# COMPUTER SCIENCE (CPSC)

**CPSC S100a / CPSC 100a, Introduction to Computing and Programming** Ozan Erat
In-person Course. Introduction to the intellectual enterprises of computer science and to the art of programming. Students learn how to think algorithmically and solve problems efficiently. Topics include abstraction, algorithms, data structures, encapsulation, resource management, security, software engineering, and web development. Languages include C, Python, SQL, and JavaScript, plus CSS and HTML. Problem sets inspired by real-world domains of biology, cryptography, finance, forensics, and gaming. 1 Credit. Session A: May 27 – June 28. Tuition: $5070. QR

* **CPSC 035b / MUSI 035b, Twenty-First Century Electronic and Computer Music Techniques** Scott Petersen
Exploration of twenty-first century electronic and computer music through the diverse subjects and issues at the intersection of technology and new music. How computers have changed and challenged the analysis, composition, production, and appreciation of music over the last fifty years. Knowledge of basic music theory and the ability to read Western musical notation is assumed. Enrollment limited to first-year students. QR

**CPSC 100a / CPSC S100a, Introduction to Computing and Programming** Ozan Erat
Introduction to the intellectual enterprises of computer science and to the art of programming. Students learn how to think algorithmically and solve problems efficiently. Topics include abstraction, algorithms, data structures, encapsulation, resource management, security, software engineering, and web development. Languages include C, Python, SQL, and JavaScript, plus CSS and HTML. Problem sets inspired by real-world domains of biology, cryptography, finance, forensics, and gaming. See CS50's website, https://cs50.yale.edu, for additional information. No previous programming experience required. Open to students of all levels and majors. QR 0 Course cr

**CPSC 110a or b, Python Programming for Humanities and Social Sciences** Staff
Introduction to computer science and Python programming with domain-specific applications. Students learn how to think algorithmically and solve problems efficiently. Topics include abstraction, algorithms, data structures, web development, and statistical tools. Students learn to apply computing techniques in the fields of social sciences & humanities by analyzing data. No previous programming experience is required. This course is intended for students of social sciences & humanities majors. QR 0 Course cr

**CPSC 112b, Introduction to Programming** Timothy Barron
Development on the computer of programming skills, problem-solving methods, and selected applications. No previous experience with computers necessary. QR 0 Course cr

**CPSC 123a or b / PLSC 351a or b / S&DS 123a or b / S&DS 523a or b, YData: An Introduction to Data Science** Ethan Meyers
Computational, programming, and statistical skills are no longer optional in our increasingly data-driven world; these skills are essential for opening doors to manifold research and career opportunities. This course aims to dramatically enhance knowledge and capabilities in fundamental ideas and skills in data science, especially computational

and programming skills along with inferential thinking. YData is an introduction to Data Science that emphasizes the development of these skills while providing opportunities for hands-on experience and practice. YData is accessible to students with little or no background in computing, programming, or statistics, but is also engaging for more technically oriented students through extensive use of examples and hands-on data analysis. Python 3, a popular and widely used computing language, is the language used in this course. The computing materials will be hosted on a special purpose web server. QR

* **CPSC 150a, Computer Science and the Modern Intellectual Agenda** David Gelernter
Introduction to the basic ideas of computer science (computability, algorithm, virtual machine, symbol processing system), and of several ongoing relationships between computer science and other fields, particularly philosophy of mind. No previous experience with computers necessary. Enrollment limited to 25. WR, HU

**CPSC 170b, AI for Future Presidents** Brian Scassellati
AI is becoming an essential tool for not only scientists and engineers, but also for physicians, judges, artists, and presidents. This course is designed for all students, with no prerequisites, and requires no programming. We look at topics that range from job loss due to automation, how machine learning systems are impacting healthcare, the impact of language models on education, and many other topics that are at the front of the headlines today. Will ChatGPT make essays obsolete? Will robots take my job? How smart will machines become? Students learn some of the basic limits of this technology, understand how to critically analyze public claims made about AI, and understand the societal impact that AI is having.

**CPSC 175b, C Programming Language and Linux** Jay Lim
We discuss the basics of the software development toolchain using the C programming language in the Linux operating system environment. Topics include an overview of C programming language including pointers, malloc, free, function pointers, recursion, and C macros. We further discuss tools useful for developing complex programs including git, compilers, Linux environment, gdb, and valgrind. Finally, we apply the language and tools to multiple fields of computer science. Familiarity with basic programming. CPSC100 or CPSC112 recommended. QR

**CPSC 183a, Law, Technology, and Culture** Brad Rosen
An exploration of the myriad ways in which law and technology intersect, with a special focus on the role of cyberspace. Topics include digital copyright, free speech, privacy and anonymity, information security, innovation, online communities, the impact of technology on society, and emerging trends. No previous experience with computers or law necessary. SO

* **CPSC 185b, Control, Privacy, and Technology** Brad Rosen
The evolution of various legal doctrines with and around technological development. Topics include criminal law, privacy, search and seizure, digital rights, and the implications of technologically permitted methods of control on the law. Special attention to case law and policy. After CPSC 183. WR, SO

**CPSC 200a, Introduction to Information Systems** Stephen Slade
The real-world artifacts and implementations that comprise the vital computational organisms that populate our world. Hardware and software and the related issues

of security, privacy, regulation, and software engineering. Examples stress practical applications of technology, as well as limitations and societal issues. After CPSC 100 or 112 or equivalent. QR

**CPSC 201a or b, Introduction to Computer Science** Stephen Slade
Introduction to the concepts, techniques, and applications of computer science. Topics include computer systems (the design of computers and their languages); theoretical foundations of computing (computability, complexity, algorithm design); and artificial intelligence (the organization of knowledge and its representation for efficient search). Examples stress the importance of different problem-solving methods. After CPSC 112 or equivalent. QR

**CPSC 202a or b, Mathematical Tools for Computer Science** Staff
Introduction to formal methods for reasoning and to mathematical techniques basic to computer science. Topics include propositional logic, discrete mathematics, and linear algebra. Emphasis on applications to computer science: recurrences, sorting, graph traversal, Gaussian elimination. QR

**CPSC 223a or b, Data Structures and Programming Techniques** Staff
Topics include programming in C; data structures (arrays, stacks, queues, lists, trees, heaps, graphs); sorting and searching; storage allocation and management; data abstraction; programming style; testing and debugging; writing efficient programs. After CPSC 200, 201 or passing an exam based on CPSC 201 content. QR

**CPSC 310b, Technology, Power, and Security: Political Challenges of the Computer Age** Joan Feigenbaum
Twenty-first century societies are faced with both threats and opportunities that combine sophisticated computation with politics and international relations in critical ways. Examples include cyber warfare; cyber espionage; cyber crime; the role of social media in democratic self-governance, authoritarian control, and election "hacking"; cryptocurrencies; and mass surveillance. This course examines the political challenges wrought by massive increases in the power of computational and communication technologies and the potential for citizens and governments to harness those technologies to solve problems. Students may not earn credit for both CPSC 210 and CPSC 310. Prerequisite: CPSC 223 or the equivalent. QR, SO 0 Course cr

**CPSC 323a or b, Introduction to Systems Programming and Computer Organization** Staff
Machine architecture and computer organization, systems programming in a high-level language, issues in operating systems, software engineering, prototyping in scripting languages. After CPSC 223. QR

**CPSC 327a or b, Object-Oriented Programming** Timothy Barron
Object-oriented programming as a means to designing and writing efficient, reliable, modular, and reusable code. Covers core concepts and features of object-oriented languages (classes, inheritance, composition, encapsulation, polymorphism, and exceptions) as well as the use of object-oriented design patterns (iterator, decorator, strategy, adapter, observer, etc.). This course was previously number CPSC 427. After CPSC 223. QR

**CPSC 334a, Creative Embedded Systems** Scott Petersen
Ubiquitous computing is creating new canvases and opportunities for creative ideas. This class explores the use of microprocessors, distributed sensor networks, IoT, and

intermedia systems for the purposes of creative expression. The course is delivered in a mixed lecture and lab format that introduces the fundamental concepts and theory behind embedded systems as well as issues particular to their creative employment. The key objective of the course is for students to conceive of and implement creative uses of computation. To this end, skills to be obtained during the course are as follows: (1) appreciate the current efforts and motivation to push the limitations of computation for creative expression, both in new application and new foundational research; (2) weigh factors such as cost, power, processing, memory, I/O capabilities, and networking capabilities when choosing a set of embedded devices and sensors; (3) contextualize unfamiliar hardware and languages through examples, documentation, and familiar design pattern; and (4) manage communication between multiple languages, devices, and protocols. Additionally, at the end of the course students will have a portfolio of their work in the form of writing, code, video, audio, and physical artifacts. Prerequisite: CPSC 223 or equivalent or by permission of instructor. QR RP

**CPSC 338b / EENG 348b, Digital Systems** Staff
Development of engineering skills through the design and analysis of digital logic components and circuits. Introduction to gate-level circuit design, beginning with single gates and building up to complex systems. Hands-on experience with circuit design using computer-aided design tools and microcontroller programming. Recommended preparation: EENG 201. QR

*** CPSC 362b / AMTH 362b / EENG 435b, Decisions and Computations across Networks** A Stephen Morse
For a long time there has been interest in distributed computation and decision making problems of all types. Among these are consensus and flocking problems, the multi-agent rendezvous problem, distributed averaging, gossiping, localization of sensors in a multi-sensor network, distributed algorithms for solving linear equations, distributed management of multi-agent formations, opinion dynamics, and distributed state estimation. The aim of this course is to explain what these problems are and to discuss their solutions. Related concepts from spectral graph theory, rigid graph theory, non-homogeneous Markov chain theory, stability theory, and linear system theory are covered. Although most of the mathematics need is covered in the lectures, students taking this course should have a working understanding of basic linear algebra. The course is open to all students. Prerequisite: Linear algebra or instructor permission. SC

**CPSC 364a, Introduction to Blockchains, Cryptocurrencies, Smart Contracts, and Decentralized Applications** Fan Zhang
This course offers an introduction to blockchain technology and its practical applications. The objective is to provide students with a comprehensive overview of the fundamental concepts and hands-on experience in building on actual blockchains. The course covers the technological foundation of the blockchain stack (consensus layer, ordering layer, execution layer, etc.), the design of representative applications (cryptocurrencies, smart contracts, Decentralized Finance, etc.), and the principles for writing secure smart contracts, and ends with an overview of the latest research directions. To provide a hands-on building experience, the course hosts a Catch-the-Flag (CTF) competition where students are asked to hack buggy smart contracts within a controlled environment. The course assumes a background in various fundamental

areas of CS, including discrete math, probability, algorithms, and data structures. Required: CPSC 202 and 223 (or equivalent). QR

**CPSC 365a or b / ECON 365a or b, Algorithms** Staff
Paradigms for algorithmic problem solving: greedy algorithms, divide and conquer, dynamic programming, and network flow. NP completeness and approximation algorithms for NP-complete problems. Algorithms for problems from economics, scheduling, network design and navigation, geometry, biology, and optimization. Provides algorithmic background essential to further study of computer science. Only one of CPSC 365 or CPSC 366 may be taken for credit. Prerequisites: CPSC 202 or MATH 244, CPSC 223. QR

**CPSC 370b, Artificial Intelligence** Stephen Slade
How can we enable computers to make rational, intelligent decisions? This course explores fundamental techniques for Artificial Intelligence (AI), covering topics such as search, planning, learning, and reasoning under uncertainty. Through hands-on programming projects, students learn conceptual, algorithmic, and practical considerations for implementing foundational AI algorithms. By the end of this class, students have an understanding of the history and breadth of AI problems and topics, and are prepared to undertake more advanced courses in robotics, computer vision, natural language processing, and machine learning. Prerequisites: CPSC 202 and CPSC 223. Students should also be familiar with basic object-oriented programming concepts in Python.

**CPSC 381b, Introduction to Machine Learning** Alex Wong
This course focuses on fundamental topics in machine learning. We begin with an overview of different components of machine learning and types of learning paradigms. We introduce a linear function, discuss how one can train a linear function on a given dataset, and utilize it to tackle classification and regression problems. We then consider kernel methods to enable us to solve nonlinear problems. Additionally, we introduce the concept of generalization error and overfitting. We discuss the role of regularization and extend linear regression to ridge regression. We also cover optimization, beginning from gradient descent and extending it to stochastic gradient descent and its momentum variant; the concept of alternating optimization; the curse of dimensionality; and topics on dimensionality reduction. We conclude the course with neural networks: how to build them using the topics discussed, how to optimize them, and how to apply them to solve a range of machine learning tasks. Students should have passed courses in data structures and object-oriented programming (e.g. CPSC 223a or equivalent courses), foundational mathematical tools such as discrete math and linear algebra (e.g. CPSC 202 or equivalent courses), calculus (e.g. MATH 112, MATH 115, MATH 120, or equivalent courses), linear algebra (e.g. MATH 225, or equivalent courses), and artificial intelligence (e.g. CPSC 370/570). A background in statistics will be useful, but not required. Experience in programming with Python and familiarity with Google Colab, and numerical and image processing packages (i.e. NumPy, SciPy) will be helpful.

**CPSC 413a, Computer System Security** Timothy Barron
Overview of the principles and practice behind analyzing, designing, and implementing secure computer systems. Covers problems that have continued to plague computer systems for years as well as recent events and research in this rapidly evolving field of computer science. Learn to think from the perspective of an adversary; to understand

systems well enough to see how their flaws could be exploited, and to consequently defend against such exploitation. Offers opportunities for hands-on exploration of attacks and defenses in the contexts of web applications, networks, and system level software. Also discusses ethical considerations and responsibilities associated with security research and practice. After CPSC 323.

**CPSC 416a, Lattices and Post-Quantum Cryptography** Katerina Sotiraki
This course explores the role of lattices in modern cryptography. In the last decades, novel computational problems, whose hardness is related to lattices, have been instrumental in cryptography by offering: (a) a basis for "post-quantum" cryptography, (b) cryptographic constructions based on worst-case hard problems, (c) numerous celebrated cryptographic protocols unattainable from other cryptographic assumptions. This course covers the foundations of lattice-based cryptography from fundamental definitions to advanced cryptographic constructions. More precisely, we introduce the Learning with Error (LWE) and the Short Integer Solutions (SIS) problems and study their unique properties, such as the fact that their average-case hardness is based on the worst-case hardness of lattice problems. Next, we cover lattice constructions of advanced cryptographic primitives, such as fully homomorphic encryption and signature schemes. Finally, we introduce some notions of quantum cryptography and explore the role of lattices in this area. Overall, this course offers insights into the foundations and recent advancements in lattice-based cryptography. Prerequisite: CPSC 467/567 or equivalent and linear algebra

* **CPSC 417a, Advanced Topics in Cryptography: Cryptography and Computation** Charalampos Papamanthou
Traditional cryptography is mostly concerned with studying the foundations of securing communication via, for example, encryption and message authentication codes. This class studies the applications of cryptography in securing *computation*. Topics include, but are not limited to, fundamental results and the most recent progress in oblivious computation and private information retrieval (PIR), zero-knowledge proofs, secure computation, consensus algorithms, searchable encryption, and lattice-based cryptography. The class focuses both on theory and applications. Prerequisite: CPSC 467 or equivalent. This course assumes prior knowledge of fundamental notions in cryptography and mathematical maturity as well as comfort with programming.

**CPSC 420b / EENG 420b, Computer Architecture** Abhishek Bhattacharjee
This course offers a treatment of computer architectures for high-performance and power/energy-efficient computer systems. Topics include the foundations of general-purpose computing, including instruction set architectures, pipelines, superscalar and out-of-order execution, speculation, support for precise exceptions, and simultaneous multi-threading. We also cover domain-specific hardware (e.g., graphics processing units), and ongoing industry efforts to elevate them to the status of first-class computing units. In tandem, we cover topics relevant to both general-purpose and domain-specific computing, including memory hierarchies, address translation and virtual memory, on-chip networks, machine learning techniques for resource management, and coherence techniques. If time permits, we will study the basics of emerging non-classical computing paradigms like neuromorphic computing. Overall, this course offers insights on how the computing industry is combating the waning of traditional technology scaling via acceleration and heterogeneity. Prerequisites:

CPSC 323, 223, and 202. This is a programming-intensive course, so comfort with large programming projects is essential.

* **CPSC 421a, Compilers and Interpreters** Jay Lim
Compiler organization and implementation: lexical analysis, formal syntax specification, parsing techniques, execution environment, storage management, code generation and optimization, procedure linkage and address binding. The effect of language-design decisions on compiler construction. After CPSC 323. QR

**CPSC 426a, Building Distributed Systems** Y. Richard Yang
Ubiquitous services such as Google, Facebook, and Amazon run on the back of massive distributed systems. This course covers the fundamental principles, abstractions, and mechanisms that inform the design of such systems, as well as the practical details of real-world implementations. Technical topics covered include properties such as consistency, availability, durability, isolation, and failure atomicity; as well as protocols such as RPC, consensus, consistent hashing, and distributed transactions. The final project involves implementing a real-world distributed service. After CPSC 323. QR

**CPSC 427a, C++ Programming for Stability, Security, and Speed** Michael Fischer
Computer programming involves both abstraction and practice. Lower-level programming courses focus on learning how to correctly implement algorithms for carrying out a task. This course treats a computer program as an artifact with additional attributes of practical importance including execution efficiency, clarity and readability, redundancy, safety in the face of unexpected or malicious environments, and longevity - the ability to evolve over time as bugs are discovered and requirements change. This course is taught using modern C++. Prerequisite CPSC 223.

**CPSC 429a, Principles of Computer System Design** Lin Zhong
Humans are stupid; computers are limited. Yet a collaboration of humans and computers has led to ever more powerful and complex computer systems. This course examines the limitations of humans and computers in this endeavor and how they shape the design, implementation, and evaluation of computer systems. It surveys the empirical knowledge reported by scholars and practitioners that overcome such limitations. The lectures, reading assignments, and classroom discussions travel through psychology and philosophy and revisit important results from theoretical computer science, with a goal of elucidating the rationales behind the best practices in computer systems research and development. Prerequisite: CPSC 323 or equivalent. Students should have the ability to write significant system programs in at least one systems programming language (e.g., C, C++ and Rust).

**CPSC 431a / MUSI 428a, Computer Music: Algorithmic and Heuristic Composition** Scott Petersen
Study of the theoretical and practical fundamentals of computer-generated music, with a focus on high-level representations of music, algorithmic and heuristic composition, and programming languages for computer music generation. Theoretical concepts are supplemented with pragmatic issues expressed in a high-level programming language. Ability to read music is assumed. After CPSC 202 and 223. QR

**CPSC 437a, Database Systems** Avi Silberschatz
Introduction to database systems. Data modeling. The relational model and the SQL query language. Relational database design, integrity constraints, functional

dependencies, and normal forms. Object-oriented databases. Database data structures: files, B-trees, hash indexes. After CPSC 223. QR

**CPSC 439a, Software Engineering** Timos Antonopoulos
Introduction to fundamental concepts in software engineering and to the development and maintenance of large, robust software systems. The process of collecting requirements and writing specifications; project planning and system design; methods for increasing software reliability, including delta debugging and automatic test-case generation; type systems, static analysis, and model checking. Students build software in teams. After CPSC 323. QR RP

**CPSC 446a, Data and Information Visualization** Holly Rushmeier
Visualization is a powerful tool for understanding data and concepts. This course provides an introduction to the concepts needed to build new visualization systems, rather than to use existing visualization software. Major topics are abstracting visualization tasks, using visual channels, spatial arrangements of data, navigation in visualization systems, using multiple views, and filtering and aggregating data. Case studies to be considered include a wide range of visualization types and applications in humanities, engineering, science, and social science. Prerequiste: CPSC 223.

**CPSC 447a, Introduction to Quantum Computing** Yongshan Ding
This course introduces the fundamental concepts in the theory and practice of quantum computation. Topics include information processing, quantum programming, quantum compilation, quantum algorithms, and error correction. The objective of the course is to engage students in applying fresh thinking to what computers can do – we establish an understanding of how quantum computers store and process data, and discover how they differ from conventional digital computers. We anticipate this course will be of interest to students working in computer science, electrical engineering, physics, or mathematics. Prerequisites: CPSC 201 and CPSC 202. Basic familiarity with discrete probability and linear algebra is recommended. Prior experience in quantum computing is useful but not required. SC

**CPSC 448a / EENG 426a / ENAS 876a, Silicon Compilation** Rajit Manohar
An upper-level course on compiling computations into digital circuits using asynchronous design techniques. Emphasis is placed on the synthesis of circuits that are robust to uncertainties in gate and wire delays by the process of program transformations. Topics include circuits as concurrent programs, delay-insensitive design techniques, synthesis of circuits from programs, timing analysis and performance optimization, pipelining, and case studies of complex asynchronous designs. Prerequisite: EENG 201 and introductory programming, or permission of instructor.

**CPSC 450a, Sustainable Computing** Robert Soule
This course covers topics at the intersection of technology and sustainability. We will be reading primary sources on a range of challenges spanning technical considerations, policy, and ethics. Some of the topics include: definitions of sustainability, global perspectives on sustainable computing, measurements and monitoring, energy grid, green data centers, green networks, green storage, edge computing, green software, regulation, policy and standards, and life-cycle analysis. Students will be evaluated based on reading reviews, class participation, and a semester-long project on a topic of their choice in sustainable computing. CPSC 223

**CPSC 454a, Software Analysis and Verification** Ruzica Piskac
Introduction to concepts, tools, and techniques used in the formal verification of software. State-of-the art tools used for program verification; detailed insights into algorithms and paradigms on which those tools are based, including model checking, abstract interpretation, decision procedures, and SMT solvers. After CPSC 202 and 323 or equivalents. QR RP

**CPSC 455a / ECON 425a, Economics and Computation** Yang Cai
A mathematically rigorous investigation of the interplay of economic theory and computer science, with an emphasis on the relationship of incentive-compatibility and algorithmic efficiency. Our main focus is on algorithmic tools in mechanism design, algorithms and complexity theory for learning and computing Nash and market equilibria, and the price of anarchy. Case studies in Web search auctions, wireless spectrum auctions, matching markets, and network routing, and social networks. Prerequisite: CPSC 365 or permission of the instructor. Familiarity with basic microeconomic theory is helpful but not required. QR

* **CPSC 459a, Building Interactive Machines** Marynel Vazquez
This advanced course brings together methods from machine learning, computer vision, robotics, and human-computer interaction to enable interactive machines to perceive and act in a variety of environments. Part of the course examines approaches for perception with different sensing devices and algorithms; the other part focuses on methods for decision making and applied machine learning for control. Understanding of probability, differential calculus, linear algebra, and planning (in Artificial Intelligence) is expected for this course. Programming assignments require proficiency in Python and high-level familiarity with C++. Prerequisites: CPSC 201, CPSC 202, and CPSC 470 (or 570), or permission of the instructor. QR

* **CPSC 464a, Algorithms and their Societal Implications** Nisheeth Vishnoi
Today's society comprises humans living in an interconnected world that is intertwined with a variety of sensing, communicating, and computing devices. Human-generated data is being recorded at unprecedented rates and scales, and powerful AI and ML algorithms, which are capable of learning from such data, are increasingly controlling various aspects of modern society: from social interactions. These data-driven decision-making algorithms have a tremendous potential to change our lives for the better, but, via the ability to mimic and nudge human behavior, they also have the potential to be discriminatory, reinforce societal prejudices, violate privacy, polarize opinions, and influence democratic processes. Thus, designing effective tools to govern modern society which reinforce its cherished values such as equity, justice, democracy, health, privacy, etc. has become paramount and requires a foundational understanding of how humans, data, and algorithms interact. This course is for students who would like to understand and address some of the key challenges and emerging topics at the aforementioned interplay between computation and society. On the one hand, we study human decision-making processes and view them through the lens of computation and on the other hand we study and address the limitations of artificial decision-making algorithms when deployed in various societal contexts. The focus is on developing solutions through a combination of foundational work such as coming up with the right definitions, modeling, algorithms, and empirical evaluation. The current focus is on bias and privacy, with additional topics including robustness, polarization, and

democratic representation. Solid mathematical and programming background is necessary to enroll in this course. CPSC 365 and S&DS 251 are recommended.

**CPSC 468a, Computational Complexity** Dylan McKay
Introduction to the theory of computational complexity. Basic complexity classes, including polynomial time, nondeterministic polynomial time, probabilistic polynomial time, polynomial space, logarithmic space, and nondeterministic logarithmic space. The roles of reductions, completeness, randomness, and interaction in the formal study of computation. After CPSC 365 or 366, or with permission of instructor. QR

*** CPSC 473a, Intelligent Robotics Laboratory** Brian Scassellati
Students work in small teams to construct novel research projects using one of a variety of robot architectures. Project topics may include human-robot interaction, adaptive intelligent behavior, active perception, humanoid robotics, and socially assistive robotics. Enrollment limited to 20. After CPSC 472. QR

**CPSC 474a, Computational Intelligence for Games** James Glenn
Introduction to techniques used for creating computer players for games, particularly board games. Topics include combinatorial and classical game theory, stochastic search methods, applications of neural networks, and procedural content generation. Prerequisites: CPSC 202 and CPSC 223. QR

**CPSC 475a / BENG 475a / EENG 475a, Computational Vision and Biological Perception** Steven Zucker
An overview of computational vision with a biological emphasis. Suitable as an introduction to biological perception for computer science and engineering students, as well as an introduction to computational vision for mathematics, psychology, and physiology students. Prerequisite: CPSC 112 and MATH 120, or with permission of instructor. QR, SC RP

**CPSC 478a, Computer Graphics** Theodore Kim
Introduction to the basic concepts of two- and three-dimensional computer graphics. Topics include affine and projective transformations, clipping and windowing, visual perception, scene modeling and animation, algorithms for visible surface determination, reflection models, illumination algorithms, and color theory. After CPSC 202 and 223. QR

**CPSC 483a, Deep Learning on Graph-Structured Data** Rex Ying
Graph structure emerges in many important domain applications, including but not limited to computer vision, natural sciences, social networks, languages and knowledge graphs. This course offers an introduction to deep learning algorithms applied to such graph-structured data. The first part of the course is an introduction to representation learning for graphs, and covers common techniques in the field, including distributed node embeddings, graph neural networks, deep graph generative models and non-Euclidean embeddings. The first part also touches upon topics of real-world significance, including auto-ML and explainability for graph learning. The second part of the course covers important applications of graph machine learning. We learn ways to model data as graphs and apply graph learning techniques to problems in domains including online recommender systems, knowledge graphs, biological networks, physical simulations and graph mining. The course covers many deep techniques (graph neural networks, graph deep generative models) catered to graph structures. We will cover basic deep learning tutorials in this course. Prerequisites:

CPSC 201, CPSC 223, and one of CPSC 365 or CPSC 366. Knowledge of graphs as a data structure, and understanding of basic graph algorithms are essential for applying machine learning to graph-structured data. Familiarity with Python and important libraries such as Numpy and Pandas are helpful. CPSC 452 and CPSC 453 are highly recommended prior because they cover the foundations of deep neural networks. Experience in machine Learning courses such as CPSC 481, and Graph Theory courses such as CPSC 462 are welcomed as well. QR

* **CPSC 490a, Senior Project** Sohee Park
Individual research intended to fulfill the senior requirement. Requires a faculty supervisor and the permission of the director of undergraduate studies. The student must submit a written report about the results of the project.