Part I

Basic Programming Skills

Chapter 1

Getting Started

Learning Objectives

- Why learn how to program/think like a computer scientist?
- What can you expect (and what should you not expect)?
- Which tools will we use?

1.1 Why learn how to program/think like a computer scientist?

There are many reasons why you might want to learn how to program. Some are more practical, others more philosophic. If you have a clear idea why you are trying to learn how to program, great! But if you ever need help to motivate you to continue, consider some of the things here.

dealing with frustration While programming you will encounter frustration. Having a very patient machine executing whatever you tell it, but not getting the result you want can be vexing. It is like getting three wishes, but not getting what you want because you didn't use the exact right words. The worst part is that you have nobody to blame except yourself.

That doesn't sound very good, but it trains you to formulate very precisely what you want. Though not always appreciated in polite conversation, this is a very useful skill. Especially when ordering (custom) software from a software engineer.

bicycle for the mind `https://youtu.be/AfbG4-guIAA`

Steve Jobs described the computer as a bicycle for the mind. By itself a computer doesn't achieve much. A computer extends our mental capabilities to allow for great things. A computer is a tool and like any tool you have to learn how to use it most effectively.

A computer is very flexible and can help you solve many problems. But you are responsible for using it right. If you use the wrong screwdriver to tighten a screw and it doesn't work, it's not because the screw or screwdriver is defective. It also doesn't mean you are wrong for trying though! (But know when you should make the effort to get the "right" screwdriver).

problem solving What differentiates a computer scientist from a programmer? There are many opinions on this. The one we'll consider here is the following.

A programmer writes programs based on a specification. A computer scientist answers (hopefully interesting) questions. Finding the answers may (they may not!) require a computer. A computer scientist knows how he might use a computer to answer his or her questions and might enlist the help of programmers to to actually do it.

modern working/living We are surrounded by computers wherever we go. Knowing how to use makes you a better candidate for a job. Allows you to solve more problems yourself (without having to pay for a specialist). And generally allows you to make more of opportunities to see around you.

I feel that if you are dependant on something in your day to day live, it's worth knowing how it works.

1.2 What can you expect?

We will provide a framework for you to learn how to "think like a computer scientist" including (but not limited to) programming. This framework consists of

space and pace How fast should you be going through the materials.

feedback and monitoring Make sure you don't fall behind.

community You are not alone and learn from each other.

These should help you, but you have to put in the work!

1.3 Which tools will we use?

This is not the time or place to go in depth into all of the resources used during the basic track. Here we give you the basics needed to get started. As time progresses you'll get familiar with these tools and their more advanced features.

"How to think like a computer scientist"

Throughout the basic track we'll refer to the materials in "How to think like a computer scientist". The book can be found here:
`https://buildmedia.readthedocs.org/media/pdf/howtothink/latest/howtothink.pdf`.

Other versions are available, but when referring it will be to this version (3rd edition, Feb 26, 2019).

The book includes code examples that you can and should run! This is not part of the assignment but a chance to see the concepts from the book in action before applying them in an assignment.

Given the extent of the book and the limitations of the basic track, we will have to skip some material. Whenever you would like or require a deeper understanding of any topic the skipped parts are a good place to start!

Python

The programming language we'll be using is Python 3.8.* [1]

MacOS comes with python built-in but that is an old version which was further limited to the bare essentials. DO NOT use this version of Python, get the one mentioned above. You might consider googling how to use *homebrew* to install Python, but that is up to you.

[1] The * here means "latest version". At the time of writing this is 3.8.1

IDLE/PyCharm

Computer programs are typically plain (no markup) text files. This means you can write programs using any text editor, for example: notepad. A large part of choosing software typically comes down to personal preference and I will refrain from make big generalizations here.

A type of software that comes with a text editor is the Integrated Development Environment (IDE). IDEs typically help you write programs faster, providing auto-complete and helping you identify problems with your code. There are many IDEs and which one is “best” differs per person and sometimes even per project.

During the basic track please use the tools we recommend, this makes it easier for us to help you. At the end, or during the projects phase we can help you experiment with other tools if you’d like.

IDLE or Integrated Development and Learning Environment comes standard with Python. If you have successfully installed Python you have access to IDLE. Though it helps you a little bit, it allows you to make your own mistakes, which can be useful while learning.

PyCharm is a much more extensive IDE, helping you write code much faster. This year we will start with PyCharm right away. There are many, many settings, but if you don’t know what you are doing you may get yourself in a lot of trouble, so be careful. You are likely to get into a situation you are not yet able to get out of by yourself.

PyCharm offers a community edition which is free for personal use. There is also an ultimate version for which students can request a free licence. The community edition should be enough for you to complete the work for this learning community, though the ultimate edition is used for the examples.

Git/GitHub

When working on larger projects with multiple programmers it is important to have a way to share code. We’ll go deeper into the Git “Workflow” and tools at a later time. There are several Git providers on the internet and it is also possible to host your own. For the purposes of these lessons we’ll use GitHub and PyCharm’s built-in Git client.

1.4 Homework

Reading

- Chapter 1 (`https://buildmedia.readthedocs.org/media/pdf/howtothink/latest/howtothink.pdf`)

Assignments

1. Install Python
2. Install PyCharm
3. Create A GitHub account
4. Do the exercises from section 1.12