Google

Careers

3uide for Technical Development

ɨaving a solid foundation in Computer Science is important to become a successful Software :ngineer. This guide is a suggested path for university students to develop their technical skills ıcademically and non-academically through self paced hands-on learning. You may use this |uide to determine courses to take, but please make sure you are taking courses required for 'our major in order to graduate. The online resources provided in this guide are not meant to eplace courses available at your university. However, they may help supplement your learnings ır provide an introduction to a topic.

Jsing this guide:

- Please use this guide at your discretion
- There may be other things you want to learn or do outside of this guide - go for it!
- Checking off all items in this guide does not guarantee a job at Google
- This guide will evolve or change - check back for updates

:ollow our Google for Students +Page to get additional tips, resources, and other students ıteresting in development.

₹ecommendations for Academic Learnings

- **Introduction to CS Course**

  Take Introduction to Computer Science courses that provide basic instructions on coding.

  Online resources: Udacity - intro to CS course, Coursera - Computer Science 101

- **Code in at least one object oriented programming language: C++, Java, or Python**

Beginner Online Resources: Coursera - Learn to Program: The Fundamentals, MIT Intro to Programming in Java, Google's Python Class, Coursera - Introduction to Python, Python Open Source E-Book

Intermediate Online Resources: Udacity's Design of Computer Programs, Coursera - Learn to Program: Crafting Quality Code, Coursera - Programming Languages, Brown University - Introduction to Programming Languages

- **Learn other Programming Languages**

  Notes: Add to your repertoire - JavaScript, CSS & HTML; Ruby; PHP; C; Perl; shell script; Lisp and Scheme.

  Online Resources: w3school.com - HTML Tutorial, Codecademy, Udacity - Mobile Web Development, Udacity - HTML5 Game Development

- **Test Your Code**

  Notes: Learn how to catch bugs, create tests, and break your software

  Online Resources: Udacity - Software Testing Methods, Udacity - Software Debugging

- **Develop logical reasoning and knowledge of discrete math**

  Online Resources: MIT Mathematics for Computer Science, Coursera - Introduction to Logic, Coursera - Linear and Discrete Optimization, Coursera - Probabilistic Graphical Models, Coursera - Game Theory

- **Develop strong understanding of Algorithms and Data Structures**

  Notes: Learn about fundamental data types (stack, queues, and bags), sorting algorithms (quicksort, mergesort, heapsort), data structures (binary search trees, red-black trees, hash tables), and Big O.

  Online Resources: MIT Introduction to Algorithms, Coursera Introduction to Algorithms Part 1 & Part 2, List of Algorithms, List of Data Structures, Book: The Algorithm Design Manual

- **Develop a strong knowledge of operating systems**

  Online Resources: UC Berkeley Computer Science 162

- **Learn UX Design**

  Online Resources: Udacity - UX Design for Mobile Developers

- **Learn Artificial Intelligence**

  Online Resources: Stanford University - Introduction to Robotics, Natural Language

Processing, Machine Learning

- **Learn how to build compilers**

  Online Resources: Coursera - Compilers

- **Learn cryptography**

  Online Resources: Coursera - Cryptography, Udacity - Applied Cryptography

- **Learn Parallel Programming**

  Online Resources: Coursera - Heterogeneous Parallel Programming

- **Work on project outside of the classroom.**

  Notes: Create and maintain a website, build your own server, or build a robot.

  Online Resources: Apache List of Projects, Google Summer of Code, Google Developer Group

- **Work on a small piece of a large system (codebase), read and understand existing code, track down documentation, and debug things.**

  Notes: GitHub is a great way to read other people's code or contribute to a project.

  Online Resources: GitHub, Kiln

- **Work on project with other programmers.**

  Notes: This will help you improve your ability to work well in a team and enable you to learn from others.

- **Practice your algorithmic knowledge and coding skills**

  Notes: Practice your algorithmic knowledge through coding competitions like CodeJam or ACM's International Collegiate Programming Contest.

  Online Resources: CodeJam, ACM ICPC

- **Become a Teaching Assistant**

  Helping to teach other students will help enhance your knowledge in the subject matter.

- **Internship experience in software engineering**

  Notes: Make sure you apply for internships well in advance of the period internships take place. In the US, internships take place during the summer, May-September. Applications are usually accepted several months in advance.