Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: Docta, Joshua Stephen
Section: CPE22S3
Performed on: 05 / 04 / 2025
Submitted on: 05 / 04 / 2025
Submitted to: Engr. Roman M. Richard

6.3 Supplementary Activities:

Exercise 1

In [3]:

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

#prints mean value of salaries
MEAN = sum(salaries)/sal_length
print('Mean: \n\t', MEAN)

#prints median value of salaries
if sal_length % 2 == 0:
    MEDIAN = (salaries[sal_length//2] + salaries[sal_length//2-1])/2
else:
    MEDIAN = salaries[sal_length//2]
print('Median: \n\t',MEDIAN)

#print mode value in salaries
num = list(set(salaries))
Num = {x:salaries.count(x) for x in num}
counts = [x for x in Num.values()]
num = [x for x in Num.keys()]
MODE = [x for x in num if Num[x] == max(counts)]
print('Mode:\n\t', MODE[0])

#Print the sample variance
VARIANCE = sum((x-MEAN)**2/(sal_length-1) for x in salaries)
print('Sample Variance:\n\t',VARIANCE)

#Print the standard De
STANDARD_DEV = VARIANCE**(1/2)
print('Sample standard deviation:\n\t',STANDARD_DEV)
```

```
Mean: 
	 585690.0
Median: 
	 589000.0
Mode:
	 477000.0
Sample Variance:
	 70664054444.44444
Sample standard deviation:
	 265827.11382484
```

In [4]:

```python
from statistics import mean
from statistics import median
from statistics import mode
from statistics import variance
from statistics import stdev
print('Mean: \n\t',mean(salaries))
print('Median: \n\t',median(salaries))
print('Mode:\n\t',mode(salaries))
print('Sample Variance:\n\t',variance(salaries))
print('Sample standard deviation:\n\t',stdev(salaries))
```

```
Mean: 
	 585690.0
Median: 
	 589000.0
Mode:
	 477000.0
Sample Variance:
	 70664054444.44444
Sample standard deviation:
	 265827.11382484
```

Exercise 2

In [6]:

```python
# Prints the range of salaries
RANGE = max(salaries) - min(salaries)
print('Range: \n\t', RANGE)

# Prints  Coefficient of variation and Interquartile range
lower = salaries[(sal_length//4-1)]
higher = salaries[((sal_length*3)//4-1)]
IQR = higher-lower # Interquartile range
CV = (STANDARD_DEV/MEAN)*100 #  Coefficient of variation
print('Coefficient of variation:\n\t', CV)
print('Interquartile range:\n\t',IQR)


#Prints Quartile coefficient of dispersion
QCD = (higher-lower)/(higher+lower)
print('Quartile coefficient of dispersion:\n\t',QCD)
```

```
Range:
         995000.0
Coefficient of variation:
         45.38699889443903
Interquartile range:
         415000.0
Quartile coefficient of dispersion:
         0.34212695795548226
```

Exercise 3

In [8]:

```
import pandas as pd
diabetes = pd.read_csv('diabetes.csv')
```

In [9]:

```
# 1. Identify the column names
diabetes.columns
```

Out[9]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [10]:

```
#2. Identify the data types of the data
diabetes.dtypes
```

Out[10]:

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

In [11]:

```
#3. Display the total number of records
diabetes.count()
```

Out[11]:

```
Pregnancies                 768
Glucose                     768
BloodPressure               768
SkinThickness               768
Insulin                     768
BMI                         768
DiabetesPedigreeFunction    768
Age                         768
Outcome                     768
dtype: int64
```

In [12]:

```
#4. Display the first 20 records
diabetes.head(20)
```

Out[12]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFur
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	

In [13]:

```
#5. Display the last 20 records
diabetes.tail(20)
```

Out[13]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
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	

In [14]:

```
#6. Change the Outcome column to Diagnosis
diabetes.rename(columns = {'Outcome':'Diagnosis'}, inplace = True)
```

In [15]:

```
#7. Create a new column Classification that display "Diabetes" if the value of outc
new_col = ['Diabetes' if x==1 else "No Diabetes" for x in diabetes.Diagnosis]
diabetes['Classification'] = new_col
diabetes
```

Out[15]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
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	

768 rows × 10 columns

In [16]:

```
#8. Create a new dataframe "withDiabetes" that gathers data with diabetes
withDiabetes = diabetes[diabetes['Diagnosis'] == 1]
withDiabetes
```

Out[16]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
0	6	148	72	35	0	33.6	
2	8	183	64	0	0	23.3	
4	0	137	40	35	168	43.1	
6	3	78	50	32	88	31.0	
8	2	197	70	45	543	30.5	
...	...	...	...	...	...	...	
755	1	128	88	39	110	36.5	
757	0	123	72	0	0	36.3	
759	6	190	92	0	0	35.5	
761	9	170	74	31	0	44.0	
766	1	126	60	0	0	30.1	

268 rows × 10 columns

In [17]:

```
#9. Create a new dataframe "noDiabetes" thats gathers data with no diabetes
noDiabetes = diabetes[diabetes['Diagnosis'] == 0]
noDiabetes
```

Out[17]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
1	1	85	66	29	0	26.6	
3	1	89	66	23	94	28.1	
5	5	116	74	0	0	25.6	
7	10	115	0	0	0	35.3	
10	4	110	92	0	0	37.6	
...	...	...	...	...	...	...	
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	
767	1	93	70	31	0	30.4	

500 rows × 10 columns

In [18]:

```
#10. Create a new dataframe "Pedia" that gathers data with age 0 to 19
Pedia = diabetes[(diabetes['Age'] >= 0) & (diabetes['Age'] <= 9)]
Pedia
```

Out[18]:

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct

In [19]:

```
Adult = diabetes[diabetes['Age'] >= 19]
Adult
```

Out[19]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
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	

768 rows × 10 columns

In [20]:

```
#12-15. Use numpy to get the...
import numpy as np
```

In [21]:

```
#12. Use numpy to get the average age and glucose value.
average_age = np.mean(diabetes.Age)
average_glucose = np.mean(diabetes.Glucose)
print('Average Age:\n\t', average_age)
print('Average Glucose:\n\t', average_glucose)
```

```
Average Age:
	 33.240885416666664
Average Glucose:
	 120.89453125
```

In [22]:

```
#13. Use numpy to get the median age and glucose value.
median_age = np.median(diabetes.Age)
median_glucose = np.median(diabetes.Glucose)
print('Median Age:\n\t', median_age)
print('Median Glucose:\n\t', median_glucose)
```

```
Median Age:
	 29.0
Median Glucose:
	 117.0
```

In [23]:

```
#14.  Use numpy to get the middle values of glucose and age.
Age = np.sort(diabetes.Age)
Glucose = np.sort(diabetes.Glucose)
Size = len(Age)
middle_age = Age[Size//2-1]
```

```
middle_glucose = Glucose[Size//2-1]
print('Middle Age:\n\t', middle_age)
print('Middle Glucose:\n\t', middle_glucose)
```

```
Middle Age:
	 29
Middle Glucose:
	 117
```

In [24]:

```
#15. Use numpy to get the standard deviation of the skinthickness.
np.std(diabetes.SkinThickness)
```

Out[24]:

```
15.941828626496978
```

6.4 Conclusion

In conclusion, from this hands on activity I was able to learn different approches on how to handle statistical data. On the first exrcise I was able to recall and practice once again various python functions reach the objectives and instructions. And through this same exercise I have expanded my knowledge learned something new, such as the python statistics which greatly helps in lessening the time and effort when coding. On exercise 2 I might still be having doubt with my answer as i wasn't able to check it using python statistics, but I think what I've written/typed in my code is right following the various statistical formulas I've reread and restudy once again. And I think exercise 3 was much easier compared to the first two because we've just recently studied python pandas(although unfortunately I wasn't able to submit the activity at the last minute) and implementing numpy is quite easier.