CSCA08 Exercise 1

Due: September 24, 2017. 5:00pm
Pre-Run: September 22, 2017. 5:00pm

We're still starting slow, but this time you actually have to write some real code. It's important that your file and function names are **exactly** as written here, otherwise the automarker won't be able to find them, and you will receive a mark of zero. We are providing sample input/output for the shell so you can make sure you're following our instructions exactly.

In a file called `ex1.py`, you will write 3 functions. You should have no global code (i.e., no code outside of the functions), and nowhere in your code should you use `print`, `input` or `import`.

We will do a pre-run on Friday night of any code that was submitted before 5:00pm on Friday. That way you get one chance to try out your code and fix any mistakes before Sunday.

Function 1: A useless function

Create a function called `useless`, that takes 3 parameters of any type, does nothing with them, and returns the string `That was a waste of time` [1]. Running your code in the shell should look like this:

```
>>> useless(1,2,3)
'That was a waste of time'
>>> useless("Hello","Goodbye",999)
'That was a waste of time'
```

Function 2: Squaring a number

Create a function called `square_me`, that takes 1 number (int or float) parameter and returns the square of that number as a result (you don't have to worry about what it does if something other than a float/int is passed in). Your code should behave as follows:

```
>>> square_me(3)
9
>>> square_me(9.5)
90.25
```

[1] Be careful, capitalization, spacing and punctuation count here

Function 3: Biographical Data

Create a function called student_data, that takes 4 parameters, a name (a string), age (an integer), student number (a string) and whether they are enrolled in CSCA08 (a boolean), and returns a string containing that information in the following format: <student number,name,age,enrolled>[2]. It may be helpful to remember that you can cast a number or a boolean to a string using the str function. Your code should work as follows:

```
>>> student_data("Brian",35,"1234567",False)
'<1234567,Brian,35,False>'
>>> student_data("Nick",97,"0000001",True)
'<0000001,Nick,97,True>'
```

[2] no spaces around the commas