Week 1
The basics

Running your first program

To run your first Python program, follow the following steps:

1. Open IDLE.
2. Click File → New File.
3. In the editor window that appears, type:

```
print("Hello World")
```

```
Hello World
```

4. Click Run → Run Module.

Arithmetic

1. In the editor window, type the following then run it.

```
print(23+2)
print(23-2)
print(23*2)
print(23/2)
```

```
25
21
46
11.5
```

What do the operators `+`, `-`, `*` and `/` do? (The meaning of these operators is given below)

WARNING

In Python 2, the `/` operator used to behave differently. If you are sharing code with someone using Python 2, you should write

```
from __future__ import division
```

at the top of your file. This will change Python 2's `/` operator to behave like it does in Python 3.

2. In the editor window, type the following then run it.

```
print(23**2)
print(23//2)
print(23%2)
```

What do the operators `**`, `//` and `%` do? If you're not sure, try changing the numbers either side of them to work it out. (The meaning of these operators is given below)

WARNING

`3^2` is not 3 to the power of 2 in Python.

3. In the editor window, type the following code and replace each `$` with one of `+`, `-`, `*`, `/`, `**`, `//` or `%`.

```
print(1 $ 2 $ 3 $ 4 $ 5)
```

Which operators do you need to replace the `$`s with to make Python print:

a. 15	**c.** 12.5	**e.** 0	**g.** 1
b. 5!	**d.** 12	**f.** 10	**h.** 3.5

Python can also deal with complex numbers:

```
print((1+2j)**2)
```

```
(-3+4j)
```

The following list gives the meaning of the operators used in this section:

- `+` Add
- `-` Subtract
- `*` Multiply
- `/` Divide
- `//` Integer division (divide then round down)
- `%` Modulo (the remainder when divided by)
- `**` Exponentation (to the power of)

`math`

There is a Python module called `math` that contains many mathematical functions and constants. You can import it by typing:

```
import math
```

You can then use all its functionality, for example:

```
import math
print(math.cos(math.pi/3))
print(math.log(3))
print(math.floor(3.5))
```

```
0.5000000000000001
1.0986122886681098
3
```

If you are working with complex numbers, you should import `cmath` instead:

```
import cmath
print(cmath.e**(1j*cmath.pi))
```

```
(-1+1.2246467991473532e-16j)
```

We will look at modules and importing in more detail later in the course.

Variables

In Python, you can define a variable by writing:

```
a = 3
```

You can then use this variable in calculations or to define other variables, for example:

```
print(a+10)
b = a*5
long_variable_name = b**a
a = a+1
```

```
13
```

The last example above sets the new value of `a` to be the old value of `a` plus 1. This is a very useful operation: so useful that there is a shorthand way to do it:

- `a += 1` is a shorthand way of writing `a = a+1`
- `a -= 5` is a shorthand way of writing `a = a-5`
- `a *= 2` is a shorthand way of writing `a = a*2`

This shorthand also works for the operators `/`, `**`, `//` and `%`.

4. You are given the following lines of Python.

```
a = 3
```

```
b = a
```

```
a += b
```

```
a -= b
```

```
b *= a
```

```
a *= -1
```

```
print(a)
```

```
print(b)
```

Re-order the lines to make scripts that print the following. (You do not always need to use all of the lines.)

a. 3 **c.** -3 **e.** 0
b. 6 **d.** -6 **f.** `NameError`

Conditions: `if`

You can set a variable equal to `True` or `False`, then use the logical operators `and`, `or` and `not` to combine them.

```
a = True
b = False
print(a or b)
print(a and b)
print(b or not b)
```

```
True
False
True
```

You can also use the comparision operators `<`, `>`, `<=`, `>=`, `==` and `!=` to compare numbers.

```
print(4 == 4)
print(4 == 3)
print(3 > 4 or 8 != 5)
```

```
True
False
True
```

You can use the `if` command to only run lines of code if something is `True`.

```
a = 3
if a%2 == 0:
    print("a is even")
```

The line(s) after the `if` that are indented will only run if `a%2 == 0`.

The above code will print nothing, because the remainder when a is divided by 2 (`a%2`) is not equal to 0. The code below, however, will print “a is even”.

```
a = 4
if a%2 == 0:
    print("a is even")
```

```
a is even
```

You can use `elif` and `else` to string together chains of `if`s:

```
a = 15
if a%2 == 0:
    print("a is even")
elif a%3 == 0:
    print("a is odd and a multiple of 3")
else:
    print("a is odd and not a multiple of 3")
```

```
a is odd and a multiple of 3
```

5. Write Python code that prints one of “a is a multiple of 3”, “a is a multiple of 5”, “a is a multiple of 3 and a multiple of 5” or “a is not a multiple of 3 or 5”.

Test your code with multiple values of `a`.

Loops: `for` and `while`

A `for` loop will run the indented lines of code following it with the variable taking every value in the given iterator. The most commonly used iterator is `range`: `range(a,b)` will include all the integers from `a` up to (and not including) `b`. `range(b)` is the same as `range(0,b)`.

6. The following code prints the square numbers between 0 and 100 (inclusive).

```
for i in range(11):
    print(i**2)
```

```
0
1
4
9
16
25
36
```

```
49
64
81
100
```

Adapt this code to:

a. Print the square numbers between 1 and 400.

b. Print the cube numbers between 1 and 1000.

c. Print the square numbers between 1 and 1000.

A `while` loop will keep running the indented lines as long as its condition is `True`.

7. The following code also prints the square numbers between 0 and 100 (inclusive).

```
i = 0
while i**2 <= 100:
    print(i**2)
    i += 1
```

```
0
1
4
9
16
25
36
49
64
81
100
```

Adapt this code to:

a. Print the square numbers between 1 and 400.

b. Print the cube numbers between 1 and 1000.

c. Print the square numbers between 1 and 1000.

8. In the previous two questions, was it easier to use `while` or `for`?

Controlling loops

There are two commands that you can use to control loops: `break` and `continue`. If the code encounters a `break`, then it will exit the innermost loop that it is in. If the code encounters a `continue`, then it will move on to the next run through the innermost loop that it is in.

In the following code, the loop will repeat until `i` reaches 15: when `i` is 15, `i` will be printed then the loop will be broken and the code will end.

```
for i in range(1,1000):
    if i%3 == 0 and i%5 == 0:
        print(i)
        break
```

```
15
```

In the following code, the loop will print all the numbers that are not multiples of three. When `i` is a multiple of three, the `continue` is run: this causes the loop to continue from to top with the next value of `i`.

```
for i in range(1000):
    if i%3 == 0:
        continue
    print(i)
```

```
1
2
4
5
7
...
```

When writing code, the `pass` command may be helpful. If you try running the following code, you will get an error because Python cannot find anything indented after the `if`.

```
for i in range(1000):
    if i%3 == 0:

    else:
        print(i)
```

```
IndentationError: expected an indented block
```

The `pass` command does nothing, but Python can see that it is indented, so the following code will run fine.

```
for i in range(1000):
    if i%3 == 0:
        pass
    else:
        print(i)
```

```
1
2
4
5
7
...
```

`pass` is mostly useful as a placeholder when you want to run the code you've written so far but not finished.

When writing loops, you may accidentally create an infinite loop, for example:

```python
i = 1
while i**2 <= 100:
    print(i**2)
```

```
1
1
1
1
1
1
...
```

If this happens, you can press CTRL+C to kill the Python that is currently running.

Putting it all together

Using all the Python commands you have learned this week, do the following questions.

9. Write a Python script that prints out all the numbers between 1 and 100 that are multiples of 3 or 5 (or both).

10. Write a Python script that prints out all the numbers between 1 and 100 that are multiples of 3 or 5 (but not both).

11. The temperature in degrees Fahrenheit (T_F) and in degrees Celsius (T_C) are related by the formula

$$\frac{9T_\text{C}}{5} + 32 = T_\text{F}.$$

Write a Python script that converts a temperature from Celsius to Fahrenheit. Use the data (given to the nearest degree) in the following table to test your code:

Celsius	0	28	100
Fahrenheit	32	82	212

12. Write a Python script that prints out all the prime numbers between 1 and 100.

13. It is known that

$$\pi = \sqrt{\frac{6}{1^2} + \frac{6}{2^2} + \frac{6}{3^2} + \frac{6}{4^2} + \frac{6}{5^2} + \ldots}$$

Write a Python script that computes this with the first 10 terms in the series. Compare the result to the exact value of π.

Adapt your Python script to include the first 100 terms. How much closer is the result?

Assessed question

When you have completed this question, you should show your code to one of the helpers in your class.

14. It is known that

$$\ln 2 = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \frac{1}{7} - \ldots$$

Write a Python script to add up the first 1000 terms in this series.

To check your code, compare it to the actual value of $\ln 2$:

```python
import math
print(math.log(2))
```