% COMP 1012 Summer 2020 -- Assignment 1

COMP 1012 Summer 2020 -- Assignment 1

Instructors: Sheikh Jubair (A01)

Due on **Friday, July 3rd at 11:30pm**.

Outcomes

- Using variables and literals
- Evaluating simple expressions
- Using the `math` module
- Printing formatted output
- Basic file I/O
- for loop
- list

Description

In this assignment, you will TBD.

Notes

- You **don't** need to use `input()` to get started on this assignment. Use the values shown in the sample sessions to get your solution started, then add input *later*.
- Many of the calculations require you to convert units (from millimeters to meters, from millimeters to inches, from hours to minutes to seconds). Make sure that you're converting the values *before* doing the calculation!
- Use 2 decimal places of precision for speed and 4 decimal places of precision for all other values in your output.
- **Do not** create your own value for π. Use the constant that comes from the `math` module.
- Part 2 covers the same outcomes as part 1, but adds complex math functions from the `math` module. While you are encouraged to complete part 2 for practice, completing part 1 shows proficiency at the outcomes defined above.

"Programmed by..."

Use the following code snippet at the **end** of your script to print out the time and end of processing message:

```
import time
print ("\nProgrammed by ******YOU******")
print ("Date: " + time.ctime())
print ("End of processing")
```

Question 1: Required

In this question, you will calculate some statistics for a csv file named **reducedweather.csv**. The file contains tabular data which includes Manitoba daily minimum and maximum temperatures from Jan until Sept 17 in 2019. The first row is a header record (row), with headings **Max Temp** and **Min Temp**. After the header row, each row is a day where maximum and minimum temperature for that day is recorded. From this file, you need to calculate mean, standard deviation, highest and lowest temperature for each column.

Step 1 - Reading file:

The first step is to read the file. The file contains two columns. Even though we already know how many columns are contained in this file, the script that you produce for reading this file must be generic. This means it will work for a .csv file with any number of columns if the initial header record is given.

To accomplish this, you cannot hard-code the data structures to work with a fixed number of columns. You can use lists. For example, after reading the header row, you can split the header with comma. As the split function returns a list, that list will automatically contain all the header but in a separate index of the list.

```
headings = infile.readline().strip().split(',')
```

Once you have read the header record, you know how many columns are contained in the file (by using `len(headings)`). Now, you can create another list to hold the data. Suppose, this list is called `observations`. The `observations` list contains lists too. The number of list inside the `observations` list should be equal to the length of `headings`. So, each list inside the `data` corresponds to a column in the file.

To obtain this data structure, you may create an empty `observations` list first using `observations = []`. Then, you can use a for loop that will iterate `n` times (considering n is the length of `headings`) to append `n` empty lists inside `observations`. Then you can use another for loop to iterate through the rest of the data and append the data to the to appropriate list inside o`bservations`.

Step 2 - Calculating mean, highest and lowest temperature:

After creating the nested list, for each of the lists in the nested list find the mean. You can use the `sum(list)` function on each of the lists and then divide each sum by each list's length to get the mean. As there will be multiple means (the number of means will be equal to number of columns), you can store the means in a list too. You can also apply `max(list)` and `min(list)` functions on each list of the nested list to obtain highest and lowest temperature of each column. You can save the highest temperatures and lowest temperatures in two different lists.

Step 3 - Calculating Standard Deviation

After calculating the mean, you need to calculate the squared differences between the mean and each of the elements of each nested lists. You can create a new nested list to store the squared differences. After finding the squared differences, you can calculate the standard deviation. The formula for standard deviation:

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}}$$

$x_i = \text{ith observation}$
$\bar{x} = \text{mean of all observation}$
$n = \text{total number of observation}$

Step 4 - Printing Output:

The output needs to be nicely formatted in a tabular form like the sample output given below. The header of the table should be center aligned and the values under each column should be right aligned.

Data source: http://climate.weather.gc.ca/climate_data/daily_data_e.html?StationID=27174

Sample output:

```
Enter file name: reducedweather.csv

 Column Names |      Mean      | Std Deviation | Highest Score | Lowest Score
--------------+----------------+---------------+---------------+---------------
      Max Temp|          10.10|          16.91|          36.60|         -29.80
--------------+----------------+---------------+---------------+---------------
      Min Temp|          -2.20|          15.17|          21.30|         -39.90
--------------+----------------+---------------+---------------+---------------



Programmed By Instructor: Sheikh Jubair
University of Manitoba
Computer Science
```

Question 2: Optional (Don't hand in)

Social Insurance Numbers (SINs) are issued by the Canadian Government for administering various government programs. For example, a SIN is required for reporting of earned income from employment. A SIN consists of 9 digits (eg. 046454286). Not all 9-digit numbers represent a valid SIN. There is a simple validation process to determine if a 9-digit number is a valid SIN. Below, we use an example will illustrate how the validation process works.

Suppose the 9-digit number to be validated is 046454286. Take the number 121212121 and write it below 046454286 to get

046454286

121212121

Next, for each of the 9 digit positions, multiply the top and bottom digit to get

0 8 6 8 5 8 2 16 6

Next, add up all the digits together (note that 16 contains the digits 1 and 6) to get

0 + 8 + 6 + 8 + 5 + 8 + 2 + 1 + 6 + 6 = 50

If the sum from the previous line is evenly divisible by 10 (that is, no remainder), then the 9-digit number is a valid SIN. If the sum from the previous line is not evenly divisible by 10, then it is not a valid SIN. For the example 9-digit number 046454286, it is a valid SIN, since 50 is evenly divisible by 10.

Write a program that reads in a 9-digit number and determine if it is a valid SIN.

Try your program with various 9-digit numbers including 630308902, 666234872, 056123456.

Handing in

You **must** complete the Blanket Honesty Declaration checklist{target="_blank"} before you can submit your assignment.

An assignment submission area has been created for Assignment 1. Click on the "Assessments" menu, then click on "Assignments".

The only thing you're required to turn in for this assignment is the `.py` script that you create. Please name your script file with the following convention:

```
FirstnameLastnameA1Q1.py
```