Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

```
"""
 Name: Masangkay, Frederick
 Section: CPE22S3
 Performed on: 04-05-2025
 Submitted on: 04-05-2025
 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 Python modules.

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

print(salaries)
```

```
[844000.0, 758000.0, 421000.0, 259000.0, 511000.0, 405000.0, 784000.0, 303000.0, 477000.0, 583000.0, 908000.0, 505000.0, 282000.0, 756000.0, 618000.0, 2
```

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

```
# Mean
def mean(n):
    mean = sum(n) / len(n) # Get the total sum, with sum(), of the salaries first then divide in on the total number of salaries with the use of len()
    return mean
mean(salaries)
```

```
585690.0
```

```
# median
def median(salaries):
    sorted_salaries = sorted(salaries)
    # Calculate median
    n = len(sorted_salaries)

    if n % 2 != 0: # the number of salaries is odd
        median = sorted_salaries[n // 2] # the value is in the middle of all the data
        return median
    else: # salaries length is even
        median = (sorted_salaries[n // 2 - 1] + sorted_salaries[n // 2]) / 2 # the value is the avergae of above and lower value of the middle value
        return median

median(salaries)
```

```
589000.0
```

```
# mode
from collections import Counter
```

```
counter = Counter(salaries)  # Count the occurrences of each element
most_common = counter.most_common(1) # Find the most common element(s)
most_common
```

```
[(477000.0, 3)]
```

```
# sample variance
def sample_variance(salaries):
    mean = sum(salaries) / len(salaries)

    squared_diff_sum = 0 # to store values for sum of squared differences

    for salary in salaries:  # Loop through each salary to calculate squared difference from the mean
        squared_diff_sum += (salary - mean) ** 2

    variance = squared_diff_sum / (len(salaries) - 1)  # Divide by (n - 1) to get sample variance
    return variance

sample_variance(salaries)
```

```
70664054444.44444
```

```
# sample standard deviation
def sample_standard_deviation(salaries):
    sample_standard_deviation = sample_variance(salaries) ** 0.5 # multiply the previous avluevalue from the variance by 0.5 to get the standard deviation
    return sample_standard_deviation

sample_standard_deviation(salaries)
```

```
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

```
# Range
def range(n):
    return max(n) - min(n)
range(salaries)
```

```
995000.0
```

```
# Coefficient of variation Interquartile range
from statistics import median, quantiles

def coefficient_vir(salaries):
    sorted_data = sorted(salaries)  # Sort the input data

    q1_q3 = quantiles(sorted_data, n = 4)  # Divide into quartiles

    iqr = q1_q3[2] - q1_q3[0]  # IQR = Q3 - Q1
    med = median(sorted_data)  # Median of the data

    return 100 * (iqr / med)

coefficient_vir(salaries)
```

```
71.60441426146011
```

```
# Quartile coefficient of dispersion
from statistics import quantiles
def quartileC_dispersion(salaries):
    sorted_data = sorted(salaries)
    quartiles = quantiles(sorted_data, n=4)  # Get Q1 and Q3

    q1 = quartiles[0]
    q3 = quartiles[2]

    return (q3 - q1) / (q3 + q1)  # Quartile Coefficient of Dispersion

quartileC_dispersion(salaries)
```

```
0.34491923941934166
```

Exercise 3

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

Perform the following tasks in the diabetes dataframe: . Identify the column names . Identify the data types of the data . Display the total number of records . Display the first 20 records . Display the last 20 records . Change the Outcome column to Diagnosis . Create a new column Classification that display "Diabetes" if the value of outcome is 1 , otherwise "No Diabetes" . Create a new dataframe "withDiabetes" that gathers data with diabetes . Create a new dataframe "noDiabetes" thats gathers data with no diabetes . Create a new dataframe "Pedia" that gathers data with age 0 to 19 . Create a new dataframe "Adult" that gathers data with age greater than 19 . Use numpy to get the average age and glucose value. . Use numpy to get the median age and glucose value. . Use numpy to get the middle values of glucose and age. . Use numpy to get the standard deviation of the skinthickness

```
from google.colab import files
files = files.upload()
```

```
Choose Files diabetes.csv
• diabetes.csv(text/csv) - 23873 bytes, last modified: 4/5/2025 - 100% done
Saving diabetes.csv to diabetes.csv
```

```
import pandas as pd
import numpy as np
```

```
df = pd.read_csv('diabetes.csv')
```

```
# 1. Identify the column names
print("Column Names:", df.columns)
```

```
Column Names: Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Diagnosis',
       'Classification'],
      dtype='object')
```

```
# 2. Identify the data types of the data
print("Data Types:", df.dtypes)
```

```
Data Types: Pregnancies                 int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Diagnosis                     int64
Classification               object
dtype: object
```

```
# 3. Display the total number of records
print("Total Records:", len(df))
```

```
Total Records: 768
```

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

	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
5	5	116	74	0	0	25.6	0.201	30	0	No Diabetes
6	3	78	50	32	88	31.0	0.248	26	1	Diabetes
7	10	115	0	0	0	35.3	0.134	29	0	No Diabetes
8	2	197	70	45	543	30.5	0.158	53	1	Diabetes
9	8	125	96	0	0	0.0	0.232	54	1	Diabetes
10	4	110	92	0	0	37.6	0.191	30	0	No Diabetes
11	10	168	74	0	0	38.0	0.537	34	1	Diabetes
12	10	139	80	0	0	27.1	1.441	57	0	No Diabetes
13	1	189	60	23	846	30.1	0.398	59	1	Diabetes
14	5	166	72	19	175	25.8	0.587	51	1	Diabetes
15	7	100	0	0	0	30.0	0.484	32	1	Diabetes
16	0	118	84	47	230	45.8	0.551	31	1	Diabetes
17	7	107	74	0	0	29.6	0.254	31	1	Diabetes
18	1	103	30	38	83	43.3	0.183	33	0	No Diabetes
19	1	115	70	30	96	34.6	0.529	32	1	Diabetes

Next steps: View recommended plots | New interactive sheet

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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classification
748	3	187	70	22	200	36.4	0.408	36	1	Diabetes
749	6	162	62	0	0	24.3	0.178	50	1	Diabetes
750	4	136	70	0	0	31.2	1.182	22	1	Diabetes
751	1	121	78	39	74	39.0	0.261	28	0	No Diabetes
752	3	108	62	24	0	26.0	0.223	25	0	No Diabetes
753	0	181	88	44	510	43.3	0.222	26	1	Diabetes
754	8	154	78	32	0	32.4	0.443	45	1	Diabetes
755	1	128	88	39	110	36.5	1.057	37	1	Diabetes
756	7	137	90	41	0	32.0	0.391	39	0	No Diabetes
757	0	123	72	0	0	36.3	0.258	52	1	Diabetes
758	1	106	76	0	0	37.5	0.197	26	0	No Diabetes
759	6	190	92	0	0	35.5	0.278	66	1	Diabetes
760	2	88	58	26	16	28.4	0.766	22	0	No Diabetes
761	9	170	74	31	0	44.0	0.403	43	1	Diabetes
762	9	89	62	0	0	22.5	0.142	33	0	No Diabetes
763	10	101	76	48	180	32.9	0.171	63	0	No Diabetes
764	2	122	70	27	0	36.8	0.340	27	0	No Diabetes
765	5	121	72	23	112	26.2	0.245	30	0	No Diabetes
766	1	126	60	0	0	30.1	0.349	47	1	Diabetes
767	1	93	70	31	0	30.4	0.315	23	0	No Diabetes

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

	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
...	...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0	No Diabetes
764	2	122	70	27	0	36.8	0.340	27	0	No Diabetes
765	5	121	72	23	112	26.2	0.245	30	0	No Diabetes
766	1	126	60	0	0	30.1	0.349	47	1	Diabetes
767	1	93	70	31	0	30.4	0.315	23	0	No Diabetes

768 rows × 10 columns

Next steps: View recommended plots | New interactive sheet

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

	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
...	...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0	No Diabetes
764	2	122	70	27	0	36.8	0.340	27	0	No Diabetes
765	5	121	72	23	112	26.2	0.245	30	0	No Diabetes
766	1	126	60	0	0	30.1	0.349	47	1	Diabetes
767	1	93	70	31	0	30.4	0.315	23	0	No Diabetes

768 rows × 10 columns

Next steps: View recommended plots | New interactive sheet

```
# 8. Create a new dataframe 'withDiabetes' that gathers data with diabetes
withDiabetes = df[df.Classification == 'Diabetes']
withDiabetes
```

	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
...	...	...	...	...	...	...	...	...	...	...
755	1	128	88	39	110	36.5	1.057	37	1	Diabetes
757	0	123	72	0	0	36.3	0.258	52	1	Diabetes
759	6	190	92	0	0	35.5	0.278	66	1	Diabetes
761	9	170	74	31	0	44.0	0.403	43	1	Diabetes
766	1	126	60	0	0	30.1	0.349	47	1	Diabetes

268 rows × 10 columns

Next steps: View recommended plots | New interactive sheet

```
# 9. Create a new dataframe 'noDiabetes' that gathers data with no diabetes
noDiabetes = df[df.Classification == 'No Diabetes']
noDiabetes
```

	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
...	...	...	...	...	...	...	...	...	...	...
762	9	89	62	0	0	22.5	0.142	33	0	No Diabetes
763	10	101	76	48	180	32.9	0.171	63	0	No Diabetes
764	2	122	70	27	0	36.8	0.340	27	0	No Diabetes
765	5	121	72	23	112	26.2	0.245	30	0	No Diabetes
767	1	93	70	31	0	30.4	0.315	23	0	No Diabetes

500 rows × 10 columns

Next steps: View recommended plots | New interactive sheet

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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classification

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

	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
...	...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0	No Diabetes
764	2	122	70	27	0	36.8	0.340	27	0	No Diabetes
765	5	121	72	23	112	26.2	0.245	30	0	No Diabetes
766	1	126	60	0	0	30.1	0.349	47	1	Diabetes
767	1	93	70	31	0	30.4	0.315	23	0	No Diabetes

768 rows × 10 columns

Next steps: View recommended plots | New interactive sheet

```
# 12. Use numpy to get the average age and glucose value
average_age = np.mean(df['Age'])
average_glucose = np.mean(df['Glucose'])
print("Average Age:", average_age)
print("Average Glucose:", average_glucose)
```

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

```
# 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
```

```python
# 14. Use numpy to get the middle values of glucose and age (i.e., the 50th percentile)
middle_age = np.percentile(df['Age'], 50)
middle_glucose = np.percentile(df['Glucose'], 50)
print("Middle Age (50th percentile):", middle_age)
print("Middle Glucose (50th percentile):", middle_glucose)
```

```
Middle Age (50th percentile): 29.0
Middle Glucose (50th percentile): 117.0
```

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