

Poynting's Python Society

W3 6/10/23 – While, For Loops and a bit of Logic

Today, we want to get you to work through the first [Project Euler](#) problems.

Project Euler is a website that *archives over 800 problems* of varying difficulty, that are intended to be solved on a computer. The puzzles start off nice and easy (finding factors, generating Fibonacci sequences etc.), but become increasingly difficult...

They provide really good practice *working with Python data types* (e.g. strings, lists, operators).

I can guarantee: *getting comfortable working with Python data types makes your life SOOOO much easier.*

If you practice working with the basic data types regularly, you'll find the coding assignments much much easier later on.

You can make a blank Jupyter notebook to code, but we recommend you use Spyder. It's an enhanced editor, where the focus is on writing Python code and not normal text. It will be more convenient for solving Project Euler problems, and it's important for you to get familiar with scripts. All university computers here have Spyder downloaded, and you can easily install it on your own laptop using Anaconda, in a similar way to installing Jupyter.

We'd like you to try Problems 1, 2 and 4 – although you're welcome (and encouraged to) work on these in your own time.

Here's problem 1:

PYSOC LXD160@BHAM.AC.UK

Multiples of 3 or 5

Problem 1



If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Find the sum of all the multiples of 3 or 5 below 1000.

Answer: **233168**

Completed on Sat, 20 Apr 2019, 22:42

[Go to the thread for problem 1 in the forum.](#)

[Download overview for problem 1.](#)

Nice and easy concept – no way you'd solve this with pen and paper: *computers are useful!!!*

General Tips:

- Keep pen and paper by you: learning Python is like learning a language; if you wanted to greet someone in Spanish – you’d have to know you’re going to say in English before you translate! Having pen and paper by you helps transform your intentions into commands you can write in Python
- For problem 1, remember the ‘mod’ operator `%` exists. E.g. `5%2=1` – it returns the REMAINDER of the division.
- For problem 4, remember that Python strings can be ‘indexed backwards’ by using `[::-1]`

```
In [52]: myString = 'hello'

In [53]: myString[::-1]
Out[53]: 'olleh'

In [54]: |
```

As a reminder, here’s the syntax for while and for loops:

‘<variable>’ in square brackets, is just a placeholder for a temporary variable that can be used inside an indented code block. To write some executable code, replace <variable> and <iterable> with e.g. `integer` and `range(1, 10)` – this will iterate through all the integers between 1 and 10. Note: it doesn’t matter what we use for <variable> - it could just be a letter e.g. `j` or even just an underscore: `_`

```
for <variable> in <iterable>:
    #### Code block here indented

# Leave the code block, going back to zero indentation
```

For a while loop:

```
while <condition>:
    ####
```

You’ll always enter the code block, so long as the <condition> evaluates to True. A possible condition could be `number % 2 == 0`, then so long as ‘number’ is an even number, then this will evaluate to True, and you’ll enter the code block!

Please don’t hesitate to ask for help!!!