Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: Valleser, Kenn Jie L.

Section: CPE22S3

Performed on: 05/04/2025

Submitted on: 05/04/2025

Submitted to: Engr. Roman M. Richard

6.1 Intended Learning Outcome . Use pandas and numpy data analysis tools. . 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 Py

In [ ]:

```
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

In [50]:

```
from statistics import mean, median, mode, variance, stdev
```

In [ ]:

```
# Write a comment per statistical function
```

In [ ]:

```
#mean
#with statistics module
avg = mean(salaries)
print(avg)
```

```
585690.0
```

In [ ]:

```
#without statistics module
manual_mean = sum(salaries) / len(salaries)
manual_mean
```

Out[ ]:

585690.0

In [ ]:

```
#median
#without statistics module
sorted_salaries = sorted(salaries)
n = len(sorted_salaries)

if n % 2 == 0:
    middle1 = sorted_salaries[n // 2 - 1]
    middle2 = sorted_salaries[n // 2]
    median_data = (middle1 + middle2) / 2
else:
    median_data = sorted_salaries[n // 2]

median_data
```

Out[ ]:

589000.0

In [ ]:

```
#with statistics module
median_salary = median(salaries)
median_salary
```

Out[ ]:

589000.0

In [ ]:

```
#mode
#Without stastatistics module
from collections import Counter
counter = Counter(salaries)
manual_mode = counter.most_common(1)[0][0]
manual_mode
```

Out[ ]:

477000.0

In [51]:

```
#with statistics module
mode_salary = mode(salaries)
mode_salary
```

Out[51]:

477000.0

In [52]:

```
#Sample Variance
#without statistics module
manual_variance = sum((x - manual_mean) ** 2 for x in salaries) / (n - 1)
manual_variance
```

Out[52]:

70664054444.44444

In [53]:

```
#with statistics module
variance_salary = variance(salaries)
variance_salary
```

Out[53]:

```
70664054444.44444
```

In [54]:

```
#Sample Standard Deviation
#without statistics module
manual_stdev = manual_variance ** 0.5
manual_stdev
```

Out[54]:

```
265827.11382484
```

In [46]:

```
#with statistics module
stdev_salary = stdev(salaries)
stdev_salary
```

Out[46]:

```
265827.11382484
```

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

In [ ]:

```
# Write a comment per statistical function
```

In [36]:

```
from statistics import mean, stdev, quantiles
```

In [45]:

```
#Range
salary_range = max(salaries) - min(salaries)
salary_range
```

Out[45]:

```
995000.0
```

In [44]:

```
#Coefficient of Variation (CV)
cv = (stdev(salaries) / mean(salaries)) * 100
cv
```

Out[44]:

```
45.38699889443903
```

In [37]:

```
covariation = (stdev(salaries) / mean(salaries)) * 100
q1, q2, q3 = quantiles(salaries, n = 4)
qrange = q3 - q1
print("The Coefficient Variation is ",covariation)
print("The Interquartile Range is ", qrange)
```

```
The Coefficient Variation is  45.38699889443903
The Interquartile Range is  421750.0
```

In [43]:

```
#Quartile Coefficient of Dispersion (QCD)
q1, q2, q3 = quantiles(salaries, n = 4)
qcd = (q3 - q1) / (q3 + q1)
```

```
qcd
```

Out[43]:

```
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**
    A. **Create a new dataframe "Adult" that gathers data with age greater than 19**
11. **Use numpy to get the average age and glucose value.**
12. **Use numpy to get the median age and glucose value.**
13. **Use numpy to get the middle values of glucose and age.**
14. **Use numpy to get the standard deviation of the skinthickness.**

In [ ]:

```
# Indicate which item you're answering with a comment
```

In [ ]:

```
import pandas as pd
diabetes = pd.read_csv("diabetes.csv")
df = pd.DataFrame(diabetes)
df.head()
```

Out[ ]:

	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

In [ ]:

```
#1. Identify the column name
df.columns
```

Out[ ]:

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

In [ ]:

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

Out[ ]:

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

dtype: object

In [ ]:

```
#3.Display the total number of records
df.shape[0]
```

Out[ ]:

768

In [ ]:

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

Out[ ]:

	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

In [ ]:

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

Out[ ]:

	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

In [ ]:

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

Out[ ]:

	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
...	...	...	...	...	...	...	...	...	...
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

768 rows × 9 columns

In [ ]:

```
# 7. Create a new column Classification that displays "Diabetes" if the value of outcome is 1, otherwise "No Diabetes"
df['Classification'] = df['Diagnosis'].apply(lambda x: 'Diabetes' if x == 1 else 'No Diabetes')
df
```

Out[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classifica
0	6	148	72	35	0	33.6	0.627	50	1	Diabe
1	1	85	66	29	0	26.6	0.351	31	0	No Diabe
2	8	183	64	0	0	23.3	0.672	32	1	Diabe
3	1	89	66	23	94	28.1	0.167	21	0	No Diabe
4	0	137	40	35	168	43.1	2.288	33	1	Diabe
...	...	...	...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.171	63	0	No Diabe
764	2	122	70	27	0	36.8	0.340	27	0	No Diabe
765	5	121	72	23	112	26.2	0.245	30	0	No Diabe
766	1	126	60	0	0	30.1	0.349	47	1	Diabe
767	1	93	70	31	0	30.4	0.315	23	0	No Diabe

768 rows × 10 columns

In [ ]:

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

Out[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classifica
0	6	148	72	35	0	33.6	0.627	50	1	Diabe
2	8	183	64	0	0	23.3	0.672	32	1	Diabe
4	0	137	40	35	168	43.1	2.288	33	1	Diabe
6	3	78	50	32	88	31.0	0.248	26	1	Diabe
8	2	197	70	45	543	30.5	0.158	53	1	Diabe
...	...	...	...	...	...	...	...	...	...	
755	1	128	88	39	110	36.5	1.057	37	1	Diabe
757	0	123	72	0	0	36.3	0.258	52	1	Diabe
759	6	190	92	0	0	35.5	0.278	66	1	Diabe
761	9	170	74	31	0	44.0	0.403	43	1	Diabe
766	1	126	60	0	0	30.1	0.349	47	1	Diabe

268 rows × 10 columns

In [ ]:

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

Out[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classifica
1	1	85	66	29	0	26.6	0.351	31	0	No Diabe
3	1	89	66	23	94	28.1	0.167	21	0	No Diabe
5	5	116	74	0	0	25.6	0.201	30	0	No Diabe
7	10	115	0	0	0	35.3	0.134	29	0	No Diabe
10	4	110	92	0	0	37.6	0.191	30	0	No Diabe
...	...	...	...	...	...	...	...	...	...	
762	9	89	62	0	0	22.5	0.142	33	0	No Diabe
763	10	101	76	48	180	32.9	0.171	63	0	No Diabe
764	2	122	70	27	0	36.8	0.340	27	0	No Diabe
765	5	121	72	23	112	26.2	0.245	30	0	No Diabe
767	1	93	70	31	0	30.4	0.315	23	0	No Diabe

500 rows × 10 columns

In [ ]:

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

Out[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classification

In [ ]:

```
# 11. Create a new dataframe "Adult" that gathers data with age greater than 19
Adult = df[df['Age'] > 19]
Adult
```

Out[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classifica
0	6	148	72	35	0	33.6	0.627	50	1	Diabe
1	1	85	66	29	0	26.6	0.351	31	0	No Diabe
2	8	183	64	0	0	23.3	0.672	32	1	Diabe
3	1	89	66	23	94	28.1	0.167	21	0	No Diabe
4	0	137	40	35	168	43.1	2.288	33	1	Diabe
...	...	...	...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.171	63	0	No Diabe
764	2	122	70	27	0	36.8	0.340	27	0	No Diabe
765	5	121	72	23	112	26.2	0.245	30	0	No Diabe
766	1	126	60	0	0	30.1	0.349	47	1	Diabe
767	1	93	70	31	0	30.4	0.315	23	0	No Diabe

768 rows × 10 columns

In [ ]:

```
# 12. Use numpy to get the average age and glucose value
```

```
import numpy as np
avg_age = np.mean(df['Age'])
avg_glucose = np.mean(df['Glucose'])
print("Average Age:", avg_age)
print("Average Glucose:", avg_glucose)
```

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

In [ ]:

```
# 13. Use numpy to get the median age and glucose value
median_age = np.median(df['Age'])
median_glucose = np.median(df['Glucose'])
print("Median Age:", median_age)
print("Median Glucose:", median_glucose)
```

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

In [ ]:

```
# 14. Use numpy to get the middle values of glucose and age
median_age = np.median(df['Age'])
median_glucose = np.median(df['Glucose'])
print("Middle Value of Age:", median_age)
print("Middle Value pf Glucose:", median_glucose)
```

```
Middle Value of Age: 29.0
Middle Value pf Glucose: 117.0
```

In [ ]:

```
# 15. Use numpy to get the standard deviation of the skin thickness
std_skin_thickness = np.std(df['SkinThickness'])
print("Standard Deviation of Skin Thickness:", std_skin_thickness)
```

```
Standard Deviation of Skin Thickness: 15.941828626496978
```

6.4 Conclusion

In this activity, I learned how to use Pandas and Numpy to analyze data. I did different tasks to get more comfortable with these tools.

In Exercise 1, I calculated statistics like the mean, median, mode, sample variance, and sample standard deviation by hand (without using the statistics module). Then, I checked my results using the statistics module to see if they matched. This helped me understand how these calculations work and how to do them manually.

In Exercise 2, I used the statistics module to find more statistics like range, coefficient of variation, interquartile range, and quartile coefficient of dispersion. This showed me how easy it is to get these values using built-in functions.

In Exercise 3, I worked with a real dataset (diabetes.csv) using Pandas. I learned how to load the data, check column names, and filter records based on certain conditions (like people with diabetes or age groups). I also calculated the average, median, and standard deviation of specific columns using Numpy. This exercise showed me how to manipulate and analyze data effectively.