Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: Sanchez, Justin Bjorn L.

Section: CPE22S3

Performed on: 05/04/2025

Submitted on: 05/04/2025

Submitted to: Engr. Roman M. Richard

6.1 Intended Learning Outcome

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(I used colab since I asked for permission)
- Internet Connection

Double-click (or enter) to edit

**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 deviatio

```
from statistics import mode, mean, median, stdev, variance #Import needed library

salaries
```

```
        933000.0,
        109000.0,
        551000.0,
        707000.0,
        547000.0,
        814000.0,
        540000.0,
        964000.0,
        603000.0,
        588000.0,
        445000.0,
        596000.0,
        385000.0,
        576000.0,
        290000.0,
        189000.0,
        187000.0,
        613000.0,
        657000.0,
        477000.0,
        90000.0,
        758000.0,
        877000.0,
        923000.0,
        842000.0,
        898000.0,
        923000.0,
        541000.0,
        391000.0,
        705000.0,
        276000.0,
        812000.0,
        849000.0,
        895000.0,
        590000.0,
        950000.0,
        580000.0,
        451000.0,
        660000.0,
        996000.0,
        917000.0,
        793000.0,
        82000.0,
        613000.0,
        486000.0]
```

```
mean = mean(salaries)
mean

#get the mean
```

```
585690.0
```

```
salary2 = sorted(salaries)
median(salaries)

#get the median sorted
```

```
589000.0
```

```
median1 = len(salary2) #How many items

median2 = median1 / 2 #Find middle

median3 = salary2[int(median2)] #Round down and find mid value

median3
```

```
590000.0
```

```
mode(salaries) #with module
```

```
477000.0
```

```
from collections import Counter #Use collections instead of statistic mod

Counter = Counter(salaries).most_common(1)[0][0] #pick value that appears most

Counter
```

```
477000.0
```

```
variance(salaries, mean)
```

```
70664054444.44444
```

```
stdev(salaries) #using module
```

```
265827.11382484
```

Exercise 2

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

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

```
from statistics import mean, stdev, quantiles


range = max(salaries) - min(salaries)
range

#using module
```

```
995000.0
```

```
CoVar = (stdev(salaries) / mean(salaries)) * 100 #find std
q1, q2, q3 = quantiles(salaries, n = 4 ) #calculate quartiles
iqr = q3 - q1

print(f'Coefficient of variation: {CoVar}')
print(f'Interquartile range: {iqr}')

#Without using stat mod
```

```
Coefficient of variation: 45.38699889443903
Interquartile range: 421750.0
```

```
q1, q2, q3 = quantiles(salaries, n = 4 ) #calculate quartiles
qcd = (q3 - q1) / (q3 + q1) #calculate qcd

qcd
```

```
0.34491923941934166
```

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.

```
import pandas as pd

diabetes = pd.read_csv('/content/diabetes.csv')
diabetes.head(5)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Next steps: Generate code with diabetes | View recommended plots | New interactive sheet

```
diabetes.columns #1
```

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

```
diabetes.dtypes #2
```

	0
Pregnancies	int64
Glucose	int64
BloodPressure	int64
SkinThickness	int64
Insulin	int64
BMI	float64
DiabetesPedigreeFunction	float64
Age	int64
Outcome	int64

dtype: object

```
diabetes.shape[0] #3
```

```
768
```

```
diabetes.head(20) #4
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1
10	4	110	92	0	0	37.6	0.191	30	0
11	10	168	74	0	0	38.0	0.537	34	1
12	10	139	80	0	0	27.1	1.441	57	0
13	1	189	60	23	846	30.1	0.398	59	1
14	5	166	72	19	175	25.8	0.587	51	1
15	7	100	0	0	0	30.0	0.484	32	1
16	0	118	84	47	230	45.8	0.551	31	1
17	7	107	74	0	0	29.6	0.254	31	1
18	1	103	30	38	83	43.3	0.183	33	0
19	1	115	70	30	96	34.6	0.529	32	1

Next steps: Generate code with diabetes | View recommended plots | New interactive sheet

```
diabetes.tail(20) #5
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
748	3	187	70	22	200	36.4	0.408	36	1
749	6	162	62	0	0	24.3	0.178	50	1
750	4	136	70	0	0	31.2	1.182	22	1
751	1	121	78	39	74	39.0	0.261	28	0
752	3	108	62	24	0	26.0	0.223	25	0
753	0	181	88	44	510	43.3	0.222	26	1
754	8	154	78	32	0	32.4	0.443	45	1
755	1	128	88	39	110	36.5	1.057	37	1
756	7	137	90	41	0	32.0	0.391	39	0
757	0	123	72	0	0	36.3	0.258	52	1
758	1	106	76	0	0	37.5	0.197	26	0
759	6	190	92	0	0	35.5	0.278	66	1
760	2	88	58	26	16	28.4	0.766	22	0
761	9	170	74	31	0	44.0	0.403	43	1
762	9	89	62	0	0	22.5	0.142	33	0
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

```
diabetes = diabetes.rename(columns = {'Outcome': 'Diagnosis'}) #6
diabetes.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Next steps: Generate code with diabetes | View recommended plots | New interactive sheet

```
diabetes['Classification'] = (diabetes['Diagnosis'] == 1).map({True: 'Diabetes', False: 'No Diabetes'}) #7
diabetes.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classification
0	6	148	72	35	0	33.6	0.627	50	1	Diabetes
1	1	85	66	29	0	26.6	0.351	31	0	No Diabetes
2	8	183	64	0	0	23.3	0.672	32	1	Diabetes
3	1	89	66	23	94	28.1	0.167	21	0	No Diabetes
4	0	137	40	35	168	43.1	2.288	33	1	Diabetes

Next steps: Generate code with diabetes | View recommended plots | New interactive sheet

```
withDiabetes = diabetes[diabetes['Diagnosis'] == 1] #8
withDiabetes.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classification
0	6	148	72	35	0	33.6	0.627	50	1	Diabetes
2	8	183	64	0	0	23.3	0.672	32	1	Diabetes
4	0	137	40	35	168	43.1	2.288	33	1	Diabetes
6	3	78	50	32	88	31.0	0.248	26	1	Diabetes
8	2	197	70	45	543	30.5	0.158	53	1	Diabetes

Next steps: Generate code with withDiabetes | View recommended plots | New interactive sheet

```
noDiabetes = diabetes[diabetes['Diagnosis'] == 0] #9
noDiabetes.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classification
1	1	85	66	29	0	26.6	0.351	31	0	No Diabetes
3	1	89	66	23	94	28.1	0.167	21	0	No Diabetes
5	5	116	74	0	0	25.6	0.201	30	0	No Diabetes
7	10	115	0	0	0	35.3	0.134	29	0	No Diabetes
10	4	110	92	0	0	37.6	0.191	30	0	No Diabetes

Next steps: Generate code with noDiabetes | View recommended plots | New interactive sheet

```
Pedia  = diabetes[(diabetes['Age'] >= 0) & (diabetes['Age'] <= 19)] #10
Pedia.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classification

```
Adult = diabetes[diabetes['Age'] > 19] #11
Adult.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classification
0	6	148	72	35	0	33.6	0.627	50	1	Diabetes
1	1	85	66	29	0	26.6	0.351	31	0	No Diabetes
2	8	183	64	0	0	23.3	0.672	32	1	Diabetes
3	1	89	66	23	94	28.1	0.167	21	0	No Diabetes
4	0	137	40	35	168	43.1	2.288	33	1	Diabetes

Next steps: Generate code with Adult | View recommended plots | New interactive sheet

```
import numpy as np #12

age = np.average(diabetes['Age'])
glucose = np.average(diabetes['Glucose'])

print(f'Average age: {age}')
print(f'Average glucose: {glucose}')
```

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

```
age = np.sort(diabetes['Age']) #13
glucose = np.sort(diabetes['Glucose'])
mage = np.median(age)
mglucose = np.median(glucose)

print(f'Median age: {mage}')
print(f'Median glucose: {mglucose}')
```

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

```
age = np.sort(diabetes['Age']) #14
glucose = np.sort(diabetes['Glucose'])
mage = np.median(age)
mglucose = np.median(glucose)

print(f'Median age: {mage}')
print(f'Median glucose: {mglucose}')
```

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

```
std = np.std(diabetes['SkinThickness']) #15
std
```

```
15.941828626496978
```

6.4 Conclusion

To conclude this, during this lab activity, I gained lots of experience using numpy with stats mod and without stats module by thinking outside the box, also I applied what I learned on sir Romans class about basic pandas here. this helped me learn statistical calculations that will help me throughout my future study in data analysis.

All in all, practicing coding is really helpful and it will help me throughout my study in TIP