Advanced Python

Module 7A.2. Part 1

1

1. What We'll Cover

Part 1 will explain…

- Programming from first principles.
- How programming relates to hardware.
- What programming languages allow us to do.
- The different types of languages.
- Where Python fits in.

The aim: to help you understand the programming you've already done in a wider context.

2

2. Progress

- So far, you've undertaken code academy tutorials, and learned the basics of writing code.
- I think it's true to say that you've made great progress!
- Tools like code academy are fantastic – they've allowed you to learn the process of programming.
- However, such tools abstract away some of the important detail - now we try to fill in some of the gaps.

3

3. Programming In a Nutshell

- Programming involves the writing of instructions that orchestrate the actions of a computer and it's hardware components.
- The instructions are written with the aim of accomplishing a specific goal or completing a specific task.
- You know this, but what does it mean?
- How is it achieved?
- We don't need to delve too deep into computer science to gain a solid understanding.

4

4. Current Experience

- For many of you, programming has consisted of the following steps:
  1. Open a web browser (Chrome, Firefox, etc).
  2. Load the code academy website.
  3. Edit the code in the browser window.
  4. Run the code in the browser window.
- Some of you may have written code in other tools, then run it via a website.

5

5. What's Really Happening?

- Although these details are abstracted away when using code academy, what you're actually doing, is controlling a CPU.
  - The code academy website is running on a computer.
  - When you run your code in the browser window, it is executed on a computer hosted somewhere.
  - That computer has it's own CPU, memory, hard disks etc. When you create a variable such as `name = "Rob"`, that variable is stored in the RAM (memory) of that computer.
  - Your instructions are passed to that computer's CPU, and executed.
- This is what's really been happening when you've been running code during your tutorials.

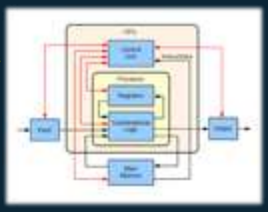
Standard, everyday CPU.

6

6. Controlling a CPU

A simple diagram of a modern CPU is shown to the right.

- It accepts input in the form of machine code, and produces outputs, also in machine code.
- The parts in between are responsible for various tasks – math operations (adding, subtracting etc), moving data, finding data, logical tests, and accessing the memory.

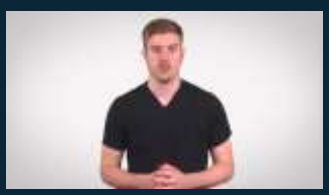


To control the CPU, the code you write is turned into "Machine Code". The CPU then reads the machine code, and ultimately does as it's told!

But how does this link to the programming you've been doing?

7

7. Microarchitecture Review



Credit: https://www.youtube.com/watch?v=FkeRMQzD-0Y

8

8. Types of Languages

- It links, as the code you write has to be converted into machine code.
- There are two different types of coding language:

"Low Level" Languages

- Hard for humans to read.
- Easy for machines to read.
- Can be processed efficiently by CPUs.
- Often written in binary or simple numerical format.
- Can be used to control specific parts of the CPU.
- The total lines of code required to do simple tasks can become very large!

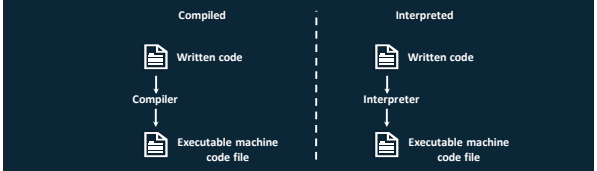
"High Level" Languages

- Easy for humans to read.
- Hard for machines to read.
- Cannot be processed efficiently by CPUs.
- Often written using a combination of human language and numbers.
- The total lines of code required to do simple tasks is small!
- Human language is after all more expressive.

9

9. Translating to Machine Code

- Machine code clearly doesn't need translating for the CPU to understand it.
- All other programming languages must be translated. There are two main approaches for doing this.



10

10. Back to Python

- Python is an interpreted programming language.
- When you run Python code in code academy, a python interpreter reads each line, converts it to machine code, and this is run by the CPU.
- You've been learning Python 3. Thus, you've been using an interpreter capable of translating Python 3 code to machine code.
- Programming languages improve over time. If you wrote code Python 2 code, a Python 3 interpreter would *not* understand it. Thus ,you'd get errors.
- What's important to understand here:
  - Python is interpreted, you therefore need an interpreter to run Python code.
  - You must use an interpreter that understand the code your writing.
  - Any computer with a suitable Python interpreter will be able to run your code.

11

11. Advantages

- There are many advantages to Python being interpreted.
- These have simplified your learning a great deal.
- These advantages include:
  - The ability to create variables without explaining what they contain (e.g. are variables numbers or strings).
  - The ability to create variables without worry where they are stored in memory.
  - The ability to run individual python commands without having to build entire programs.
  - The ability to run Python code on any machine with an interpreter (platform independence).

12

12. Disadvantages

There are some disadvantages:

- We can't control where variables are put in memory, making it difficult to write super-efficient programs.
- Interpreting Python code is slow – this means Python problems do not run as fast as programs compiled in other languages.
- Because we don't explicitly tell Python what types our variables are, we can introduce unexpected errors if we're not careful.

13

13. Review – Compiled vs. Interpreted

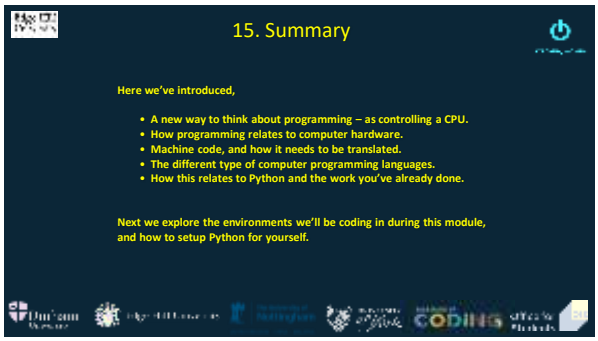
Credit: https://www.youtube.com/watch?v=I1f45REi3k4

14

14. New Understanding

- So far you've primarily written Python in code academy.
- However, now you understand Python in a new way.
- You know you can write Python code outside of code academy and use an interpreter to run it.
- When doing this, we're controlling the CPU, getting it to execute tasks.
- We'll build upon this understanding moving forward.

15


15. Summary
Here we've introduced,
• A new way to think about programming – as controlling a CPU.
• How programming relates to computer hardware.
• Machine code, and how it needs to be translated.
• The different type of computer programming languages.
• How this relates to Python and the work you've already done.
Next we explore the environments we'll be coding in during this module, and how to setup Python for yourself.

16


16. Links
Useful links:
• https://en.wikipedia.org/wiki/Computer_programming
• https://en.wikipedia.org/wiki/Machine_code
• https://en.wikipedia.org/wiki/Compiled_language
• https://en.wikipedia.org/wiki/Interpreted_language
• https://en.wikipedia.org/wiki/High-level_programming_language
• Great tutorial video: https://www.youtube.com/watch?v=rfscVS0vtbw

17