Volunteer resources

This is a new course, and unfortunately may have bugs, issues and problems. If you have ideas, comments or would like to help, come and join us on github https://github.com/CodeClub/python-curriculum

You can also email tef, (tef@codeclub.org.uk) too. We've tested the coursework, but your feedback will be invaluable for making it better, easier and more fun.

Installing python

Although a little daunting, the python download page at http://www.python.org/download/ has everything you need to get started.

We've suggesting Python 3 over Python 2, and you will need to know which operating system and hardware to use.

An all too fast introduction to Python

Python is a scripting language, with variables and objects. Objects can be numbers, `1` or strings `"abc"`, or lists `[ 1,2,3]`, or dictionaries `{'one': 123, 'two': 123}`. Variables don't need to be explicitly defined, just assigned to be used.

```
x = 1
x = x + 2
y = [ 1, 2, 3]
```

Python uses whitespace to delimit blocks of control, in functions, if, for and while loops:

```
def foo(a,b,c):
    return a * b * c

if foo(1,2,3) < 8:
    print("Hello")
else:
    print("Nope")

x = 10
while x > 0:
    x = x -1
    print(x)
```

```
for x in range(10):
    print(x)
```

One thing that will trip people up is that python variables don't store the values themselves, but a reference to the value.

```
x = [ 1,2,3 ]
y = x
y.append(4)
print(x) # prints 1,2,3,4
```

Don't worry if this seems a little confusing, we're trying to stay away from the more awkward bits of python for the introductory lessons. If you're comfortable with scratch, you should be able to start playing with Python.

There is much more to python than this, and the offical python tutorial is a good place to get started.

- http://docs.python.org/3/tutorial/introduction.html
- http://docs.python.org/2/tutorial/introduction.html

Python Problems

We don't touch on any advanced or intermediate features in this course so far, and the majority of problems we've encountered from testing are as follows:

1. Missing ':' at end of `if`, `def`, `for`

   Everyone does this. I do this. This is probably the first syntax error people will encounter

2. Using `-` instead of `_`.

   `-` is an operator and can't appear in variable names

3. Missing or wrong indentation.

   Python is very pedantic about whitespace. Tabs count as 8 spaces, but thankfully IDLE will convert tabs into 4 spaces (the python coding standard, so mixing them
   shouldn't come up). We've tried to cover indentation as much as possible without boring the students to death

4. Other classic syntax errors

   Misspelled names, missing or excess punctuation.

5. Forgetting to open a new window in IDLE

   We've repeated this instruction a couple of times, but students need two windows open in IDLE to begin
   and occasionally start writing in the output window

6. The turtle window crashes after the program finishes running

   This is ok, the student can just rerun the program and it quits and restarts.

7. The student uses a file name with the same name as a library

   For example, `turtle.py` will break if you have `from turtle import *`

Python 2 vs Python 3

The course currently suggests using Python 3, but none of the examples depend on Python 3 features. All the code should work under Python 2, although there are some minor caveats.

The Print statement

In python 2, print is a keyword, and so is normally written `print "foo"`, but in Python 3, print is an ordinary function and usually written `print("foo")`. Due to syntactic quirks, the python 3 style works in python 2, but it may lead to some confusion if a trailing comma appears, i.e `print("foo",)` works differently in Python 2 and 3.

In python 2, print("foo",) is the same as print(("Foo",)) in python three: printing a one element list containing "Foo".

Floating Point

In python 2, `1/2` is 0, as python assumes integer division. In python 3, `1/2` is 0.5. We've tried to avoid relying on one behaviour or the other, and for python 2 this can be remedied by using floating point, i.e `1.0/2` or `1/2.0`.

Other Python Courses

Learn Python the Hard Way
http://learnpythonthehardway.org/

Introduction to Computer Science 101
https://www.udacity.com/course/cs101

Codecademy python track
http://www.codecademy.com/tracks/python

Dive into Python
http://www.diveintopython.net/

This might be relevant to your interests

Cheat sheet for translating Scratch blocks to python commands
https://docs.google.com/document/d/1PExbraZ6a1yK7EG0M8nHWfIuQz7-SL8XZOmPmpF05Uo/pub