Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: Silang, Elijah Yohance
Section: CPE22S3
Performed on: 04/05/2025
Submitted on: 04/05/2025
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
- 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 numpy as np
```

```
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

Mean

```
mean_result = np.mean(salaries)
print(mean_result)

# This outputs the mean of multiple values.
# Basically its sum of all values divided by the number of values.
```

```
585690.0
```

Median

```
median_result = np.median(salaries)
median_result = np.median(salaries)
print(median_result)

# This outputs the median of multiple values.
```

```
589000.0
```

Mode

```
counter(salaries)

# I don't know. yet.
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-126-7ce7585ed784> in <cell line: 0>()
----> 1 np.counter(salaries).mostcommon(3)

/usr/local/lib/python3.11/dist-packages/numpy/__init__.py in __getattr__(attr)
    408             return char.chararray
    409 
--> 410         raise AttributeError("module {!r} has no attribute "
    411                              "{!r}".format(__name__, attr))
    412 

AttributeError: module 'numpy' has no attribute 'counter'
```

Sample Variance

```
sv_result = np.var(salaries)
print(sv_result)

# This outputs the variance of the data.
```

```
69957413900.0
```

Sample Standard Deviation

```
std_result = np.std(salaries)
print(std_result)

# This outputs the standard deviation of multiple values.
```

```
264494.6386980273
```

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

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

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

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
import numpy as np

diabetes = pd.read_csv('/content/diabetes.csv')
diabetes = diabetes.fillna(0)
diabetes.head()
```

	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: View recommended plots New interactive sheet

1. Identify the column names

```
diabetes.columns
```

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

2. Identify the data types of the data

```
diabetes.dtypes
```

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

dtype: object

3. Display the total number of records

```
diabetes.shape
```

```
(768, 9)
```

4. Display the first 20 records

```
diabetes.head(20)
```

	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: View recommended plots New interactive sheet

5. Display the last 20 records

```
diabetes.tail(20)
```

	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

6. Change the Outcome column to Diagnosis

```
new_diabetes = diabetes.rename(
  columns = {
    'Outcome' : 'Diagnosis'
  }
)
new_diabetes.columns
```

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

7. Create a new column Classification that display "Diabetes" if the value of outcome is 1 , otherwise "No Diabetes"

```
diabetes = diabetes.assign(
    Classification = lambda x: ['Diabetes' if outcome == 1 else 'No Diabetes' for outcome in x.Outcome]
)
diabetes.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	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: View recommended plots   New interactive sheet

8. Create a new dataframe "withDiabetes" that gathers data with diabetes

```
withDiabetes = diabetes[diabetes['Outcome'] == True]
withDiabetes.head()
```

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

Next steps: View recommended plots New interactive sheet

9. Create a new dataframe "noDiabetes" thats gathers data with no diabetes

```
noDiabetes = diabetes[diabetes['Outcome'] == False]
noDiabetes.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1	1	85	66	29	0	26.6	0.351	31	0
3	1	89	66	23	94	28.1	0.167	21	0
5	5	116	74	0	0	25.6	0.201	30	0
7	10	115	0	0	0	35.3	0.134	29	0
10	4	110	92	0	0	37.6	0.191	30	0

Next steps: View recommended plots New interactive sheet

10. Create a new dataframe "Pedia" that gathers data with age 0 to 19

```
Pedia = diabetes[diabetes['Age'] <= 19]
Pedia.head()
```

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome

11. Create a new dataframe "Adult" that gathers data with age greater than 19

```
Adult = diabetes[diabetes['Age'] >= 20]
Adult.head()
```

	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: View recommended plots New interactive sheet

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

```
ave_age = np.mean(diabetes['Age'])
ave_glucose = np.mean(diabetes['Glucose'])
print(f'Average age: {ave_age}\nAverage glucose: {ave_glucose}')
```

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

```
#If you prefer rounded off by 2 decimal values.
print(f'Average age: {ave_age:.2f}\nAverage glucose: {ave_glucose:.2f}')
```

```
Average age: 33.24
Average glucose: 120.89
```

13. Use numpy to get the median age and glucose value.

```
med_age = np.median(diabetes['Age'])
med_glucose = np.median(diabetes['Glucose'])
print(f'Median age: {med_age}\nMedian glucose: {med_glucose}')
```

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

14. Use numpy to get the middle values of glucose and age.

```
#sort first
sorted_age = np.sort(diabetes['Age'])
sorted_glucose = np.sort(diabetes['Glucose'])

middle_age = np.median(sorted_age)
middle_glucose = np.median(sorted_glucose)

print(f'Middle value for age: {middle_age}\nMiddle value for glucose: {middle_glucose}')
```

```
Middle value for age: 29.0
Middle value for glucose: 117.0
```

15. Use numpy to get the standard deviation of the skinthickness.

```
std_skinthickness = np.std(diabetes['SkinThickness'])
print(f'Stadard deviation of skin thickness: {std_skinthickness}')
```