**Curriculum**

**Curriculum version**: 8.0.0 (see [CHANGELOG](https://github.com/ossu/computer-science/blob/master/CHANGELOG.md))

* [Prerequisites](https://github.com/ossu/computer-science#prerequisites)
* [Intro CS](https://github.com/ossu/computer-science#intro-cs)
  + [Introduction to Programming](https://github.com/ossu/computer-science#introduction-to-programming)
  + [Introduction to Computer Science](https://github.com/ossu/computer-science#introduction-to-computer-science)
* [Core CS](https://github.com/ossu/computer-science#core-cs)
  + [Core programming](https://github.com/ossu/computer-science#core-programming)
  + [Core math](https://github.com/ossu/computer-science#core-math)
  + [CS Tools](https://github.com/ossu/computer-science#cs-tools)
  + [Core systems](https://github.com/ossu/computer-science#core-systems)
  + [Core theory](https://github.com/ossu/computer-science#core-theory)
  + [Core applications](https://github.com/ossu/computer-science#core-applications)
  + [Core security](https://github.com/ossu/computer-science#core-security)
* [Advanced CS](https://github.com/ossu/computer-science#advanced-cs)
  + [Advanced programming](https://github.com/ossu/computer-science#advanced-programming)
  + [Advanced systems](https://github.com/ossu/computer-science#advanced-systems)
  + [Advanced theory](https://github.com/ossu/computer-science#advanced-theory)
  + [Advanced math](https://github.com/ossu/computer-science#advanced-math)
* [Final project](https://github.com/ossu/computer-science#final-project)

**Prerequisites**

* [Core CS](https://github.com/ossu/computer-science#core-cs) assumes the student has already taken [high school math](https://github.com/ossu/computer-science/blob/master/FAQ.md#how-can-i-review-the-math-prerequisites), including algebra, geometry, and pre-calculus.
* [Advanced CS](https://github.com/ossu/computer-science#advanced-cs) assumes the student has already taken the entirety of Core CS and is knowledgeable enough now to decide which electives to take.
* Note that [Advanced systems](https://github.com/ossu/computer-science#advanced-systems) assumes the student has taken a basic physics course (e.g. AP Physics in high school).

**Intro CS**

**Introduction to Programming**

If you've never written a for-loop, or don't know what a string is in programming, start here. This course is self-paced, allowing you to adjust the number of hours you spend per week to meet your needs.

**Topics covered**: simple programs simple data structures

| **Courses** | **Duration** | **Effort** | **Prerequisites** | **Discussion** |
| --- | --- | --- | --- | --- |
| [Python for Everybody](https://www.py4e.com/lessons) | 10 weeks | 10 hours/week | none | [chat](https://discord.gg/syA242Z) |

**Introduction to Computer Science**

This course will introduce you to the world of computer science. Students who have been introduced to programming, either from the courses above or through study elsewhere, should take this course for a flavor of the material to come. If you finish the course wanting more, Computer Science is likely for you!

**Topics covered**: computation imperative programming basic data structures and algorithms and more

| **Courses** | **Duration** | **Effort** | **Prerequisites** | **Discussion** |
| --- | --- | --- | --- | --- |
| [Introduction to Computer Science and Programming using Python](https://www.edx.org/course/introduction-to-computer-science-and-programming-7) ([alt](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/)) | 9 weeks | 15 hours/week | [high school algebra](https://www.khanacademy.org/math/algebra-home) | [chat](https://discord.gg/jvchSm9) |

**Core CS**

All coursework under Core CS is **required**, unless otherwise indicated.

**Core programming**

**Topics covered**: functional programming design for testing program requirements common design patterns unit testing object-oriented design static typing dynamic typing ML-family languages (via Standard ML) Lisp-family languages (via Racket) Ruby and more

The How to Code courses are based on the textbook [How to Design Programs](https://htdp.org/2003-09-26/). The First Edition is available for free online and includes problem sets and solutions. Students are encouraged to do these assignments.

| **Courses** | **Duration** | **Effort** | **Prerequisites** | **Discussion** |
| --- | --- | --- | --- | --- |
| [How to Code - Simple Data](https://www.edx.org/course/how-to-code-simple-data) | 7 weeks | 8-10 hours/week | none | [chat](https://discord.gg/RfqAmGJ) |
| [How to Code - Complex Data](https://www.edx.org/course/how-to-code-complex-data) | 6 weeks | 8-10 hours/week | How to Code: Simple Data | [chat](https://discord.gg/kczJzpm) |
| [Programming Languages, Part A](https://www.coursera.org/learn/programming-languages) | 5 weeks | 4-8 hours/week | How to Code ([Hear instructor](https://www.coursera.org/lecture/programming-languages/recommended-background-k1yuh)) | [chat](https://discord.gg/8BkJtXN) |
| [Programming Languages, Part B](https://www.coursera.org/learn/programming-languages-part-b) | 3 weeks | 4-8 hours/week | Programming Languages, Part A | [chat](https://discord.gg/EeA7VR9) |
| [Programming Languages, Part C](https://www.coursera.org/learn/programming-languages-part-c) | 3 weeks | 4-8 hours/week | Programming Languages, Part B | [chat](https://discord.gg/8EZUVbA) |
| [Object-Oriented Design](https://www.coursera.org/learn/object-oriented-design) | 4 weeks | 4 hours/week | [Basic Java](https://www.youtube.com/watch?v=GoXwIVyNvX0) |  |
| [Design Patterns](https://www.coursera.org/learn/design-patterns) | 4 weeks | 4 hours/week | Object-Oriented Design |  |
| [Software Architecture](https://www.coursera.org/learn/software-architecture) | 4 weeks | 2-5 hours/week | Design Patterns |  |

**Core Math**

Discrete math (Math for CS) is a prerequisite and closely related to the study of algorithms and data structures. Calculus both prepares students for discrete math and helps students develop mathematical maturity.

**Topics covered**: discrete mathematics mathematical proofs basic statistics O-notation discrete probability and more

| **Courses** | **Duration** | **Effort** | **Notes** | **Prerequisites** | **Discussion** |
| --- | --- | --- | --- | --- | --- |
| [Calculus 1A: Differentiation](https://openlearninglibrary.mit.edu/courses/course-v1:MITx+18.01.1x+2T2019/about) ([alt](https://ocw.mit.edu/courses/mathematics/18-01sc-single-variable-calculus-fall-2010/index.htm)) | 13 weeks | 6-10 hours/week | The alternate covers this and the following 2 courses | [high school math](https://github.com/ossu/computer-science/blob/master/FAQ.md#how-can-i-review-the-math-prerequisites) | [chat](https://discord.gg/mPCt45F) |
| [Calculus 1B: Integration](https://openlearninglibrary.mit.edu/courses/course-v1:MITx+18.01.2x+3T2019/about) | 13 weeks | 5-10 hours/week | - | Calculus 1A | [chat](https://discord.gg/sddAsZg) |
| [Calculus 1C: Coordinate Systems & Infinite Series](https://openlearninglibrary.mit.edu/courses/course-v1:MITx+18.01.3x+1T2020/about) | 6 weeks | 5-10 hours/week | - | Calculus 1B | [chat](https://discord.gg/FNEcNNq) |
| [Mathematics for Computer Science](https://openlearninglibrary.mit.edu/courses/course-v1:OCW+6.042J+2T2019/about) ([alt](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-spring-2015/index.htm)) | 13 weeks | 5 hours/week | An alternate version with solutions to the problem sets is [here](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2005/assignments/). Students struggling can consider the [Discrete Mathematics Specialization](https://www.coursera.org/specializations/discrete-mathematics) first. It is more interactive but less comprehensive, and costs money to unlock full interactivity. | Calculus 1C | [chat](https://discord.gg/EuTzNbF) |

**CS Tools**

Understanding theory is important, but you will also be expected to create programs. There are a number of tools that are widely used to make that process easier. Learn them now to ease your future work writing programs.

**Topics covered**: terminals and shell scripting vim command line environments version control and more

| **Courses** | **Duration** | **Effort** | **Prerequisites** | **Discussion** |
| --- | --- | --- | --- | --- |
| [The Missing Semester of Your CS Education](https://missing.csail.mit.edu/) | 2 weeks | 12 hours/week | - | [chat](https://discord.gg/5FvKycS) |

**Core systems**

**Topics covered**: procedural programming manual memory management boolean algebra gate logic memory computer architecture assembly machine language virtual machines high-level languages compilers operating systems network protocols and more

| **Courses** | **Duration** | **Effort** | **Additional Text / Assignments** | **Prerequisites** | **Discussion** |
| --- | --- | --- | --- | --- | --- |
| [Build a Modern Computer from First Principles: From Nand to Tetris](https://www.coursera.org/learn/build-a-computer) ([alt](https://www.nand2tetris.org/)) | 6 weeks | 7-13 hours/week | - | C-like programming language | [chat](https://discord.gg/vxB2DRV) |
| [Build a Modern Computer from First Principles: Nand to Tetris Part II](https://www.coursera.org/learn/nand2tetris2) | 6 weeks | 12-18 hours/week | - | one of [these programming languages](https://user-images.githubusercontent.com/2046800/35426340-f6ce6358-026a-11e8-8bbb-4e95ac36b1d7.png), From Nand to Tetris Part I | [chat](https://discord.gg/AsUXcPu) |
| [Operating Systems: Three Easy Pieces](https://pages.cs.wisc.edu/~remzi/Classes/537/Spring2018/) | 10-12 weeks | 6-10 hours/week | - | algorithms, [familiarity with C](https://hackr.io/tutorials/learn-c?sort=upvotes&type_tags%5B%5D=1) is useful | [chat](https://discord.gg/wZNgpep) |
| [Computer Networking: a Top-Down Approach](http://gaia.cs.umass.edu/kurose_ross/online_lectures.htm) | 8 weeks | 4–12 hours/week | [Wireshark Labs](http://gaia.cs.umass.edu/kurose_ross/wireshark.php) | algebra, probability, basic CS | [chat](https://discord.gg/MJ9YXyV) |

**Core theory**

**Topics covered**: divide and conquer sorting and searching randomized algorithms graph search shortest paths data structures greedy algorithms minimum spanning trees dynamic programming NP-completeness and more

| **Courses** | **Duration** | **Effort** | **Prerequisites** | **Discussion** |
| --- | --- | --- | --- | --- |
| [Divide and Conquer, Sorting and Searching, and Randomized Algorithms](https://www.coursera.org/learn/algorithms-divide-conquer) | 4 weeks | 4-8 hours/week | any programming language, Mathematics for Computer Science | [chat](https://discord.gg/mKRS7tY) |
| [Graph Search, Shortest Paths, and Data Structures](https://www.coursera.org/learn/algorithms-graphs-data-structures) | 4 weeks | 4-8 hours/week | Divide and Conquer, Sorting and Searching, and Randomized Algorithms | [chat](https://discord.gg/Qstqe4t) |
| [Greedy Algorithms, Minimum Spanning Trees, and Dynamic Programming](https://www.coursera.org/learn/algorithms-greedy) | 4 weeks | 4-8 hours/week | Graph Search, Shortest Paths, and Data Structures | [chat](https://discord.gg/dWVvjuz) |
| [Shortest Paths Revisited, NP-Complete Problems and What To Do About Them](https://www.coursera.org/learn/algorithms-npcomplete) | 4 weeks | 4-8 hours/week | Greedy Algorithms, Minimum Spanning Trees, and Dynamic Programming | [chat](https://discord.gg/dYuY78u) |

**Core Security**

**Topics covered** Confidentiality, Integrity, Availability Secure Design Defensive Programming Threats and Attacks Network Security Cryptography and more

Note: ***These courses are provisionally recommended***. There is an open [Request For Comment](https://github.com/ossu/computer-science/issues/639) on security course selection. Contributors are encouraged to compare the various courses in the RFC and offer feedback.

| **Courses** | **Duration** | **Effort** | **Prerequisites** | **Discussion** |
| --- | --- | --- | --- | --- |
| [Information Security: Context and Introduction](https://www.coursera.org/learn/information-security-data) | 5 weeks | 3 hours/week | - | [chat](https://discord.gg/8h6Rz8g) |
| [Principles of Secure Coding](https://www.coursera.org/learn/secure-coding-principles) | 4 weeks | 4 hours/week | - | [chat](https://discord.gg/5gMdeSK) |
| [Identifying Security Vulnerabilities](https://www.coursera.org/learn/identifying-security-vulnerabilities) | 4 weeks | 4 hours/week | - | [chat](https://discord.gg/V78MjUS) |

Choose **one** of the following:

| **Courses** | **Duration** | **Effort** | **Prerequisites** | **Discussion** |
| --- | --- | --- | --- | --- |
| [Identifying Security Vulnerabilities in C/C++Programming](https://www.coursera.org/learn/identifying-security-vulnerabilities-c-programming) | 4 weeks | 5 hours/week | - | [chat](https://discord.gg/Vbxce7A) |
| [Exploiting and Securing Vulnerabilities in Java Applications](https://www.coursera.org/learn/exploiting-securing-vulnerabilities-java-applications) | 4 weeks | 5 hours/week | - | [chat](https://discord.gg/QxC22rR) |

**Core applications**

**Topics covered**: Agile methodology REST software specifications refactoring relational databases transaction processing data modeling neural networks supervised learning unsupervised learning OpenGL raytracing and more

| **Courses** | **Duration** | **Effort** | **Prerequisites** | **Discussion** |
| --- | --- | --- | --- | --- |
| [Databases: Modeling and Theory](https://www.edx.org/course/modeling-and-theory) | 2 weeks | 10 hours/week | core programming | [chat](https://discord.gg/pMFqNf4) |
| [Databases: Relational Databases and SQL](https://www.edx.org/course/databases-5-sql) | 2 weeks | 10 hours/week | core programming | [chat](https://discord.gg/P8SPPyF) |
| [Databases: Semistructured Data](https://www.edx.org/course/semistructured-data) | 2 weeks | 10 hours/week | core programming | [chat](https://discord.gg/duCJ3GN) |
| [Machine Learning](https://www.coursera.org/learn/machine-learning) | 11 weeks | 4-6 hours/week | linear algebra | [chat](https://discord.gg/NcXHDjy) |
| [Computer Graphics](https://www.edx.org/course/computer-graphics-2) | 6 weeks | 12 hours/week | C++ or Java, linear algebra | [chat](https://discord.gg/68WqMNV) |
| [Software Engineering: Introduction](https://www.edx.org/course/software-engineering-introduction) | 6 weeks | 8-10 hours/week | Core Programming, and a [sizable project](https://github.com/ossu/computer-science/blob/master/FAQ.md#why-require-experience-with-a-sizable-project-before-the-Software-Engineering-courses) | [chat](https://discord.gg/5Qtcwtz) |

**Core Ethics**

**Topics covered**: Social Context Analytical Tools Professional Ethics Intellectual Property Privacy and Civil Liberties and more

| **Courses** | **Duration** | **Effort** | **Prerequisites** | **Discussion** |
| --- | --- | --- | --- | --- |
| [Ethics, Technology and Engineering](https://www.coursera.org/learn/ethics-technology-engineering) | 9 weeks | 2 hours/week | none | [chat](https://discord.gg/6ttjPmzZbe) |
| [Intellectual Property Law in Digital Age](https://www.coursera.org/learn/intellectual-property-law-in-digital-age) | 4 weeks | 2 hours/week | none | [chat](https://discord.gg/YbuERswpAK) |
| [Data Privacy Fundamentals](https://www.coursera.org/learn/northeastern-data-privacy) | 3 weeks | 3 hours/week | none | [chat](https://discord.gg/64J34ajNBd) |

**Advanced CS**

After completing **every required course** in Core CS, students should choose a subset of courses from Advanced CS based on interest. Not every course from a subcategory needs to be taken. But students should take *every* course that is relevant to the field they intend to go into.

**Advanced programming**

**Topics covered**: debugging theory and practice goal-oriented programming parallel computing object-oriented analysis and design UML large-scale software architecture and design and more

| **Courses** | **Duration** | **Effort** | **Prerequisites** |
| --- | --- | --- | --- |
| [Parallel Programming](https://www.coursera.org/learn/scala-parallel-programming) | 4 weeks | 6-8 hours/week | Scala programming |
| [Compilers](https://www.edx.org/course/compilers) | 9 weeks | 6-8 hours/week | none |
| [Introduction to Haskell](https://www.seas.upenn.edu/~cis194/fall16/) | 14 weeks | - | - |
| [Learn Prolog Now!](https://www.let.rug.nl/bos/lpn/lpnpage.php?pageid=online) ([alt](https://github.com/ossu/computer-science/files/6085884/lpn.pdf))\* | 12 weeks | - | - |
| [Software Debugging](https://www.udacity.com/course/software-debugging--cs259) | 8 weeks | 6 hours/week | Python, object-oriented programming |
| [Software Testing](https://www.udacity.com/course/software-testing--cs258) | 4 weeks | 6 hours/week | Python, programming experience |

(\*) book by Blackburn, Bos, Striegnitz (compiled from [source](https://github.com/LearnPrologNow/lpn), redistributed under [CC license](https://creativecommons.org/licenses/by-sa/4.0/))

**Advanced systems**

**Topics covered**: digital signaling combinational logic CMOS technologies sequential logic finite state machines processor instruction sets caches pipelining virtualization parallel processing virtual memory synchronization primitives system call interface and more

| **Courses** | **Duration** | **Effort** | **Prerequisites** |
| --- | --- | --- | --- |
| [Computation Structures 1: Digital Circuits](https://learning.edx.org/course/course-v1:MITx+6.004.1x_3+3T2016) | 10 weeks | 6 hours/week | [Nand2Tetris II](https://www.coursera.org/learn/nand2tetris2) |
| [Computation Structures 2: Computer Architecture](https://learning.edx.org/course/course-v1:MITx+6.004.2x+3T2015) | 10 weeks | 6 hours/week | Computation Structures 1 |
| [Computation Structures 3: Computer Organization](https://learning.edx.org/course/course-v1:MITx+6.004.3x_2+1T2017) | 10 weeks | 6 hours/week | Computation Structures 2 |

**Advanced theory**

**Topics covered**: formal languages Turing machines computability event-driven concurrency automata distributed shared memory consensus algorithms state machine replication computational geometry theory propositional logic relational logic Herbrand logic game trees and more

| **Courses** | **Duration** | **Effort** | **Prerequisites** |
| --- | --- | --- | --- |
| [Theory of Computation](http://aduni.org/courses/theory/index.php?view=cw) ([Lectures](https://www.youtube.com/playlist?list=PLTke5lHMAdSNmi57H0DOTzClHPK6UwSTN)) | 8 weeks | 10 hours/week | discrete mathematics, logic, algorithms |
| [Computational Geometry](https://www.edx.org/course/computational-geometry) | 16 weeks | 8 hours/week | algorithms, C++ |
| [Game Theory](https://www.coursera.org/learn/game-theory-1) | 8 weeks | 3 hours/week | mathematical thinking, probability, calculus |

**Advanced math**

| **Courses** | **Duration** | **Effort** | **Prerequisites** | **Discussion** |
| --- | --- | --- | --- | --- |
| [Essence of Linear Algebra](https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab) | - | - | [high school math](https://github.com/ossu/computer-science/blob/master/FAQ.md#how-can-i-review-the-math-prerequisites) | [chat](https://discord.gg/m6wHbP6) |
| [Linear Algebra](https://ocw.mit.edu/courses/mathematics/18-06sc-linear-algebra-fall-2011/) | 14 weeks | 12 hours/week | corequisite: Essence of Linear Algebra | [chat](https://discord.gg/k7nSWJH) |
| [Introduction to Numerical Analysis](https://www.coursera.org/learn/intro-to-numerical-analysis)([alt](https://ocw.mit.edu/courses/mathematics/18-335j-introduction-to-numerical-methods-spring-2019/index.htm)) | 7 weeks | 3-4 hours/week | [Mathematics for Computer Science](https://openlearninglibrary.mit.edu/courses/course-v1:OCW+6.042J+2T2019/about), Optional: [Linear Algebra](https://ocw.mit.edu/courses/mathematics/18-06sc-linear-algebra-fall-2011/) | [chat](https://discord.gg/FNEcNNq) |
| [Introduction to Logic](https://www.coursera.org/learn/logic-introduction) | 10 weeks | 4-8 hours/week | [set theory](https://www.youtube.com/playlist?list=PL5KkMZvBpo5AH_5GpxMiryJT6Dkj32H6N) | [chat](https://discord.gg/MbM2Gg5) |
| [Probability](https://projects.iq.harvard.edu/stat110/home) | 24 weeks | 12 hours/week | [Differentiation and Integration](https://www.edx.org/course/calculus-1b-integration) | [chat](https://discord.gg/UVjs9BU) |

**Final project**

OSS University is project-focused. The assignments and exams for each course are to prepare you to use your knowledge to solve real-world problems.

After you've gotten through all of Core CS and the parts of Advanced CS relevant to you, you should think about a problem that you can solve using the knowledge you've acquired. Not only does real project work look great on a resume, but the project will also validate and consolidate your knowledge. You can create something entirely new, or you can find an existing project that needs help via websites like [CodeTriage](https://www.codetriage.com/) or [First Timers Only](https://www.firsttimersonly.com/).

Students who would like more guidance in creating a project may choose to use a series of project oriented courses. Here is a sample of options (many more are available, at this point you should be capable of identifying a series that is interesting and relevant to you):

| **Courses** | **Duration** | **Effort** | **Prerequisites** |
| --- | --- | --- | --- |
| [Fullstack Open](https://fullstackopen.com/en/) | 12 weeks | 6 hours/week | programming |
| [Modern Robotics (Specialization)](https://www.coursera.org/specializations/modernrobotics) | 26 weeks | 2-5 hours/week | freshman-level physics, linear algebra, calculus, [linear ordinary differential equations](https://www.khanacademy.org/math/differential-equations) |
| [Data Mining (Specialization)](https://www.coursera.org/specializations/data-mining) | 30 weeks | 2-5 hours/week | machine learning |
| [Big Data (Specialization)](https://www.coursera.org/specializations/big-data) | 30 weeks | 3-5 hours/week | none |
| [Internet of Things (Specialization)](https://www.coursera.org/specializations/internet-of-things) | 30 weeks | 1-5 hours/week | strong programming |
| [Cloud Computing (Specialization)](https://www.coursera.org/specializations/cloud-computing) | 30 weeks | 2-6 hours/week | C++ programming |
| [Data Science (Specialization)](https://www.coursera.org/specializations/jhu-data-science) | 43 weeks | 1-6 hours/week | none |
| [Functional Programming in Scala (Specialization)](https://www.coursera.org/specializations/scala) | 29 weeks | 4-5 hours/week | One year programming experience |
| [Game Design and Development with Unity 2020 (Specialization)](https://www.coursera.org/specializations/game-design-and-development) | 6 months | 5 hours/week | programming, interactive design |

**Evaluation**

Upon completing your final project:

* Submit your project's information to [PROJECTS](https://github.com/ossu/computer-science/blob/master/PROJECTS.md) via a pull request.
* Put the OSSU-CS badge in the README of your repository!
  + Markdown: [![Open Source Society University - Computer Science](https://img.shields.io/badge/OSSU-computer--science-blue.svg)](https://github.com/ossu/computer-science)
  + HTML: <a href="https://github.com/ossu/computer-science"><img alt="Open Source Society University - Computer Science" src="https://img.shields.io/badge/OSSU-computer--science-blue.svg"></a>
* Use our [community](https://github.com/ossu/computer-science#community) channels to announce it to your fellow students.

Solicit feedback from your OSSU peers. You will not be "graded" in the traditional sense — everyone has their own measurements for what they consider a success. The purpose of the evaluation is to act as your first announcement to the world that you are a computer scientist and to get experience listening to feedback — both positive and negative.

The final project evaluation has a second purpose: to evaluate whether OSSU, through its community and curriculum, is successful in its mission to guide independent learners in obtaining a world-class computer science education.

**Cooperative work**

You can create this project alone or with other students! **We love cooperative work**! Use our [channels](https://github.com/ossu/computer-science#community) to communicate with other fellows to combine and create new projects!

**Which programming languages should I use?**

My friend, here is the best part of liberty! You can use **any** language that you want to complete the final project.

The important thing is to **internalize** the core concepts and to be able to use them with whatever tool (programming language) that you wish.