STUDY PLAN · 2026

Three-Year Computer Science Curriculum

A BSc-style computer science programme: from the first line of Python to algorithms, software engineering and machine learning, using university courses on Coursera and free textbooks.

Year 1	Year 2	Year 3
Foundations	Programming and algorithms	Software engineering and broadening
384 h	**379 h**	**438 h**

1,201 h of core coursework

6 semesters

68 courses

11 universities

14 reference textbooks

Data collected from coursera.org in November 2026 · Textbook availability checked on the same dates

Contents

1. About the curriculum
2. Overview and workload
3. Year 1 · Foundations (S1, S2)
4. Year 2 · Programming and algorithms (S3, S4)
5. Year 3 · Software engineering and broadening (S5, S6)
6. Options
7. Bibliography
8. Method and caveats

1. About the curriculum

This curriculum reproduces, for self-study, the structure of a computer science bachelor's degree: three years, five areas and a steady rise in abstraction. It is designed for a motivated beginner with end-of-high-school mathematics who can devote about ten hours a week to study.

THE FIVE AREAS

- **Programming.** Python, C, Java, Go, JavaScript.
- **Mathematics.** proofs, combinatorics, graphs, number theory, linear algebra, statistics.
- **Algorithms.** data structures, algorithm design and analysis, complexity.
- **Software engineering.** design, architecture, agile lifecycle, testing, security.
- **Broadening.** concurrency, the web and machine learning.

THE PRINCIPLES BEHIND THE ORDER

- **Programming first.** Python (S1) is the tool for everything else, including discrete mathematics, which relies on small programs.
- **Mathematics before algorithms.** Proofs, combinatorics and graphs (S2) come before Stanford's algorithm analysis (S4).
- **From low level to abstraction.** C (S3) shows memory and the machine; Java (S3, S4) brings objects and data structures; software engineering (S5) organises it all.
- **The craft before broadening.** Design, lifecycle and testing (S5) come before security, concurrency, the web and machine learning (S6).
- **One project per semester.** Each semester ends with a capstone or a project that brings its skills together.

LANGUAGES AND TOOLS

Python (S1, S2, S6), C (S3), Java (S3 to S5), Go and JavaScript (S6). Rice uses the browser-based CodeSkulptor; the other courses need a C compiler, a Java JDK and Go installed.

2. Overview and workload

YEAR	SEM.	CONTENT	HOURS
Year 1 Foundations **384 h**	**S1**	**Programming and algorithmic thinking** Python for Everybody Specialization (79 h) Fundamentals of Computing Specialization (98 h)	177 h
	S2	**Mathematics for computing** Introduction to Discrete Mathematics for Computer Science (116 h) Linear Algebra from Elementary to Advanced (34 h) Statistics with Python (57 h)	207 h
Year 2 Programming and algorithms **379 h**	**S3**	**C and Java** Introductory C Programming Specialization (84 h) Java Programming and Software Engineering Fundamentals (84 h)	168 h
	S4	**Data structures and algorithms** Object Oriented Java Programming: Data Structures and Beyond (154 h) Algorithms Specialization (57 h)	211 h
Year 3 Software engineering and broadening **438 h**	**S5**	**Software engineering** Software Design and Architecture Specialization (52 h) Software Development Lifecycle Specialization (63 h) Software Testing and Automation Specialization (95 h)	210 h
	S6	**Security, concurrency, the web and AI** Secure Software Design Specialization (60 h) Programming with Google Go (34 h) Web Design for Everybody: Basics of Web Development & Coding Specialization (78 h) Machine Learning: Theory and Hands-on Practice with Python (56 h)	228 h
Core curriculum total			**1,201 h**
Options (optional)			+ 788 h

COURSE HOURS PER SEMESTER



SUGGESTED PACE

Over 40 study weeks a year, the curriculum takes about **10 h per week in year 1**, **9 h in year 2** and **11 h in year 3**. Lengths are Coursera's estimates; allow an extra 20 to 30% for reading and projects.

YEAR 1 · 384 HOURS

Foundations

Program in Python, think algorithmically, and acquire the mathematics of computing

S1

YEAR 1 · 177 HOURS · ≈ 9 H PER WEEK OVER 20 WEEKS

Programming and algorithmic thinking

> Learn to program in Python, from the first script to interactive programs, then tackle recursion, search and graphs.

TARGET SKILLS

- Python variables, loops, functions and data structures
- Files, the web, APIs and SQLite databases
- Event-driven and object-oriented programming
- Testing, probability and combinatorics applied to programs
- Introductory search, recursion, graphs and complexity

END-OF-SEMESTER PROJECT

The Fundamentals of Computing final exam, then a Python strategy game with a minimax AI and a test suite.

THINGS TO WATCH

- Take Fundamentals of Computing alongside the second half of Python for Everybody.
- Rice uses CodeSkulptor, a browser-based Python environment.

PROGRAMME

1

SPECIALIZATION · 5 COURSES · 79 H

Python for Everybody Specialization

University of Michigan | Beginner | Python, SQL, SQLite

Stated prerequisites: No prerequisites; designed as an on-ramp to programming

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Programming for Everybody (Getting Started with Python)	Beginner	17 h	4.8 (233,775)
2	Python Data Structures	Beginner	20 h	4.9 (97,365)
3	Using Python to Access Web Data	Beginner	19 h	4.8 (44,591)
4	Using Databases with Python	Beginner	14 h	4.8 (21,659)
5	Capstone: Retrieving, Processing, and Visualizing Data with Python	Beginner	9 h	4.7 (14,483)

2

SPECIALIZATION · 7 COURSES · 98 H

Fundamentals of Computing Specialization

Rice University | Beginner | Python

Stated prerequisites: High school mathematics; no programming required

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	An Introduction to Interactive Programming in Python (Part 1)	—	19 h	4.8 (3,322)
2	An Introduction to Interactive Programming in Python (Part 2)	Beginner	16 h	4.9 (1,214)
3	Principles of Computing (Part 1)	Intermediate	19 h	4.7 (584)
4	Principles of Computing (Part 2)	Intermediate	16 h	4.8 (365)
5	Algorithmic Thinking (Part 1)	Intermediate	14 h	4.6 (385)
6	Algorithmic Thinking (Part 2)	Intermediate	13 h	4.7 (220)
7	The Fundamentals of Computing Capstone Exam	Intermediate	1 h	4.8 (116)

FURTHER READING

Python for Everybody

Charles R. Severance

The Michigan specialization's official textbook.

Free, CC BY 4.0 · www.py4e.com/book

Think Python, 3rd edition

Allen B. Downey

A more rigorous take on functions, recursion and classes.

Free, CC BY-NC-SA 4.0 · allendowney.github.io/ThinkPython

S2

YEAR 1 · 207 HOURS · ≈ 10 H PER WEEK OVER 20 WEEKS

Mathematics for computing

> Learn to prove, count and reason about graphs, master basic linear algebra and the statistics modern computing needs.

TARGET SKILLS

- Proofs: induction, contradiction, invariants
- Combinatorics, discrete probability, graph theory
- Modular arithmetic and RSA cryptography
- Linear systems, matrices, eigenvalues
- Descriptive and inferential statistics in Python

END-OF-SEMESTER PROJECT

UCSD's Delivery Problem project (travelling salesman), then a small RSA implementation with tests.

THINGS TO WATCH

- Discrete Mathematics needs basic Python: S1 provides it.
- Linear Algebra requires no calculus.

PROGRAMME

1

SPECIALIZATION · 5 COURSES · 116 H

Introduction to Discrete Mathematics for Computer Science

University of California San Diego | Beginner | Python

Stated prerequisites: Basic math (fractions, squares), basic Python (functions, loops, recursion)

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Mathematical Thinking in Computer Science	Beginner	42 h	4.4 (2,289)
2	Combinatorics and Probability	Beginner	23 h	4.6 (874)
3	Introduction to Graph Theory	Beginner	21 h	4.5 (1,073)
4	Number Theory and Cryptography	Beginner	17 h	4.5 (610)
5	Delivery Problem	Beginner	13 h	4.7 (376)

2

SPECIALIZATION · 3 COURSES · 34 H

Linear Algebra from Elementary to Advanced

Johns Hopkins University | Beginner

Stated prerequisites: Basic algebra; no calculus needed

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Linear Algebra: Linear Systems and Matrix Equations	Beginner	10 h	4.7 (183)
2	Linear Algebra: Matrix Algebra, Determinants, & Eigenvectors	—	15 h	4.8 (79)
3	Linear Algebra: Orthogonality and Diagonalization	Intermediate	9 h	4.8 (56)

3

SPECIALIZATION · 3 COURSES · 57 H

Statistics with Python

University of Michigan | Beginner | Python, Jupyter, Matplotlib, NumPy, pandas, Seaborn, Statsmodels

Stated prerequisites: High school algebra; basic Python recommended

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Understanding and Visualizing Data with Python	Beginner	20 h	4.7 (2,741)
2	Inferential Statistical Analysis with Python	Intermediate	22 h	4.6 (938)
3	Fitting Statistical Models to Data with Python	Intermediate	15 h	4.4 (718)

FURTHER READING

Mathematics for Computer Science

E. Lehman, F. T. Leighton, A. R. Meyer (MIT)

The reference for proofs, combinatorics and graphs.

Free, CC BY-SA · ocw.mit.edu/courses/6-1200j-mathematics-for-computer-science-spring-2024/pages/readings

Linear Algebra

Jim Hefferon

A complete textbook with worked exercises.

Free · hefferon.net/linearalgebra

OpenIntro Statistics

D. Diez, M. Çetinkaya-Rundel, C. Barr

For Statistics with Python.

Free PDF · www.openintro.org/book/os

YEAR 2 · 379 HOURS

Programming and algorithms

Program close to the machine in C, program with objects in Java, then design efficient data structures and algorithms

S3

YEAR 2 · 168 HOURS · ≈ 8 H PER WEEK OVER 20 WEEKS

C and Java

> Understand memory, pointers and compilation with C, then object-oriented programming with Java.

TARGET SKILLS

- Seven-step algorithm design, writing, testing and debugging in C
- Pointers, arrays, recursion and memory management
- Interacting with the system: files, arguments, dynamic memory
- Java, classes, lists and design principles
- Simple web pages in HTML, CSS and JavaScript

END-OF-SEMESTER PROJECT

The recommender system from Duke's final course, then a mini Unix shell in C with redirections and pipes.

THINGS TO WATCH

- Start with C: memory management then sheds light on how Java works.
- Duke's first course covers the web (HTML, CSS, JavaScript) before Java.

PROGRAMME

1

SPECIALIZATION · 4 COURSES · 84 H

Introductory C Programming Specialization

Duke University | Beginner | C

Stated prerequisites: No prior programming experience required

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Programming Fundamentals	Beginner	19 h	4.7 (7,054)
2	Writing, Running, and Fixing Code in C	Beginner	20 h	4.3 (837)
3	Pointers, Arrays, and Recursion	Beginner	21 h	4.3 (352)
4	Interacting with the System and Managing Memory	Beginner	24 h	4.2 (182)

2

SPECIALIZATION · 5 COURSES · 84 H

Java Programming and Software Engineering Fundamentals

Duke University | Beginner | Java, JavaScript, HTML, CSS, BlueJ, CodePen

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Programming Foundations with JavaScript, HTML and CSS	Beginner	33 h	4.6 (14,960)
2	Java Programming: Solving Problems with Software	Beginner	18 h	4.6 (8,095)
3	Java Programming: Arrays, Lists, and Structured Data	Beginner	15 h	4.6 (3,223)
4	Java Programming: Principles of Software Design	Beginner	13 h	4.6 (1,505)
5	Java Programming: Build a Recommendation System	Intermediate	5 h	4.7 (661)

FURTHER READING

Modern C

Jens Gustedt

Modern C, from syntax to fine-grained memory management.

Free PDF, CC BY-NC-ND 4.0 · gustedt.gitlabpages.inria.fr/modern-c

Introduction to Programming Using Java

David J. Eck

A complete Java 17 textbook, alongside Duke.

Free, CC BY-NC-SA 4.0 · math.hws.edu/javanotes

S4

YEAR 2 · 211 HOURS · ≈ 11 H PER WEEK OVER 20 WEEKS

Data structures and algorithms

> Choose and implement the right data structures, then design and analyse efficient algorithms.

TARGET SKILLS

- Advanced object-oriented programming in Java
- Lists, trees, hash tables, graphs and their performance
- Divide and conquer, sorting, search and randomised algorithms
- Graph traversal, shortest paths, greedy algorithms
- Dynamic programming and NP-completeness

END-OF-SEMESTER PROJECT

The UCSD capstone (social network analysis in Java), then a benchmark comparing five sorting and shortest-path algorithms.

THINGS TO WATCH

- Stanford Algorithms assumes you can read proofs: S2 prepares for it.
- For intensive interview practice, see the Data Structures and Algorithms option (UCSD).

PROGRAMME

1

SPECIALIZATION · 5 COURSES · 154 H

Object Oriented Java Programming: Data Structures and Beyond

University of California San Diego | Intermediate | Java

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Object Oriented Programming in Java	Intermediate	40 h	4.7 (5,806)
2	Data Structures and Performance	Intermediate	42 h	4.8 (2,251)
3	Advanced Data Structures in Java	Intermediate	30 h	4.8 (1,243)
4	Mastering the Software Engineering Interview	Intermediate	22 h	4.6 (555)
5	Capstone: Analyzing (Social) Network Data	Beginner	20 h	4.7 (101)

2

SPECIALIZATION · 4 COURSES · 57 H

Algorithms Specialization

Stanford University (Stanford Online) | Intermediate | C, Java, Python

Stated prerequisites: Programming in at least one language (C, Java or Python); familiarity with proofs (induction, contradiction); discrete probability

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Divide and Conquer, Sorting and Searching, and Randomized Algorithms	Intermediate	15 h	4.8 (5,342)
2	Graph Search, Shortest Paths, and Data Structures	Intermediate	14 h	4.8 (2,002)
3	Greedy Algorithms, Minimum Spanning Trees, and Dynamic Programming	Intermediate	15 h	4.8 (1,279)
4	Shortest Paths Revisited, NP-Complete Problems and What To Do About Them	Intermediate	13 h	4.8 (832)

FURTHER READING

Open Data Structures

Pat Morin

Data structures with their code in Java.

Free, CC BY 2.5 · opendatastructures.org

Algorithms

Jeff Erickson

Alongside Stanford Algorithms: recursion, dynamic programming, graphs.

Free, CC BY 4.0 · jeffe.cs.illinois.edu/teaching/algorithms

YEAR 3 · 438 HOURS

Software engineering and broadening

Design, organise and test professional software, then security, concurrency, the web and machine learning

S5

YEAR 3 · 210 HOURS · ≈ 10 H PER WEEK OVER 20 WEEKS

Software engineering

> Move from a program that works to maintainable software: design, architecture, agile lifecycle and automated testing.

TARGET SKILLS

- Object-oriented design and design patterns
- Software architecture and service-oriented architectures
- Software processes, agile and lean
- Black-box and white-box testing, automated analysis
- Web and mobile testing

END-OF-SEMESTER PROJECT

A software project run in pairs over two sprints: design with patterns, product backlog, automated tests and continuous integration.

THINGS TO WATCH

- Software Design and Architecture uses Java and Android Studio.
- Two Software Testing courses are rated below 4: supplement with the Google book.

PROGRAMME

1

SPECIALIZATION · 4 COURSES · 52 H

Software Design and Architecture Specialization

University of Alberta | Beginner | Java, Android Studio

Stated prerequisites: Familiarity with object-oriented programming; basic Java

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Object-Oriented Design	Intermediate	17 h	4.7 (2,434)
2	Design Patterns	Intermediate	15 h	4.7 (1,382)
3	Software Architecture	Intermediate	10 h	4.5 (958)
4	Service-Oriented Architecture	Intermediate	10 h	4.5 (599)

2

SPECIALIZATION · 4 COURSES · 63 H

Software Development Lifecycle Specialization

University of Minnesota | Intermediate

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Software Development Processes and Methodologies	Beginner	21 h	4.7 (2,135)
2	Agile Software Development	Intermediate	13 h	4.7 (7,485)
3	Lean Software Development	Beginner	11 h	4.6 (433)
4	Engineering Practices for Building Quality Software	Intermediate	18 h	4.5 (476)

3

SPECIALIZATION · 4 COURSES · 95 H

Software Testing and Automation Specialization

University of Minnesota | Intermediate | Java, JUnit, Cucumber, Gherkin, Selenium, OWASP

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Introduction to Software Testing	Intermediate	30 h	4.3 (839)
2	Black-box and White-box Testing	Intermediate	26 h	3.8 (117)
3	Introduction to Automated Analysis	Intermediate	19 h	4.2 (87)
4	Web and Mobile Testing with Selenium	Intermediate	20 h	3.7 (111)

FURTHER READING

Software Engineering at Google

T. Winters, T. Manshreck, H. Wright (eds.)

Software engineering practice at scale: testing, code review, dependencies.

Free online, CC BY-NC-ND 4.0 · abseil.io/resources/swe-book

The Scrum Guide

K. Schwaber, J. Sutherland

The reference text for Scrum, in a few pages.

Free · scrumguides.org

S6

YEAR 3 · 228 HOURS · ≈ 11 H PER WEEK OVER 20 WEEKS

Security, concurrency, the web and AI

> Broaden the profile: design secure software, program concurrency in Go, build for the web and use machine learning.

TARGET SKILLS

- Threats, abstraction and secure design methods
- Functions, methods and concurrency in Go
- HTML, CSS, JavaScript and responsive design
- Supervised and unsupervised learning, introduction to deep learning

END-OF-SEMESTER PROJECT

The final project: a secure web application, served by a concurrent Go service, with a machine-learning recommendation feature.

THINGS TO WATCH

- Programming with Go needs object-oriented experience: Java (S3, S4) provides it.
- Machine Learning: Theory and Hands-on Practice has few reviews; rely on ISLP.

PROGRAMME

1

SPECIALIZATION · 4 COURSES · 60 H

Secure Software Design Specialization

University of Colorado System | Beginner | Java, NetBeans, JUnit, IBM Rhapsody

Stated prerequisites: Knowing how to code in some language; workplace software development experience helpful

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Software Design as an Element of the Software Development Lifecycle	—	11 h	4.6 (253)
2	Software Design as an Abstraction	Intermediate	16 h	4.7 (135)
3	Software Design Methods and Tools	—	14 h	—
4	Software Design Threats and Mitigations	Intermediate	19 h	4.6 (94)

2

SPECIALIZATION · 3 COURSES · 34 H

Programming with Google Go

University of California, Irvine | Intermediate | Go

Stated prerequisites: Previous programming experience, ideally object-oriented (C++ or Python)

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Getting Started with Go	Intermediate	10 h	4.6 (2,484)
2	Functions, Methods, and Interfaces in Go	Intermediate	11 h	4.6 (1,017)
3	Concurrency in Go	Intermediate	13 h	4.7 (809)

3

SPECIALIZATION · 5 COURSES · 78 H

Web Design for Everybody: Basics of Web Development & Coding Specialization

University of Michigan | Beginner | HTML, CSS, JavaScript, Bootstrap

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Introduction to HTML5	Beginner	12 h	4.7 (27,930)
2	Introduction to CSS3	Beginner	17 h	4.8 (9,671)
3	Interactivity with JavaScript	Beginner	10 h	4.7 (7,382)
4	Advanced Styling with Responsive Design	Beginner	11 h	4.7 (4,617)
5	Web Design for Everybody Capstone	Beginner	28 h	4.8 (1,790)

4

SPECIALIZATION · 3 COURSES · 56 H

Machine Learning: Theory and Hands-on Practice with Python

University of Colorado Boulder | Intermediate | Python

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Introduction to Machine Learning: Supervised Learning	Intermediate	23 h	4.5 (67)
2	Introduction to Machine Learning: Unsupervised Learning	Intermediate	15 h	4.6 (24)
3	Introduction to Deep Learning	Intermediate	18 h	4.5 (24)

FURTHER READING

A Tour of Go

The Go Authors

The official Go tutorial, concurrency included.

Free official tutorial · go.dev/tour

Eloquent JavaScript, 4th edition

Marijn Haverbeke

To go beyond the JavaScript basics of Web Design for Everybody.

Free, CC BY-NC 3.0 · eloquentjavascript.net

An Introduction to Statistical Learning (Python)

G. James, D. Witten, T. Hastie, R. Tibshirani, J. Taylor

The reference textbook for Machine Learning (CU Boulder).

Free PDF · www.statlearning.com

6. Options

These specializations are not counted in the total. They suit the end of year 3 or after the curriculum, depending on the chosen direction.

Intensive practice on data structures and algorithms through auto-graded programming exercises: ideal for technical interview preparation.

SPECIALIZATION · 6 COURSES · 182 H

Data Structures and Algorithms Specialization

University of California San Diego | Intermediate | Python, Java, C++, JavaScript, Rust

Stated prerequisites: Basic programming (loops, arrays, stacks, recursion); discrete math (induction, contradiction)

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Algorithmic Toolbox	Intermediate	41 h	4.6 (12,588)
2	Data Structures	Intermediate	23 h	4.6 (5,580)
3	Algorithms on Graphs	Intermediate	55 h	4.7 (2,272)
4	Algorithms on Strings	Intermediate	19 h	4.5 (1,091)
5	Advanced Algorithms and Complexity	Advanced	27 h	4.6 (703)
6	Genome Assembly Programming Challenge	Advanced	17 h	4.5 (355)

CU Boulder's advanced algorithms, up to linear programming and quantum algorithms. Eligible for academic credit.

SPECIALIZATION · 5 COURSES · 199 H

Foundations of Data Structures and Algorithms Specialization

University of Colorado Boulder | Advanced | Python | academic credit

Stated prerequisites: Some background in calculus, probability theory and Python programming

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Algorithms for Searching, Sorting, and Indexing	Intermediate	36 h	4.7 (544)
2	Trees and Graphs: Basics	Advanced	34 h	4.6 (194)
3	Dynamic Programming, Greedy Algorithms	Advanced	38 h	4.7 (282)
4	Approximation Algorithms and Linear Programming	Advanced	46 h	4.8 (67)
5	Advanced Data Structures, RSA and Quantum Algorithms	Advanced	45 h	4.6 (70)

Modern, safe systems programming in Rust, with data engineering and DevOps.

SPECIALIZATION · 5 COURSES · 161 H

Rust Programming Specialization

Duke University | Beginner | Rust, Python, Cargo, Docker, Jenkins, Prometheus, Hugging Face

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Rust Fundamentals	Beginner	40 h	4.0 (255)
2	Data Engineering with Rust	Intermediate	64 h	3.5 (72)
3	Rust for DevOps	Intermediate	19 h	4.1 (19)
4	Python and Rust with Linux Command Line Tools	Intermediate	21 h	5.0 (16)
5	Rust for Large Language Model Operations (LLMOps)	Intermediate	17 h	4.3 (15)

Functional and parallel programming, up to Spark, with EPFL.

SPECIALIZATION · 5 COURSES · 184 H

Functional Programming in Scala Specialization

École Polytechnique Fédérale de Lausanne | Intermediate | Scala, Apache Spark, Apache Hadoop

Stated prerequisites: At least one year of programming experience

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Functional Programming Principles in Scala	Intermediate	56 h	4.8 (8,271)
2	Functional Program Design in Scala	Intermediate	35 h	4.5 (3,137)
3	Parallel programming	Intermediate	33 h	4.4 (1,844)
4	Big Data Analysis with Scala and Spark	Intermediate	28 h	4.6 (2,600)
5	Functional Programming in Scala Capstone	—	32 h	4.4 (547)

Server-side web development in Python, following Web Design for Everybody.

SPECIALIZATION · 4 COURSES · 62 H

Django for Everybody Specialization

University of Michigan | Intermediate | Django, Python, HTML, CSS, JavaScript, jQuery, SQL, Linux

Stated prerequisites: Python for Everybody specialization or equivalent (encouraged)

#	COURSE	LEVEL	LENGTH	RATING (REVIEWS)
1	Web Application Technologies and Django	Intermediate	12 h	4.7 (1,928)
2	Building Web Applications in Django	Intermediate	14 h	4.7 (787)
3	Django Features and Libraries	Intermediate	17 h	4.7 (401)
4	Developing and Deploying a Complete Django Web Application	Intermediate	19 h	4.8 (295)

7. Bibliography

All 14 works can be read free online, under the conditions stated.

1. **Charles R. Severance**. *Python for Everybody*. Free, CC BY 4.0. www.py4e.com/book
2. **Allen B. Downey**. *Think Python, 3rd edition*. Free, CC BY-NC-SA 4.0. allendowney.github.io/ThinkPython
3. **E. Lehman, F. T. Leighton, A. R. Meyer (MIT)**. *Mathematics for Computer Science*. Free, CC BY-SA. ocw.mit.edu/courses/6-1200j-mathematics-for-computer-science-spring-2024/pages/readings
4. **Jim Hefferon**. *Linear Algebra*. Free. hefferon.net/linearalgebra
5. **D. Diez, M. Çetinkaya-Rundel, C. Barr**. *OpenIntro Statistics*. Free PDF. www.openintro.org/book/os
6. **Jens Gustedt**. *Modern C*. Free PDF, CC BY-NC-ND 4.0. gustedt.gitlabpages.inria.fr/modern-c
7. **David J. Eck**. *Introduction to Programming Using Java*. Free, CC BY-NC-SA 4.0. math.hws.edu/javanotes
8. **Pat Morin**. *Open Data Structures*. Free, CC BY 2.5. opendatastructures.org
9. **Jeff Erickson**. *Algorithms*. Free, CC BY 4.0. jeffe.cs.illinois.edu/teaching/algorithms
10. **T. Winters, T. Manshreck, H. Wright (eds.)**. *Software Engineering at Google*. Free online, CC BY-NC-ND 4.0. abseil.io/resources/swe-book
11. **K. Schwaber, J. Sutherland**. *The Scrum Guide*. Free. scrumguides.org
12. **The Go Authors**. *A Tour of Go*. Free official tutorial. go.dev/tour
13. **Marijn Haverbeke**. *Eloquent JavaScript, 4th edition*. Free, CC BY-NC 3.0. eloquentjavascript.net
14. **G. James, D. Witten, T. Hastie, R. Tibshirani, J. Taylor**. *An Introduction to Statistical Learning (Python)*. Free PDF. www.statlearning.com

8. Method and caveats

- **Data source.** Courses, lengths, levels and ratings come from coursera.org pages collected in November 2026, and from the “Computer Science Specializations” and “Data Science Specializations Coursera” files.
- **How hours are counted.** Totals add up the length stated for each course. Coursera's header estimates, which often differ, are not used.
- **What the curriculum does not cover.** Coursera offers no university specialization covering operating systems, networks, databases or compilers; the C, Go and Software Testing courses touch on part of them.
- **Catalogue changes.** Coursera regularly changes its courses, lengths and ratings. Check each page before enrolling.
- **Textbooks.** Free availability was checked on the authors' and publishers' websites; licences differ.