IOC Topic 7A.2 Part 1 – Advanced Python

Transcript & Notes

Author: Dr. Robert Lyon

Contact: robert.lyon@edgehill.ac.uk (www.scienceguyrob.com)

Institution: Edge Hill University

Version: 1.0

Topic 7A.2 Part 1, Introduction Slide

Hello and welcome to Part 1 of Topic 7A, Module 2, Advanced Python. In this module we aim to provide some background that will help you understand the programming you've already done. We eventually build upon this background and introduce some of the "advanced" features of Python. We'll also explore the Object-Orientated approach to software design. My name is Dr. Robert Lyon, and I'll be taking you through this module.

Slide 1

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.

Slide 2

- 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. This will hopefully help provide answers to those questions that crop up often when first learning to program.

Slide 3

- Programming involves the writing of instructions that orchestrate the actions of a computer and its 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.

Slide 4

- For many of you, programming has consisted of the following steps:
- Open a web browser (Chrome, Firefox, etc).
- Load the code academy website. Here we see me doing this in the video to the right. I just load up the tutorials, much like you've done in recent weeks.
- Edit the code in the browser window. You can see in the video that I write some code for myself.
- Run the code in the browser window. To do this, we have to click the run button to execute the code I've written. Then the output appears, so long as I've not made any mistakes.
- Some of you may have written code in other tools, then pasted it into the website window. But the process is pretty much the same – write the code, then run the code.

Slide 5

- Although these details are abstracted away when using code academy, what you're actually doing, is controlling a CPU.
- The code academy website is actually running on a computer.
- When you run your code in the browser window, it is executed on a computer hosted somewhere.
- That computer has its 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.

Slide 6

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?

Slide 7

Please go ahead and watch this video, which will review some of the concepts I've discussed.

Slide 8

- So how does the python code you write link to this? It links, as the code you write has to be converted into machine code. I'll explain how this happens shortly, but for now I'd like you to understand the following point.
- There are two different types of coding language:

There are "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.
- If you add up the number of lines of code required to complete a task in a low-level language, this total can become very large!

There are "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.
- If you add up the number of lines of code required to complete a task in a high-level language, this total can be very small.
- Human language is after all more expressive.

Slide 9

- 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.
- Some computer programming languages are known as compiled. For each type of CPU, a tool called a compiler is created, that understands it's machine code. When you write a piece of code, it is stored in a source code file. This file is then passed to the compiler. The compiler goes through the code file line by line and translates it to machine code. This produces an executable file, that a CPU can run.
- Some computer programming languages are known as interpreted. For each type of CPU, a tool called an interpreter is created, that understands it's machine code. When you write a piece of code, it is stored in a source code file. This file is then passed to the interpreter. The interpreter goes through the code file line by line and translates it to machine code. This produces an executable file, that a CPU can run. Interpreted languages are special, in that they can interpreted on the fly. That is, you don't need to interpret an entire source code file to create machine code. You can interpret single lines of code as you write them if needed.

Let's close the loop and bring this back to Python.

Slide 10

- 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. I suppose it would be a little like writing a book in Shakespearean English, and expecting a modern English speaker to understand it perfectly.
- 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.

Slide 11

- 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 worrying 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).

Slide 12

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 compared to compiling code – this means Python programs 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.

Slide 13

Please go ahead and watch this video, which will review some of the concepts I've discussed.

Slide 14

- 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.

Slide 15

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.

Slide 16

This slide contains some links for you to follow up in your own time. Once you've looked at those, move onto part two.

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