The Ultimate Self-Taught Developer Curriculum

Video

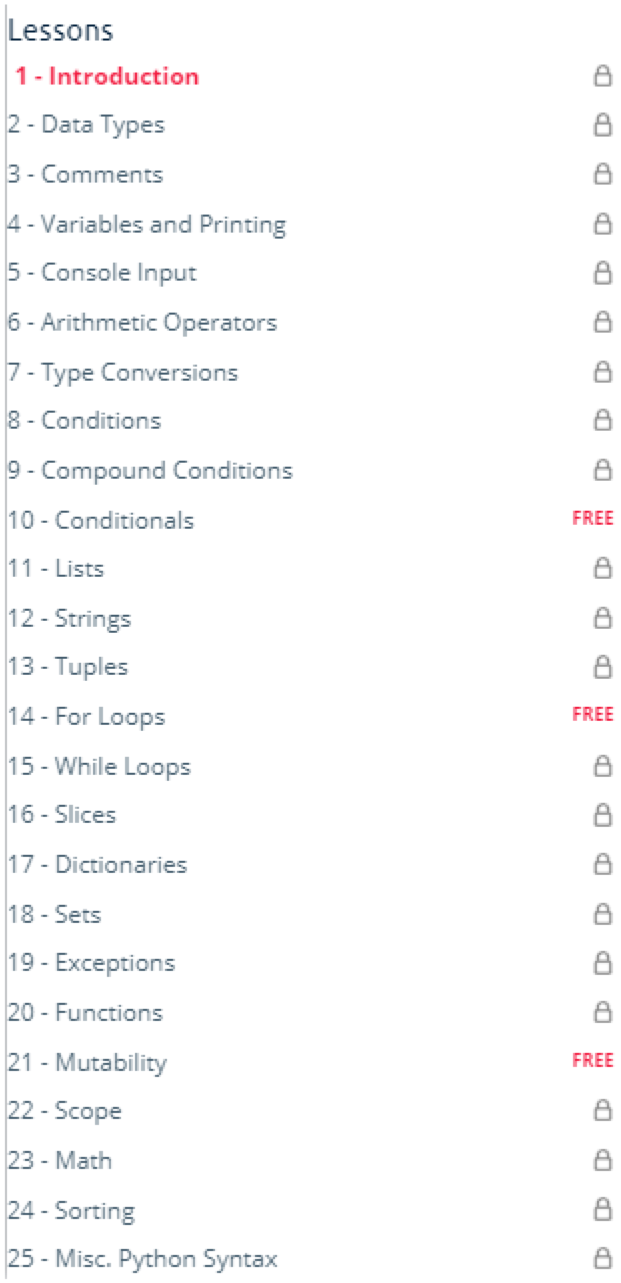
https://youtu.be/vxctuiRlmrs?si=WsTa-3I5jBlbUjZC

Notes

- Assumes you have 15-20 hours a week to study

Month 1

- Determine what type of programming you want to do (1-2 day/s)
- Determine the language based on type you have decided on
- Fundamental programming concepts (data types, variables, for loop, etc)



Lessons

1 - Introduction	🔒
2 - Data Types	🔒
3 - Comments	🔒
4 - Variables and Printing	🔒
5 - Console Input	🔒
6 - Arithmetic Operators	🔒
7 - Type Conversions	🔒
8 - Conditions	🔒
9 - Compound Conditions	🔒
10 - Conditionals	FREE
11 - Lists	🔒
12 - Strings	🔒
13 - Tuples	🔒
14 - For Loops	FREE
15 - While Loops	🔒
16 - Slices	🔒
17 - Dictionaries	🔒
18 - Sets	🔒
19 - Exceptions	🔒
20 - Functions	🔒
21 - Mutability	FREE
22 - Scope	🔒
23 - Math	🔒
24 - Sorting	🔒
25 - Misc. Python Syntax	🔒

- Problems and practice tasks

Month 2

- Object Oriented Programming
- Paradigms and topics

  Examples

  - Classes

    - Objects
    - Attributes
    - Methods
    - Static methods
    - Inheritance
    - Abstract classes
    - Interfaces
    - Operator overloading
- Start some small projects (with a couple classes)

Month 3

- Should be more comfortable, know the fundamentals and remember most the basic syntax
- Projects and exercises to practice these skills

Month 4-6

- Start looking into advanced programming concepts

  Examples

    - Decorators
    - Organising code (into modules and packages)
    - How to run code from the command line
    - How to manually compile code
    - Iterators
    - Generators
    - Underlying concepts specific to chosen programming language (how it works)
    - Asynchronous programming
    - Threading

    - Multiprocessing
    - Memory management
    - Pointers vs references
    - Language types (dynamic, etc)
- Start looking into 'Operating System concepts'

  Examples

    - How a CPU works and what is it
    - Core vs thread
    - What does it mean to run a program
    - Binary
    - Files - source code vs bytecode
- Work on harder programming problems (easy and medium problems on platforms such as LeetCode)

Month 7

- How to write 'Quality Code'

  Examples

    - What is a 'good program'?
    - Good programming habits
    - Clean code
    - Optimal solutions
    - How to make code easier for other people to read, edit, etc
    - Code bases
    - How to organise code
    - How to lay out many files
- Read other peoples code (Spend an hour on GitHub or Stack Overflow reading production level code)

- Start learning Git and GitHub basics
- Start learning the command line/bash (how to move a file, etc)

Month 8

- Learn another programming language

  Examples

  - Dynamically typed → statically typed
  - High level → low level (Python → C or C++)
- Learn the idea of using the 'best tool for the job'
- No need to be a master at the second language

Month 9-10

- Data structures and algorithms
- Time and space complexity (Big O, Big Theta, Big Omega)
- Start looking at mathematics related to programming

  Examples

  - Proof for an algorithm
  - Discrete mathematics
  - Linear algebra
- Start looking into computer architecture

  Examples

  - Transistor
  - Gates (AND, NOR, etc)
  - CPU design
- Heap
- Binary Trees
- B-Trees

Month 11-12

- Learn some modules and packages for a language
- Do a large project (over 2 months)
- System design

  Examples

  - How is something like Netflix made?
  - DevOps
  - How many servers does Netflix have?
  - How optimised is their delivery mechanism?
  - How are they serving content so quickly?
- Design patterns and program structure
- Things specific to the area of programming you want to do (start specialising)
- More practice projects and exercises