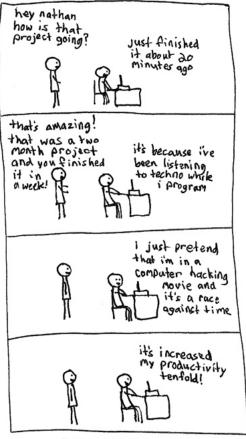# CS51A - Assignment 1

Due Thursday Jan. 31, at 11:59pm

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

1 Part B: A semester abroad in Europe [11 points]

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

For the main part of this assignment, you will write some more functions. Create a new file called *[first_name][last_name]-assignment1B.py.*

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)
   15.34
   ```

   (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
   15.34
   ```

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. 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: Don't forget about the `//` operator.

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.

2   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.

3   When you're done

Submit your .py file online using the courses submission mechanism under "assign1b". Note, you can resubmit the same assignment as many times as you would like up until the deadline.

4 Grading

	points
Part A (from lab 1)	3
Part B	16
comments and code style	3
extra credit	1
total	22 (+1)