Practical 3

Perform sorting on Series data and Data Frames

Theory

Series:

A Series is a one-dimensional array-like object that can hold various types of data, including integers, floats, strings, or Python objects. It is essentially a labeled array, where each element is associated with an index. The index is a set of labels that allows for fast lookups and alignment of data.

Key Characteristics:

- **Homogeneous data:** All elements in a Series must have the same data type.

- **Indexed:** Each element in a Series has a label, which can be used for selection and alignment.

- **Size Dynamism:** The size of a Series can be changed dynamically, similar to a Python list.

DataFrame:

A DataFrame is a two-dimensional labeled data structure with columns of potentially different types. It is similar to a spreadsheet or a SQL table, where data is organized in rows and columns. Each column in a DataFrame is a Series object, and all columns share the same index.

Key Characteristics:

- **Heterogeneous data:** Each column in a DataFrame can have a different data type.

- **Indexed:** Like Series, DataFrame objects also have an index, allowing for fast data alignment.

- **Column and Row Operations:** You can perform operations on both columns and rows of a DataFrame, such as adding or deleting columns, filtering rows, and aggregating data.

Conclusion:

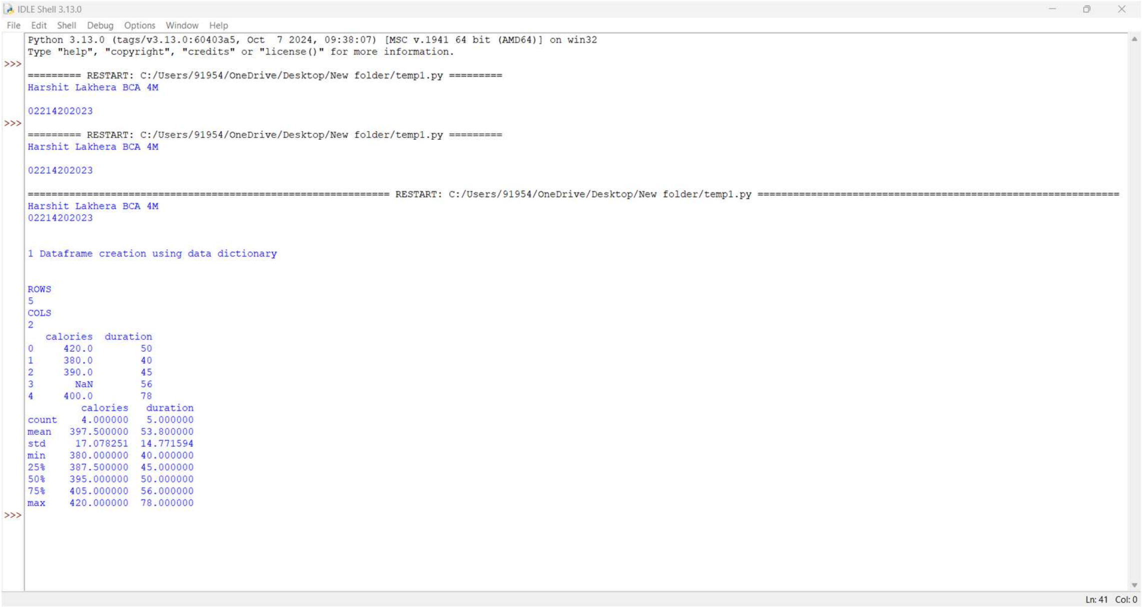
- Series and DataFrame are fundamental data structures in the Pandas library for data manipulation and analysis.

- Series is a one-dimensional array-like object with an index, while DataFrame is a two-dimensional tabular data structure with both row and column indices.

- Understanding and effectively using these data structures are essential skills for data analysis and manipulation tasks in Python.

Code

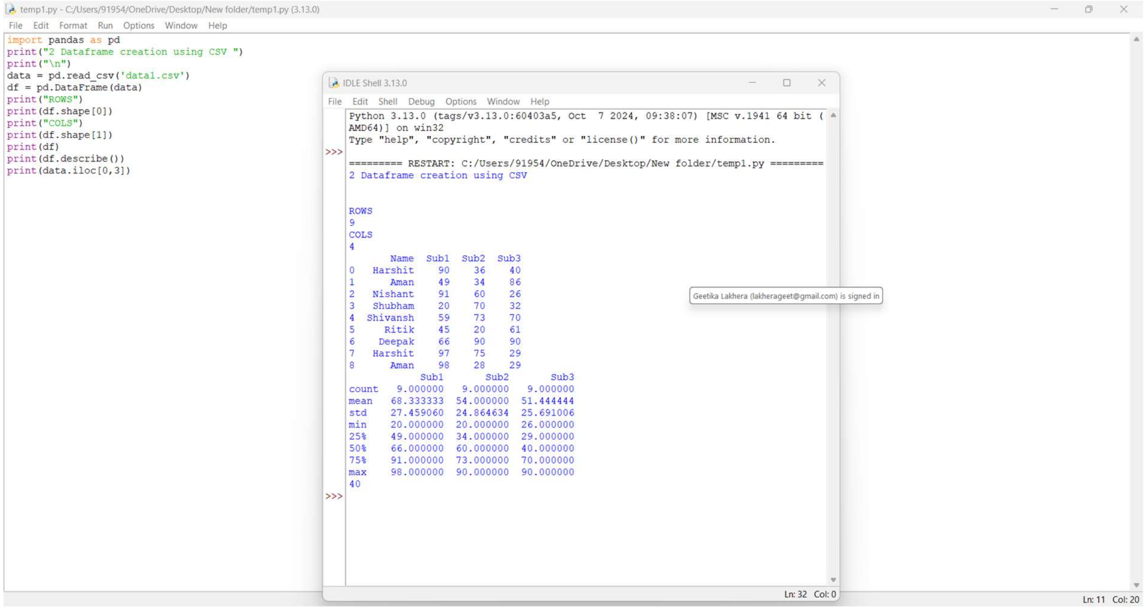
```
print("Harshit Lakhera BCA 4M")
print("02214202023")
print("\n")
print("1 Dataframe creation using data dictionary")
print("\n")
import pandas as pd
data = {"calories": [420, 380, 390,None,400],"duration": [50, 40, 45,56,78]}
df = pd.DataFrame(data)
print("ROWS")
print(df.shape[0])
print("COLS")
print(df.shape[1])
print(df)
print(df.describe())
```



```
Python 3.13.0 (tags/v3.13.0:60403a5, Oct  7 2024, 09:38:07) [MSC v.1941 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========= RESTART: C:/Users/91954/OneDrive/Desktop/New folder/temp1.py =========
Harshit Lakhera BCA 4M

02214202023
>>>
========= RESTART: C:/Users/91954/OneDrive/Desktop/New folder/temp1.py =========
Harshit Lakhera BCA 4M

02214202023

============================================================== RESTART: C:/Users/91954/OneDrive/Desktop/New folder/temp1.py ==============================================================
Harshit Lakhera BCA 4M
02214202023


1 Dataframe creation using data dictionary


ROWS
5
COLS
2
   calories  duration
0     420.0        50
1     380.0        40
2     390.0        45
3       NaN        56
4     400.0        78
         calories   duration
count    4.000000   5.000000
mean   397.500000  53.800000
std     17.078251  14.771594
min    380.000000  40.000000
25%    387.500000  45.000000
50%    395.000000  50.000000
75%    405.000000  56.000000
max    420.000000  78.000000
>>>
```

#2 Dataframe creation using CSV

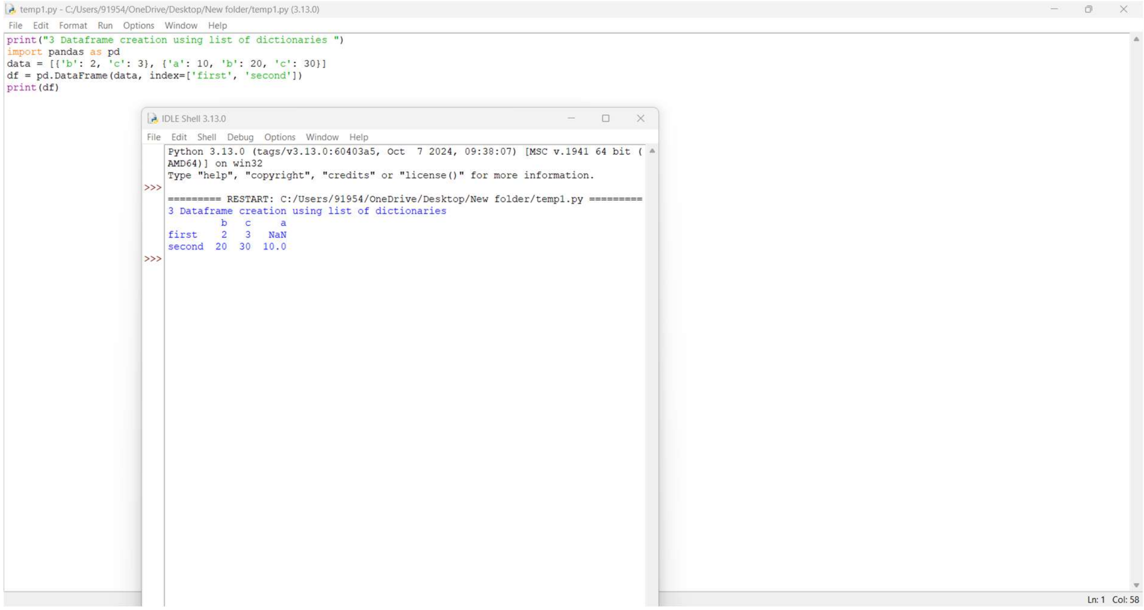
```
print("2  Dataframe creation using CSV ")
print("---------------------------------------------------------")
print("\n")
data = pd.read_csv(“data.csv”)
df = pd.DataFrame(data)
print("ROWS")
print(df.shape[0])
print("COLS")
print(df.shape[1])
print(df)
print(df.describe())
print(data.iloc[0,3])
```


temp1.py - C:/Users/91954/OneDrive/Desktop/New folder/temp1.py (3.13.0)
File Edit Format Run Options Window Help
IDLE Shell 3.13.0
File Edit Shell Debug Options Window Help
Geetika Lakhera (lakherageet@gmail.com) is signed in
Ln: 32 Col: 0
Ln: 11 Col: 20

```
import pandas as pd
print("2 Dataframe creation using CSV ")
print("\n")
data = pd.read_csv('data1.csv')
df = pd.DataFrame(data)
print("ROWS")
print(df.shape[0])
print("COLS")
print(df.shape[1])
print(df)
print(df.describe())
print(data.iloc[0,3])
```

```
Python 3.13.0 (tags/v3.13.0:60403a5, Oct  7 2024, 09:38:07) [MSC v.1941 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========= RESTART: C:/Users/91954/OneDrive/Desktop/New folder/temp1.py =========
2 Dataframe creation using CSV


ROWS
9
COLS
4
       Name  Sub1  Sub2  Sub3
0   Harshit    90    36    40
1      Aman    49    34    86
2   Nishant    91    60    26
3   Shubham    20    70    32
4  Shivansh    59    73    70
5     Ritik    45    20    61
6    Deepak    66    90    90
7   Harshit    97    75    29
8      Aman    98    28    29
            Sub1       Sub2       Sub3
count   9.000000   9.000000   9.000000
mean   68.333333  54.000000  51.444444
std    27.459060  24.864634  25.691006
min    20.000000  20.000000  26.000000
25%    49.000000  34.000000  29.000000
50%    66.000000  60.000000  40.000000
75%    91.000000  73.000000  70.000000
max    98.000000  90.000000  90.000000
40
>>>
```

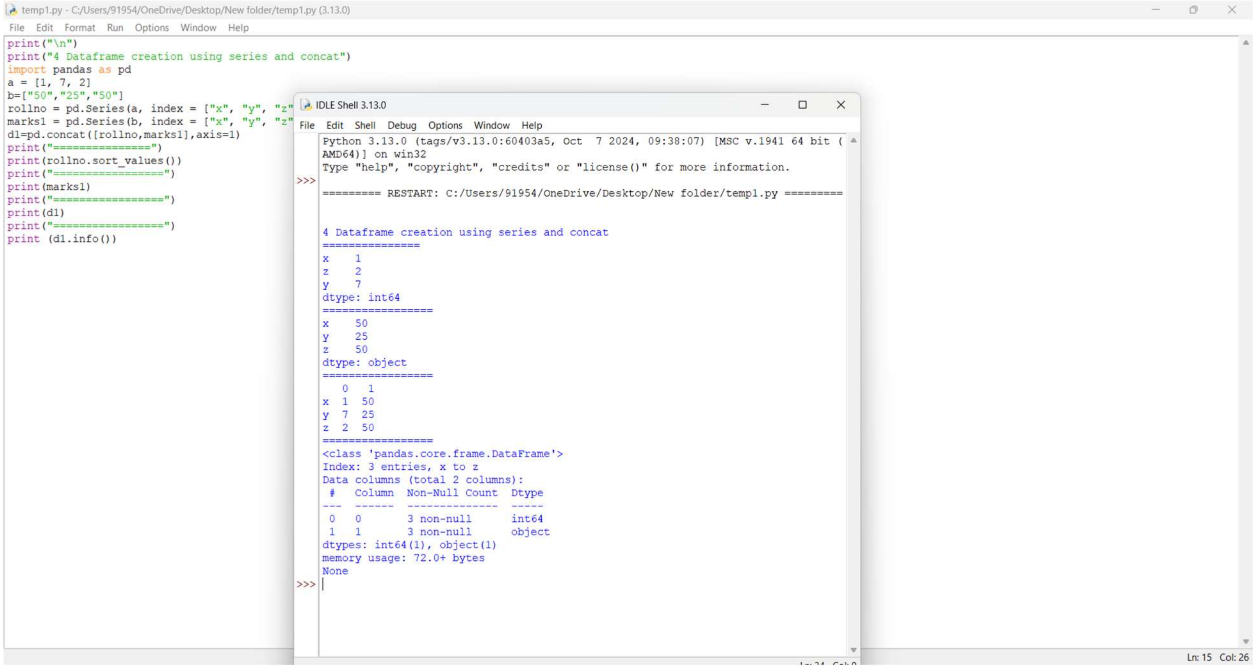
```
#3 Dataframe creation using lits of dictionaries

print("3 Dataframe creation using list of dictionaries ")
print("-------------------------------------------------------")

import pandas as pd
data = [{'b': 2, 'c': 3}, {'a': 10, 'b': 20, 'c': 30}]
df = pd.DataFrame(data, index=['first', 'second'])
print(df)
```


temp1.py - C:/Users/91954/OneDrive/Desktop/New folder/temp1.py (3.13.0)
File Edit Format Run Options Window Help
IDLE Shell 3.13.0
File Edit Shell Debug Options Window Help
Ln: 1 Col: 58

```
print("3 Dataframe creation using list of dictionaries ")
import pandas as pd
data = [{'b': 2, 'c': 3}, {'a': 10, 'b': 20, 'c': 30}]
df = pd.DataFrame(data, index=['first', 'second'])
print(df)
```

```
Python 3.13.0 (tags/v3.13.0:60403a5, Oct  7 2024, 09:38:07) [MSC v.1941 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========= RESTART: C:/Users/91954/OneDrive/Desktop/New folder/temp1.py =========
3 Dataframe creation using list of dictionaries
         b   c     a
first    2   3   NaN
second  20  30  10.0
>>>
```

```
# 4 Dataframe creation using series and conact
print("4 Dataframe creation using series and concat)
import pandas as pd
a = [1, 7, 2]
b=["50","25","50"]
rollno = pd.Series(a, index = ["x", "y", "z"])
marks1 = pd.Series(b, index = ["x", "y", "z"])
d1=pd.concat([rollno,marks1],axis=1)
print("===============")
print(rollno.sort_values())
print("=================")
print(marks1)
print("=================")
print(d1)
print("=================")
print (d1.info())
```



temp1.py - C:/Users/91954/OneDrive/Desktop/New folder/temp1.py (3.13.0)
File Edit Format Run Options Window Help

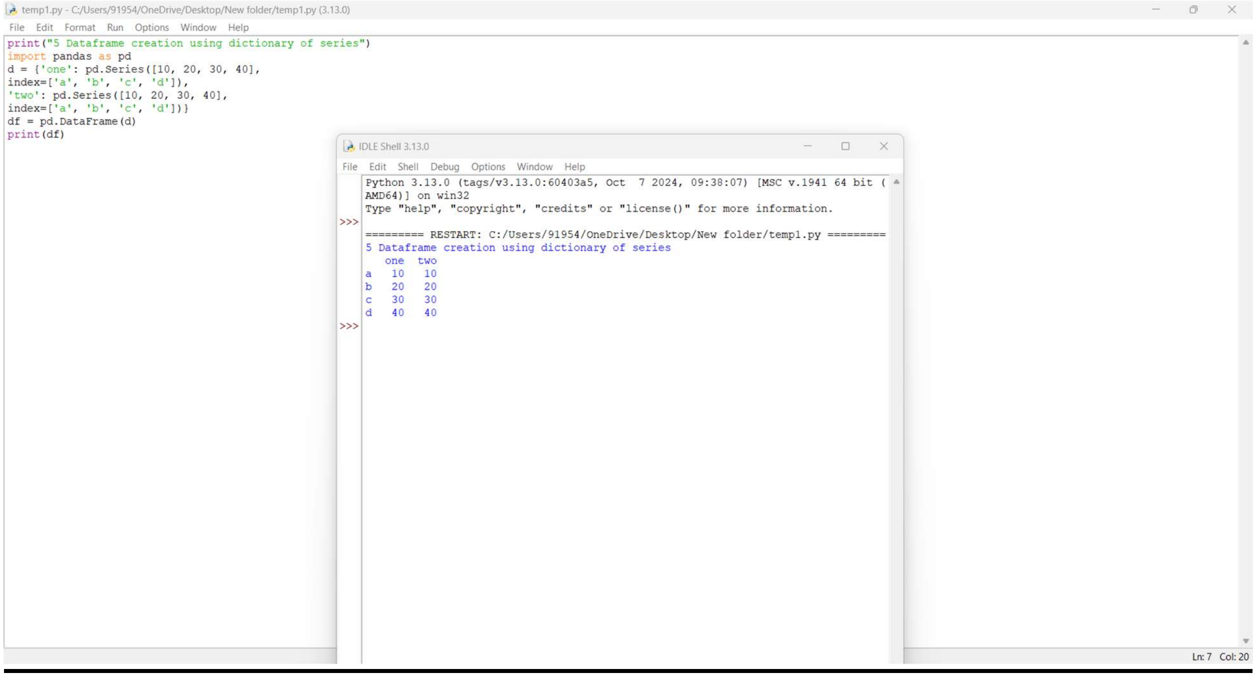
```
print("\n")
print("4 Dataframe creation using series and concat")
import pandas as pd
a = [1, 7, 2]
b=["50","25","50"]
rollno = pd.Series(a, index = ["x", "y", "z"
marks1 = pd.Series(b, index = ["x", "y", "z"
d1=pd.concat([rollno,marks1],axis=1)
print("===============")
print(rollno.sort_values())
print("=================")
print(marks1)
print("=================")
print(d1)
print("=================")
print (d1.info())
```

IDLE Shell 3.13.0
File Edit Shell Debug Options Window Help
Ln: 15 Col: 26

```
Python 3.13.0 (tags/v3.13.0:60403a5, Oct  7 2024, 09:38:07) [MSC v.1941 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========= RESTART: C:/Users/91954/OneDrive/Desktop/New folder/temp1.py =========


4 Dataframe creation using series and concat
===============
x    1
z    2
y    7
dtype: int64
=================
x    50
y    25
z    50
dtype: object
=================
   0   1
x  1  50
y  7  25
z  2  50
=================
<class 'pandas.core.frame.DataFrame'>
Index: 3 entries, x to z
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       3 non-null      int64
 1   1       3 non-null      object
dtypes: int64(1), object(1)
memory usage: 72.0+ bytes
None
>>> 
```

5 Dataframe creation using dictionary of series

```
print("5 Dataframe creation using  series”)
import pandas as pd
d = {'one': pd.Series([10, 20, 30, 40],
index=['a', 'b', 'c', 'd']),
'two': pd.Series([10, 20, 30, 40],
index=['a', 'b', 'c', 'd'])}
df = pd.DataFrame(d)
print(df)
```


temp1.py - C:/Users/91954/OneDrive/Desktop/New folder/temp1.py (3.13.0)
File Edit Format Run Options Window Help
IDLE Shell 3.13.0
File Edit Shell Debug Options Window Help
Ln: 7 Col: 20

```
print("5 Dataframe creation using dictionary of series")
import pandas as pd
d = {'one': pd.Series([10, 20, 30, 40],
index=['a', 'b', 'c', 'd']),
'two': pd.Series([10, 20, 30, 40],
index=['a', 'b', 'c', 'd'])}
df = pd.DataFrame(d)
print(df)
```

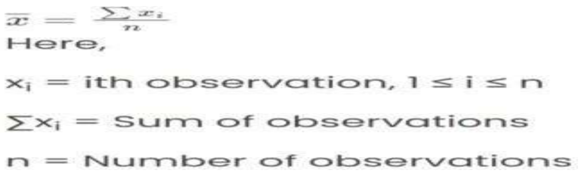
```
Python 3.13.0 (tags/v3.13.0:60403a5, Oct  7 2024, 09:38:07) [MSC v.1941 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========= RESTART: C:/Users/91954/OneDrive/Desktop/New folder/temp1.py =========
5 Dataframe creation using dictionary of series
   one  two
a   10   10
b   20   20
c   30   30
d   40   40
>>>
```

Practical 4

Write a program to demonstrate mean function, mode, median

Theory

1. **Mean:** The mean, also known as the average, is calculated by adding up all the values in a dataset and then dividing by the total number of values. It is often denoted by μ (mu) for a population or $\overline{x}$ (x-bar) for a sample. The formula for calculating the mean is:



$\overline{x} = \frac{\sum x_i}{n}$

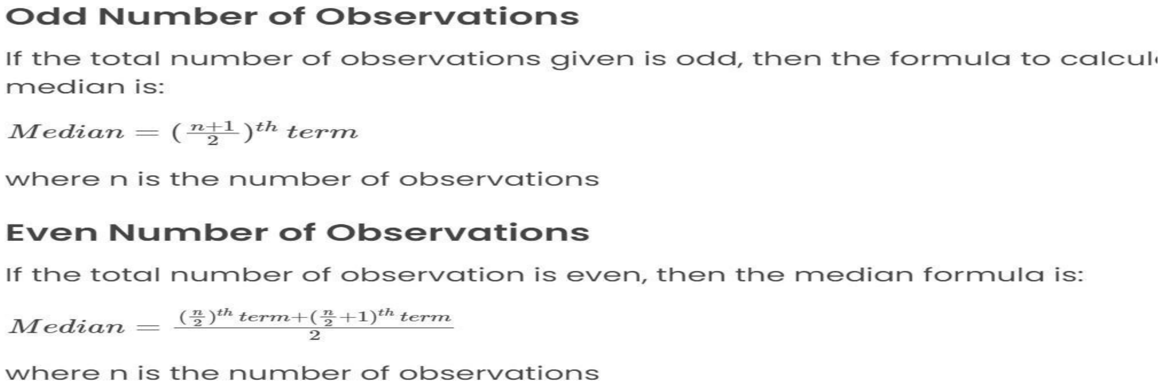
Here,

x_i = ith observation, $1 \le i \le n$

$\sum x_i$ = Sum of observations

n = Number of observations

2. **Median:** The median is the middle value of a dataset when it is ordered from least to greatest. If the dataset has an odd number of values, the median is the middle value. If the dataset has an even number of values, the median is the average of the two middle values. The median is less sensitive to outliers compared to the mean.



Odd Number of Observations

If the total number of observations given is odd, then the formula to calcul
median is:

$Median = (\frac{n+1}{2})^{th}\ term$

where n is the number of observations

Even Number of Observations

If the total number of observation is even, then the median formula is:

$Median = \frac{(\frac{n}{2})^{th}\ term + (\frac{n}{2}+1)^{th}\ term}{2}$

where n is the number of observations

3. **Mode:** The mode is the value that appears most frequently in a dataset. A dataset can have one mode (unimodal), two modes (bimodal), or more than two modes (multimodal). Unlike mean and median, mode is not affected by outliers.

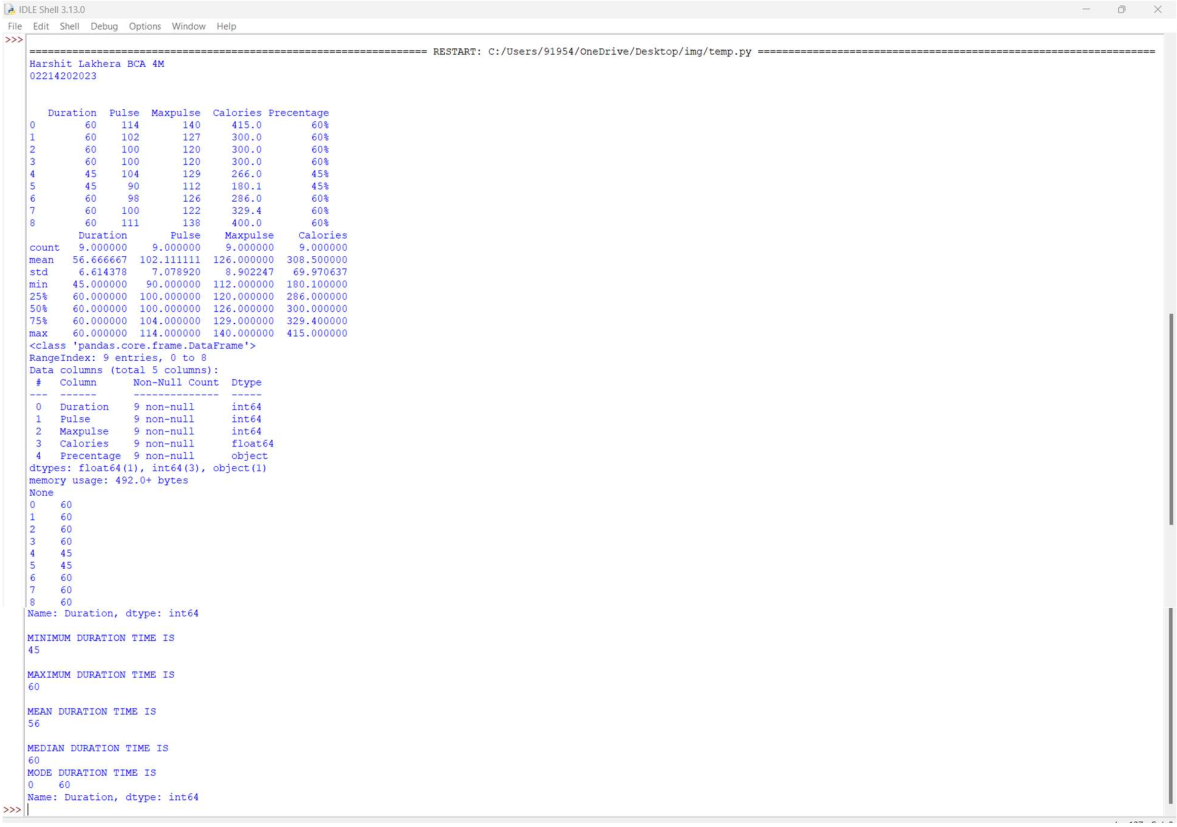
4. **Maximum Duration Time:** The maximum duration time refers to the longest duration among all the durations observed in a dataset.

5. **Minimum Duration Time:** The minimum duration time refers to the shortest duration among all the durations observed in a dataset.

Code

```
import pandas as pd
import numpy as np
data = pd.read_csv("Book1.csv")
print("Harshit Lakhera BCA 4M")
print("02214202023")
print("\n")
print(data)
print(data.describe())
print(data.info())
data_duration=data["Duration"]
print(data_duration)
mn=int(np.min(data_duration))
print("")
print("MINIMUM DURATION TIME IS")
print(mn)
mx=int(np.max(data_duration))
print("")
print("MAXIMUM DURATION TIME IS")
print(mx)
m=int(np.mean(data_duration))
print("")
print("MEAN DURATION TIME IS")
print(m)
md=int(np.median(data_duration))
print("")
print("MEDIAN DURATION TIME IS")
print(md)
mode1=data_duration.mode()
print("MODE DURATION TIME IS")
print(mode1)
```

Output:



```
>>> 
    ================================================================ RESTART: C:/Users/91954/OneDrive/Desktop/img/temp.py ================================================================
    Harshit Lakhera BCA 4M
    02214202023


       Duration  Pulse  Maxpulse  Calories Precentage
    0        60    114       140     415.0        60%
    1        60    102       127     300.0        60%
    2        60    100       120     300.0        60%
    3        60    100       120     300.0        60%
    4        45    104       129     266.0        45%
    5        45     90       112     180.1        45%
    6        60     98       126     286.0        60%
    7        60    100       122     329.4        60%
    8        60    111       138     400.0        60%
            Duration       Pulse    Maxpulse    Calories
    count   9.000000    9.000000    9.000000    9.000000
    mean   56.666667  102.111111  126.000000  308.500000
    std     6.614378    7.078920    8.902247   69.970637
    min    45.000000   90.000000  112.000000  180.100000
    25%    60.000000  100.000000  120.000000  286.000000
    50%    60.000000  100.000000  126.000000  300.000000
    75%    60.000000  104.000000  129.000000  329.400000
    max    60.000000  114.000000  140.000000  415.000000
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 9 entries, 0 to 8
    Data columns (total 5 columns):
     #   Column      Non-Null Count  Dtype  
    ---  ------      --------------  -----  
     0   Duration    9 non-null      int64  
     1   Pulse       9 non-null      int64  
     2   Maxpulse    9 non-null      int64  
     3   Calories    9 non-null      float64
     4   Precentage  9 non-null      object 
    dtypes: float64(1), int64(3), object(1)
    memory usage: 492.0+ bytes
    None
    0    60
    1    60
    2    60
    3    60
    4    45
    5    45
    6    60
    7    60
    8    60
    Name: Duration, dtype: int64

    MINIMUM DURATION TIME IS
    45

    MAXIMUM DURATION TIME IS
    60

    MEAN DURATION TIME IS
    56

    MEDIAN DURATION TIME IS
    60
    MODE DURATION TIME IS
    0    60
    Name: Duration, dtype: int64
>>> 
```

Practical 5

Write a program to implement pivot() and pivot-table() on a DataFrame.

Theory

Pivot:

The pivot operation in Pandas is a fundamental tool for reshaping and reorganizing tabular data. It allows users to transform long-form data into wide-form data and vice versa. Key features of the pivot operation include:

- **Index and Column Values:** Users can specify which columns to use as index and which ones to use as columns while reshaping the DataFrame.
- **Aggregation:** Pivot allows users to aggregate data based on index and column values, applying summary functions to compute aggregated values.
- **Reshaping:** It facilitates restructuring data into a more structured format, making it easier to analyze and interpret.

Pivot Tables:

Pivot tables are a powerful feature in Pandas and other data analysis tools. They provide a way to summarize and aggregate data in tabular form, allowing users to analyze complex datasets and derive insights efficiently. Pivot tables allow users to:

- Group and aggregate data based on one or more columns.
- Compute various summary statistics (e.g., sum, mean, count) for the grouped data.
- Perform multi-level aggregation and display results in a structured format.
- Quickly create reports and visualizations to understand the underlying data distribution and trends.

Without CSV:

- The script begins by creating a DataFrame df from a dictionary data, representing temperature and humidity data for different cities on specific dates.
- It demonstrates the usage of Pandas' pivot operation using pivot() to rearrange the data. Specifically, it pivots the DataFrame on the 'Date' and 'City' columns and displays the temperature values.
- Additionally, it creates another DataFrame df_dup from a dictionary data_dup with duplicate entries.
- Further, it showcases the utility of pivot tables through the pivot_table() function, which computes the mean temperature for each city on each date. Pivot tables offer a concise way to summarize and analyze data, providing insights into aggregated values.

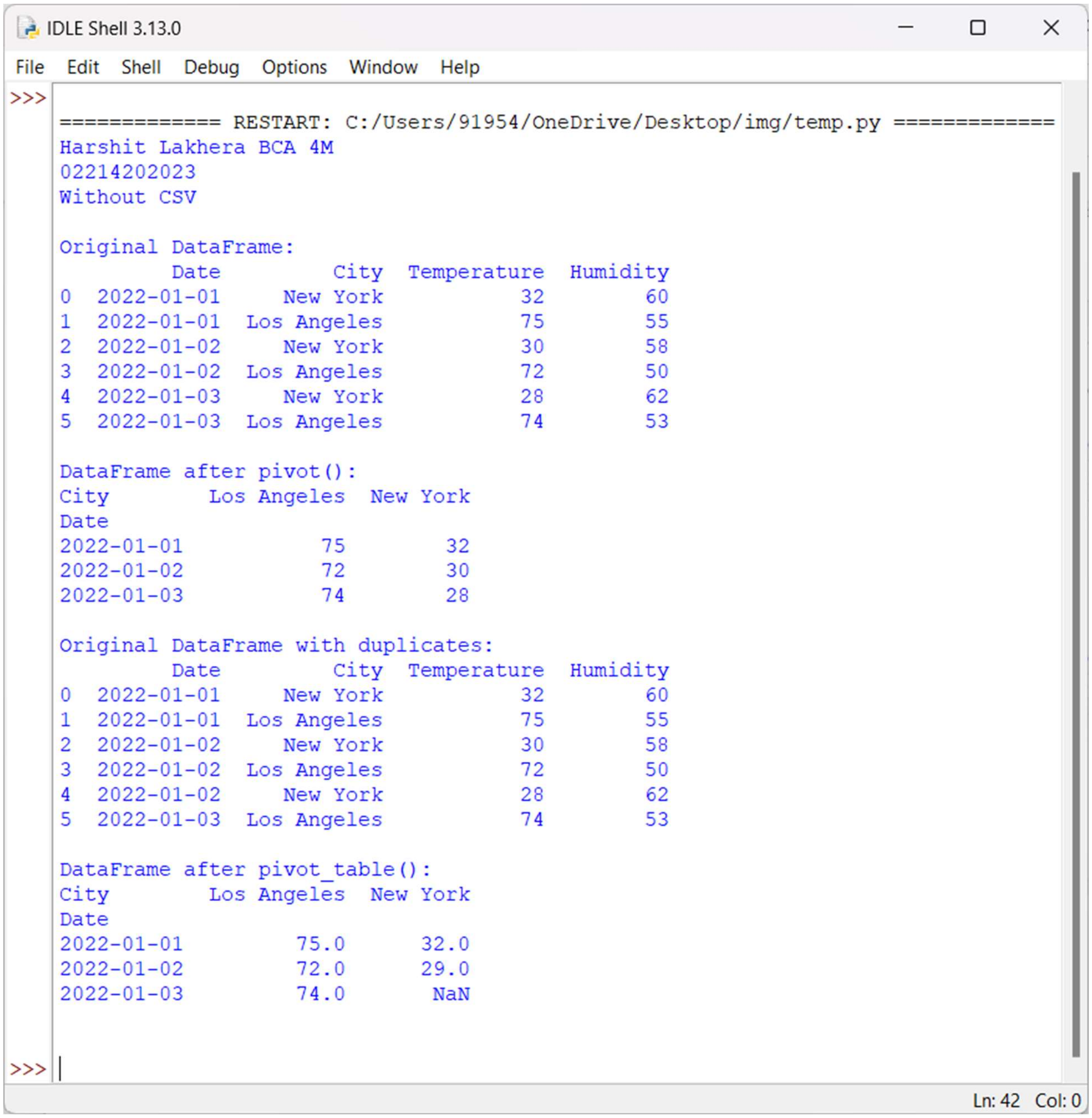
With CSV:

- The script reads a CSV file named 'sales.csv' into a DataFrame df, representing sales data with columns like Region, Country, Total Profit, and Total Cost.
- It displays the original DataFrame.
- Then, it leverages the power of pivot tables with the pivot_table() function to aggregate Total Profit and Total Cost based on Region and Country. This operation is performed using numpy's sum function as the aggregation function.
- Finally, it prints the pivot tables for Total Profit and Total Cost.

Code

```
import pandas as pd
print("Harshit Lakhera BCA 4M")
print("02214202023")
print("Without CSV \n")
data = {
'Date': ['2022-01-01', '2022-01-01', '2022-01-02', '2022-01-02', '2022-01-03',
'2022-01-03'],
'City': ['New York', 'Los Angeles', 'New York', 'Los Angeles', 'New York', 'Los Angeles'],
'Temperature': [32, 75, 30, 72, 28, 74],
'Humidity': [60, 55, 58, 50, 62, 53]
}

df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)
print()
pivot_df = df.pivot(index='Date', columns='City', values='Temperature')
print("DataFrame after pivot():")
print(pivot_df)
print()

data_dup = {
'Date': ['2022-01-01', '2022-01-01', '2022-01-02', '2022-01-02', '2022-01-02',
'2022-01-03'],
'City': ['New York', 'Los Angeles', 'New York', 'Los Angeles', 'New York', 'Los Angeles'],
'Temperature': [32, 75, 30, 72, 28, 74],
'Humidity': [60, 55, 58, 50, 62, 53]
}
df_dup = pd.DataFrame(data_dup)

print("Original DataFrame with duplicates:")
print(df_dup)
print()

pivot_table_df = df_dup.pivot_table(index='Date', columns='City', values='Temperature',
aggfunc='mean')

print("DataFrame after pivot_table():")
print(pivot_table_df)
print("\n")
```

Output:

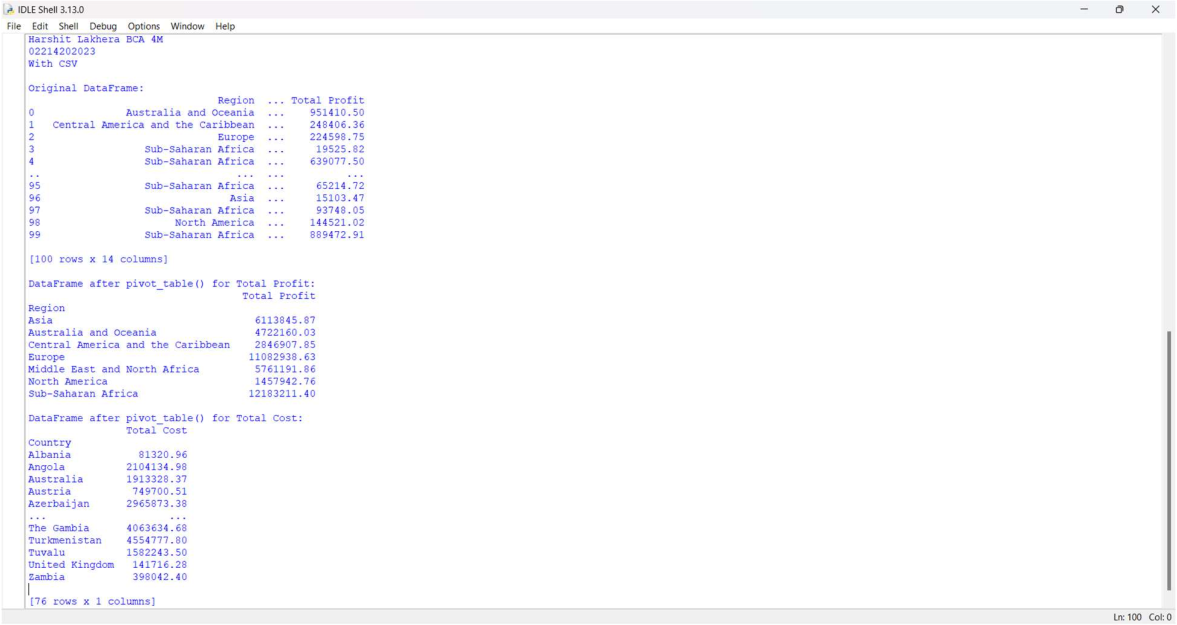


```
>>>
============= RESTART: C:/Users/91954/OneDrive/Desktop/img/temp.py =============
Harshit Lakhera BCA 4M
02214202023
Without CSV

Original DataFrame:
         Date         City  Temperature  Humidity
0  2022-01-01     New York           32        60
1  2022-01-01  Los Angeles           75        55
2  2022-01-02     New York           30        58
3  2022-01-02  Los Angeles           72        50
4  2022-01-03     New York           28        62
5  2022-01-03  Los Angeles           74        53

DataFrame after pivot():
City        Los Angeles  New York
Date
2022-01-01           75        32
2022-01-02           72        30
2022-01-03           74        28

Original DataFrame with duplicates:
         Date         City  Temperature  Humidity
0  2022-01-01     New York           32        60
1  2022-01-01  Los Angeles           75        55
2  2022-01-02     New York           30        58
3  2022-01-02  Los Angeles           72        50
4  2022-01-02     New York           28        62
5  2022-01-03  Los Angeles           74        53

DataFrame after pivot_table():
City        Los Angeles  New York
Date
2022-01-01         75.0      32.0
2022-01-02         72.0      29.0
2022-01-03         74.0       NaN

>>> |
```

#With CSV

```
import pandas as pd
import numpy as np
print("Harshit Lakhera BCA 4M")
print("02214202023")
print("With CSV\n")
df = pd.read_csv("Book1.csv")
print("Original DataFrame:")
print(df)
print()

pivot_table_profit = df.pivot_table(index='Region', values='Total Profit', aggfunc=lambda x:
getattr(np, 'sum')(x))
pivot_table_cost = df.pivot_table(index='Country', values='Total Cost', aggfunc=lambda x:
getattr(np, 'sum')(x))

print("DataFrame after pivot_table() for Total Profit:")
print(pivot_table_profit)
print("\nDataFrame after pivot_table() for Total Cost:")
print(pivot_table_cost)
```

Output:



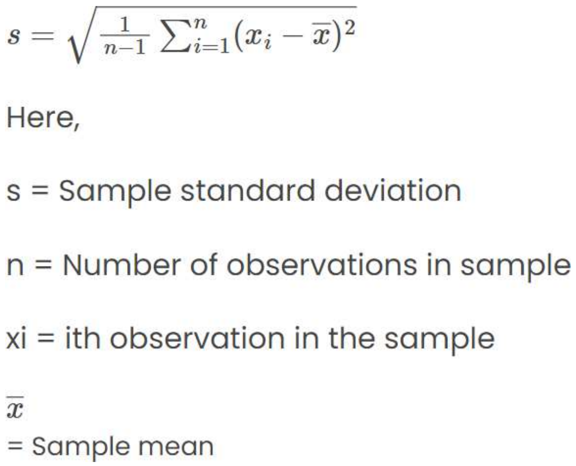
```
Harshit Lakhera BCA 4M
02214202023
With CSV

Original DataFrame:
                              Region  ... Total Profit
0               Australia and Oceania  ...    951410.50
1   Central America and the Caribbean  ...    248406.36
2                              Europe  ...    224598.75
3                  Sub-Saharan Africa  ...     19525.82
4                  Sub-Saharan Africa  ...    639077.50
..                                ...  ...          ...
95                 Sub-Saharan Africa  ...     65214.72
96                               Asia  ...     15103.47
97                 Sub-Saharan Africa  ...     93748.05
98                      North America  ...    144521.02
99                 Sub-Saharan Africa  ...    889472.91

[100 rows x 14 columns]

DataFrame after pivot_table() for Total Profit:
                                   Total Profit
Region
Asia                                 6113845.87
Australia and Oceania                4722160.03
Central America and the Caribbean    2846907.85
Europe                              11082938.63
Middle East and North Africa         5761191.86
North America                        1457942.76
Sub-Saharan Africa                  12183211.40

DataFrame after pivot_table() for Total Cost:
                Total Cost
Country
Albania           81320.96
Angola          2104134.98
Australia       1913328.37
Austria          749700.51
Azerbaijan      2965873.38
...                    ...
The Gambia      4063634.68
Turkmenistan    4554777.80
Tuvalu          1582243.50
United Kingdom   141716.28
Zambia           398042.40

[76 rows x 1 columns]
```

Practical 6

Write a program to demonstrate standard deviation and variance.

Theory

1. Standard Deviation- Standard Deviation is defined as the degree of dispersion of the data point to the mean value of the data point. It tells us how the value of the data points varies to the mean value of the data point and it tells us about the variation of the data point in the sample of the data.



$$s = \sqrt{\tfrac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

Here,

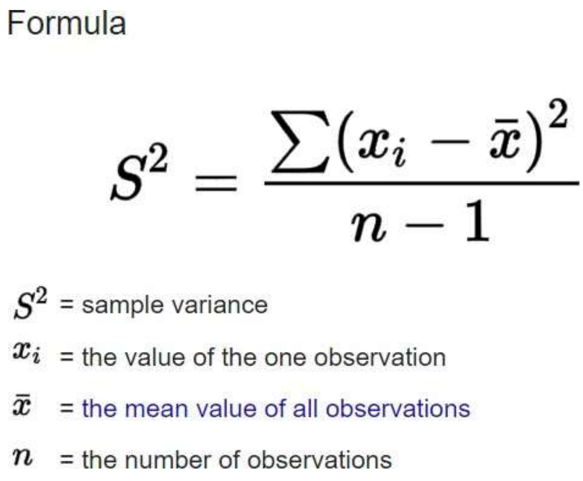
s = Sample standard deviation

n = Number of observations in sample

xi = ith observation in the sample

$\overline{x}$
= Sample mean

2. Variance- Variance is the expected value of the squared variation of a random variable from its mean value, in probability and statistics. Informally, variance estimates how far a set of numbers (random) are spread out from their mean value.



Formula

$$S^2 = \frac{\sum(x_i - \bar{x})^2}{n-1}$$

S^2 = sample variance

x_i = the value of the one observation

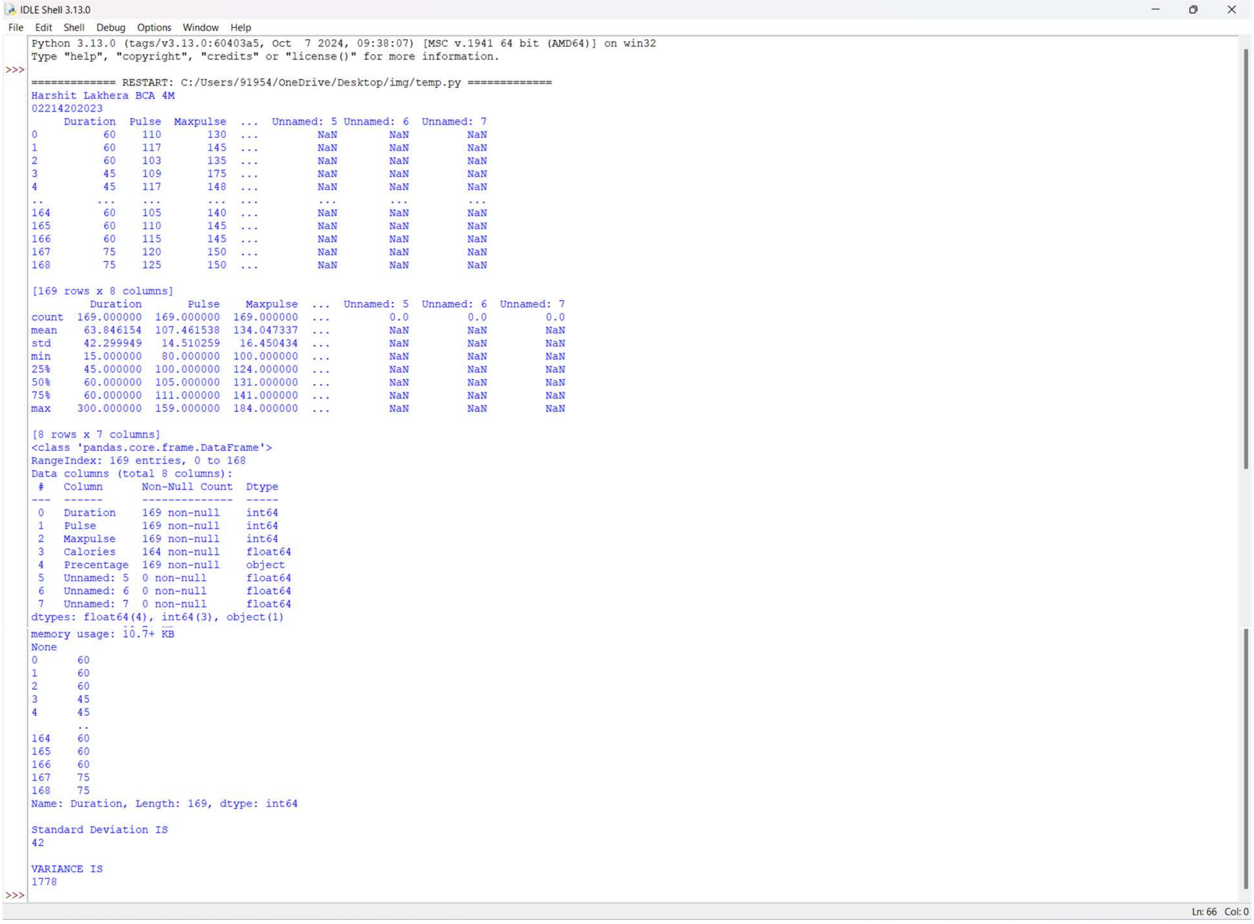
$\bar{x}$ = the mean value of all observations

n = the number of observations

Code

```
import pandas as pd
import numpy as np
print("Harshit Lakhera BCA 4M")
print("02214202023")
data=pd.read_csv("Book1.csv")
print(data)
print(data.describe())
print(data.info())

data_duration=data["Duration"]
print(data_duration)

sd=int(np.std(data_duration))
print("")
print("Standard Deviation IS")
print(sd)

va=int(np.var(data_duration))
print("")
print("VARIANCE IS")
print(va)
```

Output:



```
    Python 3.13.0 (tags/v3.13.0:60403a5, Oct  7 2024, 09:38:07) [MSC v.1941 64 bit (AMD64)] on win32
    Type "help", "copyright", "credits" or "license()" for more information.
>>> 
    ============= RESTART: C:/Users/91954/OneDrive/Desktop/img/temp.py =============
    Harshit Lakhera BCA 4M
    02214202023
         Duration  Pulse  Maxpulse  ...  Unnamed: 5 Unnamed: 6  Unnamed: 7
    0          60    110       130  ...         NaN        NaN         NaN
    1          60    117       145  ...         NaN        NaN         NaN
    2          60    103       135  ...         NaN        NaN         NaN
    3          45    109       175  ...         NaN        NaN         NaN
    4          45    117       148  ...         NaN        NaN         NaN
    ..        ...    ...       ...  ...         ...        ...         ...
    164        60    105       140  ...         NaN        NaN         NaN
    165        60    110       145  ...         NaN        NaN         NaN
    166        60    115       145  ...         NaN        NaN         NaN
    167        75    120       150  ...         NaN        NaN         NaN
    168        75    125       150  ...         NaN        NaN         NaN

    [169 rows x 8 columns]
             Duration       Pulse    Maxpulse  ...  Unnamed: 5  Unnamed: 6  Unnamed: 7
    count  169.000000  169.000000  169.000000  ...         0.0         0.0         0.0
    mean    63.846154  107.461538  134.047337  ...         NaN         NaN         NaN
    std     42.299949   14.510259   16.450434  ...         NaN         NaN         NaN
    min     15.000000   80.000000  100.000000  ...         NaN         NaN         NaN
    25%     45.000000  100.000000  124.000000  ...         NaN         NaN         NaN
    50%     60.000000  105.000000  131.000000  ...         NaN         NaN         NaN
    75%     60.000000  111.000000  141.000000  ...         NaN         NaN         NaN
    max    300.000000  159.000000  184.000000  ...         NaN         NaN         NaN

    [8 rows x 7 columns]
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 169 entries, 0 to 168
    Data columns (total 8 columns):
     #   Column      Non-Null Count  Dtype  
    ---  ------      --------------  -----  
     0   Duration    169 non-null    int64  
     1   Pulse       169 non-null    int64  
     2   Maxpulse    169 non-null    int64  
     3   Calories    164 non-null    float64
     4   Precentage  169 non-null    object 
     5   Unnamed: 5  0 non-null      float64
     6   Unnamed: 6  0 non-null      float64
     7   Unnamed: 7  0 non-null      float64
    dtypes: float64(4), int64(3), object(1)
    memory usage: 10.7+ KB
    None
    0      60
    1      60
    2      60
    3      45
    4      45
           ..
    164    60
    165    60
    166    60
    167    75
    168    75
    Name: Duration, Length: 169, dtype: int64

    Standard Deviation IS
    42

    VARIANCE IS
    1778
>>> 
```