CPE311 Computational Thinking with Python

Name: Ballesteros, Angelo

Section: CPE22S2

Performed on: 6/20/2024

Submitted on: 6/20/2024

Submitted to: Engr. Roman M. Richard

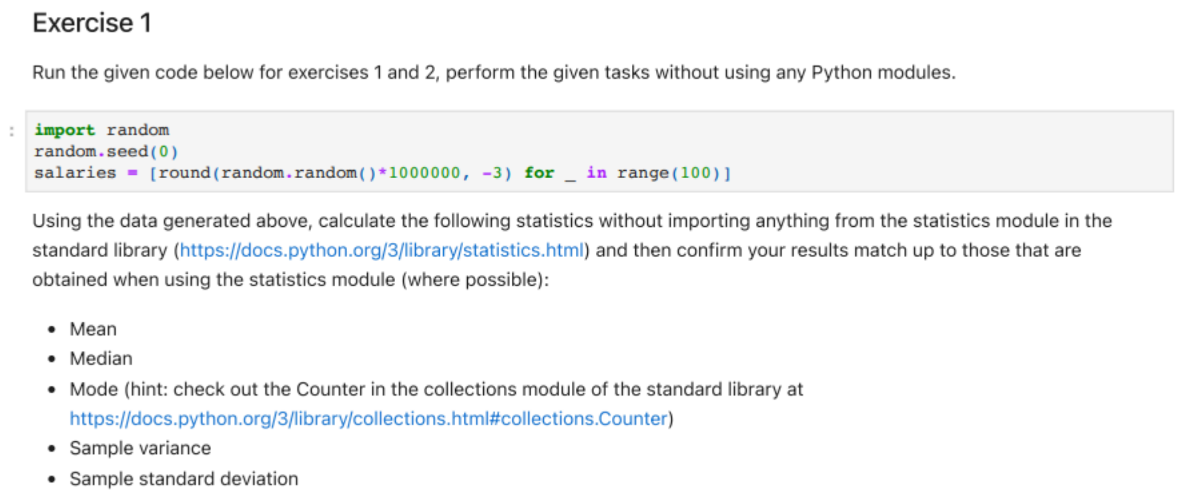
6.1 Intended Learning Outcomes

1. Use pandas and numpy data analysis tools.
2. Demonstrate how to analyze data using numpy and pandas

6.2 Resources:

- Personal Computer
- Jupyter Notebook
- Internet Connection

6.3 Supplementary Activities:



Exercise 1

Run the given code below for exercises 1 and 2, perform the given tasks without using any Python modules.

```
import random
random.seed(0)
salaries = [round(random.random()*1000000, -3) for _ in range(100)]
```

Using the data generated above, calculate the following statistics without importing anything from the statistics module in the standard library (https://docs.python.org/3/library/statistics.html) and then confirm your results match up to those that are obtained when using the statistics module (where possible):

- Mean
- Median
- Mode (hint: check out the Counter in the collections module of the standard library at https://docs.python.org/3/library/collections.html#collections.Counter)
- Sample variance
- Sample standard deviation

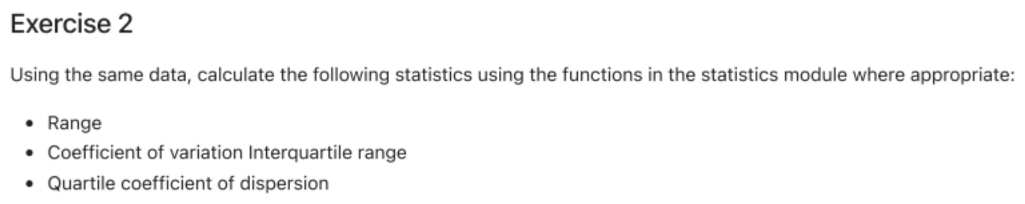
```
import random # data generated
random.seed(0)
salaries = [round(random.random()*1000000, -3) for _ in range(100)]


from statistics import median
from math import isnan
from itertools import filterfalse


def cal_mean(data): # mean function
    return sum(data) / len(data)

mean = cal_mean(salaries)
```

```
def cal_median(data): # median function
    sorted_data = sorted(data)
    n = len(sorted_data)
    midpoint = n // 2

    if n % 2 == 1:
        return sorted_data[midpoint]
    else:
        return (sorted_data[midpoint - 1] + sorted_data[midpoint]) / 2

median = cal_median(salaries)


from collections import Counter

def cal_mode(data): # mode function
    frequency = Counter(data)
    mode_data = frequency.most_common(1)
    return mode_data[0][0] if mode_data else None

mode = cal_mode(salaries)


def cal_sampleVar(data): # sample variance function
    mean = cal_mean(data)
    squared_diff = [(x - mean) ** 2 for x in data]
    return sum(squared_diff) / (len(data) - 1)

sampleVariance = cal_sampleVar(salaries)


def cal_standardDev(data): # sample standard deviation function
    variance = cal_sampleVar(data)
    return variance ** 0.5

sample_standardDev = cal_standardDev(salaries)


# output
print(f"Mean:", mean)
print(f"Median:", median)
print(f"Mode:", mode)
print(f"Sample Variance:", sampleVariance)
print(f"Sample Standard Deviation:", sample_standardDev)
```

```
Mean: 585690.0
Median: 589000.0
Mode: 477000.0
Sample Variance: 70664054444.44444
Sample Standard Deviation: 265827.11382484
```

```
# reference of output

import pandas as pd

df = pd.DataFrame(salaries)
df.describe()
```

	0
count	100.000000
mean	585690.000000
std	265827.113825
min	1000.000000
25%	403500.000000
50%	589000.000000
75%	816750.000000
max	996000.000000



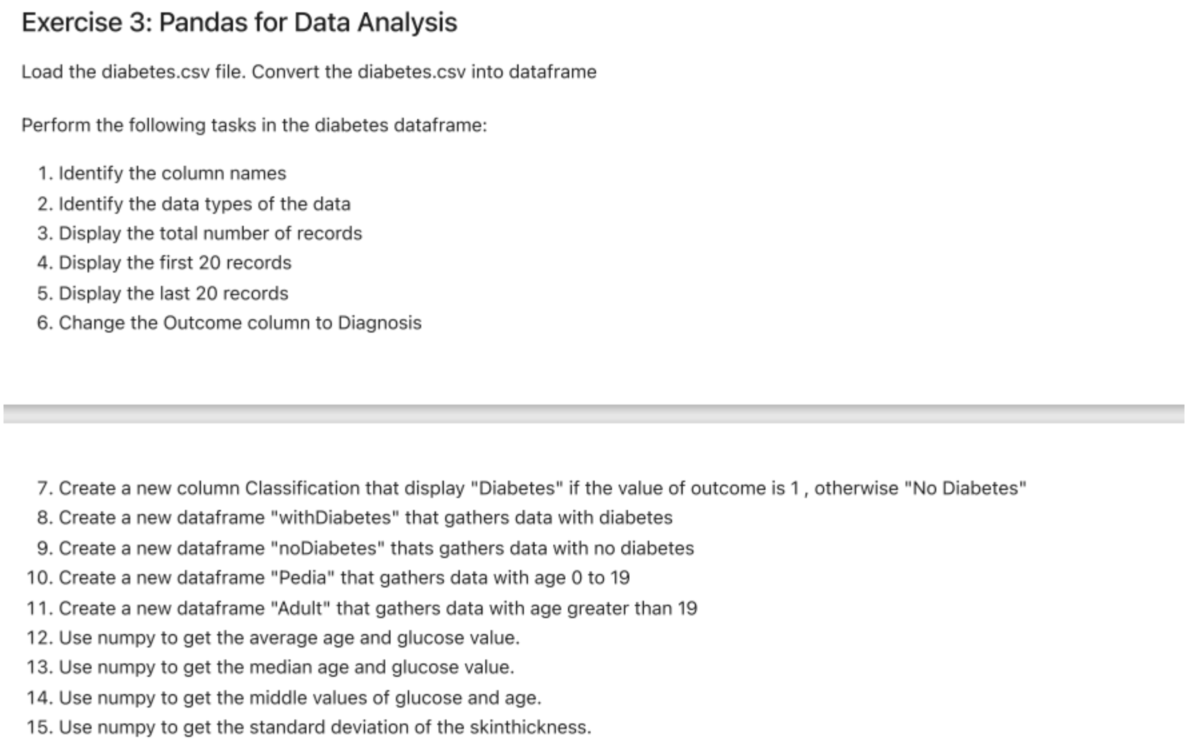
Exercise 2

Using the same data, calculate the following statistics using the functions in the statistics module where appropriate:

- Range
- Coefficient of variation Interquartile range
- Quartile coefficient of dispersion

```
range = max(salaries) - min(salaries) # range function


mean = cal_mean(salaries) # COV function
standardDev = cal_standardDev(salaries)
COV = (standardDev / mean) * 100


def cal_iqr(data): # interquartile range function
    sorted_data = sorted(data)
    q1 = cal_median(sorted_data[:len(sorted_data) // 2])
    q3 = cal_median(sorted_data[(len(sorted_data) + 1) // 2:])
    return q3 - q1

iqr = cal_iqr(salaries)


def cal_qd(data): # quartile dispersion function
    sorted_data = sorted(data)
    q1 = cal_median(sorted_data[:len(sorted_data) // 2])
    q3 = cal_median(sorted_data[(len(sorted_data) + 1) // 2:])
    return (q3 - q1) / (q3 + q1)

qd = cal_qd(salaries)


# output

print(f"Range:", range)
print(f"Coefficient of Variation:%", COV)
print(f"Interquartile Range:", iqr)
print(f"Quartile Coefficient of Dispersion:", qd)
```

```
Range: 995000.0
Coefficient of Variation:% 45.38699889443903
Interquartile Range: 417500.0
Quartile Coefficient of Dispersion: 0.3417928776094965
```



Exercise 3: Pandas for Data Analysis

Load the diabetes.csv file. Convert the diabetes.csv into dataframe

Perform the following tasks in the diabetes dataframe:

1. Identify the column names
2. Identify the data types of the data
3. Display the total number of records
4. Display the first 20 records
5. Display the last 20 records
6. Change the Outcome column to Diagnosis
7. Create a new column Classification that display "Diabetes" if the value of outcome is 1 , otherwise "No Diabetes"
8. Create a new dataframe "withDiabetes" that gathers data with diabetes
9. Create a new dataframe "noDiabetes" thats gathers data with no diabetes
10. Create a new dataframe "Pedia" that gathers data with age 0 to 19
11. Create a new dataframe "Adult" that gathers data with age greater than 19
12. Use numpy to get the average age and glucose value.
13. Use numpy to get the median age and glucose value.
14. Use numpy to get the middle values of glucose and age.
15. Use numpy to get the standard deviation of the skinthickness.

```
# uploading csv and convert it into dataframe

import pandas as pd
from google.colab import files

uploaded = files.upload()

diabetes = list(uploaded.keys())[0]

df = pd.read_csv(diabetes)

print(df)
```

Choose Files diabetes.csv

- **diabetes.csv**(text/csv) - 23873 bytes, last modified: 6/20/2024 - 100% done

```
Saving diabetes.csv to diabetes (2).csv
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

```
column_names = list(df.columns) # identify column names
print(column_names)
```

```
['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
```

```
df_types = df.dtypes # identify the data types
print(df_types)
```

```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

```
print(f"Number of records: {df.shape[0]}") # total number of records
```

```
Number of records: 768
```

```
print(df.head(20)) # first 20 records
```

```
    Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0             6      148             72             35        0  33.6
1             1       85             66             29        0  26.6
2             8      183             64              0        0  23.3
3             1       89             66             23       94  28.1
4             0      137             40             35      168  43.1
5             5      116             74              0        0  25.6
```

```
6            3      78            50            32       88  31.0
7           10     115             0             0        0  35.3
8            2     197            70            45      543  30.5
9            8     125            96             0        0   0.0
10           4     110            92             0        0  37.6
11          10     168            74             0        0  38.0
12          10     139            80             0        0  27.1
13           1     189            60            23      846  30.1
14           5     166            72            19      175  25.8
15           7     100             0             0        0  30.0
16           0     118            84            47      230  45.8
17           7     107            74             0        0  29.6
18           1     103            30            38       83  43.3
19           1     115            70            30       96  34.6

    DiabetesPedigreeFunction  Age  Outcome
0                      0.627   50        1
1                      0.351   31        0
2                      0.672   32        1
3                      0.167   21        0
4                      2.288   33        1
5                      0.201   30        0
6                      0.248   26        1
7                      0.134   29        0
8                      0.158   53        1
9                      0.232   54        1
10                     0.191   30        0
11                     0.537   34        1
12                     1.441   57        0
13                     0.398   59        1
14                     0.587   51        1
15                     0.484   32        1
16                     0.551   31        1
17                     0.254   31        1
18                     0.183   33        0
19                     0.529   32        1
```

```
df.tail(20) # last 20 records
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
748	3	187	70	22	200	36.4	
749	6	162	62	0	0	24.3	
750	4	136	70	0	0	31.2	
751	1	121	78	39	74	39.0	
752	3	108	62	24	0	26.0	
753	0	181	88	44	510	43.3	
754	8	154	78	32	0	32.4	
755	1	128	88	39	110	36.5	
756	7	137	90	41	0	32.0	
757	0	123	72	0	0	36.3	
758	1	106	76	0	0	37.5	
759	6	190	92	0	0	35.5	
760	2	88	58	26	16	28.4	
761	9	170	74	31	0	44.0	
762	9	89	62	0	0	22.5	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

```
df = df.rename(columns={'Outcome': 'Diagnosis'}) # change outcome to diagnosis
```

```
# create new column Classification that display Diabetes if outcome is 1, otherwise No Diabetes

df['Classification'] = df['Diagnosis'].apply(lambda x: 'Diabetes' if x == 1 else 'No Diabetes')
print(df.head(20))
```

```
    Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0             6      148             72             35        0  33.6
1             1       85             66             29        0  26.6
2             8      183             64              0        0  23.3
3             1       89             66             23       94  28.1
4             0      137             40             35      168  43.1
5             5      116             74              0        0  25.6
6             3       78             50             32       88  31.0
7            10      115              0              0        0  35.3
8             2      197             70             45      543  30.5
9             8      125             96              0        0   0.0
10            4      110             92              0        0  37.6
11           10      168             74              0        0  38.0
12           10      139             80              0        0  27.1
13            1      189             60             23      846  30.1
14            5      166             72             19      175  25.8
15            7      100              0              0        0  30.0
16            0      118             84             47      230  45.8
17            7      107             74              0        0  29.6
18            1      103             30             38       83  43.3
19            1      115             70             30       96  34.6

    DiabetesPedigreeFunction  Age  Diagnosis Classification
0                      0.627   50          1       Diabetes
1                      0.351   31          0    No Diabetes
2                      0.672   32          1       Diabetes
3                      0.167   21          0    No Diabetes
4                      2.288   33          1       Diabetes
5                      0.201   30          0    No Diabetes
6                      0.248   26          1       Diabetes
7                      0.134   29          0    No Diabetes
8                      0.158   53          1       Diabetes
9                      0.232   54          1       Diabetes
10                     0.191   30          0    No Diabetes
11                     0.537   34          1       Diabetes
12                     1.441   57          0    No Diabetes
13                     0.398   59          1       Diabetes
14                     0.587   51          1       Diabetes
15                     0.484   32          1       Diabetes
16                     0.551   31          1       Diabetes
17                     0.254   31          1       Diabetes
18                     0.183   33          0    No Diabetes
19                     0.529   32          1       Diabetes
```

```
withDiabetes = df.loc[df['Diagnosis'] == 1] # dataframe withDiabetes
```

```
noDiabetes = df.loc[df['Diagnosis'] == 0] # dataframe noDiabetes
```

```
pedia = df.loc[(df['Age'] >= 0) & (df['Age'] <= 19)] # dataframe Pedia
```

```
adult = df.loc[(df['Age'] > 19)] # dataframe Adult
```

```
# average age and glucose value

import numpy as np

average_age = np.mean(df['Age'])
average_glucose = np.mean(df['Glucose'])

print(f"Average age:", average_age)
print(f"Average glucose value:", average_glucose)
```

```
Average age: 33.240885416666664
Average glucose value: 120.89453125
```

```
# median age and glucose value

median_age = np.median(df['Age'])
median_glucose = np.median(df['Glucose'])

print(f"Median age:", median_age)
print(f"Median glucose value:", median_glucose)
```

```
Median age: 29.0
Median glucose value: 117.0
```

```
# middle values of glucose and age

mid_glucose = np.median(df['Glucose'])
mid_age = np.median(df['Age'])

print(f"Middle glucose value:", mid_glucose)
print(f"Middle age:", mid_age)
```

```
Middle glucose value: 117.0
Middle age: 29.0
```

```
# standard deviation of skin thickness

import numpy as np

skinthickness_std = np.std(df['SkinThickness'])

print(f"Standard deviation of skinthickness:", skinthickness_std)
```

```
Standard deviation of skinthickness: 15.941828626496939
```

6.4 Conclusion

In conclusion, I have applied the intended learning outcomes for the activity. I have use the pandas and numpys for importing and analyzing the given data which is the diabetes file. I got a little problem in exercise 1 and 2 because the statistics value does not match in the statistics table of the random generated data. Furthermore, this activity does connect in my previous course which is MATH 019A - Engineering Data Analysis so it is easy for me in understanding the measures of central tendency but in terms of coding it in Python, it was hard so I did get some guides. Lastly, I thought it was not possible in converting the csv file into dataframe since Google Colab does not have some uploading button, turns out there was a code to upload files and it displays the dataframe of the uploaded file.

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```