

Computational Thinking

Dr. Christopher Gwilliams

gwilliamsc@cardiff.ac.uk

Computational Thinking?

Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent
Jeannette Wing

TL;DR Problem solving by you, or a machine, in a way that is simple to reproduce

Course Overview

- Python!
- Variables & Types
- Types
- Operators & Conditionals
- Flow Control
- Data Structures

Course Overview

- Loops and Functions
- Methods and Classes
- Input/Output
- Maths & Binary Arithmetic
- Sorting algorithms and problem decomposition

Course Breakdown

Coursework - Week 3

Coding exercise with report (60%)

Exam (1hr) January

Theory based written exam (40%)

What Is This Not?

This is not a quick course to learn Python. First and foremost, this is Computational Thinking through the medium of Python.

This could (almost) as easily be taught using any language. There will be much more to learn about Python after this course but you will be better equipped to learn it.

Intro to Python

```
"Hello, Python"
```

```
Out[2]:
```

```
'Hello, Python'
```

Why Python?

- Open Source (Free as in free speech, not free beer)
- Widely used (bear with me)
- *MASSIVE* standard library
- Cross-platform
- Used in Academia and Industry
- Powerful yet easy to learn

What is this Python?

- Made by Guido Van Rossum in 1991
- Continues to be overseen by him
- Interpreted
- Object-oriented
- Strongly Typed
- Dynamically Typed



Benevolent Dictator for Life

Who Uses Python?

- Academia (all disciplines)
- Google (search)
- Battlefield (Web APIs)
- Disney (animation)
- NASA (shuttle launch planning)
- Instagram
- Mozilla
- Dropbox (EVERYTHING)
- Government
- You?

ya-di-ya-da

Python 2 vs. Python 3

Python 2

- Legacy
- Largest number of libraries
- Coming to end of life soon



Python 3

- Like the Xbox One versus the 360, it is partially backwards compatible
- Almost a complete rewrite
- Now the de facto version
- Libraries in the process of being rewritten for support

How Does It Work?

- Interpreted (not compiled)

Ever written in Java or C++?

Name	
	Average3.class
	Average3.java

Java files compiled to Class files before they are run

- Python (and others) are not compiled, they are interpreted
- What other languages are interpreted?

Interpeted Languages

HIGH LEVEL OVERVIEW Files are run line by line at runtime (no need to compile). Some checks are done before the file runs (spacing, module imports etc), but errors are typically not caught until that line of the file is executed.

Benefits:

- (Generally) faster development time
- Platform independent (source)
- Dynamic types

Drawbacks:

- Speed (kind of, we will look at that in this course)

What Can Python Do?

- Websites (Django, Flask, Bottle)
- Math (algorithms, optimisation and simulations)
- Desktop apps (Kivy, Tkinter, WxPython)
- Mobile apps (kivy)
- Command line tools

Python Interpreter

- Open up a command prompt (terminal) and type `python`
- POW, you are in the python interpreter. Try this:

```
"Hello, Python" #press enter here
"My name is <your name here>"
```

Out[6]:

```
'My name is <your name here>'
```

REPL

What does this stand for?

READ EVAL PRINT LOOP

```
[encima:...]$ python3
Python 3.4.3 (default, Aug 11 2015, 08:57:25)
[GCC 4.2.1 Compatible Apple LLVM 6.1.0 (clang-602.0.53)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, ASE Peeps")    READ AND EVALUATE
Hello, ASE Peeps                PRINT
>>>                             LOOP
```

NOTE: Whitespace

Ever used a different language with semicolons and curly braces?

Python does not have that. EVERYTHING revolves around whitespace. Indentation is key and enforces clean code

We will cover this as we go on but be careful about spacing errors

Do not worry about the code for now, but take a look at how this works:

```
for i in range(0,100):
print("bloop")
```

```
File "<ipython-input-4-4f6a00436454>", line 2
  print("bloop")
    ^
```

IndentationError: expected an indented block

```
for i in range(0,3):
    print("bloop")
```

```
bloop
bloop
bloop
```

Side note: Using the interpreter for errors

The example in the previous slide told you the error and the line it occurred on, like this:

```
[encima:~/Comp_Thinking_in_Python]$ python error.py (master*)
djfkld
Traceback (most recent call last):
  File "error.py", line 2, in <module>
    print("fkjflgh " + 1344)
TypeError: cannot concatenate 'str' and 'int' objects
```

Annotations in the image:

- Line number: points to "line 2"
- Problem line (not always): points to the `print` statement
- Error: points to the `TypeError` message

Always use this **before** you ask/search for the problem. The error text is what you should search for as well. We cannot answer questions like: "Why is my code not working?" any more than you can.

"I am trying to print a string at line 11 and it is giving me an indentation error" Saying something like this out loud (or in your head) may actually help you resolve the problem. That being said: **we are here to help and never be afraid to ask for it, it is what you are paying for**

Python REPL

When you type `python` in to your command prompt, this is the Python interpreter and it is an example of a REPL. Whatever you type is evaluated immediately and run, allowing you to write a new line.

What are the benefits of this? How else can this be done?

REPL Benefits

- Rapid prototyping
- Check it works on your system
- Command line scripting
- Slides, talks and demos!

But we don't always want to be writing one liners in the command line, do we?

Exercise: Python Scripts

- Open up a text editor (atom)
- Type `print("hello, world")`
- Save the file as `hello.py`
- In your command prompt, run it by typing `python hello.py` in the directory it was saved in

What happens?

REPL vs Scripts

What happens when you remove the print and the brackets?

A REPL is designed to show you the state of things with each command. So, typing: `print me` will print that string. Scripts (or files) will not do this unless you tell it to.

This is partly because printing to the command prompt (also called: `stdout`) is one of the slowest operations you can perform in Python!

Learning Python

First port of call: This course and recommended reading

Second port of call: the interpreter (Learn by making mistakes. Fixing bugs and errors will increase your knowledge exponentially)

Third port of call: The Python docs (<https://docs.python.org>)

Fourth port of call: The Python docs (<https://docs.python.org>)

Fifth port of call: Us. Stop us and ask and we will do our best to help!

Asking for Help

1. Do you know what the problem is?
2. Do you know where the problem is?
3. What have you tried?
4. We can help you better if you have helped yourself first.

Still really stuck?

Sure?

Fine: Stack Overflow (<https://stackoverflow.com>)

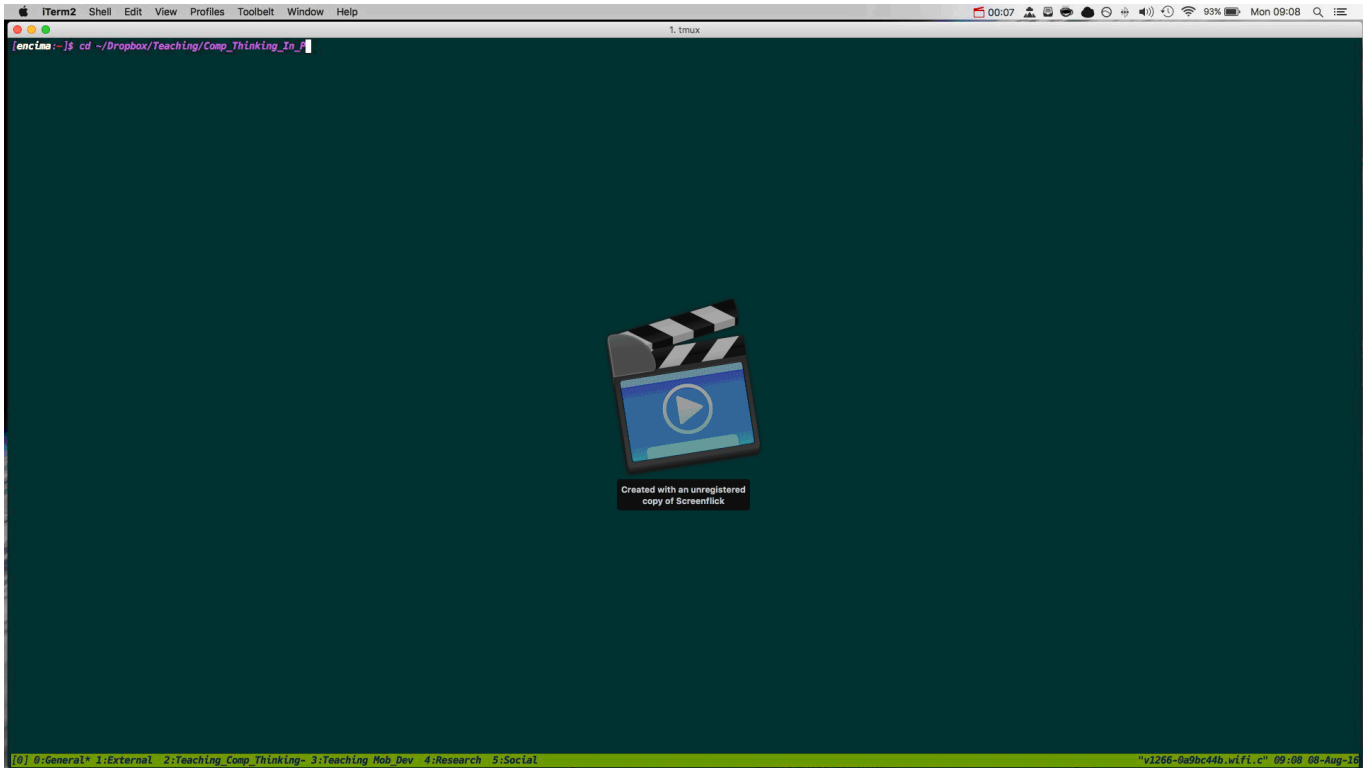
Beware, Python 3 is new and not all the code you find online will run. Any code written in Python 2.* will **not** be marked or accepted.

Jupyter Usage

The Jupyter notebook is a way of running code in the slides, providing answers to exercises and experimenting with any code samples provided. This first lecture was provided as a PDF but there is an `ipynb` file also included on Learning Central.

Every session will have at least a PDF and an `ipynb` file. In order to run that file (and interact with the code samples), you need to run:

- Open the directory you have the notebooks downloaded to in Command Prompt/Terminal
- Run the following command: `jupyter notebook`
- If it does not open automatically, point your browser to localhost:8888 (localhost:8888/tree)
- Click on the relevant notebook and run the code blocks with the Play arrow or by pressing Ctrl-Return



Let's Get Hands On!

Below, there are a few lines of Python code. Fill them in and we can go through what they mean.

```
language = "" #insert the language we are coding in here
print(language)
```

Here, you have set the variable `language` to the value `Python`. What do you note that is different here over a language like C#? Give me 2 differences.

```
char x = 'y';
Console.WriteLine(x);
```

C# requires you write the type of the variable as well as the name. Why do you think Python doesn't?

There is a semicolon at the end of the line. Python relies on whitespace instead of expression terminators.

Finally, Python calls `print` while C# calls `Console.WriteLine`. A chocolate bar for anyone that can tell me the difference:

Python has `print` as a function but `WriteLine` is a method attached to the `Console` Object.

Do not worry, all of this will be covered in depth in the course.

Fin

OK, that serves as your intro to the module. Ask questions now or forever hold your peace.

Homework

Download the 1_Homework and follow the instructions in this lecture to view it in your browser.

Complete all sections **BEFORE** the next session!

Instructions for submission:

- Open the homework notebook
- Complete the exercises
- Save the file with your student number: 1_Homework_C111111