment due at the first day of class. Please see https://cs181.stanford.edu for more information. Same as: WIM

CS 182. Ethics, Public Policy, and Technological Change. 5 Units.
Examination of recent developments in computing technology and platforms through the lenses of philosophy, public policy, social science, and engineering. Course is organized around four main units: algorithmic decision-making and bias; data privacy and civil liberties; artificial intelligence and autonomous systems; and the power of private computing platforms. Each unit considers the promise, perils, rights, and responsibilities at play in technological developments. Prerequisite: CS106A. Same as: COMM 180, ETHICSCOS 182, PHIL 82, POLISCI 182, PUBLPOL 182

CS 182W. Ethics, Public Policy, and Technological Change. 5 Units.
Writing-intensive version of CS182. Satisfies the WIM requirement for Computer Science, Engineering Physics,STS, and Math/Comp Sci undergraduates. Prerequisite: CS106A. See CS182 for room location. Enroll in either CS 182 or CS 182W, not both. Enrollment in WIM version of the course is limited to 100 students, who will be selected via weighted lottery (weighted modestly to favor juniors and seniors). To sign up for the selection process, please complete this web form by Friday, December 6th; https://forms.gle/XMsN5Kgdt2UJ3Hv36. Same as: WIM

Stanford Bulletin 2019-20
CS 183E. Effective Leadership in High-Tech. 1 Unit.
You will undoubtedly leave Stanford with the technical skills to excel in your first few jobs. But non-technical skills are just as critical to making a difference. This seminar is taught by two industry veterans in engineering leadership and product management. In a small group setting, we will explore how you can be a great individual contributor (communicating with clarity, getting traction for your ideas, resolving conflict, and delivering your best work) and how you can transition into leadership roles (finding leadership opportunities, creating a great team culture, hiring and onboarding new team members). We will end by turning back to your career (picking your first job and negotiating your offer, managing your career changes, building a great network, and succeeding with mentors). Prerequisites: Preference given to seniors and co-terms in Computer Science and related majors. Enrollment limited and application required for admittance.

CS 184. Bridging Policy and Tech Through Design. 3 Units.
This project-based course aims to bring together students from computer science and the social sciences to work with external partner organizations at the nexus of digital technology and public policy. Students will collaborate in interdisciplinary teams on a problem with a partner organization. Along with the guidance of faculty mentors and the teaching staff, students will produce a set of deliverables ranging from policy memos and white papers to data visualizations and software. Possible projects suggested by partner organizations will be presented at an information session on Tuesday, March 3rd at 6:30 pm in Gates 415 - RSVP http://bit.ly/InfoCS184. Following the info session, a course application will open for teams to be selected before the start of Spring Quarter. Students may apply to a project with a partner organization or with a pre-formed team and their own idea to be reviewed for approval by the course staff. Applications close late March and successful applicants will be notified by Tuesday, March 24th. The course meets twice a week: Once with all students taking the course and a second time with the project-based team mentors. The time below is for the cross-team meetings, as the team meetings will be scheduled according to member availability. Prerequisites: Appropriate preparation depends on the nature of the project proposed, and will be verified by the teaching staff based on your application.

CS 190. Software Design Studio. 3 Units.
This course teaches the art of software design: how to decompose large complex systems into classes that can be implemented and maintained easily. Topics include the causes of complexity, modular design, techniques for creating deep classes, minimizing the complexity associated with exceptions, in-code documentation, and name selection. The class involves significant system software implementation and uses an iterative approach consisting of implementation, review, and revision. The course is taught in a studio format with in-class discussions and code reviews in addition to lectures. Prerequisite: CS 140 or equivalent. Apply at: https://web.stanford.edu/class/cs190.

CS 191. Senior Project. 1-6 Unit.
Restricted to Computer Science and Computer Systems Engineering students. Group or individual projects under faculty direction. Register using instructor's section number. A project can be either a significant software application or publishable research. Software application projects include substantial programming and modern user-interface technologies and are comparable in scale to shareware programs or commercial applications. Research projects may result in a paper publishable in an academic journal or presentable at a conference. Required public presentation of final application or research results. Prerequisite: Completion of at least 135 units.

CS 191W. Writing Intensive Senior Project. 3-6 Units.
Restricted to Computer Science and Computer Systems Engineering students. Writing-intensive version of CS191. Register using the section number of an Academic Council member. Prerequisite: Completion of at least 135 units.
Same as: WIM CS 192. Programming Service Project. 1-4 Unit.
Restricted to Computer Science students. Appropriate academic credit (without financial support) is given for volunteer computer programming work of public benefit and educational value.

CS 193A. Android Programming. 3 Units.
Introduction to building applications for Android platform. Examines key concepts of Android programming: tool chain, application life-cycle, views, controls, intents, designing mobile UIs, networking, threading, and more. Features weekly lectures and a series of small programming projects. Phone not required, but a phone makes the projects more engaging. Prerequisites: 106B or Java experience at 106B level. Enrollment limited and application required.

CS 193C. Client-Side Internet Technologies. 3 Units.
Client-side technologies used to create web sites such as Google maps or Gmail. Includes HTML5, CSS, JavaScript, the Document Object Model (DOM), and Ajax. Prerequisite: programming experience at the level of CS106A.

CS 193P. iOS Application Development. 3 Units.
Tools and APIs required to build applications for the iPhone and iPad platforms using the iOS SDK. User interface design for mobile devices and unique user interactions using multi-touch technologies. Object-oriented design using model-view-controller paradigm, memory management, Swift programming language. Other topics include: object-oriented database API, animation, multi-threading, networking and performance considerations. Prerequisites: C language and object-oriented programming experience exceeding 106B or X level. Previous completion of any one of the following is required: CS 107, 108 (preferred) or 110. Recommended: UNIX, graphics, databases.

CS 193Q. Introduction to Python Programming. 1 Unit.
CS193Q teaches basic Python programming with a similar end-condition to CS106AP: strings, lists, numbers, dicts, loops, logic, functions, testings, decomposition and style, and modules. CS193Q assumes knowledge of some programming language, and proceeds by showing how each common programming idea is expressed in Python. CS193Q moves very quickly, meeting 3 times for 4 hours for a total of 12 hours which is a mixture of lecture and lab time.

CS 193X. Web Programming Fundamentals. 3 Units.
Introduction to full-stack web development with an emphasis on fundamentals. Client-side topics include layout and rendering through HTML and CSS, event-driven programming through JavaScript, and single-threaded asynchronous programming techniques including Promises. Focus on modern standardized APIs and best practices. Server-side topics include the development of RESTful APIs, JSON services, and basic server-side storage techniques. Covers desktop and mobile web development. Enrollment is limited. Application required. Prerequisite: 106B or equivalent.

CS 194. Software Project. 3 Units.
Design, specification, coding, and testing of a significant team programming project under faculty supervision. Documentation includes capture of project rationale, design and discussion of key performance indicators, a weekly progress log and a software architecture diagram. Public demonstration of the project at the end of the quarter. Preference given to seniors. May be repeated for credit. Prerequisites: CS 110 and CS 161.

CS 194A. Android Programming Workshop. 1 Unit.
Learn basic, foundational techniques for developing Android mobile applications and apply those toward building a single or multi page, networked Android application.

CS 194H. User Interface Design Project. 3-4 Units.
Advanced methods for designing, prototyping, and evaluating user interfaces to computing applications. Novel interface technology, advanced interface design methods, and prototyping tools. Substantial, quarter-long course project that will be presented in a public presentation. Prerequisites: CS 147, or permission of instructor.
CS 194W. Software Project. 3 Units.
Restricted to Computer Science and Electrical Engineering undergraduates. Writing-intensive version of CS194. Preference given to seniors.
Same as: WIM

CS 195. Supervised Undergraduate Research. 3-4 Units.
Directed research under faculty supervision. Students are required to submit a written report and give a public presentation on their work.

CS 196. Computer Consulting. 2 Units.
Focus is on Macintosh and Windows operating system maintenance, and troubleshooting through hardware and software foundation and concepts. Topics include operating systems, networking, security, troubleshooting methodology with emphasis on Stanford's computing environment. Final project. Not a programming course.
Same as: VPTL 196

CS 197. Computer Science Research. 3 Units.
An onramp for students interested in breaking new ground in the frontiers of computer science. Students select a research area (AI, HCI, Systems, etc.), and are matched with a quarter-long project and a Ph.D. student mentor. Lectures by faculty introduce the fundamentals of computer science research; special interest group meetings provide peer mentorship and feedback. Alumni of the course are given the opportunity to be connected to faculty for ongoing research, or to repeat the class under CS197A for credit (but no lecture component) to continue work on their projects. Prerequisites: Enrollment is by application. CS106B is required; CS107 is strongly recommended. Team projects will involve programming.

CS 197A. Computer Science Research Seminar. 1 Unit.
Students attend weekly CS 197 Special Interest Group meetings and remain actively involved in research. Prerequisite: CS 197.

CS 198. Teaching Computer Science. 3-4 Units.
Students lead a discussion section of 106A while learning how to teach a programming language at the introductory level. Focus is on teaching skills, techniques, and course specifics. Application and interview required; see http://cs198.stanford.edu.

CS 198B. Additional Topics in Teaching Computer Science. 1 Unit.
Students build on the teaching skills developed in CS198. Focus is on techniques used to teach topics covered in CS106B. Prerequisite: successful completion of CS198.

CS 199. Independent Work. 1-6 Unit.
Special study under faculty direction, usually leading to a written report. Letter grade; if not appropriate, enroll in 199R

CS 199P. Independent Work. 1-6 Unit.
(Staff).

CS 1C. Introduction to Computing at Stanford. 1 Unit.
For those who want to learn more about Stanford's computing environment. Topics include: computer maintenance and security, computing resources, Internet privacy, and copyright law. One-hour lecture/demonstration in dormitory clusters prepared and administered weekly by the Resident Computer Consultant (RCC). Final project. Not a programming course.
Same as: VPTL 1

CS 1U. Practical Unix. 1 Unit.
A practical introduction to using the Unix operating system with a focus on Linux command line skills. Class will consist of video tutorials and weekly hands-on lab sections. Topics include: grep and regular expressions, ZSH, Vim and Emacs, basic and advanced GDB features, permissions, working with the file system, revision control, Unix utilities, environment customization, and using Python for shell scripts. Topics may be added, given sufficient interest. Course website: http://cs1u.stanford.edu.

CS 202. Law for Computer Science Professionals. 1 Unit.
An overview of intellectual property law as it relates to computer science and other disciplines, including discussions of patents, trademarks, copyrights, trade secrets, computer fraud litigation and interesting historical tidbits. Emphasis on topics of current interest such as software and business method patents, copyright issues concerning software, music, art and artificial intelligence, and current disputes of note including the recently-settled Waymo v. Uber lawsuit and the ongoing Oracle v. Google, Apple v. Samsung and hiQ v. LinkedIn sagas. Guest lectures typically have covered open source and the free software movement, practical issues for business founders (including corporate formation issues and non-disclosure, non-compete, work-made-for-hire and license agreements), and other pertinent topics. Classes are presented in an open discussion format broadly directed to students with both technical and non-technical backgrounds.

CS 203. Cybersecurity: A Legal and Technical Perspective. 2 Units.
(Formerly IPS 251) This class will use the case method to teach basic computer, network, and information security from technology, law, policy, and business perspectives. Using real world topics, we will study the technical, legal, policy, and business aspects of an incident or issue and its potential solutions. The case studies will be organized around the following topics: vulnerability disclosure, state sponsored sabotage, corporate and government espionage, credit card theft, theft of embarrassing personal data, phishing and social engineering attacks, denial of service attacks, attacks on weak session management and URLS, security risks and benefits of cloud data storage, wiretapping on the Internet, and digital forensics. Students taking the class will learn about the techniques attackers use, applicable legal prohibitions, rights, and remedies, the policy context, and strategies in law, policy and business for managing risk. Grades will be based on class participation, two reflection papers, and a final exam. Special Instructions: This class is limited to 65 students, with an effort made to have students from Stanford Law School (30 students will be selected by lottery) and students from Computer Science (30 students) and International Policy Studies (5 students). Elements used in grading: Class Participation (20%), Written Assignments (40%), Final Exam (40%). Cross-listed with the Law School (Law 4004) and International Policy Studies (IPS course number TBD).
Same as: INTLPSOL 251

CS 204. Computational Law. 2-3 Units.
Legal informatics based on representation of regulations in computable form. Encoding regulations facilitate creation of legal information systems with significant practical value. Convergence of technological trends, growth of the Internet, advent of semantic web technology, and progress in computational logic make computational law prospects better. Topics: current state of computational law, prospects and problems, philosophical and legal implications. This course is *Cross* listed with LAW 4019. Prerequisite: basic concepts of programming.

CS 205L. Continuous Mathematical Methods with an Emphasis on Machine Learning. 3 Units.
A survey of numerical approaches to the continuous mathematics with emphasis on machine and deep learning. Although motivated from the standpoint of machine learning, the course will focus on the underlying mathematical methods including computational linear algebra and optimization, as well as special topics related to training/using neural networks including automatic differentiation via backward propagation, steepest/gradient decent, momentum methods and adaptive time stepping for ordinary differential equations, etc. Students have the option of doing written homework and either a take-home or in class exams with no programming required, or may skip the exams and instead do a programming project. (Replaces CS205A, and satisfies all similar requirements.) Prerequisites: Math 51; Math 104 or 113 or equivalent or comfortable with the associated material.
CS 206. Exploring Computational Journalism. 3 Units.
This project-based course will explore the field of computational journalism, including the use of Data Science, Info Visualization, AI, and emerging technologies to help journalists discover and tell stories, understand their audience, advance free speech, and build trust. Admission by application; please email R.B. Brenner at rrbrenner@stanford.edu to request application.
Same as: COMM 281

CS 208E. Great Ideas in Computer Science. 3 Units.
Great Ideas in Computer Science Covers the intellectual tradition of computer science emphasizing ideas that reflect the most important milestones in the history of the discipline. Topics include programming and problem solving; implementing computation in hardware; algorithmic efficiency; the theoretical limits of computation; cryptography and security; computer networks; machine learning; and the philosophy behind artificial intelligence. Readings will include classic papers along with additional explanatory material.

CS 209. Law, Bias, & Algorithms. 3 Units.
Human decision making is increasingly being displaced by predictive algorithms. Judges sentence defendants based on statistical risk scores; regulators take enforcement actions based on predicted violations; advertisers target materials based on demographic attributes; and employers evaluate applicants and employees based on machine-learned models. A predominant concern with the rise of such algorithmic decision making is that it may replicate or exacerbate human bias. Algorithms might discriminate, for instance, based on race or gender. This course surveys the legal and ethical principles for assessing the equity of algorithms, describes techniques for designing fair systems, and considers how antidiscrimination law and the design of algorithms may need to evolve to account for machine bias. Concepts will be developed in part through guided in-class coding exercises. Admission is by consent of instructor and is limited to 20 students. Grading is based on response papers, class participation, and a final project. Prerequisite: CS 106A or equivalent knowledge of coding.
Same as: CSRE 230, MS&E 330, SOC 279

CS 210A. Software Project Experience with Corporate Partners. 3-4 Units.
Two-quarter project course. Focus is on real-world software development. Corporate partners seed projects with loosely defined challenges from their R&D labs; students innovate to build their own compelling software solutions. Student teams are treated as start-up companies with a budget and a technical advisory board comprised of the instructional staff and corporate liaisons. Teams will typically travel to the corporate headquarters of their collaborating partner, meaning some teams will travel internationally. Open loft classroom format such as found in Silicon Valley software companies. Exposure to: current practices in software engineering; techniques for stimulating innovation; significant development experience with creative freedoms; working in groups; real world software engineering challenges; public presentation of technical work; creating written descriptions of technical work. Prerequisites: CS 109 and 110.

CS 210B. Software Project Experience with Corporate Partners. 3-4 Units.
Continuation of CS210A. Focus is on real-world software development. Corporate partners seed projects with loosely defined challenges from their R&D labs; students innovate to build their own compelling software solutions. Student teams are treated as start-up companies with a budget and a technical advisory board comprised of the instructional staff and corporate liaisons. Teams will typically travel to the corporate headquarters of their collaborating partner, meaning some teams will travel internationally. Open loft classroom format such as found in Silicon Valley software companies. Exposure to: current practices in software engineering; techniques for stimulating innovation; significant development experience with creative freedoms; working in groups; real world software engineering challenges; public presentation of technical work; creating written descriptions of technical work. Prerequisites: CS 210A.

CS 213. Creating Great VR: From Ideation to Monetization. 1 Unit.
Covering everything from VR fundamentals to futurecasting to launch management, this course will expose you to best practices and guidance from VR leaders that helps positions you to build great VR experiences.

CS 217. Hardware Accelerators for Machine Learning. 3-4 Units.
This course provides in-depth coverage of the architectural techniques used to design accelerators for training and inference in machine learning systems. This course will cover classical ML algorithms such as linear regression and support vector machines as well as DNN models such as convolutional neural nets, and recurrent neural nets. We will consider both training and inference for these models and discuss the impact of parameters such as batch size, precision, sparsity and compression on the accuracy of these models. We will cover the design of accelerators for ML model inference and training. Students will become familiar with hardware implementation techniques for using parallelism, locality, and low precision to implement the core computational kernels used in ML. To design energy-efficient accelerators, students will develop the intuition to make trade-offs between ML model parameters and hardware implementation techniques. Students will read recent research papers and complete a design project. Prerequisites: CS 149 or EE 180. CS 229 is ideal, but not required.

CS 218. Deep Learning Systems Design. 3-4 Units.
This course will cover classical ML algorithms such as linear regression and support vector machines as well as DNN models such as convolutional neural nets, and recurrent neural nets. We will consider both training and inference for these models and discuss the impact of parameters such as batch size, precision, sparsity and compression on the accuracy of these models. We will cover the design of accelerators for ML model inference and training. Students will become familiar with hardware implementation techniques for using parallelism, locality, and low precision to implement the core computational kernels used in ML. To design energy-efficient accelerators, students will develop the intuition to make trade-offs between ML model parameters and hardware implementation techniques. Students will read recent research papers and complete a design project. Prerequisites: CS 149 or EE 180. CS 229 is ideal, but not required.

CS 219. Algorithms. 3 Units.
Algorithms is centered around algorithms and data structures, and the ability to analyze and implement them. Topics include: graphs, sorting, search, strings, and data compression. Weekly coding assignments reinforcing machine learning concepts and applications. Prerequisites: programming experience at the level of CS107, mathematical fluency at the level of CS103, comfort with probability at the level of CS109 (or equivalent). Application required for enrollment.

CS 221. Artificial Intelligence: Principles and Techniques. 3-4 Units.
Artificial intelligence (AI) has had a huge impact in many areas, including medical diagnosis, speech recognition, robotics, web search, advertising, and scheduling. This course focuses on the foundational concepts that drive these applications. In short, AI is the mathematics of making good decisions given incomplete information (hence the need for probability) and limited computation (hence the need for algorithms). Specific topics include search, constraint satisfaction, game playing, Markov decision processes, graphical models, machine learning, and logic. Prerequisites: CS 103 or CS 103B/X, CS 106B or CS 106X, CS 107, and CS 109 (algorithms, probability, and programming experience).
CS 223A. Introduction to Robotics. 3 Units.
Robotic foundations in modeling, design, planning, and control. Class
covers relevant results from geometry, kinematics, statics, dynamics,
motion planning, and control, providing the basic methodologies and
tools in robotics research and applications. Concepts and models are
illustrated through physical robot platforms, interactive robot simulations,
and video segments relevant to historical research developments or to
emerging application areas in the field. Recommended: matrix algebra.
Same as: ME 320

CS 224N. Natural Language Processing with Deep Learning. 3-4 Units.
Methods for processing human language information and the underlying
computational properties of natural languages. Focus on deep learning
approaches: understanding, implementing, training, debugging,
visualizing, and extending neural network models for a variety of
language understanding tasks. Exploration of natural language
tasks ranging from simple word level and syntactic processing to
coreference, question answering, and machine translation. Examination
of representative papers and systems and completion of a final project
applying a complex neural network model to a large-scale NLP problem.
Prerequisites: calculus and linear algebra; CS124 or CS121/221.
Same as: LINGUIST 284

CS 224S. Spoken Language Processing. 2-4 Units.
Introduction to spoken language technology with an emphasis on
dialogue and conversational systems. Deep learning and other methods
for automatic speech recognition, speech synthesis, affect detection,
dialogue management, and applications to digital assistants and spoken
language understanding systems. Prerequisites: CS 124, CS 221, CS
224N, or CS 229.
Same as: LINGUIST 285

CS 224U. Natural Language Understanding. 3-4 Units.
Project-oriented class focused on developing systems and algorithms for
robust machine understanding of human language. Draws on theoretical
concepts from linguistics, natural language processing, and machine
learning. Topics include lexical semantics, distributed representations of
meaning, relation extraction, semantic parsing, sentiment analysis, and
dialogue agents, with special lectures on developing projects, presenting
research results, and making connections with industry. Prerequisites:
one of LINGUIST 180, CS 124, CS 224N, CS224S, or CS221.
Same as: LINGUIST 188, LINGUIST 288

CS 224W. Machine Learning with Graphs. 3-4 Units.
Networks are a fundamental tool for modeling complex social,
technological, and biological systems. Coupled with the emergence
of online social networks and large-scale data availability in biological
sciences, this course focuses on the analysis of massive networks which
provide several computational, algorithmic, and modeling challenges.
Students are introduced to machine learning techniques and data
mining tools apt to reveal insights on the social, technological, and
natural worlds, by means of studying their underlying network structure
and interconnections. Topics include: robustness and fragility of food
webs and financial markets; algorithms for the World Wide Web; graph
neural networks and representation learning; identification of functional
modules in biological networks; disease outbreak detection.

CS 225A. Experimental Robotics. 3 Units.
Hands-on laboratory course experience in robotic manipulation. Topics
include robot kinematics, dynamics, control, compliance, sensor-based
collision avoidance, and human-robot interfaces. Second half of class
is devoted to final projects using various robotic platforms to build and
demonstrate new robot task capabilities. Previous projects include the
development of autonomous robot behaviors of drawing, painting, playing
air hockey, yoyo, basketball, ping-pong or xylorphone. Prerequisites: 223A
or equivalent.

CS 227B. General Game Playing. 3 Units.
A general game playing system accepts a formal description of a game
to play it without human intervention or algorithms designed for specific
games. Hands-on introduction to these systems and artificial intelligence
techniques such as knowledge representation, reasoning, learning,
and rational behavior. Students create GGP systems to compete with
each other and in external competitions. Prerequisite: programming
experience. Recommended: 103 or equivalent.

CS 228. Probabilistic Graphical Models: Principles and Techniques. 3-4
Units.
Probabilistic graphical modeling languages for representing complex
domains, algorithms for reasoning using these representations, and
inference and learning from data. Topics include: Bayesian and
Markov networks, extensions to temporal modeling, approximate inference
algorithms, and methods for learning models from data. Also included are
sample applications to various domains including speech recognition,
bioinformatics, medicine, and machine learning. Prerequisites:
breadth first search, and fundamental probability theory.

CS 229. Machine Learning. 3-4 Units.
Topics: statistical pattern recognition, linear and non-linear regression,
non-parametric methods, exponential family, GLMs, support vector
machines, kernel methods, deep learning, model/feature selection,
learning theory, ML advice, clustering, dimensionality reduction, EM,
data mining, density estimation, ICA, PCA, reinforcement learning and
adaptive control, Markov decision processes, approximate dynamic programming,
and policy search. Prerequisites: knowledge of basic computer science
principles and skills at a level sufficient to write a reasonably non-trivial
computer program in Python/numpy, familiarity with probability theory to
the equivalency of CS109 or STATS116, and familiarity with multivariable
calculus and linear algebra to the equivalency of MATH51.
Same as: STATS 229

CS 229T. Statistical Learning Theory. 3 Units.
How do we formalize what it means for an algorithm to learn from
data? How do we use mathematical thinking to design better machine
learning methods? This course focuses on developing mathematical
tools for answering these questions. We will present various learning
algorithms and prove theoretical guarantees about them. Topics
include generalization bounds, implicit regularization, the theory of deep
learning, spectral methods, and online learning and bandits problems.
Prerequisites: A solid background in linear algebra (Math 104, Math 113
or CS205) and probability theory (CS109 or STAT 116), statistics and
machine learning (STATS 315A, CS 229 or STATS 216).
Same as: STATS 231

CS 22A. The Social & Economic Impact of Artificial Intelligence. 1 Unit.
Recent advances in computing may place us at the threshold of a
unique turning point in human history. Soon we are likely to entrust
management of our environment, economy, security, infrastructure, food
production, healthcare, and to a large degree even our personal activities,
to artificially intelligent computer systems. The prospect of “turning over
production, healthcare, and to a large degree even our personal activities,
to artificially intelligent computer systems. The prospect of “turning over
the keys” to increasingly autonomous systems raises many complex
and troubling questions. How will society respond as versatile robots
and machine-learning systems displace an ever-expanding spectrum of
blue- and white-collar workers? Will the benefits of this technological
revolution be broadly distributed or accrue to a lucky few? How can
we ensure that these systems are free of algorithmic bias and respect
human ethical principles? What role will they play in our system of justice
and the practice of law? How will they be used or abused in democratic
societies and autocratic regimes? Will they alter the geopolitical balance
of power, and change the nature of warfare? The goal of CS22a is to equip
students with the intellectual tools, ethical foundation, and psychological
framework to successfully navigate the coming age of intelligent
machines.
Same as: INTLPOL 200
CS 230. Deep Learning. 3-4 Units.
Deep Learning is one of the most highly sought after skills in AI. We will help you become good at Deep Learning. In this course, you will learn the foundations of Deep Learning, understand how to build neural networks, and learn how to lead successful machine learning projects. You will learn about Convolutional networks, RNNs, LSTM, Adam, Dropout, BatchNorm, Xavier/He initialization, and more. You will work on case studies from healthcare, autonomous driving, sign language reading, music generation, and natural language processing. You will master not only the theory, but also see how it is applied in industry. You will practice all these ideas in Python and in TensorFlow, which we will teach. AI is transforming multiple industries. After this course, you will likely find creative ways to apply it to your work. This class is taught in the flipped-classroom format. You will watch videos and complete in-depth programming assignments and online quizzes at home, then come in to class for advanced discussions and work on projects. This class will culminate in an open-ended final project, which the teaching team will help you on. Prerequisites: Familiarity with programming in Python and Linear Algebra (matrix / vector multiplications). CS 229 may be taken concurrently.

CS 231A. Computer Vision: From 3D Reconstruction to Recognition. 3-4 Units.
(Formerly 223B) An introduction to the concepts and applications in computer vision. Topics include: cameras and projection models, low-level image processing methods such as filtering and edge detection, mid-level vision topics such as segmentation and clustering; shape reconstruction from stereo, as well as high-level vision tasks such as object recognition, scene recognition, face detection and human motion categorization. Prerequisites: linear algebra, basic probability and statistics.

CS 231C. Computer Vision and Image Analysis of Art. 3 Units.
This course presents the application of rigorous image processing, computer vision, machine learning, computer graphics and artificial intelligence techniques to problems in the history and interpretation of fine art paintings, drawings, murals and other two-dimensional works, including abstract art. The course focuses on the aspects of these problems that are unlike those addressed widely elsewhere in computer image analysis applied to physics-constrained images in photographs, videos, and medical images, such as the analysis of brushstrokes and marks, medium, inferring artists¿ working methods, compositional principles, stymetry (quantification of style), the tracing of artistic influence, and art attribution and authentication. The course revisits classic problems, such as image-based object recognition, but in highly non-realistic, stylized artworks. Recommended: One of CS 131 or EE 168 or equivalent; ARTHIST 1B. Prerequisites: Programming proficiency in at least one of C, C++, Python, Matlab or Mathematica and tools/ frameworks such as OpenCV or Matlab's Image Processing toolbox.

CS 231N. Convolutional Neural Networks for Visual Recognition. 3-4 Units.
Computer Vision has become ubiquitous in our society, with applications in search, image understanding, apps, mapping, medicine, drones, and self-driving cars. Core to many of these applications are visual recognition tasks such as image classification and object detection. Recent developments in neural network approaches have greatly advanced the performance of these state-of-the-art visual recognition systems. This course is a deep dive into details of neural-network based deep learning methods for computer vision. During this course, students will learn to implement, train and debug their own neural networks and gain a detailed understanding of cutting-edge research in computer vision. We will cover learning algorithms, neural network architectures, and practical engineering tricks for training and fine-tuning networks for visual recognition tasks. Prerequisites: Proficiency in Python; familiarity with C/C++; CS 131 and CS 229 or equivalents; Math 21 or equivalent, linear algebra.

CS 232. Digital Image Processing. 3 Units.
Image sampling and quantization color; point operations, segmentation, morphological image processing, linear image filtering and correlation, image transforms, eigenimages, multiresolution image processing, noise reduction and restoration, feature extraction and recognition tasks, image registration. Emphasis is on the general principles of image processing. Students learn to apply material by implementing and investigating image processing algorithms in Matlab and optionally on Android mobile devices. Term project. Recommended: EE261, EE278. Same as: EE 368

CS 233. Geometric and Topological Data Analysis. 3 Units.
Mathematical computational tools for the analysis of data with geometric content, such images, videos, 3D scans, GPS traces -- as well as for other data embedded into geometric spaces. Global and local geometry descriptors allowing for various kinds of invariances. The rudiments of computational topology and persistent homology on sampled spaces. Clustering and other unsupervised techniques. Spectral methods for geometric data analysis. Non-linear dimensionality reduction. Alignment, matching, and map computation between geometric data sets. Function spaces and functional maps. Networks of data sets and joint analysis for segmentation and labeling. The emergence of abstractions or concepts from data. Prerequisites: discrete algorithms at the level of 161; linear algebra at the level of CME103. Same as: CME 251

CS 234. Reinforcement Learning. 3 Units.
To realize the dreams and impact of AI requires autonomous systems that learn to make good decisions. Reinforcement learning is one powerful paradigm for doing so, and it is relevant to an enormous range of tasks, including robotics, game playing, consumer modeling and healthcare. This class will briefly cover background on Markov decision processes and reinforcement learning, before focusing on some of the central problems, including scaling up to large domains and the exploration challenge. One key tool for tackling complex RL domains is deep learning and this class will include at least one homework on deep reinforcement learning. Prerequisites: proficiency in python, CS 229 or equivalents or permission of the instructor; linear algebra, basic probability.

CS 235. Computational Methods for Biomedical Image Analysis and Interpretation. 3-4 Units.
The latest biological and medical imaging modalities and their applications in research and medicine. Focus is on computational analytic and interpretive approaches to optimize extraction and use of biological and clinical imaging data for diagnostic and therapeutic translational medical applications. Topics include major image databases, fundamental methods in image processing and quantitative extraction of image features, structured recording of image information including semantic features and ontologies, indexing, search and content-based image retrieval. Case studies include linking image data to genomic, phenotypic and clinical data, developing representations of image phenotypes for use in medical decision support and research applications and the role that biomedical imaging informatics plays in new questions in biomedical science. Includes a project. Enrollment for 3 units requires instructor consent. Prerequisites: programming ability at the level of CS 106A, familiarity with statistics, basic biology. Knowledge of Matlab or Python highly recommended. Same as: BIOMEDIN 260, RAD 260
CS 236. Deep Generative Models. 3 Units.
Generative models are widely used in many subfields of AI and Machine Learning. Recent advances in parameterizing these models using neural networks, combined with progress in stochastic optimization methods, have enabled scalable modeling of complex, high-dimensional data including images, text, and speech. In this course, we will study the probabilistic foundations and learning algorithms for deep generative models, including Variational Autoencoders (VAE), Generative Adversarial Networks (GAN), and flow models. The course will also discuss application areas that have benefitted from deep generative models, including computer vision, speech and natural language processing, and reinforcement learning. Prerequisites: Basic knowledge about machine learning from at least one of CS 221, 228, 229 or 230. Students will work with computational and mathematical models and should have a basic knowledge of probabilities and calculus. Proficiency in some programming language, preferably Python, required.

CS 237A. Principles of Robot Autonomy I. 3-4 Units.
Basic principles for endowing mobile autonomous robots with perception, planning, and decision-making capabilities. Algorithmic approaches for robot perception, localization, and simultaneous localization and mapping; control of non-linear systems, learning-based control, and robot motion planning; introduction to methodologies for reasoning under uncertainty, e.g., (partially observable) Markov decision processes. Extensive use of the Robot Operating System (ROS) for demonstrations and hands-on activities. Prerequisites: CS 106A or equivalent, CME 100 or equivalent (for linear algebra), and CME 106 or equivalent (for probability theory). Same as: AA 174A, AA 274A, EE 260A

CS 237B. Principles of Robot Autonomy II. 3-4 Units.
This course teaches advanced principles for endowing mobile autonomous robots with capabilities to autonomously learn new skills and to physically interact with the environment and with humans. It also provides an overview of different robot system architectures. Concepts that will be covered in the course are: Reinforcement Learning and its relationship to optimal control, contact and dynamics models for prehensile and non-prehensile robot manipulation, imitation learning and human intent inference, as well as different system architectures and their verification. Students will earn the theoretical foundations for these concepts and implement them on mobile manipulation platforms. In homeworks, the Robot Operating System (ROS) will be used extensively for demonstrations and hands-on activities. Prerequisites: CS106A or equivalent, CME 100 or equivalent (for linear algebra), CME 106 or equivalent (for probability theory), and AA 171/274. Same as: AA 174B, AA 274B, EE 260B

CS 238. Decision Making under Uncertainty. 3-4 Units.
This course is designed to increase awareness and appreciation for why uncertainty matters, particularly for aerospace applications. It introduces decision making under uncertainty from a computational perspective and provides an overview of the necessary tools for building autonomous and decision-support systems. Following an introduction to probabilistic models and decision theory, the course will cover computational methods for solving decision problems with stochastic dynamics, model uncertainty, and imperfect state information. Topics include: Bayesian networks, influence diagrams, dynamic programming, reinforcement learning, and partially observable Markov decision processes. Applications cover: air traffic control, aviation surveillance systems, autonomous vehicles, and robotic planetary exploration. Prerequisites: basic probability and fluency in a high-level programming language. Same as: AA 228

CS 239. Advanced Topics in Sequential Decision Making. 3-4 Units.
Survey of recent research advances in intelligent decision making for dynamic environments from a computational perspective. Efficient algorithms for single and multiagent planning in situations where a model of the environment may or may not be known. Partially observable Markov decision processes, approximate dynamic programming, and reinforcement learning. New approaches for overcoming challenges in generalization from experience, exploration of the environment, and model representation so that these methods can scale to real problems in a variety of domains including aerospace, air traffic control, and robotics. Students are expected to produce an original research paper on a relevant topic. Prerequisites: AA 228/CS 238 or CS 221. Same as: AA 229

CS 240. Advanced Topics in Operating Systems. 3 Units.
Recent research. Classic and new papers. Topics: virtual memory management, synchronization and communication, file systems, protection and security, operating system extension techniques, fault tolerance, and the history and experience of systems programming. Prerequisite: 140 or equivalent.

CS 241. Embedded Systems Workshop. 3 Units.
Project-centric building hardware and software for embedded computing systems. Students work on an existing project of their own or join one of these projects. Syllabus topics will be determined by the needs of the enrolled students and projects. Examples of topics include: interrupts and concurrent programming, deterministic timing and synchronization, state-based programming models, filters, frequency response, and high-frequency signals, low power operation, system and PCB design, security, and networked communication. Prerequisite: CS107 (or equivalent).

CS 242. Programming Languages. 3-4 Units.
This course explores models of computation, both old, like functional programming with the lambda calculus (circa 1930), and new, like memory-safe systems programming with Rust (circa 2010). Topics include type systems (polymorphism, algebraic data types, static vs. dynamic), control flow (exceptions, continuations), concurrency/parallelism, metaprogramming, and the semantic gap between computational models and modern hardware. The study of programming languages is equal parts systems and theory, looking at how a rigorous understanding of the syntax, structure, and semantics of computation enables formal reasoning about the behavior and properties of complex real-world systems. In light of today's Cambrian explosion of new programming languages, this course also seeks to provide a conceptual clarity on how to compare and contrast the multitude of programming languages, models, and paradigms in the modern programming landscape. Prerequisites: 103, 110.

CS 243. Program Analysis and Optimizations. 3-4 Units.
Program analysis techniques used in compilers and software development tools to improve productivity, reliability, and security. The methodology of applying mathematical abstractions such as graphs, fixpoint computations, binary decision diagrams in writing complex software, using compilers as an example. Topics include data flow analysis, instruction scheduling, register allocation, parallelism, data locality, interprocedural analysis, and garbage collection. Prerequisites: 103 or 103B, and 107.

CS 244. Advanced Topics in Networking. 3-4 Units.
Classic papers, new ideas, and research papers in networking. Architectural principles: why the Internet was designed this way? Congestion control. Wireless and mobility; software-defined networks (SDN) and network virtualization; content distribution networks; packet switching; data-center networks. Prerequisite: 144 or equivalent.
CS 244B. Distributed Systems. 3 Units.
Distributed operating systems and applications issues, emphasizing high-level protocols and distributed state sharing as the key technologies. Topics: distributed shared memory, object-oriented distributed system design, distributed directory services, atomic transactions and time synchronization, application-sufficient consistency, file access, process scheduling, process migration, and storage/communication abstractions on distribution, scale, robustness in the face of failure, and security. Prerequisites: CS 144.

CS 245. Principles of Data-Intensive Systems. 3 Units.
Most important computer applications have to reliably manage and manipulate datasets. This course covers the architecture of modern data storage and processing systems, including relational databases, cluster computing frameworks, streaming systems and machine learning systems. Topics include storage management, query optimization, transactions, concurrency, fault recovery, and parallel processing, with a focus on the key design ideas shared across many types of data-intensive systems. Prerequisites: CS 145, 161.

CS 246. Mining Massive Data Sets. 3-4 Units.
Availability of massive datasets is revolutionizing science and industry. This course discusses data mining and machine learning algorithms for analyzing very large amounts of data. Topics include: Big data systems (Hadoop, Spark); Link Analysis (PageRank, spam detection); Similarity search (locality-sensitive hashing, shingling, minhashing, random hyperplanes); Stream data processing; Analysis of social-network graphs; Association rules; Dimensionality reduction (UV, SVD, and CUR decompositions); Algorithms for very-large-scale mining (clustering, nearest-neighbor search); Large-scale machine learning (gradient descent, decision tree ensembles); Multi-armed bandit, Computational advertising. We also offer a sister class CS246H (Hadoop Labs) and a follow-up project-based class CS341 (Project in Mining Massive Datasets). Prerequisites: At least one of CS107 or CS145.

CS 246H. Mining Massive Data Sets Hadoop Lab. 1 Unit.
Supplement to CS 246 providing additional material on the Apache Hadoop family of technologies. Students will learn how to implement data mining algorithms using Hadoop and Apache Spark, how to implement and debug complex data mining and data transformations, and how to use two of the most popular big data SQL tools. Topics: data mining, machine learning, data ingest, and data transformations using Hadoop, Spark, Apache Impala, Apache Hive, Apache Kafka, Apache Sqoop, Apache Flume, Apache Avro, and Apache Parquet. Prerequisite: CS 107 or equivalent.

CS 247A. Design for Artificial Intelligence. 3-4 Units.
A project-based course that builds on the introduction to design in CS147 by focusing on advanced methods and tools for research, prototyping, and user interface design. Studio based format with intensive coaching and iteration to prepare students for tackling real world design problems. This course takes place entirely in studios; you must plan on attending every studio to take this class. The focus of CS247A is design for human-centered artificial intelligence experiences. What does it mean to design for AI? What is HAI? Let’s explore what AI actually is and the constraints, opportunities and specialized processes necessary to create AI systems that work effectively for the humans involved. Prerequisites: 147 or equivalent background in design thinking.

CS 247G. Introduction to Game Design. 3-4 Units.
A project-based course that builds on the introduction to design in CS147 by focusing on advanced methods and tools for research, prototyping, and user interface design. Studio based format with intensive coaching and iteration to prepare students for tackling real world design problems. This course takes place entirely in studios; please plan on attending every studio to take this class. The focus of CS247G is an introduction to theory and practice of the design of games. We will make digital and paper games, do rapid iteration and run user research studies appropriate to game design. This class has multiple short projects, allowing us to cover a variety of genres, from narrative to pure strategy. Prerequisites: 147 or equivalent background.

CS 247I. Design for Understanding. 3-4 Units.
Complex problems require sophisticated approaches. In this project-based hands-on course, students explore the design of systems, information and interface for human use. We will model the flow of interactions, data and context, and crafting a design that is useful, appropriate and robust. Students will create utility apps or games as a response to the challenges presented. We will also examine the ethical consequences of design decisions and explore current issues arising from unintended consequences. Prerequisite: CS 147 or equivalent.

CS 247S. Service Design. 3-4 Units.
A project-based course that builds on the introduction to design in CS147 by focusing on advanced methods and tools for research, prototyping, and user interface design. Studio based format with intensive coaching and iteration to prepare students for tackling real world design problems. This course takes place entirely in studios; you must plan on attending every studio to take this class. The focus of CS247S is Service Design. In this course we will be looking at experiences that address the needs of multiple types of stakeholders at different touchpoints - digital, physical, and everything in between. If you’ve ever taken an Uber, participated in the Draw, engaged with your bank, or ordered a coffee through the Starbucks app, you’ve experienced a service that must have a coordinated experience for the customer, the service provider, and any other stakeholders involved. Let’s explore what specialized tools and processes are required to create these multi-faceted interactions. Prerequisites: 147 or equivalent background in design thinking.

CS 248. Interactive Computer Graphics. 3-4 Units.
This course provides a comprehensive introduction to interactive computer graphics, focusing on fundamental concepts and techniques, as well as their cross-cutting relationship to multiple problem domains in interactive graphics (such as rendering, animation, geometry, image processing). Topics include: 2D and 3D drawing, sampling theory, interpolation, rasterization, image compositing, the real-time GPU graphics pipeline (and parallel rendering), VR rendering, geometric transformations, curves and surfaces, geometric data structures, subdivision, meshing, spatial hierarchies, image processing, time integration, physically-based animation, and inverse kinematics. The course will involve several in-depth programming assignments and a self-selected final project that explores concepts covered in the class. Prerequisite: CS 107, MATH 51.

CS 250. Algebraic Error Correcting Codes. 3 Units.
Introduction to the theory of error correcting codes, emphasizing algebraic constructions, and diverse applications throughout computer science and engineering. Topics include basic bounds on error correcting codes; Reed-Solomon and Reed-Muller codes; list-decoding, list-recovery and locality. Applications may include communication, storage, complexity theory, pseudorandomness, cryptography, streaming algorithms, group testing, and compressed sensing. Prerequisites: Linear algebra, basic probability (at the level of, say, CS109, CME106 or EE178) and “mathematical maturity” (students will be asked to write proofs). Familiarity with finite fields will be helpful but not required. Same as: EE 387

CS 251. Cryptocurrencies and blockchain technologies. 3 Units.
For advanced undergraduates and for graduate students. The potential applications for Bitcoin-like technologies is enormous. The course will cover the technical aspects of cryptocurrencies, blockchain technologies, and distributed consensus. Students will learn how these systems work and how to engineer secure software that interacts with the Bitcoin network and other cryptocurrencies. Prerequisite: CS110. Recommended: CS255.
CS 252. Analysis of Boolean Functions. 3 Units.
Boolean functions are among the most basic objects of study in theoretical computer science. This course is about the study of boolean functions from a complexity-theoretic perspective, with an emphasis on analytic methods. We will cover fundamental concepts and techniques in this area, including influence and noise sensitivity, polynomial approximation, hypercontractivity, probabilistic invariance principles, and Gaussian analysis. We will see connections to various areas of theoretical computer science, including circuit complexity, pseudorandomness, classical and quantum query complexity, learning theory, and property testing. Prerequisites: CS 103 and CS 109 or equivalents. CS 154 and CS 161 recommended.

CS 253. Web Security. 3 Units.
Principles of web security. The fundamentals and state-of-the-art in web security. Attacks and countermeasures. Topics include: the browser security model, web app vulnerabilities, injection, denial-of-service, TLS attacks, privacy, fingerprinting, same-origin policy, cross site scripting, authentication, JavaScript security, emerging threats, defense-in-depth, and techniques for writing secure code. Course projects include writing security exploits, defending insecure web apps, and implementing emerging web standards. Prerequisite: CS 142 or equivalent web development experience.

CS 254. Computational Complexity. 3 Units.
An introduction to computational complexity theory. Topics include the P versus NP problem and other major challenges of complexity theory; Space complexity: Savitch's theorem and the Immerman-Szelepcsényi theorem; P, NP, coNP, and the polynomial hierarchy; The power of randomness in computation; Non-uniform computation and circuit complexity; Interactive proofs. Prerequisites: 154 or equivalent; mathematical maturity.

CS 254B. Computational Complexity II. 3 Units.
A continuation of CS254 (Computational Complexity). Topics include circuit complexity, proof complexity, communication and information complexity, average-case complexity, and complexity barriers. Prerequisite: CS 254.

CS 255. Introduction to Cryptography. 3 Units.
For advanced undergraduates and graduate students. Theory and practice of cryptographic techniques used in computer security. Topics: encryption (symmetric and public key), digital signatures, data integrity, authentication, key management, PKI, zero-knowledge protocols, and real-world applications. Prerequisite: basic probability theory.

CS 257. Logic and Artificial Intelligence. 2-4 Units.
This is a course at the intersection of philosophic logic and artificial intelligence. After reviewing recent work in AI that has leveraged ideas from logic, we will slow down and study in more detail various components of high-level intelligence and the tools that have been designed to capture those components. Specific areas will include: reasoning about belief and action, causality and counterfactuals, legal and normative reasoning, natural language inference, and Turing-complete logical formalisms including (probabilistic) logic programming and lambda calculus. Our main concern will be understanding the logical tools themselves, including their formal properties and how they relate to other tools such as probability and statistics. At the end, students should expect to have learned a lot more about logic, and also to have a sense for how logic has been and can be used in AI applications. Prerequisites: A background in logic, at least at the level of Phil 151, will be expected. In case a student is willing to put in the extra work to catch up, it may be possible to take the course with background equivalent to Phil 150 or CS 157. A background in AI, at the level of CS 221, would also be very helpful and will at times be expected. 2 unit option only for PhD students past the second year. Course website: http://web.stanford.edu/class/cs257/. Same as: PHIL 356C

CS 259Q. Quantum Computing. 3 Units.
The course introduces the basics of quantum algorithms, quantum computational complexity, quantum information theory, and quantum cryptography, including the models of quantum circuits and quantum Turing machines. Shor's factoring algorithms, Grover's search algorithm, the adiabatic algorithms, quantum error-correction, impossibility results for quantum algorithms, Bell's inequality, quantum information transmission, and quantum coin flipping. Prerequisites: knowledge of linear algebra, discrete probability and algorithms.

CS 260. Geometry of Polynomials in Algorithm Design. 3 Units.
Over the years, many powerful algorithms have been built via tools such as linear programming relaxations, spectral properties of graphs, and others, that all bridge the discrete and continuous worlds. This course will cover another such tool recently gaining popularity: polynomials, their roots, and their analytic properties, collectively known as geometry of polynomials. The course will cover fundamental properties of polynomials that are useful in designing algorithms, and then will showcase applications in several areas of algorithm design: combinatorial optimization, graph sparsification, high-dimensional expanders, analysis of random walks on combinatorial objects, and counting algorithms. Prerequisites: CS161 or equivalent. nBasic knowledge of probability, linear algebra, and calculus.

CS 261. Optimization and Algorithmic Paradigms. 3 Units.
Algorithms for network optimization: max-flow, min-cost flow, matching, assignment, and min-cut problems. Introduction to linear programming. Use of LP duality for design and analysis of algorithms. Approximation algorithms for NP-complete problems such as Steiner Trees, Traveling Salesman, and scheduling problems. Randomized algorithms. Introduction to sub-linear algorithms and decision making under uncertainty. Prerequisite: 161 or equivalent.

CS 264. Beyond Worst-Case Analysis. 3 Units.
This course is motivated by problems for which the traditional worst-case analysis of algorithms fails to differentiate meaningfully between different solutions, or recommends an intuitively "wrong" solution over the "right" one. This course studies systematically alternatives to traditional worst-case analysis that nevertheless enable rigorous and robust guarantees on the performance of an algorithm. Topics include: instance optimality; smoothed analysis; parameterized analysis and condition numbers; models of data (pseudorandomness, locality, diffuse adversaries, etc.); average-case analysis; robust distributional analysis; resource augmentation; planted and semi-random graph models. Motivating problems will be drawn from online algorithms, online learning, constraint satisfaction problems, graph partitioning, scheduling, linear programming, hashing, machine learning, and auction theory. Prerequisites: CS161 (required). CS261 is recommended but not required.

CS 265. Randomized Algorithms and Probabilistic Analysis. 3 Units.
Randomness pervades the natural processes around us, from the formation of networks, to genetic recombination, to quantum physics. Randomness is also a powerful tool that can be leveraged to create algorithms and data structures which, in many cases, are more efficient and simpler than their deterministic counterparts. This course covers the key tools of probabilistic analysis, and application of these tools to understand the behaviors of random processes and algorithms. Emphasis is on theoretical foundations, though we will apply this theory broadly, discussing applications in machine learning and data analysis, networking, and systems. Topics include tail bounds, the probabilistic method, Markov chains, and martingales, with applications to analyzing random graphs, metric embeddings, random walks, and a host of powerful and elegant randomized algorithms. Prerequisites: CS 161 and STAT 116, or equivalents and instructor consent. Same as: CME 309
CS 268. Geometric Algorithms. 3 Units.

CS 269G. Almost Linear Time Graph Algorithms. 3 Units.
Over the past decade there has been an explosion in activity in designing new provably efficient fast graph algorithms. Leveraging techniques from disparate areas of computer science and optimization researchers have made great strides on improving upon the best known running times for fundamental optimization problems on graphs, in many cases breaking long-standing barriers to efficient algorithm design. In this course we will survey these results and cover the key algorithmic tools they leverage to achieve these breakthroughs. Possible topics include but are not limited to, spectral graph theory, sparsification, oblivious routing, local partitioning, Laplacian system solving, and maximum flow. Prerequisites: calculus and linear algebra.
Same as: MS&E 313

CS 269O. Introduction to Optimization Theory. 3 Units.
Introduction of core algorithmic techniques and proof strategies that underlie the best known provable guarantees for minimizing high dimensional convex functions. Focus on broad canonical optimization problems and survey results for efficiently solving them, ultimately providing the theoretical foundation for further study in optimization. In particular, focus will be on first-order methods for both smooth and non-smooth convex function minimization as well as methods for structured convex function minimization, discussing algorithms such as gradient descent, accelerated gradient descent, mirror descent, Newton’s method, interior point methods, and more. Prerequisite: multivariable calculus and linear algebra.
Same as: MS&E 213

CS 269R. Elements of Quantum Computer Programming. 3 Units.
For advanced undergraduates and for graduate students. Quantum computing is an emerging computational paradigm with vast potential. This course is an introduction to modern quantum programming for students who want to work with quantum computing technologies and learn about new paradigms of computation. A physics / quantum mechanics background is not required. Students will learn the model of quantum computation, quantum programming languages, hybrid quantum/classical programming, quantum algorithms, quantum error correction, and applications. The course is hands on using open source Python packages for working with publicly available quantum processors. Prerequisites: linear algebra and programming at the undergraduate level.

CS 270. Modeling Biomedical Systems: Ontology, Terminology, Problem Solving. 3 Units.
Methods for modeling biomedical systems and for building model-based software systems. Emphasis is on intelligent systems for decision support and Semantic Web applications. Topics: knowledge representation, controlled terminologies, ontologies, reusable problem solvers, and knowledge acquisition. Students learn about current trends in the development of advanced biomedical software systems and acquire hands-on experience with several systems and tools. Prerequisites: CS106A, basic familiarity with biology, probability, and logic.
Same as: BIOMEDIN 210

CS 271. Artificial Intelligence in Healthcare. 3-4 Units.
Healthcare is one of the most exciting application domains of artificial intelligence, with transformative potential in areas ranging from medical image analysis to electronic health records-based prediction and precision medicine. This course will involve a deep dive into recent advances in AI in healthcare, focusing in particular on deep learning approaches for healthcare problems. We will start from foundations of neural networks, and then study cutting-edge deep learning models in the context of a variety of healthcare data including image, text, multimodal and time-series data. In the latter part of the course, we will cover advanced topics on open challenges of integrating AI in a societal application such as healthcare, including interpretability, robustness, privacy and fairness. The course aims to provide students from diverse backgrounds with both conceptual understanding and practical grounding of cutting-edge research on AI in healthcare.
Same as: BIODS 220, BIOMEDIN 220

CS 272. Introduction to Biomedical Informatics Research Methodology. 3-5 Units.
Capstone Biomedical Informatics (BMI) experience. Hands-on software building. Student teams conceive, design, specify, implement, evaluate, and report on a software project in the domain of biomedicine. Creating written proposals, peer review, providing status reports, and preparing final reports. Issues related to research reproducibility. Guest lectures from professional biomedical informatics systems builders on issues related to the process of project management. Software engineering basics. Because the team projects start in the first week of class, attendance that week is strongly recommended. Prerequisites: BIOMEDIN 210 or 214 or 215 or 217 or 260. Preference to BMI graduate students. Consent of instructor required.
Same as: BIOE 212, BIOMEDIN 212, GENE 212

CS 273A. The Human Genome Source Code. 3 Units.
A computational introduction to the most amazing programming language on the planet: your genome. Topics include genome sequencing (assembling source code from code fragments); the human genome functional landscape: variable assignments (genes), control-flow logic (gene regulation) and run-time stack (epigenomics); human disease and personalized genomics (as a hunt for bugs in the human code); genome editing (code injection) to cure the incurable; and the source code behind amazing animal adaptations. Algorithmic approaches will introduce ideas from computational genomics, machine learning and natural language processing. Course includes primers on molecular biology, and text processing languages. Prerequisites: CS106B or equivalent.
Same as: BIOMEDIN 273A, DBIO 273A

CS 273B. Deep Learning in Genomics and Biomedicine. 3 Units.
Recent breakthroughs in high-throughput genomic and biomedical data are transforming biological sciences into “big data” disciplines. In parallel, progress in deep neural networks are revolutionizing fields such as image recognition, natural language processing and, more broadly, AI. This course explores the exciting intersection between these two advances. The course will start with an introduction to deep learning and overview the relevant background in genomics and high-throughput biotechnology, focusing on the available data and their relevance. It will then cover the ongoing developments in deep learning (supervised, unsupervised and generative models) with the focus on the applications of these methods to biomedical data, which are beginning to produce dramatic results. In addition to predictive modeling, the course emphasizes how to visualize and extract interpretable, biological insights from such models. Recent papers from the literature will be presented and discussed. Students will be introduced to and work with popular deep learning software frameworks. Students will work in groups on a final class project using real world datasets. Prerequisites: College calculus, linear algebra, basic probability and statistics such as CS 109, and basic machine learning such as CS 229. No prior knowledge of genomics is necessary.
Same as: BIODS 237, BIOMEDIN 273B, GENE 236
CS 273C. Cloud Computing for Biology and Healthcare. 3 Units.
Big Data is radically transforming healthcare. To provide real-time personalized healthcare, we need hardware and software solutions that can efficiently store and process large-scale biomedical datasets. In this class, students will learn the concepts of cloud computing and parallel systems' architecture. This class prepares students to understand how to design parallel programs for computationally intensive medical applications and how to run these applications on computing frameworks such as Cloud Computing and High Performance Computing (HPC) systems. Prerequisites: familiarity with programming in Python and R. Same as: BIOMEDIN 222, GENE 222

CS 274. Representations and Algorithms for Computational Molecular Biology. 3-4 Units.
Topics: introduction to bioinformatics and computational biology, algorithms for alignment of biological sequences and structures, computing with strings, phylogenetic tree construction, hidden Markov models, basic structural computations on proteins, protein structure prediction, protein threading techniques, homology modeling, molecular dynamics and energy minimization, statistical analysis of 3D biological data, integration of data sources, knowledge representation and controlled terminologies for molecular biology, microarray analysis, machine learning (clustering and classification), and natural language text processing. Prerequisite: CS 106B; recommended: CS161; consent of instructor for 3 units. Same as: BIOE 214, BIOMEDIN 214, GENE 214

CS 275. Translational Bioinformatics. 4 Units.
Computational methods for the translation of biomedical data into diagnostic, prognostic, and therapeutic applications in medicine. Topics: multi-scale omics data generation and analysis, utility and limitations of public biomedical resources, machine learning and data mining, issues and opportunities in data discovery, and mobile/digital health solutions. Case studies and course project. Prerequisites: programming ability at the level of CS 106A and familiarity with biology and statistics. Same as: BIOE 217, BIOMEDIN 217, GENE 217

CS 275A. Symbolic Musical Information. 2-4 Units.
Focus on symbolic data for music applications including advanced notation systems, optical music recognition, musical data conversion, and internal structure of MIDI files. Same as: MUSIC 253

CS 275B. Music Query, Analysis, and Style Simulation. 2-4 Units.
Leveraging off three synchronized sets of symbolic data resources for notation and analysis, the lab portion introduces students to the open-source Humdrum Toolkit for music representation and analysis. Issues of data content and quality as well as methods of information retrieval, visualization, and summarization are considered in class. Grading based primarily on student projects. Prerequisite: 253 or consent of instructor. Same as: MUSIC 254

CS 276. Information Retrieval and Web Search. 3 Units.
Text information retrieval systems; efficient text indexing; Boolean, vector space, and probabilistic retrieval models; ranking and rank aggregation; evaluating IR systems; text clustering and classification; Web search engines including crawling and indexing, link-based algorithms, web metadata, and question answering; distributed word representations. Prerequisites: CS 107, CS 109, CS 161. Same as: LINGUIST 286

CS 278. Social Computing. 3 Units.
Today we interact with our friends and enemies, our team partners and romantic partners, and our organizations and societies, all through computational systems. How do we design these social computing systems to be effective and responsible? This course covers design patterns for social computing and crowdsourcing systems, and the foundational ideas that underpin them. Students will engage in the creation of new computationally-mediated social environments.

CS 279. Computational Biology: Structure and Organization of Biomolecules and Cells. 3 Units.
Computational techniques for investigating and designing the three-dimensional structure and dynamics of biomolecules and cells. These computational methods play an increasingly important role in drug discovery, medicine, bioengineering, and molecular biology. Course topics include protein structure prediction, protein design, drug screening, molecular simulation, cellular-level simulation, image analysis for microscopy, and methods for solving structures from crystallography and electron microscopy data. Prerequisites: elementary programming background (CS 106A or equivalent) and an introductory course in biology or biochemistry. Same as: BIOE 279, BIOMEDIN 279, BIOPHYS 279, CME 279

CS 28. Artificial Intelligence, Entrepreneurship and Society in the 21st Century and Beyond. 2 Units.
Technical developments in artificial intelligence (AI) have opened up new opportunities for entrepreneurship, as well as raised profound longer term questions about how human societal and economic systems may be reorganized to accommodate the rise of intelligent machines. In this course, closely co-taught by a Stanford professor and a leading Silicon Valley venture capitalist, we will examine the current state of the art capabilities of existing artificial intelligence systems, as well as economic challenges and opportunities in early stage startups and large companies that could leverage AI. We will focus on gaps between business needs and current technical capabilities to identify high impact directions for the development of future AI technology. Simultaneously, we will explore the longer term societal impact of AI driven by inexorable trends in technology and entrepreneurship. The course includes guest lectures from leading technologists and entrepreneurs who employ AI in a variety of fields, including healthcare, education, self-driving cars, computer security, natural language interfaces, computer vision systems, and hardware acceleration.

CS 294A. Research Project in Artificial Intelligence. 3 Units.
Student teams under faculty supervision work on research and implementation of a large project in AI. State-of-the art methods related to the problem domain. Prerequisites: AI course from 220 series, and consent of instructor.

CS 294S. Research Project in Software Systems and Security. 3 Units.
Topics vary. Focus is on emerging research themes such as programmable open mobile Internet that spans multiple system topics such as human-computer interaction, programming systems, operating systems, networking, and security. May be repeated for credit. Prerequisites: CS 103 and 107.

CS 294W. Writing Intensive Research Project in Computer Science. 3 Units.
Restricted to Computer Science and Computer Systems Engineering undergraduates. Students enroll in the CS 294W section attached to the CS 294 project they have chosen.

CS 298. Seminar on Teaching Introductory Computer Science. 1 Unit.
Faculty, undergraduates, and graduate students interested in teaching discuss topics raised by teaching computer science at the introductory level. Prerequisite: consent of instructor. Same as: EDUC 298

CS 300. Departmental Lecture Series. 1 Unit.
Priority given to first-year Computer Science Ph.D. students. CS Masters students admitted if space is available. Presentations by members of the department faculty, each describing informally his or her current research interests and views of computer science as a whole.

CS 309. Industrial Lectureships in Computer Science. 1 Unit.
Guest computer scientist. By arrangement. May be repeated for credit.
CS 309A. Cloud Computing Seminar. 1 Unit.
For science, engineering, computer science, business, education, medicine, and law students. Cloud computing is bringing information systems out of the back office and making it core to the entire economy. Furthermore with the advent of smarter machines cloud computing will be integral to building a more precision planet. This class is intended for all students who want to begin to understand the implications of this technology. Guest industry experts are public company CEOs who are either delivering cloud services or using cloud services to transform their businesses.

CS 315B. Parallel Computing Research Project. 3 Units.
Advanced topics and new paradigms in parallel computing including parallel algorithms, programming languages, runtime environments, library debugging/tuning tools, and scalable architectures. Research project. Prerequisite: consent of instructor.

CS 316. Advanced Multi-Core Systems. 3 Units.
In-depth coverage of the architectural techniques used in modern, multi-core chips for mobile and server systems. Advanced processor design techniques (superscalar cores, VLIW cores, multi-threaded cores, energy-efficient cores), cache coherence, memory consistency, vector processors, graphics processors, heterogeneous processors, and hardware support for security and parallel programming. Students will become familiar with complex trade-offs between performance-power-complexity and hardware-software interactions. A central part of CS316 is a project on an open research question on multi-core technologies. Prerequisites: EE 180 (formerly 108B) and EE 282. Recommended: CS 149.

CS 319. Topics in Digital Systems. 3 Units.
Advanced material is often taught for the first time as a topics course, perhaps by a faculty member visiting from another institution. May be repeated for credit.

CS 320. Value of Data and AI. 3 Units.
Many of the most valuable companies in the world and the most innovative startups have business models based on data and AI, but our understanding about the economic value of data, networks and algorithmic assets remains at an early stage. For example, what is the value of a new dataset or an improved algorithm? How should investors value a data-centric business such as Netflix, Uber, Google, or Facebook? And what business models can best leverage data and algorithmic assets in settings as diverse as e-commerce, manufacturing, biotech and humanitarian organizations? In this graduate seminar, we will investigate these questions by studying recent research on these topics and by hosting in-depth discussions with experts from industry and academia. Key topics will include value of data quantity and quality in statistics and AI, business models around data, networks, scaling effects, economic theory around data, and emerging data protection regulations. Students will also conduct a group research projects in this field. Prerequisites: Sufficient mathematical maturity to follow the technical content; some familiarity with data mining and machine learning and at least an undergraduate course in statistics are recommended.

CS 323. Automated Reasoning: Theory and Applications. 3-4 Units.
Intelligent computer agents must reason about complex, uncertain, and dynamic environments. This course is a graduate level introduction to automated reasoning techniques and their applications, covering logical and probabilistic approaches. Topics include: logical and probabilistic foundations, backtracking strategies and algorithms behind modern SAT solvers, stochastic local search and Markov Chain Monte Carlo algorithms, variational techniques, classes of reasoning tasks and reductions, and applications.

CS 325B. Data for Sustainable Development. 3-5 Units.
The sustainable development goals (SDGs) encompass many important aspects of human and ecosystem well-being that are traditionally difficult to measure. This project-based course will focus on ways to use inexpensive, unconventional data streams to measure outcomes relevant to SDGs, including poverty, hunger, health, governance, and economic activity. Students will apply machine learning techniques to various projects outlined at the beginning of the quarter. The main learning goals are to gain experience conducting and communicating original research. Prior knowledge of machine learning techniques, such as from CS 221, CS 229, CS 231N, STAT 202, or STAT 216 is required. Open to both undergraduate and graduate students. Enrollment limited to 24. Students must apply for the class by filling out the form at https://goo.gl/forms/9LSZF7iPkhAdtxS1. A permission code will be given to admitted students to register for the class. Same as: EARTHSYS 162, EARTHSYS 262

CS 326. Topics in Advanced Robotic Manipulation. 3-4 Units.
This course provides a survey of the most important and influential concepts in autonomous robotic manipulation. It includes classical concepts that are still widely used and recent approaches that have changed the way we look autonomous manipulation. We cover approaches towards motion planning and control using visual and tactile perception as well as machine learning. This course is especially concerned with new approaches for overcoming challenges in generalization from experience, exploration of the environment, and learning representation so that these methods can scale to real problems. Students are expected to present one paper in a tutorial, debate a paper once from the Pro and once from the Con side. They are also expected to propose an original research project and work on it towards a research paper. Recommended: CS 131, 223A, 229 or equivalents.

CS 327A. Advanced Robotic Manipulation. 3 Units.
Advanced control methodologies and novel design techniques for complex human-like robotic and bio mechanical systems. Class covers the fundamentals in operational space dynamics and control, elastic planning, human motion synthesis. Topics include redundancy, inertial properties, haptics, simulation, robot cooperation, mobile manipulation, human-friendly robot design, humanoids and whole-body control. Additional topics in emerging areas are presented by groups of students at the end-of-quarter mini-symposium. Prerequisites: 223A or equivalent.

CS 328. Topics in Computer Vision. 3 Units.
Fundamental issues of, and mathematical models for, computer vision. Sample topics: camera calibration, texture, stereo, motion, shape representation, image retrieval, experimental techniques. May be repeated for credit. Prerequisites: 205, 223B, or equivalents.

CS 329. Topics in Artificial Intelligence. 3 Units.
Advanced material is often taught for the first time as a topics course, perhaps by a faculty member visiting from another institution. May be repeated for credit.

CS 329M. Topics in Artificial Intelligence: Algorithms of Advanced Machine Learning. 3 Units.
This advanced graduate course explores in depth several important classes of algorithms in modern machine learning. We will focus on understanding the mathematical properties of these algorithms in order to gain deeper insights on when and why they perform well. We will also study applications of each algorithm on interesting, real-world settings. Topics include: spectral clustering, tensor decomposition, Hamiltonian Monte Carlo, adversarial training, and variational approximation. Students will learn mathematical techniques for analyzing these algorithms and hands-on experience in using them. We will supplement the lectures with latest papers and there will be a significant research project component to the class. Prerequisites: Probability (CS 109), linear algebra (Math 113), machine learning (CS 229), and some coding experience.
CS 330. Deep Multi-task and Meta Learning. 3 Units.  
While deep learning has achieved remarkable success in supervised and reinforcement learning problems, such as image classification, speech recognition, and game playing, these models are, to a large degree, specialized for the single task they are trained for. This course will cover the setting where there are multiple tasks to be solved, and study how the structure arising from multiple tasks can be leveraged to learn more efficiently or effectively. This includes: goal-conditioned reinforcement learning techniques that leverage the structure of the provided goal space to learn many tasks significantly faster; meta-learning methods that aim to learn efficient learning algorithms that can learn new tasks quickly; curriculum and lifelong learning, where the problem requires a sequence of tasks, leveraging their shared structure to enable knowledge transfer. This is a graduate-level course. By the end of the course, students should be able to understand and implement the state-of-the-art multi-task learning algorithms and be ready to conduct research on these topics. Prerequisites: CS 229 or equivalent. Familiarity with deep learning, reinforcement learning, and machine learning is assumed.

CS 331B. Representation Learning in Computer Vision. 3 Units.  
A representation performs the task of converting an observation in the real world (e.g. an image, a recorded speech signal, a word in a sentence) into a mathematical form (e.g. a vector). This mathematical form is then used by subsequent steps (e.g. a classifier) to produce the output, such as classifying an image or recognizing a spoken word. Forming the proper representation for a task is an essential problem in modern AI. In this course, we focus on 1) establishing why representations matter, 2) classical and modern methods of forming representations in Computer Vision, 3) methods of analyzing and probing representations, 4) portraying the future landscape of representations with generic and comprehensive AI/vision systems over the horizon, and finally 5) going beyond computer vision by talking about non-visual representations, such as the ones used in NLP or neuroscience. The course will heavily feature systems based on deep learning and convolutional neural networks. We will have several teaching lectures, a number of prominent external guest speakers, as well as presentations by the students on recent papers and their projects. nnRequired Prerequisites: CS131A, CS231A, CS231B, or CS231N. If you do not have the required prerequisites, please contact a member of the course staff before enrolling in this course.

CS 332. Advanced Survey of Reinforcement Learning. 3 Units.  
This class will provide a core overview of essential topics and new research frontiers in reinforcement learning. Planned topics include: model free and model based reinforcement learning, policy search, Monte Carlo Tree Search planning methods, off policy evaluation, exploration, imitation learning, temporal abstraction/hierarchical approaches, safety and risk sensitivity, human-in-the-loop RL, inverse reinforcement learning, learning to communicate, and insights from human learning. Students are expected to create an original research paper on a related topic. Prerequisites: CS 221 or AA 238/CS 238 or CS 234 or CS 229 or similar experience.

CS 333. Algorithms for Interactive Robotics. 3-4 Units.  
Once confined to the manufacturing floor, robots are quickly entering the public space at multiple levels: drones, surgical robots, service robots, and self-driving cars are becoming tangible technologies impacting the human experience. Our goal in this class is to learn about and design algorithms that enable robots to reason about their actions, interact with one another, the humans, and the environment they live in, as well as plan safe strategies that humans can trust and rely on. This is a project-based graduate course that covers a broad set of algorithms in robotics, machine learning, and control theory for the goal of developing interactive human-robot systems. Recommended: Introductory course in AI, machine learning, and robotics.

CS 334A. Convex Optimization I. 3 Units.  
Convex sets, functions, and optimization problems. The basics of convex analysis and theory of convex programming: optimality conditions, duality theory, theorems of alternative, and applications. Least-squares, linear and quadratic programs, semidefinite programming, and geometric programming. Numerical algorithms for smooth and equality constrained problems; interior-point methods for inequality constrained problems. Applications to signal processing, communications, control, analog and digital circuit design, computational geometry, statistics, machine learning, and mechanical engineering. Prerequisite: linear algebra such as EE263, basic probability. Same as: CME 364A, EE 364A

CS 335. Fair, Accountable, and Transparent (FAT) Deep Learning. 3 Units.  
Deep learning-based AI systems have demonstrated remarkable learning capabilities. A growing field in deep learning research focuses on improving the Fairness, Accountability, and Transparency (FAT) of a model in addition to its performance. Although FAT will be difficult to achieve, emerging technical approaches in this topic show promise in making better FAT AI systems. In this course, we will study the rigorous computer science necessary for FAT deep learning and dive into the technical underpinnings of topics including fairness, robustness, interpretability, common sense, AI deception, and privacy. These topics reflect state-of-the-art research in FAT, are socially important, and they have strong industrial interest due to government and other policy regulation. This course will focus on the algorithmic and statistical methods needed to approach FAT AI from a deep learning perspective. We will also discuss several application areas where we can apply these techniques. Prerequisites: Intermediate knowledge of statistics, machine learning, and AI. Qualified students will have taken any one of the following, or their advanced equivalents: CS224N, CS230, CS231N, CS236, CS273B. Alternatively, students who have taken CS229 or have equivalent knowledge can be admitted with the permission of the instructors.

CS 336. Robot Perception and Decision-Making: Optimal and Learning-based Approaches. 3-4 Units.  
How can robots perceive the world and their own motion so that they can accomplish navigation and manipulation tasks? In this course, we will study how this question has been approached specifically if the robot is equipped with visual sensing capabilities. We focus on studying how a robot can make decisions based on raw, high-dimensional sensory data that represents only partial, noisy observations of the environment. Therefore, the course is divided into two main themes (i) Estimation and (ii) Decision-Making and Control where in each theme we will study traditional approaches, learning-based methods and combinations of those. Prerequisites: CS106B, MATH 51 or CME 100, CS109, CS 221 or CS 229.

CS 337. AI-Assisted Care. 1 Unit.  
AI has been advancing quickly, with its impact everywhere. In healthcare, innovation in AI could help transforming our healthcare system. This course offers a diverse set of research projects focusing on cutting edge computer vision and machine learning technologies to solve some of healthcare’s most important problems. The teaching team and teaching assistants will work closely with students on research projects in this area. Research projects include Care for Senior at Senior Home, Surgical Quality Analysis, AI Assisted Parenting, Burn Analysis & Assessment and more. AI areas include Video Understanding, Image Classification, Object Detection, Segmentation, Action Recognition, Deep Learning, Reinforcement Learning, HCI and more. The course is open to students in both school of medicine and school of engineering. Same as: MED 277

CS 340. Topics in Computer Systems. 3-4 Units.  
Topics vary every quarter, and may include advanced material being taught for the first time. May be repeated for credit.
CS 341. Project in Mining Massive Data Sets. 3 Units.
Students work in teams of three to solve a problem involving the analysis of a massive dataset. A proposal, early in March is required. There will be an information session (announced in CS246) explaining the datasets available in early March and this information will also be on the CS341 course website in late February. Each accepted team will be assigned a mentor who will work with them regularly throughout the quarter. Teams will also be provided access to significant computing resources on a commercial public cloud.

CS 342. Building for Digital Health. 3 Units.
This project-based course will provide a comprehensive overview of key requirements in the design and full-stack implementation of a digital health research application. Several pre- vetted and approved projects from the Stanford School of Medicine will be available for students to select from and build. Student teams learn about all necessary approval processes to deploy a digital health solution (data privacy clearance/IRB approval, etc.) and be guided in the development of front-end and back-end infrastructure using best practices. The final project will be the presentation and deployment of a fully approved digital health research application. CS106A, CS106B. Recommended: CS193P/A, CS142, CS47, CS110. Limited enrollment for this course. Students need to submit their application online via: https://docs.google.com/forms/d/e/1FAIpQLSfENfanSF1T8LvfcCS9RLOQ90g_NF2jETx3pQ8Y8bjxToR7g/viewform.
Same as: MED 253

CS 344. Topics in Computer Networks. 3 Units.
This class could also be called "Build an Internet Router": Students work in teams of two to build a fully functioning Internet router, gaining hands-on experience building the hardware and software of a high-performance network system. Students design the control plane in C on a linux host and design the data plane in the new P4 language on the NetFPGA 4 x 10GE switch. For the midterm milestone, teams must demonstrate that their routers can interoperate with the other teams by building a small scale datacenter topology. In the final 3-4 weeks of the class, teams will participate in an open-ended design challenge. Prerequisites: At least one student in each team must have taken CS144 at Stanford and completed Lab 3 (static router). No Verilog or FPGA programming experience is required. May be repeated for credit.

CS 345S. Data-intensive Systems for the Next 1000x. 3-4 Units.
The last decade saw enormous shifts in the design of large-scale data-intensive systems due to the rise of Internet services, cloud computing, and Big Data processing. Where will we see the next 1000x increases in scale and data volume, and how should data-intensive systems accordingly evolve? This course will critically examine a range of trends, including the Internet of Things, drones, smart cities, and emerging hardware capabilities, through the lens of software systems research and design. Students will perform a comparative analysis by reading and discussing cutting-edge research while performing their own original research. Prerequisites: Strong background in software systems, especially databases (CS 245) and distributed systems (CS 244B), and/or machine learning (CS 229). Undergraduates who have completed CS 245 are strongly encouraged to attend.

CS 347. Human-Computer Interaction Research. 3-4 Units.
(Previously numbered CS 376.) Prepares students to conduct original HCI research by reading and discussing seminal and cutting-edge research papers. Main topics are interaction, social computing, and design; breadth topics include AI+HCI, media tools, access, programming tools, and visualization. Student groups perform a quarter-long research project. Prerequisites: For CS and Symbolic Systems undergraduates/masters students, CS 147 or CS 247. No prerequisite for PhD students or students outside of CS and Symbolic Systems.

CS 348A. Computer Graphics: Geometric Modeling & Processing. 3-4 Units.

CS 348B. Computer Graphics: Image Synthesis Techniques. 3-4 Units.
Intermediate level, emphasizing high-quality image synthesis algorithms and systems issues in rendering. Topics include: Reyes and advanced rasterization, including motion blur and depth of field; ray tracing and physically based rendering; Monte Carlo algorithms for rendering, including direct illumination and global illumination; path tracing and photon mapping; surface reflection and light source models; volume rendering and subsurface scattering; SIMD and multi-core parallelism for rendering. Written assignments and programming projects. Prerequisite: 248 or equivalent. Recommended: Fourier analysis or digital signal processing.

CS 348C. Computer Graphics: Animation and Simulation. 3 Units.

CS 348E. Character Animation: Modeling, Simulation, and Control of Human Motion. 3 Units.
This course introduces technologies and mathematical tools for simulating, modeling, and controlling human/animal movements. Students will be exposed to integrated knowledge and techniques across computer graphics, robotics, machine learning and biomechanics. The topics include numerical integration, 3D character modeling, keyframe animation, skinning/rigging, multi-body dynamics, human kinematics, muscle dynamics, trajectory optimization, learning policies for motor skills, and motion capture. Students who successfully complete this course will be able to use and modify physics simulator for character animation or robotic applications, to design/train control policies for locomotion or manipulation tasks on virtual agents, and to leverage motion capture data for synthesizing realistic virtual humans. The evaluation of this course is based on three assignments and an open-ended research project. Prerequisite: CS 148.
CS 349K. Visual Computing Systems. 3-4 Units.
Visual computing tasks such as computational photography, image/ video understanding, and real-time 3D graphics are key responsibilities of modern computer systems ranging from sensor-rich smart phones, autonomous robots, and large data centers. These workloads demand exceptional system efficiency and this course examines the key ideas, techniques, and challenges associated with the design of parallel, heterogeneous systems that execute and accelerate visual computing applications. This course is intended for graduate and advanced undergraduate-level students interested in architecting efficient graphics, image processing, and computer vision systems (both new hardware architectures and domain-optimized programming frameworks) and for students in graphics, vision, and ML that seek to understand throughput computing concepts so they can develop scalable algorithms for these platforms. Students will perform daily research paper readings, complete simple programming assignments, and compete a self-selected term project. Prerequisites: CS 107 or equivalent. Highly recommended: Parallel Computing (CS149) or Computer Architecture (EE 282). Students will benefit from some background in deep learning (CS 230, CS 231N), computer vision (CS 231A), digital image processing (CS 232) or computer graphics (CS248).

CS 349. Topics in Programming Systems. 3 Units.
Advanced material is often taught for the first time as a topics course, perhaps by a faculty member visiting from another institution. May be repeated for credit.

CS 349D. Cloud Computing Technology. 3 Units.
The largest change in the computer industry over the past five years has arguably been the emergence of cloud computing: organizations are increasingly moving their workloads to managed public clouds and using new, global-scale services that were simply not possible in private datacenters. However, both building and using cloud systems remains a black art with many difficult research challenges. This research seminar will cover industry and academic work on cloud computing and survey challenges including programming interfaces, cloud native applications, resource management, pricing, availability and reliability, privacy and security. Students will also propose and develop an original research project. Prerequisites: For graduate students, background in computer systems (CS 240, 244, 244B or 245) is strongly recommended. Undergrads will need instructor's approval.

CS 349F. Technology for Financial Systems. 2 Units.
Financial systems have spurred technological innovation and, in turn, are driven by cutting-edge technological developments. This course explores the synergy students will learn from faculty and industry experts how to build faster and fairer financial systems. Topics include network infrastructure: data center fabrics, ultra-low latency trading systems; cloud computing infrastructure: building large-scale risk computation platforms using virtual machines, containers and serverless computing. A particular focus will be on challenges and opportunities presented by cloud-native financial exchanges: the course will provide such an exchange and student groups will write programs for high-frequency and algorithmic trading. Recommended: Knowledge of basic Networking, OS, or Distributed Systems (CS 144, 140, or equivalent), as well as basic EE courses (EE 178) will be useful.

CS 350. Secure Compilation. 3 Units.
This course explores the field of secure compilation, which sits at the intersection between security and programming languages. The course covers the following topics: threat models for secure compilers, formal criteria for secure compilers to adhere to, security relevance of secure compilation criteria, security architectures employed to achieve secure compilation, proof techniques for secure compilation with a focus on backtranslation.

CS 352. Pseudo-Randomness. 3-4 Units.
Pseudorandomness is the widely applicable theory of efficiently generating objects that look random, despite being constructed using little or no randomness. Since pseudorandom objects can replace uniformly distributed ones (in a well-defined sense), one may view pseudorandomness as an extension of our understanding of randomness through the computational lens. We will study the basic tools pseudorandomness, such as limited independence, randomness extractors, expander graphs, and pseudorandom generators. We will also discuss the applications of pseudorandomness to derandomization, cryptography and more. We will cover classic result as well as cutting-edge techniques. Prerequisites: CS 154 and CS 161, or equivalents.

CS 353. The Practice of Theory Research. 3 Units.
Introduction to research in the Theory of Computing, with an emphasis on research methods (the practice of research), rather than on any particular body of knowledge. The students will participate in a limited-scope research project: starting from reading research papers from a critical point of view and conducting bibliography searches, through suggesting new research directions, identifying relevant technical areas, and finally producing and communicating new insights. The course will accompany the projects with basic insights on the main ingredients of research. Research experience is not required, but basic theory knowledge and mathematical maturity are expected. The target participants are advanced undergrads as well as MS students with interest in CS theory. Prerequisites: CS 161 and CS 154. Limited class size.

CS 354. Topics in Intractability: Unfulfilled Algorithmic Fantasies. 3 Units.
Over the past 45 years, understanding NP-hardness has been an amazingly useful tool for algorithm designers. This course will expose students to additional ways to reason about obstacles for designing efficient algorithms. Topics will include unconditional lower bounds (query- and communication-complexity), total problems, Unique Games, average-case complexity, and fine-grained complexity. Prerequisites: CS 161 or equivalent. CS 254 recommended but not required.

CS 355. Advanced Topics in Cryptography. 3 Units.
Topics: Pseudo randomness, multiparty computation, pairing-based and lattice-based cryptography, zero knowledge protocols, and new encryption and integrity paradigms. May be repeated for credit. Prerequisite: CS255.

CS 356. Topics in Computer and Network Security. 3 Units.
Research seminar covering foundational work and current topics in computer and network security. Students will read and discuss published research papers as well as complete an original research project in small groups. Open to Ph.D. and masters students as well as advanced undergraduate students. Prerequisites: While the course has no official prerequisites, students need a mature understanding of software systems and networks to be successful. We strongly encourage students to first take CS155: Computer and Network Security.

CS 357. Advanced Topics in Formal Methods. 3 Units.
Topics vary annually. Recent offerings have covered the foundations of static analysis, including decision procedures for important theories (SAT, linear integer constraints, SMT solvers), model checking, abstract interpretation, and constraint-based analysis. May be repeated for credit.

CS 358. Topics in Programming Language Theory. 3 Units.
Advanced material is often taught for the first time as a topics course, perhaps by a faculty member visiting from another institution. May be repeated for credit.

CS 358A. Programming Language Foundations. 3 Units.
This course introduces advanced formal systems and programming languages as well as techniques to reason formally about them. Possible systems of study include: the lambda calculus, System F, the Pi and Spi calculi, simply-typed languages, security type systems for non-interference, robust safety, linear types, ownership types, session types, logical relations and semantic models etc.
CS 359. Topics in the Theory of Computation. 3 Units.
Advanced material is often taught for the first time as a topics course, perhaps by a faculty member visiting from another institution. May be repeated for credit.

CS 360. Simplicity and Complexity in Economic Theory. 2-5 Units.
Technology has enabled the emergence of economic systems of formerly inconceivable complexity. Nevertheless, some technology-related economic problems are so complex that either supercomputers cannot solve them in a reasonable time, or they are too complex for humans to comprehend. Thus, modern economic designs must still be simple enough for humans to understand, and must address computationally complex problems in an efficient fashion. This topics course explores simplicity and complexity in economics, primarily via theoretical models. We will focus on recent advances. Key topics include (but are not limited to) resource allocation in complex environments, communication complexity and information aggregation in markets, robust mechanisms, dynamic matching theory, influence maximization in networks, and the design of simple (user-friendly) mechanisms. Some applications include paired kidney exchange, auctions for electricity and for radio spectrum, ride-sharing platforms, and the diffusion of information. Prerequisites: Econ 203 or equivalent.
Same as: ECON 284

CS 361. Engineering Design Optimization. 3-4 Units.
Design of engineering systems within a formal optimization framework. This course covers the mathematical and algorithmic fundamentals of optimization, including derivative and derivative-free approaches for both linear and non-linear problems, with an emphasis on multidisciplinary design optimization. Topics will also include quantitative methodologies for addressing various challenges, such as accommodating multiple objectives, automating differentiation, handling uncertainty in evaluations, selecting design points for experimentation, and principled methods for optimization when evaluations are expensive. Applications range from the design of aircraft to automated vehicles. Prerequisites: some familiarity with probability, programming, and multivariable calculus.
Same as: AA 222

CS 366. Computational Social Choice. 3 Units.
An in-depth treatment of algorithmic and game-theoretic issues in social choice. Topics include common voting rules and impossibility results; ordinal vs cardinal voting; market approaches to large scale decision making; voting in complex elections, including multi-winner elections and participatory budgeting; protocols for large scale negotiation and deliberation; fairness in societal decision making; algorithmic approaches to governance of modern distributed systems such as blockchains and community-mediated social networks; opinion dynamics and polarization. Prerequisites: algorithms at the level of 212 or CS 161, probability at the level of 221, and basic game theory, or consent of instructor.
Same as: MS&E 336

CS 368. Algorithmic Techniques for Big Data. 3 Units.
Designing algorithms for efficient processing of large data sets poses unique challenges. This course will discuss algorithmic paradigms that have been developed to efficiently process data sets that are much larger than available memory. We will cover streaming algorithms and sketching methods that produce compact data structures, dimension reduction methods that preserve geometric structure, efficient algorithms for numerical linear algebra, graph sparsification methods, as well as impossibility results for these techniques.

CS 369. Topics in Analysis of Algorithms. 3 Units.
Advanced material is often taught for the first time as a topics course, perhaps by a faculty member visiting from another institution. May be repeated for credit.

CS 369H. Hierarchies of Integer Programming Relaxations. 3 Units.
Mathematical programming relaxations of integer programming formulations are a popular way to apply convex optimization techniques to hard combinatorial optimization problems. Such relaxations can be made closer to their integer programming counterparts by adding constraints; a systematic way to achieve this is via hierarchies of relaxations. Several such hierarchies are well-studied in the literature: Lovasz-Schrijver, Sherali-Adams and the Parrilo-Lasserre sum-of-squares (SoS) hierarchy. Recently, these hierarchies have received a lot of attention due to their potential to make progress on long standing algorithmic questions, and connections to various other areas such as computational complexity, combinatorial and polynomial optimization, quantum computing, proof complexity and so on. In this course we will cover recent research results in this area for problems arising from optimization, machine learning, computational complexity and more, discussing both lower and upper bounds. Prerequisites: Mathematical maturity (required), exposure to algorithms (strongly recommended), and optimization (recommended).

CS 369L. Algorithmic Perspective on Machine Learning. 3 Units.
Many problems in machine learning are intractable in the worst case, and pose a challenge for the design of algorithms with provable guarantees. In this course, we will discuss several success stories at the intersection of algorithm design and machine learning, focusing on devising appropriate models and mathematical tools to facilitate rigorous analysis.

CS 369M. Metric Embeddings and Algorithmic Applications. 3 Units.
Low distortion embeddings of finite metric spaces is a topic at the intersection of mathematics and theoretical computer science. Much progress in this area in recent years has been motivated by algorithmic applications. Mapping complicated metrics of interest to simpler metrics (normed spaces, trees, and so on) gives access to a powerful algorithmic toolkit for approximation algorithms, online algorithms as well as for efficient search and indexing of large data sets. In a different vein, convex relaxations are a useful tool for graph partitioning problems; central to the analysis are metric embedding questions for certainly computationally defined metrics. In this course, we will see several classical and recent results on metric embeddings with a focus on algorithmic applications. Students will be expected to have a strong background in algorithms and probability.

CS 371. Computational Biology in Four Dimensions. 3 Units.
Cutting-edge research on computational techniques for investigating and designing the three-dimensional structure and dynamics of biomolecules, cells, and everything in between. These techniques, which draw on approaches ranging from physics-based simulation to machine learning, play an increasingly important role in drug discovery, medicine, bioengineering, and molecular biology. Course is devoted primarily to reading, presentation, discussion, and critique of papers describing important recent research developments. Prerequisite: CS 106A or equivalent, and an introductory course in biology or biochemistry. Recommended: some experience in mathematical modeling (does not need to be a formal course).
Same as: BIOMEDIN 371, BIOPHYS 371, CME 371
CS 372. Artificial Intelligence for Disease Diagnosis and Information Recommendations. 3 Units.
Artificial intelligence, specifically deep learning, stands out as one of the most transformative technologies of the past decade. AI can already outperform humans in several computer vision and natural language processing tasks. However, we still face some of the same limitations and obstacles that led to the demise of the first AI boom phase five decades ago. This research-oriented course will first review and reveal the limitations (e.g., iid assumption on training and testing data, voluminous training data requirement, and lacking interpretability) of some widely used AI algorithms, including convolutional neural networks (CNNs), transformers, reinforcement learning, and generative adversarial networks (GANs). To address these limitations, we will then explore topics including transfer learning for remediying data scarcity, knowledge-guided multimodal learning for improving data diversity, out of distribution generalization, attention mechanisms for enabling Interpretability, meta learning, and privacy-preserving training data management. The course will be taught through a combination of lecture and project sessions. Lectures on specialized AI applications (e.g., cancer/depression diagnosis and treatment, AI/VR for surgery, and health education) will feature guest speakers from academia and industry. Students will be assigned to work on an extensive project that is relevant to their fields of study (e.g., CS, Medicine, and Data Science). Projects may involve conducting literature surveys, formulating ideas, and implementing these ideas. Example project topics are but not limited to 1) knowledge guided GANs for improving training data diversity, 2) disease diagnosis via multimodal symptom checking, and 3) fake and biased news/information detection.

CS 373. Statistical and Machine Learning Methods for Genomics. 3 Units.
Introduction to statistical and computational methods for genomics. Sample topics include: expectation maximization, hidden Markov model, Markov chain Monte Carlo, ensemble learning, probabilistic graphical models, kernel methods and other modern machine learning paradigms. Rationales and techniques illustrated with existing implementations used in population genetics, disease association, and functional regulatory genomics studies. Instruction includes lectures and discussion of readings from primary literature. Homework and projects require implementing some of the algorithms and using existing toolkits for analysis of genomic datasets.

Same as: BIO 268, BIOMEDIN 245, GENE 245, STATS 345

CS 375. Large-Scale Neural Network Modeling for Neuroscience. 1-3 Unit.
Introduction to designing, building, and training large-scale neural networks for modeling brain and behavioral data, including: deep convolutional neural network models of sensory systems (vision, audition, somatosensation); variational and generative methods for neural interpretation; recurrent neural networks for dynamics, memory and attention; interactive agent-based deep reinforcement learning for cognitive modeling; and methods and metrics for comparing such models to real-world neural data. Attention will be given both to established methods as well as cutting-edge techniques. Students will learn conceptual bases for deep neural network models and will also implement learn to implement and train large-scale models in Tensorflow using GPUs. Requirements: Fluency in Unix shell and Python programming; familiarity with differential equations, linear algebra, and probability theory; prior experience with modern machine learning concepts (e.g. CS229) and basic neural network training tools (eg. CS230 and/or CS231n). Prior knowledge of basic cognitive science or neuroscience not required but helpful.

Same as: PSYCH 249

CS 377. Topics in Human-Computer Interaction. 2-3 Units.
Contents change each quarter. May be repeated for credit. See http://hci.stanford.edu/academics for offerings.

CS 377E. Designing Solutions to Global Grand Challenges. 3-4 Units.
In this course we creatively apply information technologies to collectively attack Global Grand Challenges (e.g., global warming, rising healthcare costs and declining access, and ensuring quality education for all). Interdisciplinary student teams will carry out need-finding within a target domain, followed by brainstorming to propose a quarter long project. Teams will spend the rest of the quarter applying user-centered design methods to rapidly iterate through design, prototyping, and testing of their solutions. This course will interweave a weekly lecture with a weekly studio session where students apply the techniques hands-on in a small-scale, supportive environment.

CS 377G. Designing Serious Games. 3-4 Units.
Over the last few years we have seen the rise of "serious games" to promote understanding of complex social and ecological challenges, and to create passion for solving them. This project-based course provides an introduction to game design principals while applying them to games that teach. Run as a hands-on studio class, students will design and prototype games for social change and civic engagement. We will learn the fundamentals of games design via lecture and extensive reading in order to make effective games to explore issues facing society today. The course culminates in an end-of-quarter open house to showcase our games. Prerequisite: CS147 or equivalent. 247G recommended, but not required.

CS 377N. Introduction to the Design of Smart Products. 3-4 Units.
This course will focus on the technical mechatronic skills as well as the human factors and interaction design considerations required for the design of smart products and devices. Students will learn techniques for rapid prototyping of smart devices, best practices for physical interaction design, fundamentals of affordances and signifiers, and interaction across networked devices. Students will be introduced to design guidelines for integrating electrical components such as PCBs into mechanical assemblies and consider the physical form of devices, not just as enclosures but also as a central component of the smart product. Prerequisites include: CS106A and E40 highly recommended, or instructor approval.

Same as: ME 216M

CS 377P. Advanced User Interface Design Patterns. 3 Units.
User interface design is about creating the most effective, intuitive design possible to help users achieve a specific goal. While understanding users is one part of the equation, the other part is a strong understanding of user interface design rules and patterns that you can apply to solve their needs. This course will deep dive into user interface design across mobile, desktop, and wearable platforms covering common patterns, when to use them, and when to break them. Each week will cover a different user interface design challenge and explore the patterns in areas such as data input, search & filters, tables and lists, content organization, navigation, dark patterns and more. Through the use of in class exercises, integrated design challenges, and an exploration of examples, students will leave the class knowing how to integrate user interface patterns into their design work to create powerful, effective digital experiences. Prerequisite: CS 147 or equivalent. 247 recommended but not required.

CS 377Q. Designing for Accessibility. 3-4 Units.
Designing for accessibility is a valuable and important skill in the UX domain, followed by brainstorming to propose a quarter long project. Teams will spend the rest of the quarter applying user-centered design methods to rapidly iterate through design, prototyping, and testing of their solutions. This course will interweave a weekly lecture with a weekly studio session where students apply the techniques hands-on in a small-scale, supportive environment.
CS 377T. Topics in Human-Computer Interaction: Teaching Studio Classes. 3 Units.

Studio teaching is a practice that dates back to the apprentice days of art studios. In this course, you will learn to teach project-based classes that include critique. We will also cover effective coaching, design of projects and exercises, and curating material in order to maximize the effectiveness of a flipped classroom. Recommended for TAs in HCI.

CS 377U. Understanding Users. 3-4 Units.

This project-based class focuses on understanding the use of technology in the world. Students will learn generative and evaluative research methods to explore how systems are appropriated into everyday life in a quarter-long project where they design, implement and evaluate a novel mobile application. Quantitative (e.g. A/B testing, instrumentation, analytics, surveys) and qualitative (e.g. diary studies, contextual inquiry, ethnography) methods and their combination will be covered along with practical experience applying these methods in their project. Prerequisites: CS 147, 193A/193P (or equivalent mobile programming experience).

CS 379. Interdisciplinary Topics. 3 Units.

Advanced material is often taught for the first time as a topics course, perhaps by a faculty member visiting from another institution. May be repeated for credit.

CS 379C. Computational Models of the Neocortex. 3 Units.

This course emphasizes approaches to scaling the technologies of computer science and systems neuroscience to take advantage of the exponential trend in computational power known as Moore’s Law. Modern methods in signal processing and machine learning are combined with technologies for managing large datasets common in industry. Classes feature scientists presenting novel approaches for analyzing the structure and function of complex neural circuits. Grading is based on class participation (30%), a project proposal due at midterm (20%), and a final project demonstration and report due by the end of finals (50%). Team projects are encouraged, especially multi-disciplinary collaborations. Prerequisites are basic high-school biology, good math skills and familiarity with machine learning. Some background in computer vision and signal processing is important for projects in structural analysis. Familiarity with modern artificial neural network technologies is a plus for projects in functional analysis. For more detail, see http://www.stanford.edu/class/cs379c/ with special attention to the CALENDAR and DISCUSSION tabs from past classes available by following the ARCHIVES link.

CS 384. Seminar on Ethical and Social Issues in Natural Language Processing. 3-4 Units.

Seminar covering issues in natural language processing related to ethical and social issues and the overall impact of these algorithms on people and society. Topics include: bias in data and models, privacy and computational profiling, measuring civility and toxicity online, computational propaganda, manipulation and framing, fairness/equity, power, recommendations and filter bubbles, applications to social good, and philosophical foundations of ethical investigation. Prerequisites: CS 224N and 224U.

CS 390A. Curricular Practical Training. 1 Unit.

Educational opportunities in high technology research and development labs in the computing industry. Qualified computer science students engage in internship work and integrate that work into their academic program. Students register under their faculty advisor during the quarter they are employed and complete a research report outlining their work activity, problems investigated, results, and follow-on projects they expect to perform. CS390A, CS390B, and CS390C may each be taken once.

CS 390B. Curricular Practical Training. 1 Unit.

Educational opportunities in high technology research and development labs in the computing industry. Qualified computer science students engage in internship work and integrate that work into their academic program. Students register under their faculty advisor during the quarter they are employed and complete a research report outlining their work activity, problems investigated, results, and follow-on projects they expect to perform. CS390A, CS390B, and CS390C may each be taken once.

CS 390C. Curricular Practical Training. 1 Unit.

Educational opportunities in high technology research and development labs in the computing industry. Qualified computer science students engage in internship work and integrate that work into their academic program. Students register under their faculty advisor during the quarter they are employed and complete a research report outlining their work activity, problems investigated, results, and follow-on projects they expect to perform. CS390A, CS390B, and CS390C may each be taken once.

CS 390D. Part-time Curricular Practical Training. 1 Unit.

For qualified computer science PhD students only. Permission number required for enrollment; see the CS PhD program administrator in Gates room 195. Educational opportunities in high technology research and development labs in the computing industry. Qualified computer science PhD students engage in research and integrate that work into their academic program. Students register under their faculty advisor during the quarter they are employed and complete a research report outlining their work activity, problems investigated, results, and follow-on projects they expect to perform. CS 390A, CS390B, and CS390C may each be taken once.

CS 399. Independent Project. 1-9 Unit.

Letter grade only. This course is for graduate students only. Undergraduate students should enroll in CS199.

CS 399P. Independent Project. 1-9 Unit.

Graded satisfactory/no credit. This course is for graduate students only. Undergraduate students should enroll in CS199P.
CS 402. Beyond Bits and Atoms: Designing Technological Tools. 3-4 Units.
This course is a practicum in the design of technology-enabled curricula and hands-on learning environments. It focuses on the theories, concepts, and practices necessary to design effective, low-cost educational technologies that support learning in all contexts for a variety of diverse learners. We will explore theories and design frameworks from constructivist and constructionist learning perspectives, as well as the lenses of critical pedagogy, Universal Design for Learning (UDL), and interaction design for children. The course will concretize theories, concepts, and practices in weekly presentations (including examples) from industry experts with significant backgrounds and proven expertise in designing successful, evidence-based, educational technology products. The Practicum provides the design foundation for EDUC 211 / CS 402 L, a hands-on lab focused on introductory prototyping and the fabrication of incipient interactive, educational technologies. (No prior prototyping experience required.) Interested students must also register for either EDUC 211 or CS 402L, complete the application at bit.ly/BBA-Winter2020 by January 4 at 5 p.m., and come to the first class at 8:30 a.m. in CERAS 108.
Same as: EDUC 236

CS 402L. Beyond Bits and Atoms - Lab. 1-3 Unit.
This lab course is a hands-on introduction to the prototyping and fabrication of tangible, interactive technologies, with a special focus on learning and education. (No prior prototyping experience required.) It focuses on the design and prototyping of low-cost technologies that support learning in all contexts for a variety of diverse learners. You will be introduced to, and learn how to use state-of-the-art fabrication machines (3D printers, laser cutters, Go Go Boards, Sensors, etc.) to design educational toolkits, educational toys, science kits, and tangible user interfaces. The lab builds on the theoretical and evidence-based foundations explored in the EDUC 236 / CS 402 Practicum. Interested students must also register for either EDUC 236 or CS 402, complete the application at bit.ly/BBA-Winter2020 by January 4 at 5 p.m., and come to the first class at 8:30 a.m. in CERAS 108.
Same as: EDUC 211

CS 41. Hap.py Code: The Python Programming Language. 2 Units.
The fundamentals and contemporary usage of the Python programming language. Primary focus is on developing best practices in writing Python and exploring the extensible and unique parts of the Python language. Topics include: Pythonic conventions, data structures such as list comprehensions, anonymous functions, iterables, and powerful built-ins (e.g. map, filter, zip). We will also focus on data analysis tools including NumPy, Pandas, Matplotlib, and Scikit-learn for their application in machine learning. Prerequisite: CS106B, CS106X, or equivalent. Application required.

CS 421. Designing AI to Cultivate Human Well-Being. 2 Units.
Artificial Intelligence (AI) has the potential to drive us towards a better future for all of humanity, but it also comes with significant risks and challenges. At its best, AI can help humans mitigate climate change, diagnose and treat diseases more effectively, enhance learning, and improve access to capital throughout the world. But it also has the potential to exacerbate human biases, destroy trust in information flow, displace entire industries, and amplify inequality throughout the world. We have arrived at a pivotal moment in the development of the technology in which we must establish a foundation for how we will design AI to capture the positive potential and mitigate the negative risks. To do this, we must be intentional about human-centered design because, ¿Only once we have thought hard about what sort of future we want, will we be able to begin steering a course toward a desirable future. If we don’t know what we want, we’re unlikely to get it.¿ Thus, building AI must be an inclusive, interactive, and introspective process guided by an affirmative vision of a beneficial AI-future. The goal of this interdisciplinary class is to bridge the gap between technological and societal objectives: How do we design AI to promote human well-being? The ultimate aim is to provide tools and frameworks to build a more harmonious human society based on cooperation toward a shared vision. Thus, students are trained in basic science to understand what brings about the conditions for human flourishing and will create meaningful AI technologies that aligns with the PACE framework: Has a clear and meaningful purpose, augments human dignity and autonomy, creates a feeling of inclusivity and collaboration, creates shared prosperity and a sense of forward movement (excellence). Toward this end, students work in interdisciplinary teams on a final project and propose a solution that tackles a significant societal challenge by leveraging technology and frameworks on human thriving.

This course will introduce the probabilistic approach to cognitive science, in which learning and reasoning are understood as inference in complex probabilistic models. Examples will be drawn from areas including concept learning, causal reasoning, social cognition, and language understanding. Formal modeling ideas and techniques will be discussed in concert with relevant empirical phenomena.
Same as: PSYCH 204

CS 43. Functional Programming Abstractions. 2 Units.
This course covers the fundamentals of functional programming and algebraic type systems, and explores a selection of related programming paradigms and current research. Haskell is taught and used throughout the course, though much of the material is applicable to other languages. Material will be covered from both theoretical and practical points of view, and topics will include higher order functions, immutable data structures, algebraic data types, type inference, lenses and optics, effect systems, concurrency and parallelism, and dependent types. Prerequisites: Programming maturity and comfort with math proofs, at the levels of CS107 and CS103.

CS 431. High-level Vision: From Neurons to Deep Neural Networks. 1-3 Unit.
Interdisciplinary seminar focusing on understanding how computations in the brain enable rapid and efficient object perception. Covers topics from multiple perspectives drawing on recent research in Psychology, Neuroscience, and Computer Science. Emphasis on discussing recent empirical findings, methods and theoretical debates in the field.
Same as: PSYCH 250

CS 448. Topics in Computer Graphics. 3-4 Units.
Topic changes each quarter. Recent topics: computational photography, datavisualization, character animation, virtual worlds, graphics architectures, advanced rendering. See http://graphics.stanford.edu/ courses for offerings and prerequisites. May be repeated for credit.
CS 448B. Data Visualization. 4 Units.
Techniques and algorithms for creating effective visualizations based on principles from graphic design, visual art, perceptual psychology, and cognitive science. Topics: graphical perception, data and image models, visual encoding, graph and tree layout, color, animation, interaction techniques, automated design. Lectures, reading, and project. Prerequisite: one of CS147, CS148, or equivalent.

CS 448H. Topics in Computer Graphics: Agile Hardware Design. 3 Units.
Topic changes each quarter. Recent topics: computational photography, data visualization, character animation, virtual worlds, graphics architectures, advanced rendering. See http://graphics.stanford.edu/ for offerings and prerequisites. May be repeated for credit.

CS 448I. Computational Imaging and Display. 3 Units.
Spawned by rapid advances in optical fabrication and digital processing power, a new generation of imaging technology is emerging: computational cameras at the convergence of applied mathematics, optics, and high-performance computing. Similar trends are observed for modern displays pushing the boundaries of resolution, contrast, 3D capabilities, and immersive experiences through the co-design of optics, electronics, and computation. This course serves as an introduction to the emerging field of computational imaging and displays. Students will learn to master bits and photons.

Same as: EE 367

CS 448M. Making Making Machines for Makers. 3-4 Units.
An introductory, project-based exploration of systems and processes for making things using computer-aided design and manufacturing, and an introduction to machines and machine tools. Emphasis will be placed on building novel machines and related software for use by "makers" and interactive machines. Course projects will encourage students to understand, build and modify/hack a sequence of machines: (1) an embroidery machine for custom textiles, (2) a paper cutting machine (with drag knife) for ornamental design, and (3) an XY plotter with Arduino controller. Through these projects students explore both (i) principles of operation (mechanical, stepper motors and servos, electrical control, computer software), and (ii) computer algorithms (trajectory, tool path, design). Current trends in interactive machines will be surveyed. The course will culminate in a final student-selected project. Prerequisite: CS106A or equivalent programming experience. Students should have a desire to make things.

CS 448V. Topics in Computer Graphics: Computational Video Manipulation. 3 Units.
The goal of this graduate (advanced undergraduate also welcome) course is to survey recent work on computational video analysis and manipulation techniques. We will learn how to acquire, represent, edit and remix video. Several popular video manipulation algorithms will be presented, with an emphasis on using these techniques to build practical systems. Students will have the opportunity to acquire their own video and implement the processing tools needed to computationally analyze and manipulate it. The course will be project based with a substantial final project.

CS 45N. Computers and Photography: From Capture to Sharing. 4 Units.
Preference to freshmen with experience in photography and use of computers. Elements of photography, such as lighting, focus, depth of field, aperture, and composition. How a photographer makes photos available for computer viewing, reliably stores them, organizes them, tags them, searches them, and distributes them online. No programming experience required. Digital SLRs and editing software will be provided to those students who do not wish to use their own.

CS 468. Topics in Geometric Algorithms: Riemannian Methods in Computer Vision and Biomedical Imaging. 3 Units.
Contents of this course change with each offering. Past offerings have included geometric matching, surface reconstruction, collision detection, computational topology, and 3D machine learning. May be repeated for credit. The Spring 2019/20 iteration will focus on data analysis and machine learning in non-flat spaces, with applications to computer vision and biomedical imaging. Prerequisites: linear algebra and basic probability theory, coding.

CS 47. Cross-Platform Mobile Development. 2 Units.
The fundamentals of cross-platform mobile application development using the React Native framework (RN). Primary focus on developing best practices in creating apps for both iOS and Android by using Javascript and existing web + mobile development paradigms. Students will explore the unique aspects that made RN a primary tool for mobile development within Facebook, Instagram, Walmart, Tesla, and UberEats. Skills developed over the course will be consolidated by the completion of a final project. Required Prerequisites: CS106A or CS106B. Website: web.stanford.edu/class/cs47/. To enroll in the class, please show up to the first day of class and fill the following application: https://forms.gle/WM3SDd3qyF3eQC3x5.

CS 476A. Music, Computing, Design I: The Art of Design. 3-4 Units.
Creative design for computer music software. Programming, audiovisual design, as well as software design for musical tools, instruments, toys, and games. Provides paradigms and strategies for designing and building music software, with emphases on interactive systems, aesthetics, and artistic product design. Course work includes several programming assignments and a "design+implement" final project. Prerequisite: experience in C/C++ and/or Java. See https://ccrma.stanford.edu/courses/256a/.

Same as: MUSIC 256A

CS 476B. Music, Computing, Design II: Virtual and Augmented Reality for Music. 3-4 Units.
Aesthetics, design, and exploration of creative musical applications of virtual reality (VR) and augmented reality (AR), centered around VR and mobile technologies. Comparison between AR, VR, and traditional software design paradigms for music. Topics include embodiment, interaction design, novel instruments, social experience, software design + prototyping. Prerequisite: MUSIC 256A / CS 476A.

Same as: MUSIC 256B

CS 499. Advanced Reading and Research. 1-15 Unit.
Letter grade only. Advanced reading and research for CS graduate students. Register using the section number associated with the instructor. Prerequisite: consent of instructor. This course is for graduate students only. Undergraduate students should enroll in CS199.

CS 499P. Advanced Reading and Research. 1-15 Unit.
Graded satisfactory/no credit. Advanced reading and research for CS graduate students. Register using the section number associated with the instructor. Prerequisite: consent of instructor. This course is for graduate students only. Undergraduate students should enroll in CS199P.
CS 49N. Using Bits to Control Atoms. 3 Units.
This is a crash course in how to use a stripped-down computer system about the size of a credit card (the raspberry pi computer) to control as many different sensors as we can implement in ten weeks, including LEDs, motion sensors, light controllers, and accelerometers. The ability to fearlessly grab a set of hardware devices, examine the data sheet to see how to use it, and stitch them together using simple code is a secret weapon that software-only people lack, and allows you to build many interesting gadgets. We will start with a “bare metal” system — no operating system, no support — and teach you how to read device data sheets describing sensors and write the minimal code needed to control them (including how to debug when things go wrong, as they always do). This course differs from most in that it is deliberately mostly about what and why rather than how — our hope is that the things you are able at the end will inspire you to follow the rest of the CS curriculum to understand better how things you’ve used work. Prerequisites: knowledge of the C programming language. A Linux or Mac laptop that you are comfortable coding on.

CS 50. Using Tech for Good. 2 Units.
Students in the class will work in small teams to implement high-impact projects for partner organizations. Taught by the CS+Social Good team, the aim of the class is to empower you to leverage technology for social good by inspiring action, facilitating collaboration, and forging pathways towards global change. Recommended: CS 106B, CS 42 or 142. Class is open to students of all years. May be repeated for credit. Cardinal Course certified by the Haas Center.

CS 51. CS + Social Good Studio: Designing Social Impact Projects. 2 Units.
Get real-world experience researching and developing your own social impact project! Students work in small teams to develop high-impact projects around problem domains provided by partner organizations, under the guidance and support of design/technical coaches from industry and non-profit domain experts. Main class components are workshops, community discussions, guest speakers and mentorship. Studio provides an outlet for students to create social change through CS while engaging in the full product development cycle on real-world projects. The class culminates in a showcase where students share their project ideas and Minimum Viable Product prototypes with stakeholders and the public. Application required; please see cs51.stanford.edu for more information.

CS 52. CS + Social Good Studio: Implementing Social Good Projects. 2 Units.
Continuation of CS51 (CS + Social Good Studio). Teams enter the quarter having completed and tested a minimal viable product (MVP) with a well-defined target user, and a community partner. Students will learn to apply scalable technical frameworks, methods to measure social impact, tools for deployment, user acquisition techniques and growth/exit strategies. The purpose of the class is to facilitate students to build a sustainable infrastructure around their product idea. CS52 will host mentors, guest speakers and industry experts for various workshops and coaching-sessions. The class culminates in a showcase where students share their projects with stakeholders and the public. Prerequisite: CS 51, or consent of instructor.

CS 520. Knowledge Graphs. 1 Unit.
Knowledge graphs have emerged as a compelling abstraction for organizing world structured knowledge over the internet, capturing relationships among key entities of interest to enterprises, and a way to integrate information extracted from multipledata sources. Knowledge graphs have also started to play a central role in machine learning and natural language processing as a method to incorporate world knowledge, as a target knowledge representation for extracted knowledge, and for explaining what is being learned. This class is a graduate level research seminar featuring prominent researchers and industry practitioners working on different aspects of knowledge graphs. It will showcase how latest research in AI, database systems and HCI is coming together in integrated intelligent systems centered around knowledge graphs.

CS 521. Seminar on AI Safety. 1 Unit.
In this seminar, we will focus on the challenges in the design of safe and verified AI-based systems. We will explore some of the major problems in this area from the viewpoint of industry and academia. We plan to have a weekly seminar speaker to discuss issues such as verification of AI systems, reward misalignment and hacking, secure and attack-resilient AI systems, diagnosis and repair, issues regarding policy and ethics, as well as the implications of AI safety in automotive industry. Prerequisites: There are no official prerequisites but an introductory course in artificial intelligence is recommended.

CS 522. Seminar in Artificial Intelligence in Healthcare. 1 Unit.
Artificial intelligence is poised to make radical changes in healthcare, transforming areas such as diagnosis, genomics, surgical robotics, and drug discovery. In the coming years, artificial intelligence has the potential to lower healthcare costs, identify more effective treatments, and facilitate prevention and early detection of diseases. This class is a seminar series featuring prominent researchers, physicians, entrepreneurs, and venture capitalists, all sharing their thoughts on the future of healthcare. We highly encourage students of all backgrounds to enroll (no AI/healthcare background necessary). Speakers and more at shift.stanford.edu/healthai.

CS 529. Robotics and Autonomous Systems Seminar. 1 Unit.
Seminar talks by researchers and industry professionals on topics related to modern robotics and autonomous systems. Broadly, talks will cover robotic design, perception and navigation, planning and control, and learning for complex robotic systems. May be repeated for credit. Same as: AA 289

CS 544. INTERACTIVE MEDIA AND GAMES. 1 Unit.
Interactive media and games increasingly pervade and shape our society. In addition to their dominant roles in entertainment, video games play growing roles in education, arts, and science. This seminar series brings together a diverse set of experts to provide interdisciplinary perspectives on these media regarding their history, technologies, scholarly research, industry, artistic value, and potential future. Same as: BIOE 196, BIOPHYS 196

CS 547. Human-Computer Interaction Seminar. 1 Unit.
Weekly speakers on human-computer interaction topics. May be repeated for credit.

CS 549. Human-Computer Interaction in the Real World. 1 Unit.
Intended for students who are pursuing a focus on HCI, this course focuses on showing students how HCI gets applied in industry across different types of companies. The course consists of on-site visits to large companies (for example Google, Yahoo, Square, Tesla) and to startups to talk to the HCI practitioners at these companies and learn first hand how HCI and design fits in at different companies. The objective of this class is to have students understand how HCI practitioners fit into organizations, the roles they play, and what skills they need in the real world to be able to do their magic.
CS 56N. Great Discoveries and Inventions in Computing. 3 Units.
This seminar will explore some of both the great discoveries that underlie computer science and the inventions that have produced the remarkable advances in computing technology. Key questions we will explore include: What is computable? How can information be securely communicated? How do computers fundamentally work? What makes computers fast? Our exploration will look both at the principles behind the discoveries and inventions, as well as the history and the people involved in those events. Some exposure to programming is required.

CS 571. Surgical Robotics Seminar. 1 Unit.
Surgical robots developed and implemented clinically on varying scales. Seminar goal is to expose students from engineering, medicine, and business to guest lecturers from academia and industry. Engineering and clinical aspects connected to design and use of surgical robots, varying in degree of complexity and procedural role. May be repeated for credit. Same as: ME 571

CS 57N. Randomness: Computational and Philosophical Approaches. 3 Units.
Is it ever reasonable to make a decision randomly? For example, would you ever let an important choice depend on the flip of a coin? Can randomness help us answer difficult questions more accurately or more efficiently? What is randomness anyway? Can an object be random? Are there genuinely random processes in the world, and if so, how can we tell? In this seminar, we will explore these questions through the lenses of philosophy and computation. By the end of the quarter students should have an appreciation of the many roles that randomness plays in both humanities and sciences, as well as a grasp of some of the key analytical tools used to study the concept. The course will be self-contained, and no prior experience with randomness/probability is necessary. Same as: PHIL 3N

CS 58. You Say You Want a Revolution. 2 Units.
This project-based course will give creative students an opportunity to work together on revolutionary change leveraging blockchain technology. The course will provide opportunities for students to become operationally familiar with blockchain concepts, supported by presentation of blockchain fundamentals at a level accessible to those with or without a strong technical background. Specific topics include: incentives, ethics, crypto-commons, values, FOMO 3D, risks, implications and social good. Students will each discover a new possible use-case for blockchain and prototype their vision for the future accordingly. Application and impact areas may come from medicine, law, economics, history, anthropology, or other sectors. Student diversity of background will be valued highly. Same as: Blockchain Edition

CS 581. Media Innovation. 1 Unit.
This course will introduce students interested in computer science, engineering, and media to what's possible and probable when it comes to media innovation. nnSpeakers from multiple disciplines and industry will discuss a range of topics in the context of evolving media with a focus on the technical trends, opportunities and challenges surfacing in the unfolding media ecosystem. Speakers will underscore the need to innovate to survive in the media and information industries. Open to both undergraduates and graduate students.

CS 58N. The Blockchain Revolution Will Not Be Televised. 3 Units.
This seminar will explore the nature of revolutions supported and enabled by technological change, using the Internet and smart phone as two historical examples and focusing on blockchain technology and potential applications such as money, banking, supply chain and market trading. In this project-based course, one meeting per week will bring in new information, including visiting experts. Other class meetings will involve team work, presentations, and discussion. Each student will help lead a section; the class collectively will produce a final book/movie/blog, in a medium selected by the class.

CS 7. Personal Finance for Engineers. 1 Unit.
Introduction to the fundamentals and analysis specifically needed by engineers to make informed and intelligent financial decisions. Course will focus on actual industry-based financial information from technology companies and realistic financial issues. Topics include: behavioral finance, budgeting, debt, compensation, stock options, investing and real estate. No prior finance or economics experience required.

CS 801. TGR Project. 0 Units.

CS 802. TGR Dissertation. 0 Units.

CS 80Q. Race and Gender in Silicon Valley. 3 Units.
Join us as we go behind the scenes of some of the big headlines about trouble in Silicon Valley. We'll start with the basic questions like who decides who gets to see themselves as "a computer person," and how do early childhood and educational experiences shape our perceptions of our relationship to technology? Then we'll see how those questions are fundamental to a wide variety of recent events from #metoo in tech companies, to the ways the under-representation of women and people of color in tech companies impacts the kinds of products that Silicon Valley brings to market. We'll see how data and the coming age of AI raise the stakes on these questions of identity and technology. How can we ensure that AI technology will help reduce bias in human decision-making in areas from marketing to criminal justice, rather than amplify it? Same as: AFRICAAM 80Q

CS 81SI. AI Interpretability and Fairness. 1 Unit.
As black-box AI models grow increasingly relevant in human-centric applications, explainability and fairness becomes increasingly necessary for trust in adopting AI models. This seminar class introduces students to major problems in AI explainability and fairness, and explores key state-of-the-art methods. Key technical topics include surrogate methods, feature visualization, network dissection, adversarial debiasing, and fairness metrics. There will be a survey of recent legal and policy trends. Each week a guest lecturer from AI research, industry, and related policy fields will present an open problem and solution, followed by a roundtable discussion with the class. Students have the opportunity to present a topic of interest/or application to their own projects (solo or in teams) in the final class. Code examples of each topic will be provided for students interested in a particular topic, but there will be no required coding components. Students who will benefit most from this class have exposure to AI, such as through projects and related coursework (e.g. statistics, CS221, CS230, CS229). Students who are pursuing subjects outside of the CS department (e.g. sciences, social sciences, humanities) with sufficient mathematical maturity are welcomed to apply. Enrollment limited to 20.

CS 82SI. Workshop for Wellness in Tech. 1 Unit.
This course is an interactive seminar focused on tackling and re-framing questions about the intersection of technology and wellness. What are the principles of human flourishing, and what is technology’s role in promoting public wellness? How can an appreciation of diversity and inclusion lead to the development of products that make people happier and healthier? How can we mindfully design tech to embrace user stories? Once we’ve taken a holistic look at the intersection of technology and wellness, we’ll invite leaders from industry to speak on how they’ve put the theory in action. They will help us tackle the question: how do you start and build a venture focused on improving human flourishing? This course will attempt to balance lecture and hands-on activities so that each session both delivers new concepts and serves as a workshop on flourishing - honing your awareness of what makes you tick.
CS 83. Playback Theater. 3 Units.
Playback combines elements of theater, community work and storytelling. In a playback show, a group of actors and musicians create an improvised performance based on the audience's personal stories. A playback show brings about a powerful listening and sharing experience. During the course, we will tell, listen, play together, and train in playback techniques. We will write diaries to process our experience in the context of education and research. The course is aimed to strengthen listening abilities, creativity and the collaborative spirit, all integral parts of doing great science. In playback, as in research, we are always moving together, from the known, to the unknown, and back. There is limited enrollment for this class. Application is required.

CS 84. Emotional Intelligence. 2 Units.
This hands-on course is aimed at Stanford engineers who wish to be successful in start-ups or engineering-focused organizations. It is based on decades of observations by the instructors, witnessing that fresh graduates routinely struggle to survive and create an impact in the corporate world. A key objective is for students to develop a basic set of skills to master day-to-day personal interactions, and to understand the dynamics of work environments. The course then aims to guide students with more complex tasks, such as how to run effective meetings or how to work in multi-disciplinary teams. Whether you wish to become a start-up founder and CEO; a manager at a tech-centric company; or an individual contributor at Facebook or Google: if you wish to hit the ground running and be highly effective from your first day at work, this course is for you.

CS 9. Problem-Solving for the CS Technical Interview. 1 Unit.
This course will prepare students to interview for software engineering and related internships and full-time positions in industry. Drawing on multiple sources of actual interview questions, students will learn key problem-solving strategies specific to the technical/coding interview. Students will be encouraged to synthesize information they have learned across different courses in the major. Emphasis will be on the oral and combination written-oral modes of communication common in coding interviews, but which are unfamiliar settings for problem solving for many students. Prerequisites: CS 106B or X.

CS 91SI. Digital Canvas: An Introduction to UI/UX Design. 2 Units.
Become familiar with prototype-design tools like Sketch and Marvel while also learning important design concepts in a low-stress environment. Focus is on the application of UI/UX design concepts to actual user interfaces: the creation of wireframes, high-fidelity mockups, and clickable prototypes. We will look at what makes a good or bad user interface, effective design techniques, and how to employ these techniques using Sketch and Marvel to make realistic prototypes. This course is ideal for anyone with little to no visual design experience who would like to build their skill set in UI/UX for app or web design. Also ideal for anyone with experience in front or back-end web development or human-computer interaction that would want to sharpen their visual design and analysis skills for UI/UX.

CS 93. Teaching AI. 1 Unit.
For graduate students who are TA-ing an AI course. This course prepares new AI section leaders to teach, write, and evaluate AI content. In class, you will be evaluating final projects individually and as a group. You will have discussions criticizing papers and assigning grades to them. You will analyze and solve discussion session problems on the board, explain algorithms like backpropagation, and learn how to give constructive feedback to students. The class will also include a guest speaker who will give teaching advice and talk about AI. Focus is on teaching skills, techniques, and final projects grading. The class meets once a week for the first 6 weeks of the quarter.