CIS 122 Fall 2015 Project 8
Due Friday, December 4, 11:59 PM

Briefly
Your programs are worth a total of 30 points

Test your programs -- did they work right? -- before uploading to Canvas.

P8_CountWords.py 15 points

Learning Objectives:
Work with text, break each line up into a series of words
Count how many times each word is used, keeping a words_to_counts dictionary of your results.

The file
For quick testing and debugging
Canvas > Files > Projects > Week10 > **alice0.txt**
For final submission
Canvas > Files > Projects > Week10 > **alice1.txt**

If you prefer, you can get the files from the web
"**http://www.cs.uoregon.edu/Classes/15F/cis122/data/alice0.txt**"
and
"**http://www.cs.uoregon.edu/Classes/15F/cis122/data/alice1.txt**"

Rubric 15 points

10 points
Your program correctly keeps counts of how often each word is used, working with each word converted to lower case, and prints a report of the words and their frequency of use
-1 words not converted to lower case
-4 input line of text not split on blanks
-5 no report of words and frequency of words or incorrect counts

1 point Words are sorted alphabetically in the report of words and how many times they are used.

4 points Words counts data stored using a dictionary

Hint: This assignment is similar to the birds_to_counts dictionary work, but each line typically has a number of words not just one.

Here's a sample of the text - it contains no punctuation, but does have both upper and lower case (ADVENTURES, sitting) words. You will need to convert the text to all lower-case.

```
ALICES ADVENTURES IN WONDERLAND

Lewis Carroll

CHAPTER I Down the Rabbit-Hole

Alice was beginning to get very tired of sitting
by her sister on the bank and of having nothing
to do once or twice she had peeped into the book
her sister was reading but it had no pictures or
conversations in it and what is the use of a
book thought Alice without pictures or
conversations
```

Running your program, your report will look somewhat like this (shown using words from alice0.txt)

```
1 a
1 adventures
2 alice
1 alices
2 and
1 bank
1 beginning
2 book
1 but
```

P8_FunctionCall.py 15 points

Learning Objective

Refresh your understanding of how to call a function that returns a value. Review the rules for calling, and for using a return value.

Notice that you **do not** need to use the same variable names when you call a function as you use in the 1st line of the function definition.

```
def calc_wind_chill(f_temp, wind_speed):
    """ float, float -> float
        returns wind_chill_temperature (F)
        given farenheit temperature and
        wind_speed in miles per hour
    """
    velocity = wind_speed ** 0.16
    chill = 35.74 + (0.6215 * f_temp) - \
            (35.75 * velocity) +        \
            (0.4275 * f_temp * velocity)
    return chill
#end def
```

Note: the wind chill function is only valid for winds over 3 mph, and for temperatures not higher than 50 degrees F.

Your program will ask whether to calculate wind chill or quit; it will continue to ask for temperature and wind speed until the user chooses to quit.

Ask a user to enter 2 values

Temperature (F or Fahrenheit)

Wind speed (in miles per hour)

Assign the temperature to a variable **temperature**

Assign the wind speed to a variable **wind**

Call the **calc_wind_chill** function using **temperature** and **wind**; assign the result to a variable **wind_chill**

Also:

Call the **calc_wind_chill** function using **temperature** and the **expression wind + 10** ; assign the result to a variable **wind_chill_2**

Example of wind chill user interaction

```
Calculate wind chill (c), or quit (q): c
Temperature in degrees F: 20
Wind speed in miles per hour: 5
Wind chill 13.0 F  20.0 F  wind speed 5.0 mph
Wind chill 6.2 F  20.0 F  wind speed 15.0 mph
```

```
Calculate wind chill (c), or quit (q): c
Temperature in degrees F: 20
Wind speed in miles per hour: 10
Wind chill 8.9 F  20.0 F  wind speed 10.0 mph
Wind chill 4.2 F  20.0 F  wind speed 20.0 mph
```

This continues until your user chooses **q** to quit.

Rubric 15 points total

9 points

You use the exact names described earlier for the calc_wind_chill function's parameters

```
def calc_wind_chill(f_temp, wind_speed):
```

and for its `return chill` final statement

and you use different names for both the temperature and wind speed when calling the function and for saving the return value.

```
wind_chill = calc_wind_chill(temperature, wind)
wind_chill_2 = calc_wind_chill(temperature, wind + 10)
```

The whole point of this review exercise is to emphasize that the names a function uses internally for the 2 arguments you provide can differ from the variable names you provide when calling the function:

The names between parentheses ( ) in a calldon't have to be the same as what the function uses.

Expressions like `wind + 10` instead of variable names' are also OK in a call to a function.

3 points

You have a **while** loop that asks whether to calculate or quit, and asks for **temperature** and **wind speed** when you choose to calculate.

The while loop must **allow you to quit** and not get stuck in a loop.

3 points

Your program asks for temperature and converts it to a float, and similarly for wind speed.

Your program then calls the function using those values.

Finally, your program prints the results, somewhat like the example shown earlier.

Optional Alternate Assignment

Going on to CIS 210 next term?

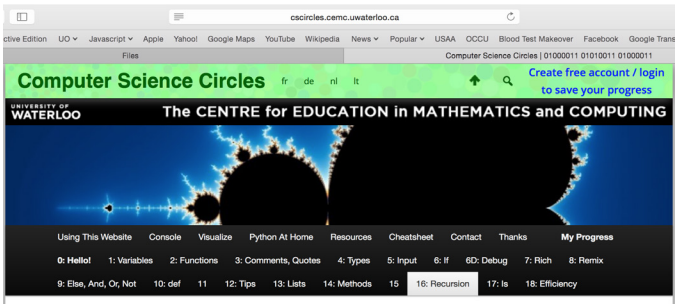
Submit this, or study it on your own during winter break.

Recursion

Earlier in the course, you wrote a function **draw_triangle(size)**

then you defined a **color_triangle(size, color)** which set up a fillcolor and began a fill, then needed to draw a triangle and did so by a call to **draw_triangle(size)**.

http://cscircles.cemc.uwaterloo.ca has a nice introduction to "recursion" – a function calling itself (instead of calling some other function).



Best bet: work through all the examples.

For credit:

P8-DigitalSum.py 5 points Function works;
digital sum of 2019 is 12:
2 + 0 + 1 + 9

P8-DigitalRoot.py 10 points
digital root of 2019 is 3
2 + 0 + 1 + 9 is 12
summing further 1 + 2 is 3

Note that digital root depends on having a well-defined digital sum function available.

P8-Hailstone.py 5 points

P8-NestedListSum 10 points x = [[4,[5,6]], 3, [[[5]]]]
ans = NestedListSum(x) # 23

P8-DigitalRuler.py 3 points XC

As you code and test a function, use the Visualizer to watch your program in action.

You do not necessarily need to submit these projects, but I do recommend doing them on your own **before** *getting in to CIS 210; they will help you get prepared for the more technically advanced topics and faster pace of that class.*