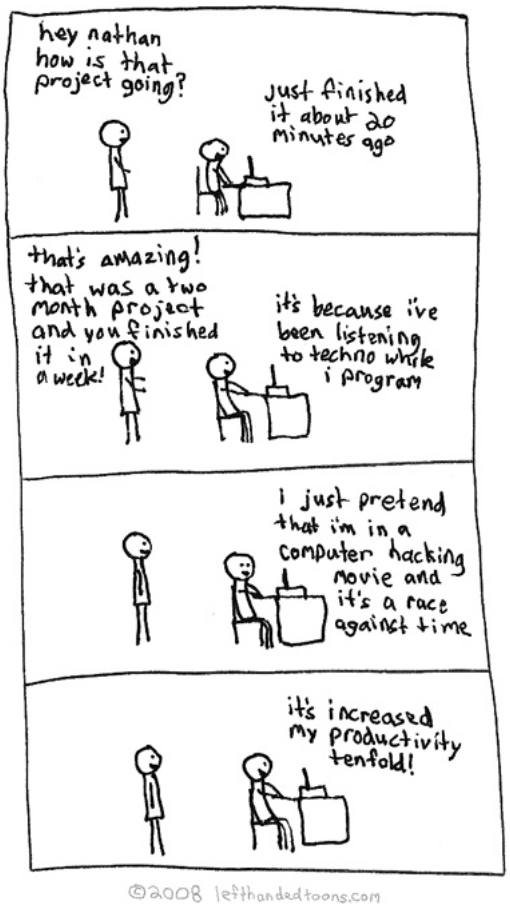
Introduction to Programming Using Python–Assignment 1 Functions

Due Wednesday, July 5th, 2023



http://www.lefthandedtoons.com/245/

1 Initial Setup

Make sure that PyCharm is installed according to the installation instructions. Open up PyCharm if it's closed. If you see the welcome screen again, use the `Create New Project` button, or else pick `New Project` from the `File` menu. You can call this one `assign1`; make sure its path is inside of your `pays2023` folder (which is on your Desktop folder) and that it uses a Python 3 base

environment. After you create the project, right click the project name `assign1` in the left pane, select `New` and click `Python File` to create the Python source file (or you could use `File/New...` while `assign1` is selected in the project tab). This new file should be named as `assign1.py`.

2 Your first function [3 points]

Define a function called `sphere_surface_area` that takes the radius as a parameter and returns the surface area of a sphere with that radius.

Remember the basic structure for defining a function is:

```
def function_name(parameter1, parameter2, ...):
    statement1
    statement2
    ...
    return something # Not all functions will have return statements!
```

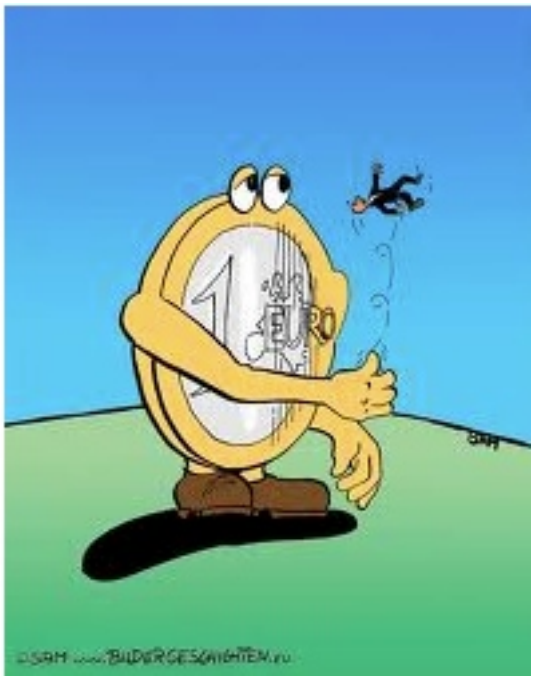
Remember that the way that Python can tell what is part of the function is based on the indentation.

After you've run your program (be sure you run it in the Python Console—see the Lab 1 handout for a reminder), you can still interact with the Python shell! For example, you can click into the Python Console and type:

```
>>> sphere_surface_area(12.4)
1932.2035136000002
```

Once you're comfortable with running Python programs with your own defined functions and with playing with them in PyCharm, move on to the main part of this assignment. If you can, it's worth getting used to using a keyboard shortcut to run your program so you don't need to switch over to the mouse so often.

3 A semester abroad in Europe



http://www.bildergeschichten.eu/euro_cartoon_witz.htm

You're going to do a semester abroad in Europe and have decided to write a few functions that will help you out with some common questions you might find yourself asking while you're there.

1. [**2 points**] Write a function called `euros_to_dollars`, with a single parameter, the price in euros, and the function will return the price in dollars. Lookup the current price exchange from euros to dollars online. For example, after running your program you could type:

   ```
   >>>  euros_to_dollars(13.5)
   15.4
   ```

   (Note: depending on the exchange rate you use, your value will be slightly different)

   Make sure that you are using the `return` statement and not printing the answer in your function. In particular, try running the following and make sure you get something identical:

   ```
   >>> dollars =  euros_to_dollars(13.5)
   >>> dollars
   15.4
   ```

2. [**2 points**] Write a function called `welcome` that doesn't take any parameters and returns "welcome" in some European language (English doesn't count!). For example:

```
>>> welcome()
'te lutem'
```

Remember that you can create functions with zero parameters. To call a function without any parameters, you still need to put the parenthesis at the end.

3. [**2 points**] Write a function called `kilometers_to_miles` that takes as input the number of kilometers and returns the distance in miles. For example, after running your program you could type:

```
>>> kilometers_to_miles(100)
62.137
```

4. [**3 points**] Write a function called `mpg_from_metric` that takes *two* parameters: first the number of kilometers and second the number of liters. The function should return the miles per gallon (i.e. miles divided by gallons) by converting the kilometers and liters appropriately. Remember, to have multiple parameters for functions, you separate them with commas.

```
>>> mpg_from_metric(400, 30)
31.358472666666668
```

5. [**2 points**] Write a function of your own that takes one or more parameters and returns something interesting/useful. Think of differences in distances, weights, volumes, speeds, recipes, etc. Brownie points for creative functions :)

6. **Extra credit** [**1 point**]: The right way to calculate the `mpg_from_metric` function is to utilize other functions you write that do some of the work for you. To calculate the mpg, write an additional function `liters_to_gallons` and then use this function and your `kilometers_to_miles` function inside the `mpg_from_metric` function.

7. [**2 points**] Write a function called `kilograms_to_pounds` that takes as input a weight in kilograms and returns the equivalent weight in pounds as a number rounded down.

```
>>> kilograms_to_pounds(13)
28
```

Hint: You can use either the `//` operator or the `int` function.

8. [**3 points**] Write a function called `kgs_to_lbs_and_oz` that takes as input a weight in kilograms and returns the weight in pounds and ounces as a string. The ounces calculated should never be more than 15.

```
>>> kgs_to_lbs_and_oz(13)
'13 kilograms is 28 lbs and 10 oz.'
```

Advice

- Work through a few examples on paper to make sure you understand the calculations.
- Use an online tool to double check your answers.
- Break the calculation down into different steps and use variables to store these intermediary values.

4 Commenting and documentation

An important part of programming is making sure that your code is well documented so that when someone else looks at it (or, more likely, you look at it a month later) you can get information about the program without having to read all of the code.

You should have the following in your program:

- Comments at the very beginning of the file stating your name, course, assignment number and the date.
- Other miscellaneous comments to make things clear. Don't go overboard, but do include comments where things are complicated or if you have a block of code that does something. For this assignment, you won't need much since the functions should all be pretty short.

In addition, make sure that you've used good *style*. This includes:

- Following naming conventions, e.g. all variables and functions should be lowercase.
- Using good variable names.
- Proper use of whitespace, including indenting and use of blank lines to separate chunks of code that belong together.

5 When you're done

Submit your `.py` file online using Gradescope; be sure you're uploading them to the "assign1" assignment. Note that you can resubmit the same assignment as many times as you would like up until the deadline.

6 Grading

	points
`sphere_surface_area`	3
A semester abroad	16
comments and code style	3
file submitted correctly	1
extra credit	1
total	23 (+1)