Technological Institute of the Philippines	Quezon City - Computer Engineering
Course Code:	CPE 019
Code Title:	Emerging Technologies 2 in CpE
2nd Semester	AY 2023-2024

ACTIVITY NO. 3.1	**Data Analysis**
Name	Serrano, Vincent Gon B.
Section	CPE32s3
Date Performed:	2-7-2024
Date Submitted:	2-7-2024
Instructor:	Engr. Roman M. Richard

1. **Objectives**

Part 1: The Dataset Part 2: Scatterplot Graphs and Correlatable Variables Part 3: Calculating Correlation with Python Part 4: Visualizing

Scenario/Background

Correlation is an important statistical relationship that can indicate whether the variable values are linearly related.

In this lab, you will learn how to use Python to calculate correlation. In Part 1, you will setup the dataset. In Part 2, you will learn how to identify if the variables in a given dataset are correlatable. Finally, in Part 3, you will use Python to calculate the correlation between two sets of variable.

Required Resources

- 1 PC with Internet access
- Raspberry Pi version 2 or higher
- Python libraries: pandas, numpy, matplotlib, seaborn
- Datafiles: brainsize.txt

Part 1: The Dataset

You will use a dataset that contains a sample of 40 right-handed Anglo Introductory Psychology students at a large Southwestern university. Subjects took four subtests (Vocabulary, Similarities, Block Design, and Picture Completion) of the Wechsler (1981) Adult Intelligence Scale-Revised. The researchers used Magnetic Resonance Imaging (MRI) to determine the brain size of the subjects. Information about gender and body size (height and weight) are also included. The researchers withheld the weights of two subjects and the height of one subject for reasons of confidentiality. Two simple modifications were applied to the dataset:

1. Replace the quesion marks used to represent the withheld data points described above by the 'NaN' string. The substitution was done because Pandas does not handle the question marks correctly.

2. Replace all tab characters with commas, converting the dataset into a CSV dataset.

The prepared dataset is saved as brainsize.txt

Step 1: Loading the Dataset From a File.

Before the dataset can be used, it must be loaded onto memory. In the code below, The first line imports the pandas modules and defines pd as a descriptor that refers to the module. The second line loads the dataset CSV file into a variable called brainFile. The third line uses read_csv(), a pandas method, to convert the CSV dataset stored in brainFile into a dataframe. The dataframe is then stored in the brainFrame variable. Run the cell below to execute the described functions.

```
# Code cell 1
import pandas as pd
brainFile = '/content/brainsize.txt'
brainFrame = pd.read_csv(brainFile, delim_whitespace=True)
```

Step 2: Verifying the dataframe.

To make sure the dataframe has been correctly loaded and created, use the head() method. Another Pandas method, head() displays the first five entries of a dataframe.

```
# Code cell 2
brainFrame.head()
```

	Gender	FSIQ	VIQ	PIQ	Weight	Height	MRI_Count
0	Female	133	132	124	118.0	64.5	816932
1	Male	140	150	124	NaN	72.5	1001121
2	Male	139	123	150	143.0	73.3	1038437
3	Male	133	129	128	172.0	68.8	965353
4	Female	137	132	134	147.0	65.0	951545

```
<google.colab._quickchart_helpers.SectionTitle at 0x799d56ee8e20>

from matplotlib import pyplot as plt
_df_0['FSIQ'].plot(kind='hist', bins=20, title='FSIQ')
plt.gca().spines[['top', 'right',]].set_visible(False)

from matplotlib import pyplot as plt
_df_1['VIQ'].plot(kind='hist', bins=20, title='VIQ')
plt.gca().spines[['top', 'right',]].set_visible(False)

from matplotlib import pyplot as plt
_df_2['PIQ'].plot(kind='hist', bins=20, title='PIQ')
plt.gca().spines[['top', 'right',]].set_visible(False)

from matplotlib import pyplot as plt
_df_3['Weight'].plot(kind='hist', bins=20, title='Weight')
plt.gca().spines[['top', 'right',]].set_visible(False)

<google.colab._quickchart_helpers.SectionTitle at 0x799d56ee87c0>
```

```
from matplotlib import pyplot as plt
import seaborn as sns
_df_4.groupby('Gender').size().plot(kind='barh',
color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)

<google.colab._quickchart_helpers.SectionTitle at 0x799d56eebee0>

from matplotlib import pyplot as plt
_df_5.plot(kind='scatter', x='FSIQ', y='VIQ', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)

from matplotlib import pyplot as plt
_df_6.plot(kind='scatter', x='VIQ', y='PIQ', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)

from matplotlib import pyplot as plt
_df_7.plot(kind='scatter', x='PIQ', y='Weight', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)

from matplotlib import pyplot as plt
_df_8.plot(kind='scatter', x='Weight', y='Height', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)

<google.colab._quickchart_helpers.SectionTitle at 0x799d56eeb190>

from matplotlib import pyplot as plt
_df_9['FSIQ'].plot(kind='line', figsize=(8, 4), title='FSIQ')
plt.gca().spines[['top', 'right']].set_visible(False)

from matplotlib import pyplot as plt
_df_10['VIQ'].plot(kind='line', figsize=(8, 4), title='VIQ')
plt.gca().spines[['top', 'right']].set_visible(False)

from matplotlib import pyplot as plt
_df_11['PIQ'].plot(kind='line', figsize=(8, 4), title='PIQ')
plt.gca().spines[['top', 'right']].set_visible(False)

from matplotlib import pyplot as plt
_df_12['Weight'].plot(kind='line', figsize=(8, 4), title='Weight')
plt.gca().spines[['top', 'right']].set_visible(False)

<google.colab._quickchart_helpers.SectionTitle at 0x799d56eea200>

<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.
```

```
from matplotlib import pyplot as plt
import seaborn as sns
figsize = (12, 1.2 * len(_df_13['Gender'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(_df_13, x='FSIQ', y='Gender', inner='stick',
palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.
```

```
from matplotlib import pyplot as plt
import seaborn as sns
figsize = (12, 1.2 * len(_df_14['Gender'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(_df_14, x='VIQ', y='Gender', inner='stick',
palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.
```

```
from matplotlib import pyplot as plt
import seaborn as sns
figsize = (12, 1.2 * len(_df_15['Gender'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(_df_15, x='PIQ', y='Gender', inner='stick',
palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.
```

```
from matplotlib import pyplot as plt
import seaborn as sns
figsize = (12, 1.2 * len(_df_16['Gender'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(_df_16, x='Weight', y='Gender', inner='stick',
```

```
palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

Part 2: Scatterplot Graphs and Correlatable Variables

Step 1: The pandas describe() method.

The pandas module includes the describe() method which performs same common calculations against a given dataset. In addition to provide common results including count, mean, standard deviation, minimum, and maximum, describe() is also a great way to quickly test the validity of the values in the dataframe. Run the cell below to output the results computed by describe() against the brainFrame dataframe.

```
# Code cell 3
brainFrame.describe()
```

```
             FSIQ         VIQ        PIQ      Weight     Height
MRI_Count
count   40.000000   40.000000   40.00000   38.000000  39.000000
4.000000e+01
mean   113.450000  112.350000  111.02500  151.052632  68.525641
9.087550e+05
std     24.082071   23.616107   22.47105   23.478509   3.994649
7.228205e+04
min     77.000000   71.000000   72.00000  106.000000  62.000000
7.906190e+05
25%     89.750000   90.000000   88.25000  135.250000  66.000000
8.559185e+05
50%    116.500000  113.000000  115.00000  146.500000  68.000000
9.053990e+05
75%    135.500000  129.750000  128.00000  172.000000  70.500000
9.500780e+05
max    144.000000  150.000000  150.00000  192.000000  77.000000
1.079549e+06
```

Step 2: Scatterplot graphs Scatterplot graphs are important when working with correlations as they allow for a quick visual verification of the nature of the relationship between the variables. This lab uses the Pearson correlation coefficient, which is sensitive only to a linear relationship between two variables. Other more robust correlation methods exist but are out of the scope of this lab.

a. Load the required modules.

Before graphs can be plotted, it is necessary to import a few modules, namely numpy and matplotlib. Run the cell below to load these modules.

```
# Code cell 4
import numpy as np
import matplotlib.pyplot as plt
```

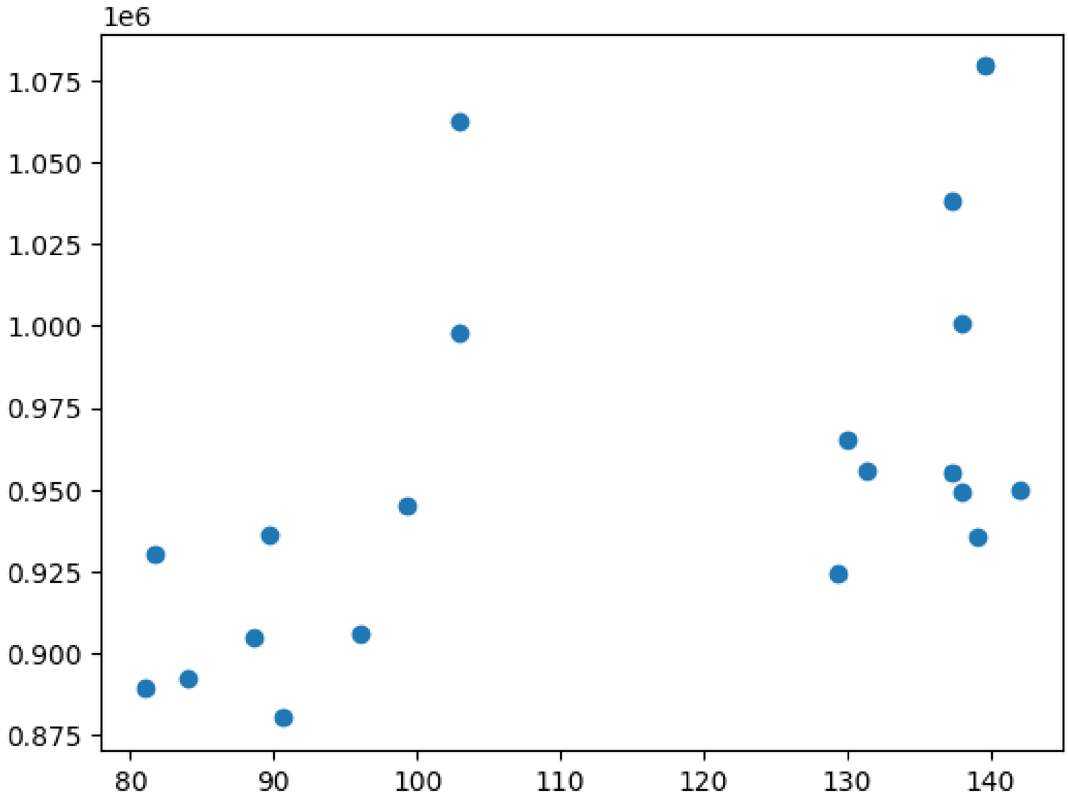
b. Separate the data. To ensure the results do not get skewed because of the differences in male and female bodies, the dateframe is split into two dataframes: one containing all male entries and another with only female instances.

Running the cell below creates the two new dataframes, menDf and womenDf, each one containing the respective entries.

```
#Code cell 5
menDf = brainFrame[(brainFrame.Gender == 'Male')]
womenDf = brainFrame[(brainFrame.Gender == 'Female')]
```

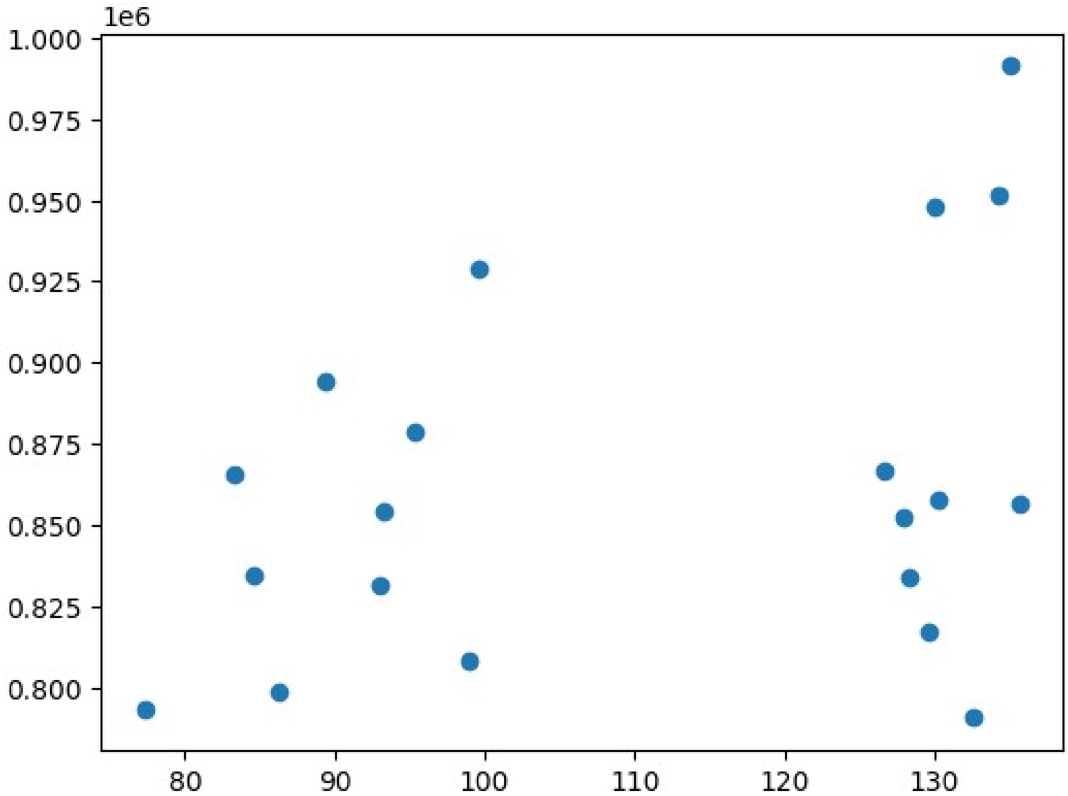
c. Plot the graphs. Because the dataset includes three different measures of intelligence (PIQ, FSIQ, and VIQ), the first line below uses Pandas mean() method to calculate the mean value between the three and store the result in the menMeanSmarts variable. Notice that the first line also refers to the menDf, the filtered dataframe containing only male entries. The second line uses the matplotlib method scatter() to create a scatterplot graph between the menMeanSmarts variable and the MRI_Countattribute. The MRI_Count in this dataset can be thought as of a measure of the physical size of the subjects' brains. The third line simply displays the graph. The fourth line is used to ensure the graph will be displayed in this notebook.

```
# Code cell 6
menMeanSmarts = menDf[["PIQ", "FSIQ", "VIQ"]].mean(axis=1)
plt.scatter(menMeanSmarts, menDf["MRI_Count"])
plt.show()
%matplotlib inline
```



Similarly, the code below creates a scatterplot graph for the women-only filtered dataframe.

```
# Code cell 7
# Graph the women-only filtered dataframe
womenMeanSmarts = womenDf[["PIQ", "FSIQ", "VIQ"]].mean(axis=1)
plt.scatter(womenMeanSmarts, womenDf["MRI_Count"])

plt.show()
%matplotlib inline
```



Part 3: Calculating Correlation with Python

Step 1: Calculate correlation against brainFrame. The pandas corr() method provides an easy way to calculate correlation against a dataframe. By simply calling the method against a dataframe, one can get the correlation between all variables at the same time.

```
#Code cell 8
brainFrame.corr(method = 'pearson')
```

```
<ipython-input-11-2e6e27112e97>:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  brainFrame.corr(method = 'pearson')
```

	FSIQ	VIQ	PIQ	Weight	Height	MRI_Count
FSIQ	1.000000	0.946639	0.934125	-0.051483	-0.086002	0.357641
VIQ	0.946639	1.000000	0.778135	-0.076088	-0.071068	0.337478
PIQ	0.934125	0.778135	1.000000	0.002512	-0.076723	0.386817
Weight	-0.051483	-0.076088	0.002512	1.000000	0.699614	0.513378
Height	-0.086002	-0.071068	-0.076723	0.699614	1.000000	0.601712
MRI_Count	0.357641	0.337478	0.386817	0.513378	0.601712	1.000000

Notice at the left-to-right diagonal in the correlation table generated above. Why is the diagonal filled with 1s? Is that a coincidence? Explain.

- No, since the 1s from the correlation table generated in a perfect diagonal manner, it strongly suggest that each variable on the table has perfect correlation to itself.

Still looking at the correlation table above, notice that the values are mirrored; values below the 1 diagonal have a mirrored counterpart above the 1 diagonal. Is that a coincidence? Explain.

- No, same logic with the former question, the table has built a perfect correlation between each variables and has made a mirrored values above and below the 1 diagonal.

Using the same corr() method, it is easy to calculate the correlation of the variables contained in the female-only dataframe:

```
# Code cell 9
womenDf.corr(method='pearson')
```

```
<ipython-input-10-a6271751808a>:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  womenDf.corr(method='pearson')
```

	FSIQ	VIQ	PIQ	Weight	Height	MRI_Count
FSIQ	1.000000	0.955717	0.939382	0.038192	-0.059011	0.325697
VIQ	0.955717	1.000000	0.802652	-0.021889	-0.146453	0.254933
PIQ	0.939382	0.802652	1.000000	0.113901	-0.001242	0.396157
Weight	0.038192	-0.021889	0.113901	1.000000	0.552357	0.446271
Height	-0.059011	-0.146453	-0.001242	0.552357	1.000000	0.174541
MRI_Count	0.325697	0.254933	0.396157	0.446271	0.174541	1.000000

And the same can be done for the male-only dataframe:

```
# Code cell 10
# Use corr() for the male-only dataframe with the pearson method
menDf.corr(method = 'pearson')
```

```
<ipython-input-13-e6099c4c20d5>:3: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  menDf.corr(method = 'pearson')
```

	FSIQ	VIQ	PIQ	Weight	Height	MRI_Count
FSIQ	1.000000	0.944400	0.930694	-0.278140	-0.356110	0.498369
VIQ	0.944400	1.000000	0.766021	-0.350453	-0.355588	0.413105
PIQ	0.930694	0.766021	1.000000	-0.156863	-0.287676	0.568237
Weight	-0.278140	-0.350453	-0.156863	1.000000	0.406542	-0.076875
Height	-0.356110	-0.355588	-0.287676	0.406542	1.000000	0.301543
MRI_Count	0.498369	0.413105	0.568237	-0.076875	0.301543	1.000000

Part 4: Visualizing

Step 1: Install Seaborn. To make it easier to visualize the data correlations, heatmap graphs can be used. Based on colored squares, heatmap graphs can help identify correlations in a glance.

The Python module named seaborn makes it very easy to plot heatmap graphs.

First, run the cell below to download and install the seaborn module.

```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in
/usr/local/lib/python3.10/dist-packages (0.13.1)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in
/usr/local/lib/python3.10/dist-packages (from seaborn) (1.23.5)
Requirement already satisfied: pandas>=1.2 in
/usr/local/lib/python3.10/dist-packages (from seaborn) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in
/usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4-
>seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4-
>seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4-
>seaborn) (4.47.2)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4-
>seaborn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4-
>seaborn) (23.2)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4-
>seaborn) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4-
>seaborn) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4-
>seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn)
(2023.4)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7-
>matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
```

Step 2: Plot the correlation heatmap.

Now that the dataframes are ready, the heatmaps can be plotted. Below is a breakdown of the code in the cell below:

Line 1: Generates a correlation table based on the womenNoGenderDf dataframe and stores it on wcorr.

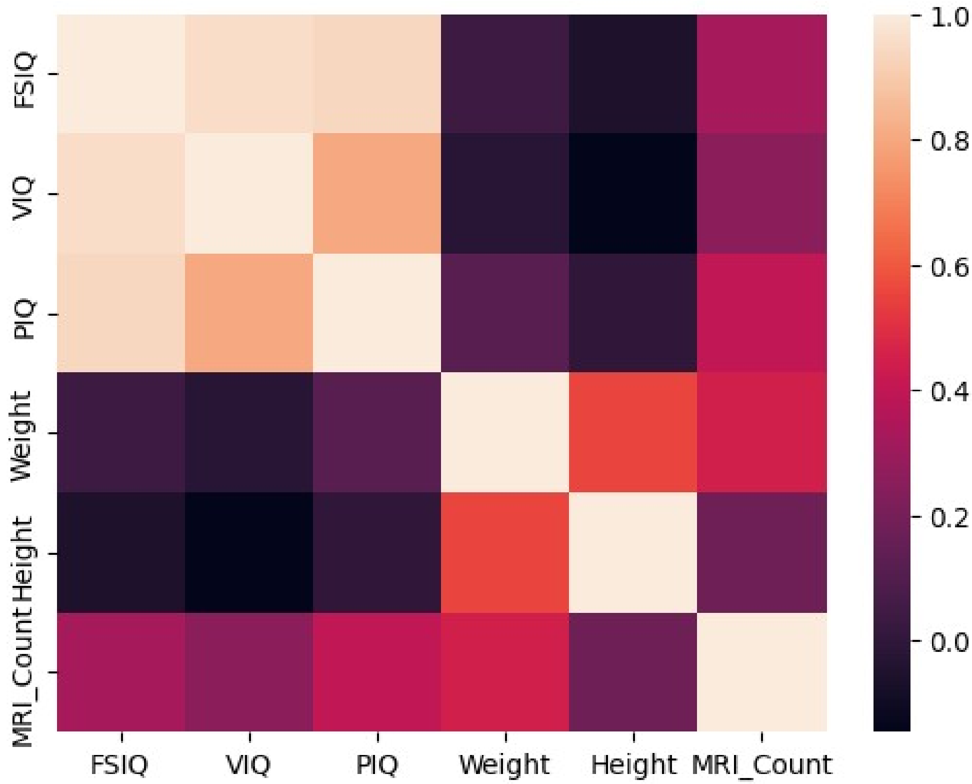
Line 2: Uses the seaborn heatmap() method to generate and plot the heatmap. Notice that heatmap() takes wcorr as a parameter.

Line 3: Use to export and save the generated heatmap as a PNG image. While the line 3 is not active (ithas the comment # character preceding it, forcing the interpreter to ignore it), it was kept forinformational purposes.

```
#Code cell 12
import seaborn as sns

wcorr = womenDf.corr()
sns.heatmap(wcorr)
#plt.savefig('attribute_correlations.png', tight_layout=True)
```

```
<ipython-input-15-101ad2c7f9b0>:4: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  wcorr = womenDf.corr()
```

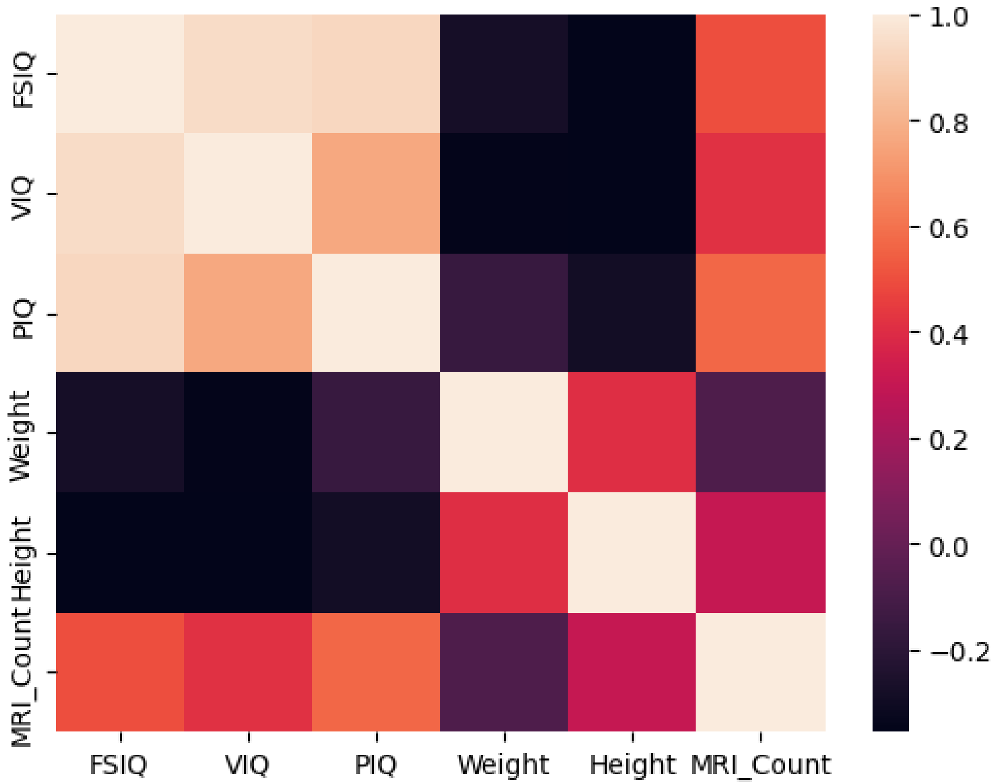
```
<Axes: >
```



Similarly, the code below creates and plots a heatmap for the male-only dataframe.

```
#Code cell 13
import seaborn as sns

mcorr = menDf.corr()
sns.heatmap(mcorr)
#plt.savefig('attribute_correlations.png', tight_layout=True)
```

```
<ipython-input-16-ae3fd096f905>:4: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  mcorr = menDf.corr()

<Axes: >
```



Many variable pairs present correlation close to zero. What does that mean?

- Variable pairs presenting correlation that is close to zero means that the pair have weak correlation. Hence, they have no weak-to-no relation to each other.

Why separate the genders?

- I observed that if we want to have a more reliable data analysis, we should identify and classify the given data. Thus, we separated the mean from women to provide insights into gender-specific patterns and concerns.

What variables have stronger correlation with brain size (MRI_Count)? Is that expected? Explain.

- The strongest correlation that the brain size has is the variable itself producing a value of 1, a perfect correlation to itself and it is expected. However, the if the MRI count has to correlate with other variables than itself, the correlation between PIQ and MRI count got the stronger correlation than the other variables.

Supplementary Activity

For supplementary activity: Look for (any) real-world dataset and perform exploratory and statistical analysis.

Loading the necessary libraries to be used and inserting the real-world dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
FactPerf = pd.read_csv('/content/SupplementaryDataset.csv')
```

For this part, I used the .head() to display the first five entries on the dataset I uploaded.

```
FactPerf.head()

  school sex  age  class_failures  family_relationship  free_time
social  \
0     GP   F   18               0                    4          3
4
1     GP   F   17               0                    5          3
3
2     GP   F   15               3                    4          3
2
3     GP   F   15               0                    3          2
2
4     GP   F   16               0                    4          3
2

   weekday_alcohol  weekend_alcohol  health  absences  grade_1
grade_2  \
0                1                1       3         6        5
6
1                1                1       3         4        5
5
2                2                3       3        10        7
8
3                1                1       5         2       15
14
4                1                2       5         4        6
10

   final_grade
0            6
1            6
2           10
3           15
4           10
```

The code below will perform and display the results of some common calculations that will describe the statistics of the given dataset. For this I have at least 395 students' data to analyze.

```
FactPerf.describe()
```

```
                age  class_failures  family_relationship   free_time  \
count  395.000000      395.000000           395.000000  395.000000   
mean    16.696203        0.334177             3.944304    3.235443   
std      1.276043        0.743651             0.896659    0.998862   
min     15.000000        0.000000             1.000000    1.000000   
25%     16.000000        0.000000             4.000000    3.000000   
50%     17.000000        0.000000             4.000000    3.000000   
75%     18.000000        0.000000             5.000000    4.000000   
max     22.000000        3.000000             5.000000    5.000000   

           social  weekday_alcohol  weekend_alcohol      health   
absences  \
count  395.000000       395.000000       395.000000  395.000000   
395.000000   
mean     3.108861         1.481013         2.291139    3.554430   
5.708861   
std      1.113278         0.890741         1.287897    1.390303   
8.003096   
min      1.000000         1.000000         1.000000    1.000000   
0.000000   
25%      2.000000         1.000000         1.000000    3.000000   
0.000000   
50%      3.000000         1.000000         2.000000    4.000000   
4.000000   
75%      4.000000         2.000000         3.000000    5.000000   
8.000000   
max      5.000000         5.000000         5.000000    5.000000   
75.000000   

          grade_1     grade_2  final_grade  
count  395.000000  395.000000   395.000000  
mean    10.908861   10.713924    10.415190  
std      3.319195    3.761505     4.581443  
min      3.000000    0.000000     0.000000  
25%      8.000000    9.000000     8.000000  
50%     11.000000   11.000000    11.000000  
75%     13.000000   13.000000    14.000000  
max     19.000000   19.000000    20.000000  
```

Separating the dataset by gender, just like in the procedure

```
menFP = FactPerf[(FactPerf.sex == 'M')]
womenFP = FactPerf[(FactPerf.sex == 'F')]
```

Using the person method to provide table of correlation for men between variables: age, class failures, family relationship, free time, absences, point grades for 1st period. 2nd period, and final period, and on a scale of 1-5 social, weekday alcohol, weekend alcohol, health.

```
menFP.corr(method = 'pearson')
```

```
<ipython-input-40-eb5df9105ca6>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  menFP.corr(method = 'pearson')

                          age  class_failures  family_relationship
free_time  \
age                  1.000000        0.375306             0.030901
0.036182
class_failures       0.375306        1.000000             0.030384
0.247977
family_relationship  0.030901        0.030384             1.000000
0.173285
free_time            0.036182        0.247977             0.173285
1.000000
social               0.176930        0.278513            -0.026370
0.203239
weekday_alcohol      0.245704        0.148218            -0.155329
0.145842
weekend_alcohol      0.247321        0.198477            -0.194270
0.002490
health              -0.053293        0.021507             0.164898
0.085530
absences             0.244087        0.060493            -0.067068
0.028586
grade_1             -0.225840       -0.476230             0.015881
0.011189
grade_2             -0.274882       -0.465862            -0.046433  -
0.038583
final_grade         -0.272040       -0.478148             0.046613  -
0.024129

                       social  weekday_alcohol  weekend_alcohol
health  \
age                  0.176930         0.245704         0.247321 -
0.053293
class_failures       0.278513         0.148218         0.198477
0.021507
family_relationship -0.026370        -0.155329        -0.194270
0.164898
free_time            0.203239         0.145842         0.002490
0.085530
social               1.000000         0.333063         0.539459
0.037746
weekday_alcohol      0.333063         1.000000         0.675284
0.041666
weekend_alcohol      0.539459         0.675284         1.000000
0.124722
health               0.037746         0.041666         0.124722
```

```
1.000000
absences                 0.228589         0.198429         0.241033 -
0.041253
grade_1                 -0.228247        -0.194556        -0.310212 -
0.051764
grade_2                 -0.248277        -0.143289        -0.263048 -
0.077297
final_grade             -0.234775        -0.145485        -0.235717 -
0.063893

                     absences   grade_1   grade_2  final_grade
age                  0.244087 -0.225840 -0.274882    -0.272040
class_failures       0.060493 -0.476230 -0.465862    -0.478148
family_relationship -0.067068  0.015881 -0.046433     0.046613
free_time            0.028586  0.011189 -0.038583    -0.024129
social               0.228589 -0.228247 -0.248277    -0.234775
weekday_alcohol      0.198429 -0.194556 -0.143289    -0.145485
weekend_alcohol      0.241033 -0.310212 -0.263048    -0.235717
health              -0.041253 -0.051764 -0.077297    -0.063893
absences             1.000000 -0.109746 -0.118761    -0.037133
grade_1             -0.109746  1.000000  0.862252     0.830741
grade_2             -0.118761  0.862252  1.000000     0.931665
final_grade         -0.037133  0.830741  0.931665     1.000000
```

Doing the same process for female...

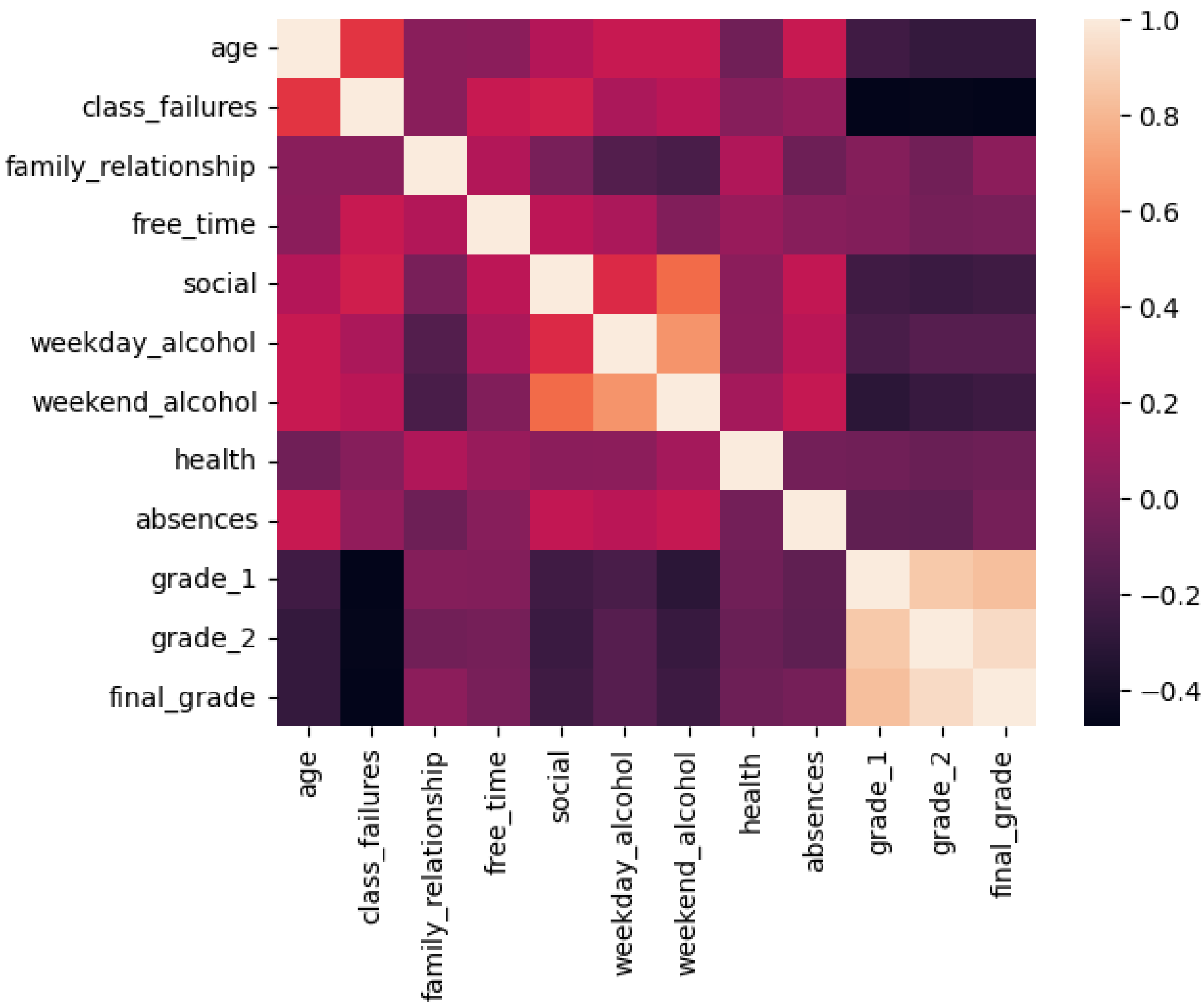
```
womenFP.corr(method = 'pearson')

<ipython-input-41-3e6340d07094>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  womenFP.corr(method = 'pearson')

                          age  class_failures  family_relationship
free_time  \
age                  1.000000        0.097176             0.081230
0.010824
class_failures       0.097176        1.000000            -0.125721  -
0.092057
family_relationship  0.081230       -0.125721             1.000000
0.110040
free_time            0.010824       -0.092057             0.110040
1.000000
social               0.079739       -0.043070             0.141890
0.347690
weekday_alcohol     -0.031917        0.103378            -0.012635
0.184811
weekend_alcohol     -0.028273        0.050395            -0.066263
```

```
0.199468
health               -0.064362        0.097717             0.019327
0.004620
absences              0.140070        0.075343            -0.027006  -
0.089203
grade_1               0.112836       -0.238695             0.017781  -
0.030500
grade_2               0.001995       -0.250549            -0.002124  -
0.034865
final_grade          -0.049473       -0.260475             0.044696  -
0.004360

                        social  weekday_alcohol  weekend_alcohol
health  \
age                   0.079739        -0.031917        -0.028273 -
0.064362
class_failures       -0.043070         0.103378         0.050395
0.097717
family_relationship  0.141890        -0.012635        -0.066263
0.019327
free_time             0.347690         0.184811         0.199468
0.004620
social                1.000000         0.152299         0.269888 -
0.073032
weekday_alcohol       0.152299         1.000000         0.516895
0.043969
weekend_alcohol       0.269888         0.516895         1.000000 -
0.021821
health               -0.073032         0.043969        -0.021821
1.000000
absences             -0.054111         0.108605         0.120142 -
0.009970
grade_1              -0.087386        -0.014723         0.031973 -
0.120103
grade_2              -0.092920        -0.012554         0.073527 -
0.144730
final_grade          -0.057349        -0.002133         0.091787 -
0.088674

                      absences   grade_1   grade_2  final_grade
age                   0.140070  0.112836  0.001995    -0.049473
class_failures        0.075343 -0.238695 -0.250549    -0.260475
family_relationship -0.027006  0.017781 -0.002124     0.044696
free_time            -0.089203 -0.030500 -0.034865    -0.004360
social               -0.054111 -0.087386 -0.092920    -0.057349
weekday_alcohol       0.108605 -0.014723 -0.012554    -0.002133
weekend_alcohol       0.120142  0.031973  0.073527     0.091787
health               -0.009970 -0.120103 -0.144730    -0.088674
absences              1.000000  0.024380  0.028996     0.086359
```

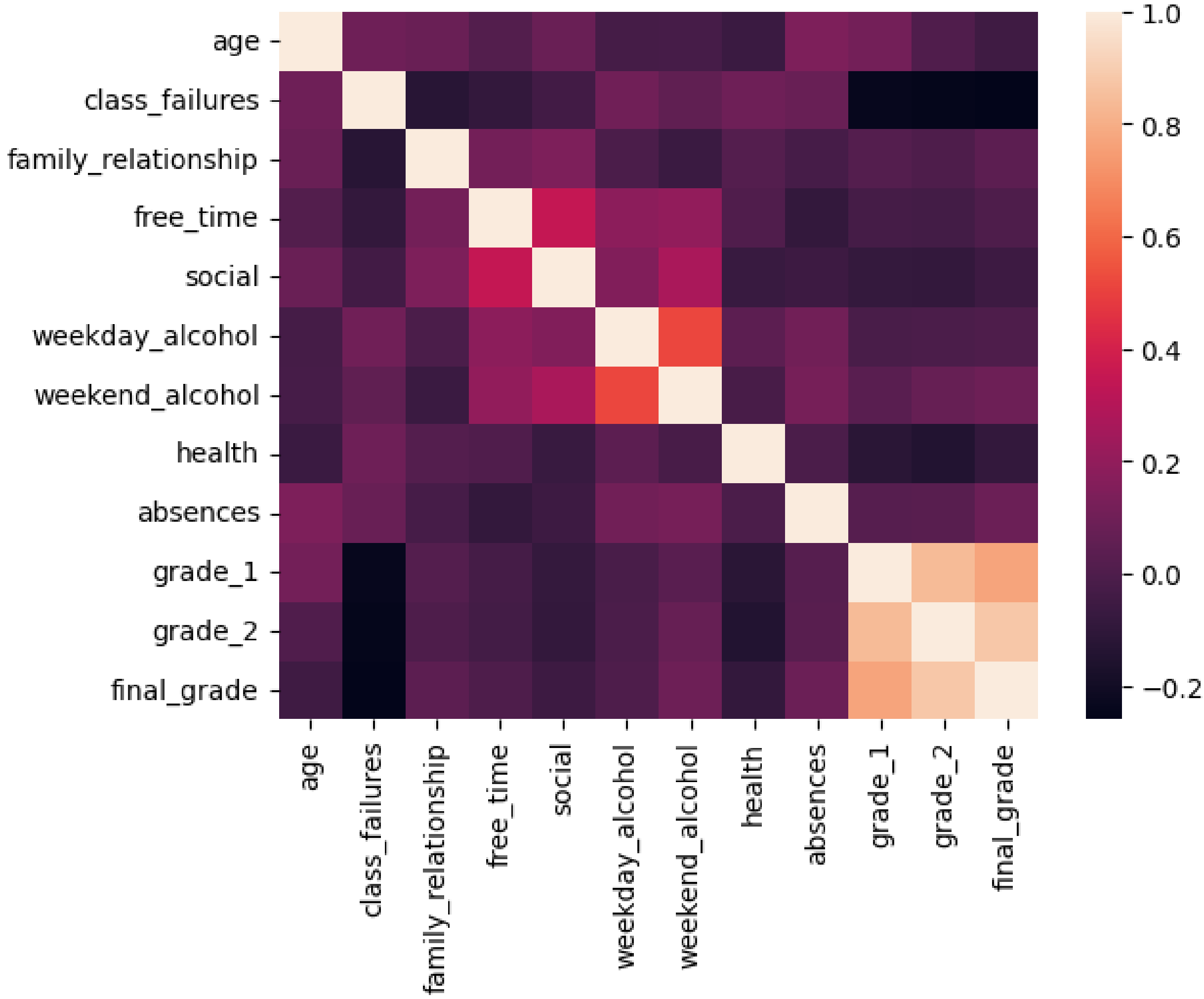
```
grade_1                    0.024380  1.000000  0.839500      0.772186
grade_2                    0.028996  0.839500  1.000000      0.880077
final_grade                0.086359  0.772186  0.880077      1.000000
```

The code below and its results show us the graphical visualization for easy analyzation of the correlation of the variable inside the given dataset, separated by gender.

```
bcorr = menFP.corr()
sns.heatmap(bcorr)

<ipython-input-42-6b3e014edcbd>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  bcorr = menFP.corr()

<Axes: >
```



```
gcorr = womenFP.corr()
sns.heatmap(gcorr)

<ipython-input-43-b2b98bed4636>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  gcorr = womenFP.corr()

<Axes: >
```



CONCLUSION AND ANALYSIS

Upon performing the activity, it is possible to analyze data with ease through correlation analysis using Python. There are many ways I can process and present the given data to analyze it properly and easily.

The supplementary activity lets me find a real-word data which I find it hard to decide which dataset should I use and how can it be processed using Python. Through simple modifications to the dataset, like leaving only the variables which I think they can correlate to each other.

Finally, the results showed that there aren't any strong correlation of factors like drinking alcohol, absences, health, social, family relationship, and their free time to the grades they have achieved. However, there are strong positive correlation between the 3 grading periods which is expected.

My realization for this activity is that I still need to practice my Python skills until I can confidently write the syntax I have in mind.