Python Basic Track

LCfP Staff

September 29, 2017

Contents

Introduction

0.1 About this book

0.2 About the authors

0.2.1 Vincent Velthuizen

0.2.2 Niels Wouda

0.2.3 Nick Szirbik

0.3 Acknowledgements

Chapter 1

What it is and what it isn't

TL;DR

- Computers are not smart!
- Programming is not a silver bullet,
- and it can be more art then science.
- Software engineering is hard.
- We will allow you to be experience the above to show you that they are true.

1.1 Computers

"I think one of the things that really separates us from the high primates is that we're tool builders.

I read a study that measured the efficiency of locomotion for various species on the planet. The Condor used the least energy to move a kilometer and humans came in with a rather unimpressive showing about a third of the way down the list. It was not not too proud of a showing for the crown of creation, so that didn't look so good. But then somebody at Scientific American had the insight to test the efficiency of locomotion for a man on a bicycle. And a man on a bicycle or human on a bicycle blew the Condor away, completely off the top of the charts.

That's what a computer is to me. It's the most remarkable tool that we've ever come up with. It's the equivalent of a bicycle for our minds"

Steve Jobs (1955-2011)

`https://www.youtube.com/watch?v=AfbG4-guIAA`

The above metaphor is very accurate. A computer can be a very powerful tool, accelerating you capabilities to great heights. But by it self a computer doesn't do very much, it needs to be both powered and steered by someone. It also takes some effort to learn how to use it.

Unfortunately, unlike bicycles, computers evolve *very* quickly. You might not forget how to use the one you are working with, but keeping your skills relevant requires upkeep!

1.2 Programming

A computer program is basically a recipe for a computer. Do this, then do this, etc. Learning to program is also similar to learning how to cook. You will start with the basics, think boiling and baking eggs. Moving on to more complicated techniques (for example poaching), or combining ingredients into a more complicated dish (a cake).

An import part of programming is problem solving. Computers 'think' differently from humans. It is up to the programmer to transform a 'human' problem into one the computer can solve for you. In the beginning you might think "this is easier to do myself". When you get good at programming however you can save yourself a lot of time.

Once you have become a capable program you'll start to see ways to automate the repetitive tasks around you. A common occurrence when your reach that stage is creating

overly complicated solutions to simple tasks. We will show you some interesting cases of these as the course progresses.

1.3 Software engineering

The possibilities for software seem almost limitless. This causes expectations to be high and programs to become complicated. When programs become very big and complicated it can be very difficult to predict its outcome. This usually isn't what we want. A program should reliably produce 'correct' results.

Techniques were developed to help create large, mostly predictable, programs. Mistakes (bugs) still happen, but multi-million line programs are run everyday, everywhere (think Windows, Android, iOS, Linux, etc.). The fields that deals with creating large programs is called 'Software engineering'. You are very likely to be part of a software engineering effort at some point in your life, and the techniques from this field are being used outside of software development more and more.

1.4 This course

During the course we will teach you how to use and talk to a computer. We will show you many 'ingredients' and 'techniques' that can be used when programming. You will learn to work with software engineering tools and be part of at least one project to experience the tools and techniques for yourself. We encourage you to make, and learn from your, mistakes.

The basis for the course is a self-paced course from `Codecademy.com`. This is to encourage you to practice outside of the sessions we organise. We would like to expand to borders of your knowledge each week. It is then up to you to discover the new areas available to you. Once you get the lay of the land, we can increase the borders again!

Part I

Codecademy course

Chapter 2

Python syntax

2.1 Variables

2.1.1 Datatypes

int, float, bool

2.1.2 Duck typing

`https://en.wikipedia.org/wiki/Duck_typing`

2.2 Whitespace

2.2.1 Keep code together

2.3 Comments

2.4 Arithmetic operations

2.5 Apply these concepts

2.5.1 Tip calculator

Chapter 3

Strings & Console Output

Chapter 4

Conditionals and Control Flow

Chapter 5

Functions

Chapter 6

Lists & Dictionaries

Index