Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: Carigo, Naira Jezreel

Section: CPE22S3

Performed on: 04/05/2025

Submitted on: 04/05/2025

Submitted to: Engr. Roman M. Richard

Exercise 1

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

In [5]:

```python
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 [7]:

```python
import random
from collections import Counter

# Generate 100 salary values using random
random.seed(0)
salaries = [round(random.random() * 1000000, -3) for _ in range(100)]

# Mean
# sum of all values divided by the number of values
total = 0
for s in salaries:
    total += s
mean = total / len(salaries)

# Median
# sort the list and get the middle value(s)
sorted_salaries = sorted(salaries)
n = len(sorted_salaries)
if n % 2 == 0:
    median = (sorted_salaries[n // 2 - 1] + sorted_salaries[n // 2]) / 2
```

```
else:
    median = sorted_salaries[n // 2]

# Mode
# use Counter to count frequencies and find the most common value(s)
frequency = Counter(salaries)
max_count = max(frequency.values())
mode = [val for val, count in frequency.items() if count == max_count]

# Sample Variance
# use the formula: variance = Σ(x - mean)² / (n - 1)
sum_squared_diff = 0
for s in salaries:
    sum_squared_diff += (s - mean) ** 2
variance = sum_squared_diff / (n - 1)

# Sample Standard Deviation
# square root of the variance (using **0.5 instead of math.sqrt)
std_dev = variance ** 0.5

# Display results
print("Exercise 1 Results:")
print("Mean:", mean)
print("Median:", median)
print("Mode:", mode)
print("Sample Variance:", variance)
print("Sample Standard Deviation:", std_dev)
```

```
Exercise 1 Results:
Mean: 585690.0
Median: 589000.0
Mode: [477000.0]
Sample Variance: 70664054444.44444
Sample Standard Deviation: 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 [12]:

```
import statistics

# Range
# highest salary minus lowest salary
range_val = max(salaries) - min(salaries)

# Coefficient of Variation
# (Standard Deviation / Mean) × 100
cv = (statistics.stdev(salaries) / statistics.mean(salaries)) * 100

# Interquartile Range (IQR)
# use statistics.quantiles with n=4 for quartiles
quartiles = statistics.quantiles(salaries, n=4)
```

```
q1 = quartiles[0]  # 25th percentile
q3 = quartiles[2]  # 75th percentile
iqr = q3 - q1

# Quartile Coefficient of Dispersion (QCD)
# (Q3 - Q1) / (Q3 + Q1)
qcd = (q3 - q1) / (q3 + q1)

# Display results
print("\nExercise 2 Results:")
print("Range:", range_val)
print("Coefficient of Variation: {:.2f}%".format(cv))
print("Interquartile Range (IQR):", iqr)
print("Quartile Coefficient of Dispersion (QCD):", round(qcd, 4))
```

```
Exercise 2 Results:
Range: 995000.0
Coefficient of Variation: 45.39%
Interquartile Range (IQR): 421750.0
Quartile Coefficient of Dispersion (QCD): 0.3449
```

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.

In [20]:

```
import pandas as pd
import numpy as np

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

Out[20]:

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

In [18]:

```
# 1. display all column names in the dataset
df.columns
```

Out[18]:

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

In [22]:

```
# 2. display data types for each column
df.dtypes
```

Out[22]:

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

In [24]:

```
# 3. get the total number of rows in the dataset
len(df)
```

Out[24]:

```
768
```

In [26]:

```
# 4. show the first 20 rows of the DataFrame
df.head(20)
```

Out[26]:

	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 [28]:

```
# 5. show the last 20 rows of the DataFrame
df.tail(20)
```

Out[28]:

	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 [58]:

```
# 6. rename the column 'Outcome' to 'Diagnosis'
df = df.rename(columns={"Outcome": "Diagnosis"})
df
```

Out[58]:

	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 [68]:

```
# 7. add a new column that shows 'Diabetes' if Diagnosis is 1, otherwise 'No Diabet
df["Classification"] = ["Diabetes" if x == 1 else "No Diabetes" for x in df["Diagno
df[["Diagnosis", "Classification"]]
```

Out[68]:

	Diagnosis	Classification
0	1	Diabetes
1	0	No Diabetes
2	1	Diabetes
3	0	No Diabetes
4	1	Diabetes
...	...	...
763	0	No Diabetes
764	0	No Diabetes
765	0	No Diabetes
766	1	Diabetes
767	0	No Diabetes

768 rows × 2 columns

In [64]:
```
# 8. filter and store rows where Diagnosis is 1 (has diabetes)
withDiabetes = df[df["Diagnosis"] == 1]
withDiabetes.head()
```

Out[64]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0
2	8	183	64	0	0	23.3	0
4	0	137	40	35	168	43.1	2
6	3	78	50	32	88	31.0	0
8	2	197	70	45	543	30.5	0

In [70]:
```
# 9. filter and store rows where Diagnosis is 0 (no diabetes)
noDiabetes = df[df["Diagnosis"] == 0]
noDiabetes.head()
```

Out[70]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFur
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	

In [72]:
```
# 10. filter and store rows for pediatric patients (age 0 to 19)
Pedia = df[df["Age"] <= 19]
Pedia.head()
```

Out[72]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct

In [74]:
```
# 11. filter and store rows for adult patients (age greater than 19)
Adult = df[df["Age"] > 19]
Adult.head()
```

Out[74]:

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

In [44]:

```
# 12.
#calculate the average (mean) of Age using numpy
avg_age = np.mean(df["Age"])

# calculate the average (mean) of Glucose using numpy
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 [46]:

```
# 13.
# Calculate the median Age using numpy
median_age = np.median(df["Age"])

# Calculate the median Glucose using numpy
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 [52]:

```
# 14.
# get the middle index of the dataset
middle_index = len(df) // 2

# get the age value at the middle index
middle_age = df["Age"].iloc[middle_index]

# get the glucose value at the middle index
middle_glucose = df["Glucose"].iloc[middle_index]

print("Middle Age:", middle_age)
print("Middle Glucose:", middle_glucose)
```

```
Middle Age: 25
Middle Glucose: 125
```

In [78]:

```
# 15. use numpy to calculate the standard deviation of SkinThickness
std_skin = np.std(df["SkinThickness"], ddof=1)
```

```
print("Standard Deviation of SkinThickness:", std_skin)
```

```
Standard Deviation of SkinThickness: 15.952217567727677
```