Beginner Programming

Jarnis

August 27, 2020

Preface

This is basically a run-down of learning python (so that you can start using APIs to build cool stuff in discord, twitter etc) in a way that will help with the long term journey which generally involves a lot of self-learning and trials and errors.

All the links in this document were visited with an ad-blocker extension, so I can't tell if these links have ads or not. Open with caution
If you can't open this link to google by clicking/double clickling 'this link' then you should try to use another PDF viewer.

Chapter 1

Starting out with python

1.1 Before we Begin

Advice on how to take Advice

> "Listen to all, plucking a feather from every passing goose, but, follow no one absolutely." Chinese Proverb

So, I read, watch and listen to a lot of what might very generally be called "advice" and a lot of advice-givers, myself included, can seem to be telling you:

> **"What you've been doing SUCKS! You're messing up! Do it THIS way!"**

.
And this makes you angry, dissapointed, depressed and despicable with yourself. Which leads you down the path of *struggling to fit your* `square` *self into whatever* `round` *hole your advice giver has prepared for you*

So I'm not going to tell you what to do. I will just type it out here and you can read it if you want. I don't think the point of every altruistic, amenable, attentive advice-giver is to down you and get you to aggressively start rearranging your life as per their image. Ofcourse, it's not my intention either. `Good advice is intended to make your life easier, not harder`.

I would like to propose a simple guideline to reading any sort of advice.
`Step 1` Read the advice
`Step 2` Try to implement it, i.e. give it a shot for like an hour or a day,

whatever *you* feel is a good duration.
`Step 3` Do whatever you want, whether or not it matches the advice or not.

Some Advice

Before starting on this conquest, I recommend you learn a few basic notions to make things easier in the later stages. Please go through the following links to start with so that you don't lose your way. Learn about a fellow's journey and another fellow.

Then go through these links next Something we all should listen to and what I wish I knew before I started learning. Also why learning code is hard is a good resource to keep you self-aware. A bit of understang of mastery is also helpful.
For the math portions (if you encounter any) Khan Acadmey is always a good and generally sufficient resource to learn.

1.2 Learning a bit of broad knowledge

This article will give you a basic rundown of what programming languages are.
Knowing a bit about object oriented programming from the wikipedia article is also helpful. Just skim through if you don't understand. Make sure to watch this video as well.
Watch this what is an API video as well.
This interactive RegEx(Regular Expressions) tutorial is also quite nice.
I feel for starting out all this is quite sufficient.

1.3 Python Basics

Start out with the first 11 chapters of automate-the-boring-stuff, the link gives you the free access to the Nth edition of the book. After finishing chapter 3 start solving the Project Euler problems using the programming knowledge you have gained. This will also be helpful in getting familiar with basic maths.
After finishing the 11 chapters I recommend you start making some projects. Some project ideas are `Hangman game, Blackjack, Text-based adventure game, Strong-Password generator`; additionally you can do the rest of the book which will give more project ideas you can implement.
I would recommend you make atleast 3 projects before proceeding.

Chapter 2

Getting a grasp of Objects

This will be a short chapter. If you have forgotten you should go back to chapter 1 and go through the OOP (Object Oriented Programming) links posted there.
Start off with the short intro to the syntax.
Now you should go a bit indepth with the syntax of OOPs in real python.
If this is feeling a bit more complex, then go through this video playlist and then try the website article again.
Then try out `Think Complexity by Allen B. Downey` for some practical implementations and some much needed graph theoretic background. I recommend you do atleast 3 chapters of this book and do all the exercises in this book of the chapters.

After this you can either try to re-implement the projects you wrote earlier or make new projects with the OOPs concepts you have learnt.

Chapter 3

Getting Adept at Python

At this point, if you haven't already, I recommend you download the individual edition of Anaconda for a variety of python packages you can use. At this point you can do the sections in any order other than the recommended order present here.

3.1 Learning the Standard Library

Learning the python standard library will enhance your python skills to amazing levels and allow you to do much more with python.
Do `Programming Python - Mark Lutz`, this is a very technical and dry book. I would recommend you *try to do* atleast parts I,II and III.

3.2 Learn something different

Learn Django

Learn how the internet works here, this is important, do not skip.
Do the Basic HTML and Basic CSS of the free Responsive Web Design Certification from here and make a basic website to know the structure of websites.
You should just get started now.

Chapter 4

Learn C++

Learning C++ will enhance your python understanding because python was built on this.
Honestly, at this point just do all/ as many chapters of this online book for all the C++ knowledge you might need.