Submission-3

COLLEGE OF ENGINEERING
(AUTONOMOUS)

Name: Bongu Hari Hara Charan

Roll no: 323103310032

Branch: CSE-1

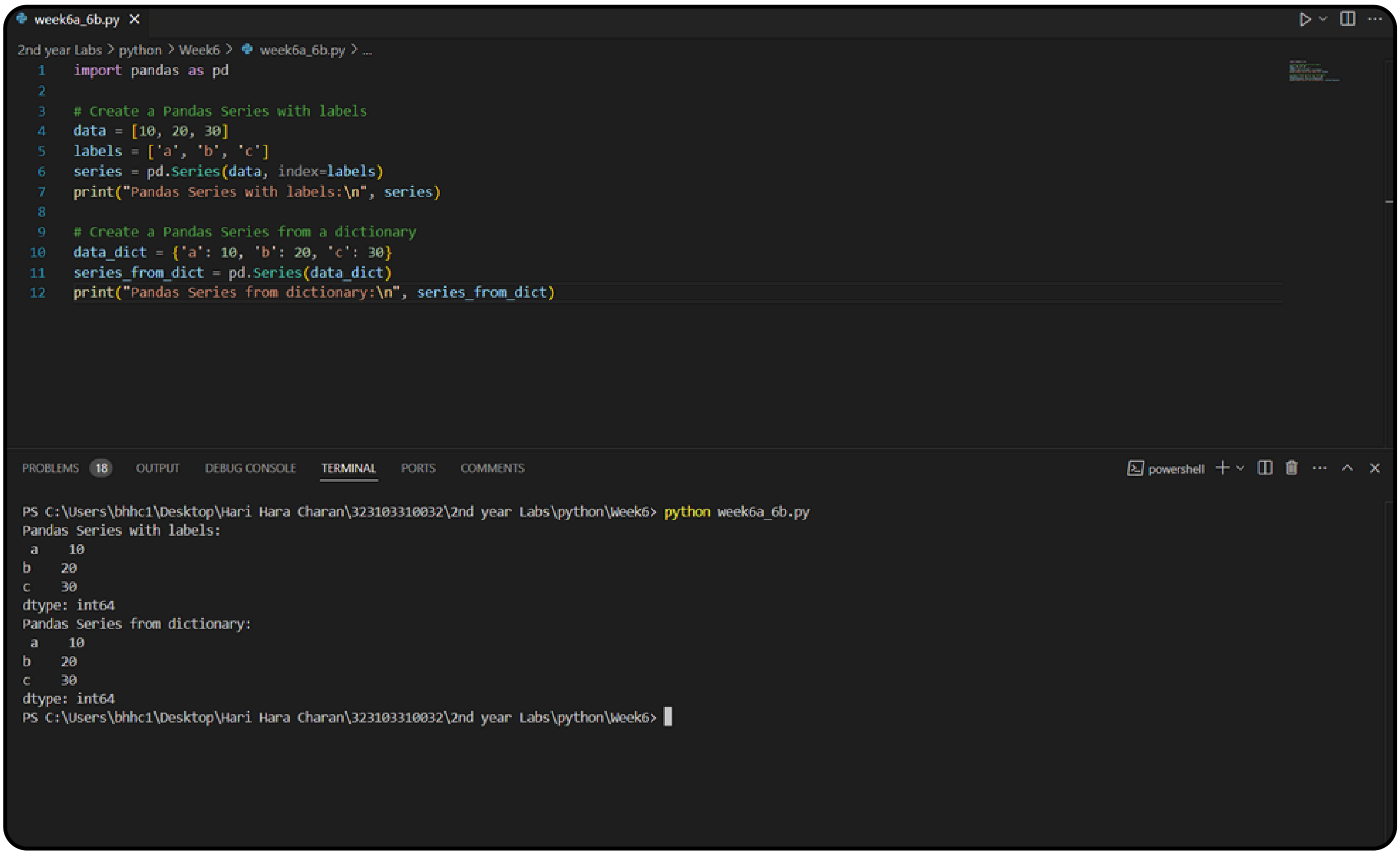
Semistor: 3

Name of the lab: Adv Python lab

Professor: Satya Keerthi Gorripati

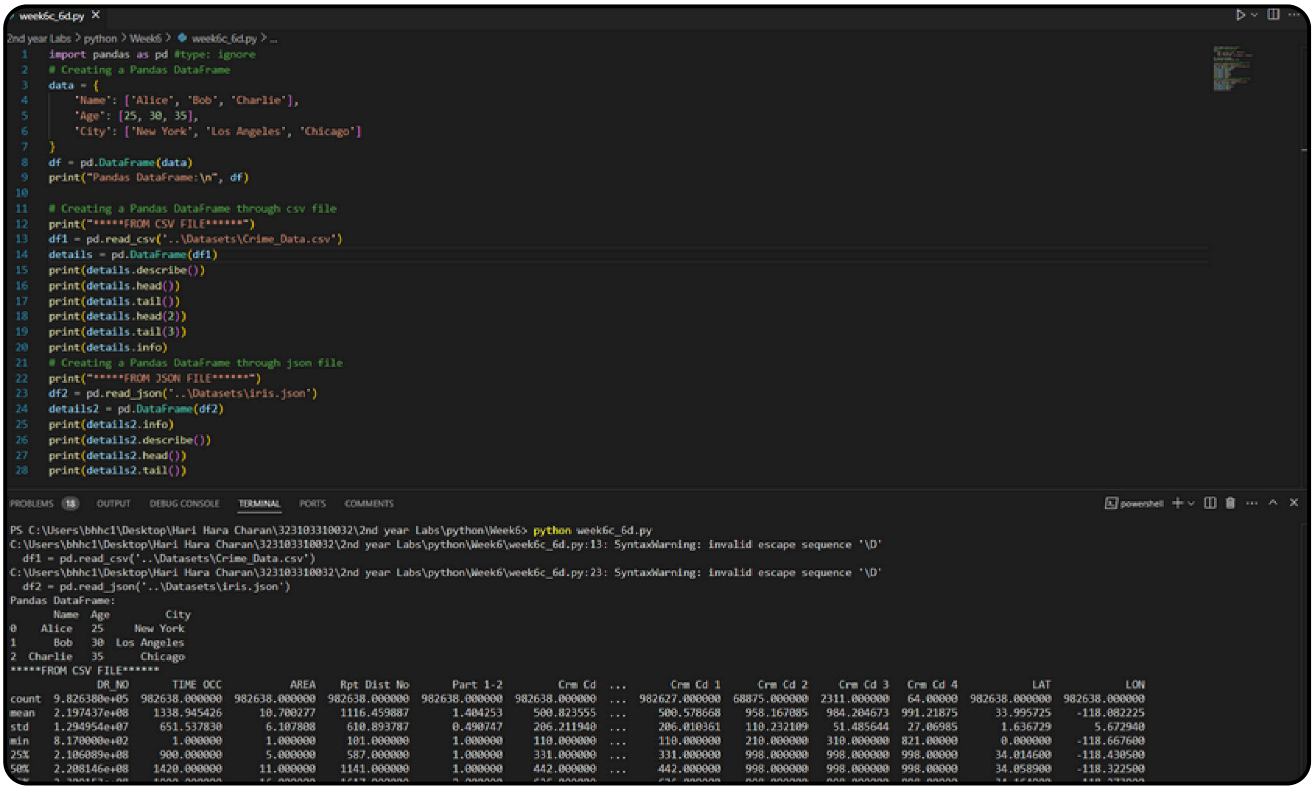
Links:

Week6
Pandas Library:



```python
import pandas as pd

# Create a Pandas Series with labels
data = [10, 20, 30]
labels = ['a', 'b', 'c']
series = pd.Series(data, index=labels)
print("Pandas Series with labels:\n", series)

# Create a Pandas Series from a dictionary
data_dict = {'a': 10, 'b': 20, 'c': 30}
series_from_dict = pd.Series(data_dict)
print("Pandas Series from dictionary:\n", series_from_dict)
```

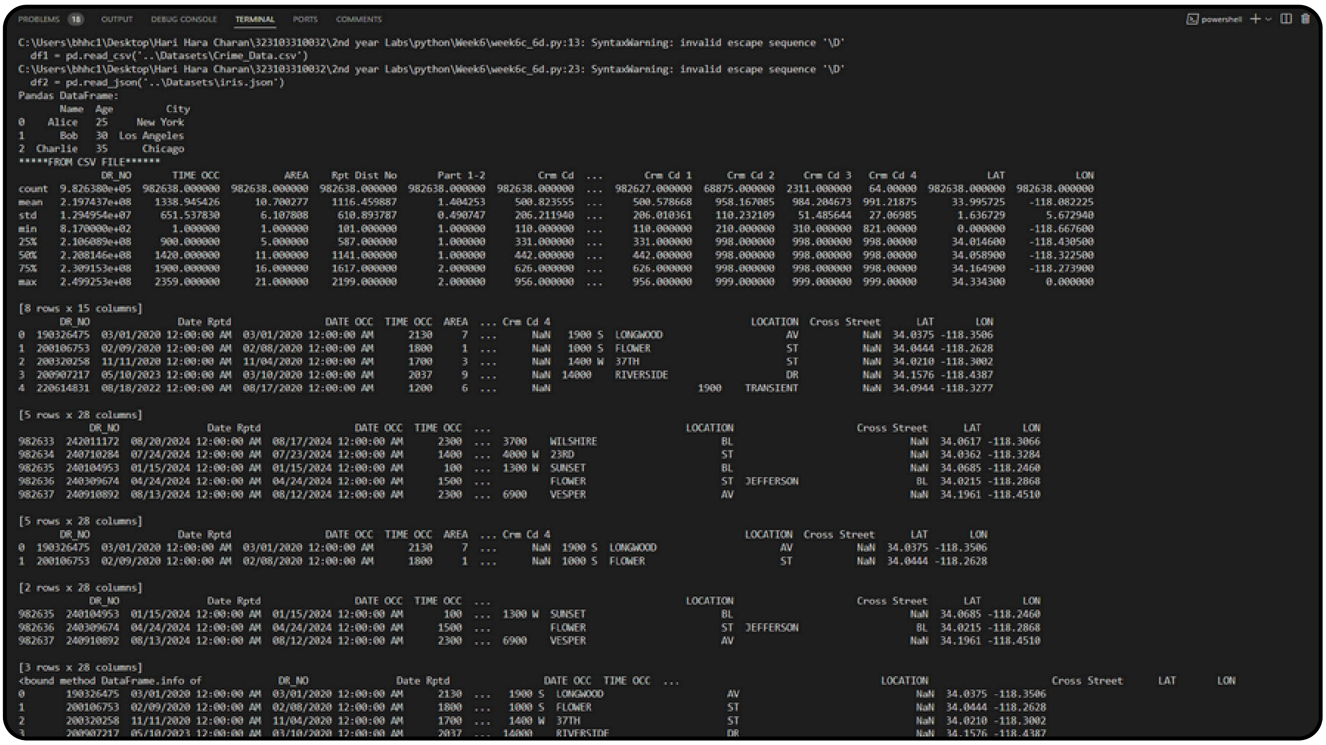
```
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week6> python week6a_6b.py
Pandas Series with labels:
 a    10
b    20
c    30
dtype: int64
Pandas Series from dictionary:
 a    10
b    20
c    30
dtype: int64
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week6>
```



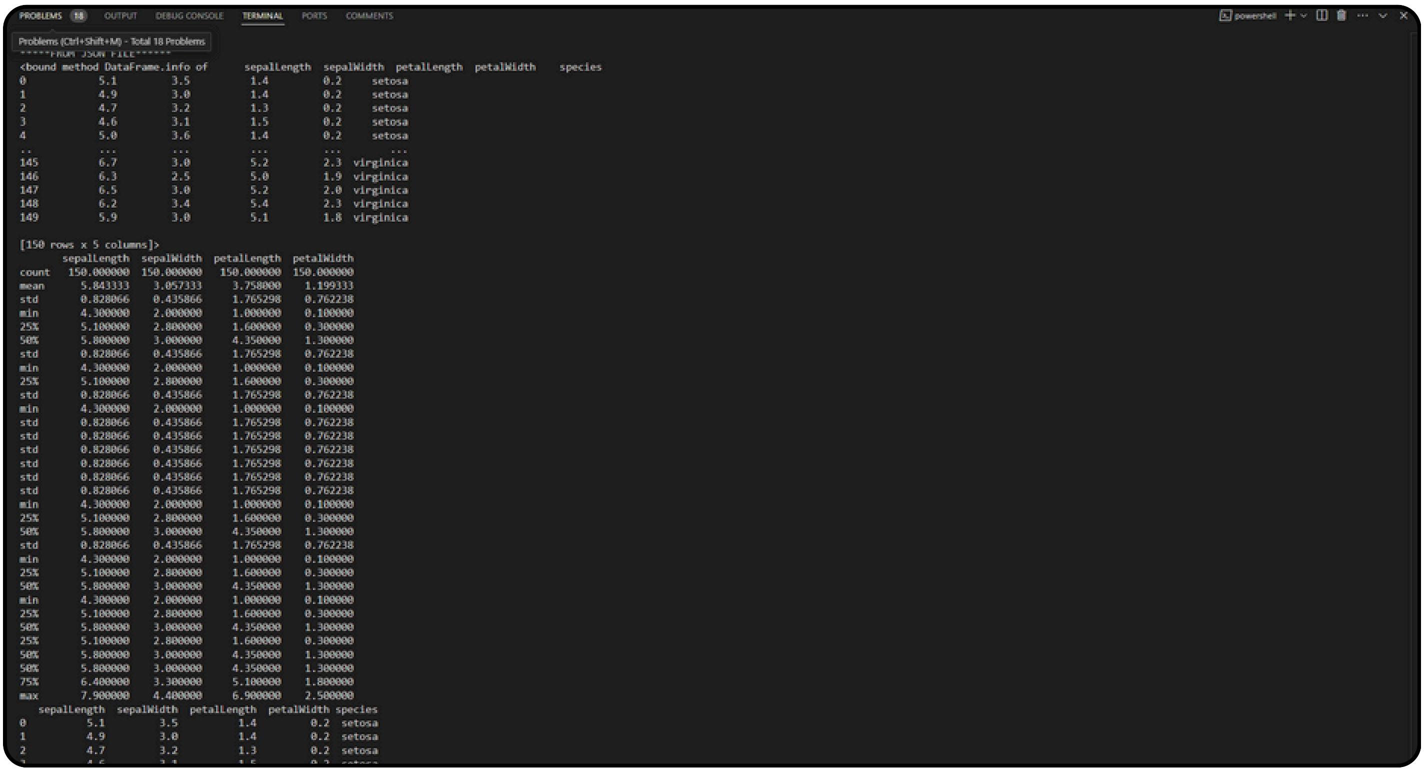
```python
import pandas as pd #type: ignore
# Creating a Pandas DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['New York', 'Los Angeles', 'Chicago']
}
df = pd.DataFrame(data)
print("Pandas DataFrame:\n", df)

# Creating a Pandas DataFrame through csv file
print("*****FROM CSV FILE******")
df1 = pd.read_csv('..\Datasets\Crime_Data.csv')
details = pd.DataFrame(df1)
print(details.describe())
print(details.head())
print(details.tail())
print(details.head(2))
print(details.tail(3))
print(details.info)
# Creating a Pandas DataFrame through json file
print("*****FROM JSON FILE******")
df2 = pd.read_json('..\Datasets\iris.json')
details2 = pd.DataFrame(df2)
print(details2.info)
print(details2.describe())
print(details2.head())
print(details2.tail())
```

```
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week6> python week6c_6d.py
C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week6\week6c_6d.py:13: SyntaxWarning: invalid escape sequence '\D'
  df1 = pd.read_csv('..\Datasets\Crime_Data.csv')
C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week6\week6c_6d.py:23: SyntaxWarning: invalid escape sequence '\D'
  df2 = pd.read_json('..\Datasets\iris.json')
Pandas DataFrame:
      Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   35      Chicago
*****FROM CSV FILE******
              DR_NO       TIME OCC           AREA    Rpt Dist No      Part 1-2         Crm Cd  ...       Crm Cd 1      Crm Cd 2     Crm Cd 3   Crm Cd 4            LAT            LON
count  9.826380e+05  982638.000000  982638.000000  982638.000000  982638.000000  982638.000000  ...  982627.000000  68875.000000  2311.000000  64.00000  982638.000000  982638.000000
mean   2.197437e+08    1338.945426      10.700277    1116.459887       1.404253     500.823555  ...     500.578668    958.167085   984.204673  991.21875      33.995725    -118.082225
std    1.294954e+07     651.537830       6.107808     610.893787       0.490747     206.211940  ...     206.010361    110.232109    51.485644   27.06985       1.636729       5.672940
min    8.170000e+02       1.000000       1.000000     101.000000       1.000000     110.000000  ...     110.000000    210.000000   310.000000  821.00000       0.000000    -118.667600
25%    2.106089e+08     900.000000       5.000000     587.000000       1.000000     331.000000  ...     331.000000    998.000000   998.000000  998.00000      34.014600    -118.430500
50%    2.208146e+08    1420.000000      11.000000    1141.000000       1.000000     442.000000  ...     442.000000    998.000000   998.000000  998.00000      34.058900    -118.322500
```

Output:

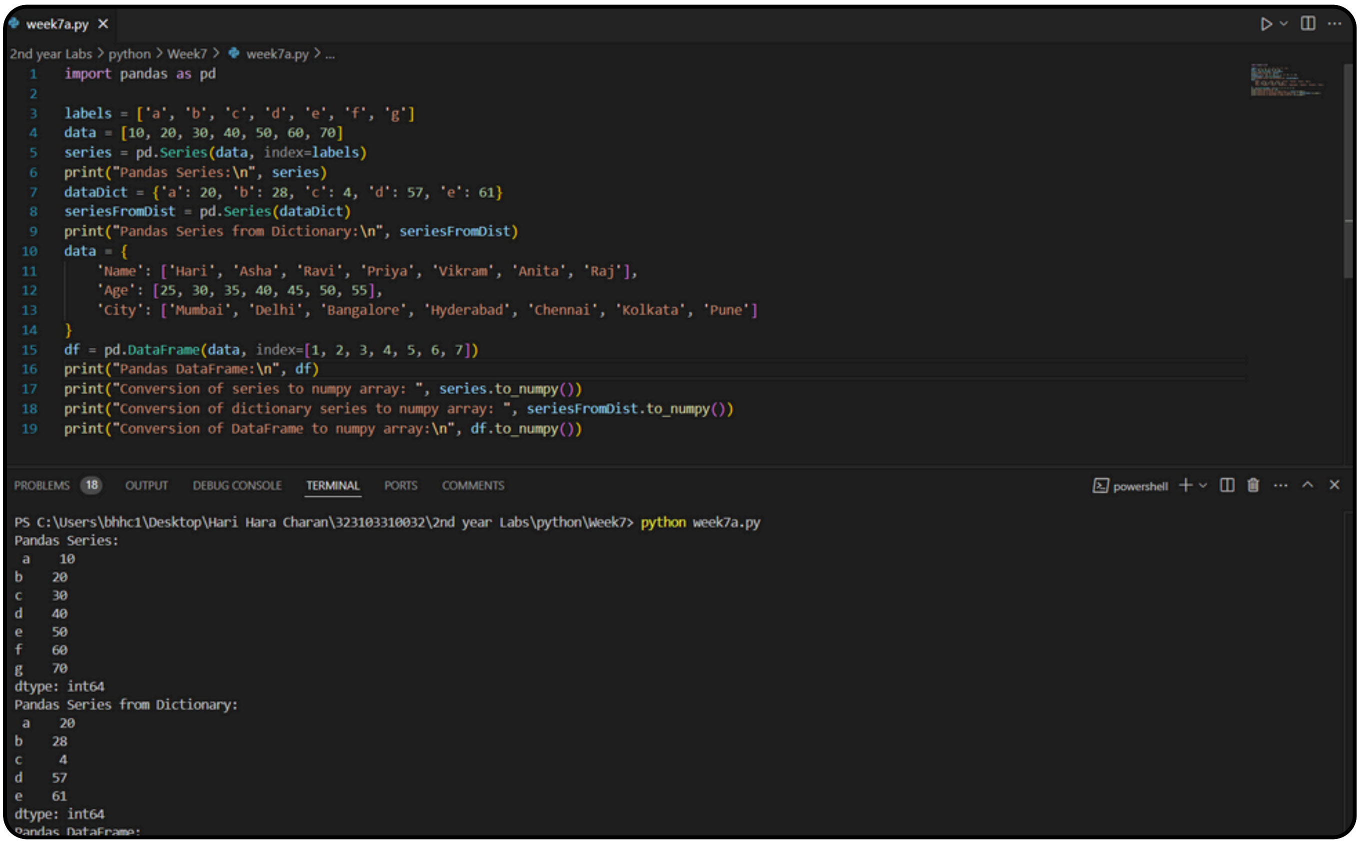


```
C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week6\week6c_6d.py:13: SyntaxWarning: invalid escape sequence '\D'
  df1 = pd.read_csv('..\Datasets\Crime_Data.csv')
C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week6\week6c_6d.py:23: SyntaxWarning: invalid escape sequence '\D'
  df2 = pd.read_json('..\Datasets\iris.json')
Pandas DataFrame:
      Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   35      Chicago
*****FROM CSV FILE******
              DR_NO       TIME OCC           AREA    Rpt Dist No       Part 1-2         Crm Cd  ...       Crm Cd 1      Crm Cd 2     Crm Cd 3  Crm Cd 4            LAT            LON
count  9.826380e+05  982638.000000  982638.000000  982638.000000  982638.000000  982638.000000  ...  982627.000000  68875.000000  2311.000000  64.00000  982638.000000  982638.000000
mean   2.197437e+08    1338.945426      10.700277    1116.459887       1.404253     500.823555  ...     500.578668    958.167085   984.204673 991.21875      33.995725    -118.082225
std    1.294954e+07     651.537830       6.107808     610.893787       0.490747     206.211940  ...     206.010361    110.232109    51.485644  27.06985       1.636729       5.672940
min    8.170000e+02       1.000000       1.000000     101.000000       1.000000     110.000000  ...     110.000000    210.000000   310.000000 821.00000       0.000000    -118.667600
25%    2.106089e+08     900.000000       5.000000     587.000000       1.000000     331.000000  ...     331.000000    998.000000   998.000000 998.00000      34.014600    -118.430500
50%    2.208146e+08    1420.000000      11.000000    1141.000000       1.000000     442.000000  ...     442.000000    998.000000   998.000000 998.00000      34.058900    -118.322500
75%    2.309153e+08    1900.000000      16.000000    1617.000000       2.000000     626.000000  ...     626.000000    998.000000   998.000000 998.00000      34.164900    -118.273900
max    2.499253e+08    2359.000000      21.000000    2199.000000       2.000000     956.000000  ...     956.000000    999.000000   999.000000 999.00000      34.334300       0.000000

[8 rows x 15 columns]
       DR_NO               Date Rptd                DATE OCC  TIME OCC  AREA  ... Crm Cd 4                                  LOCATION  Cross Street      LAT       LON
0  190326475  03/01/2020 12:00:00 AM  03/01/2020 12:00:00 AM      2130     7  ...      NaN   1900 S  LONGWOOD                    AV           NaN  34.0375 -118.3506
1  200106753  02/09/2020 12:00:00 AM  02/08/2020 12:00:00 AM      1800     1  ...      NaN   1000 S  FLOWER                      ST           NaN  34.0444 -118.2628
2  200320258  11/11/2020 12:00:00 AM  11/04/2020 12:00:00 AM      1700     3  ...      NaN   1400 W  37TH                        ST           NaN  34.0210 -118.3002
3  200907217  05/10/2023 12:00:00 AM  03/10/2020 12:00:00 AM      2037     9  ...      NaN  14000    RIVERSIDE                   DR           NaN  34.1576 -118.4387
4  220614831  08/18/2022 12:00:00 AM  08/17/2020 12:00:00 AM      1200     6  ...      NaN                       1900   TRANSIENT           NaN  34.0944 -118.3277

[5 rows x 28 columns]
            DR_NO               Date Rptd                DATE OCC  TIME OCC  ...                                  LOCATION                    Cross Street      LAT       LON
982633  242011172  08/20/2024 12:00:00 AM  08/17/2024 12:00:00 AM      2300  ...  3700    WILSHIRE                        BL                             NaN  34.0617 -118.3066
982634  240710284  07/24/2024 12:00:00 AM  07/23/2024 12:00:00 AM      1400  ...  4000 W  23RD                            ST                             NaN  34.0362 -118.3284
982635  240104953  01/15/2024 12:00:00 AM  01/15/2024 12:00:00 AM       100  ...  1300 W  SUNSET                          BL                             NaN  34.0685 -118.2460
982636  240309674  04/24/2024 12:00:00 AM  04/24/2024 12:00:00 AM      1500  ...          FLOWER                          ST  JEFFERSON                   BL  34.0215 -118.2868
982637  240910892  08/13/2024 12:00:00 AM  08/12/2024 12:00:00 AM      2300  ...  6900    VESPER                          AV                             NaN  34.1961 -118.4510

[5 rows x 28 columns]
       DR_NO               Date Rptd                DATE OCC  TIME OCC  AREA  ... Crm Cd 4                                 LOCATION  Cross Street      LAT       LON
0  190326475  03/01/2020 12:00:00 AM  03/01/2020 12:00:00 AM      2130     7  ...      NaN  1900 S  LONGWOOD                    AV           NaN  34.0375 -118.3506
1  200106753  02/09/2020 12:00:00 AM  02/08/2020 12:00:00 AM      1800     1  ...      NaN  1000 S  FLOWER                      ST           NaN  34.0444 -118.2628

[2 rows x 28 columns]
            DR_NO               Date Rptd                DATE OCC  TIME OCC  ...                                  LOCATION                    Cross Street      LAT       LON
982635  240104953  01/15/2024 12:00:00 AM  01/15/2024 12:00:00 AM       100  ...  1300 W  SUNSET                          BL                             NaN  34.0685 -118.2460
982636  240309674  04/24/2024 12:00:00 AM  04/24/2024 12:00:00 AM      1500  ...          FLOWER                          ST  JEFFERSON                   BL  34.0215 -118.2868
982637  240910892  08/13/2024 12:00:00 AM  08/12/2024 12:00:00 AM      2300  ...  6900    VESPER                          AV                             NaN  34.1961 -118.4510

[3 rows x 28 columns]
<bound method DataFrame.info of             DR_NO               Date Rptd                DATE OCC  TIME OCC  ...                                  LOCATION                    Cross Street      LAT       LON
0       190326475  03/01/2020 12:00:00 AM  03/01/2020 12:00:00 AM      2130  ...   1900 S  LONGWOOD                       AV                             NaN  34.0375 -118.3506
1       200106753  02/09/2020 12:00:00 AM  02/08/2020 12:00:00 AM      1800  ...   1000 S  FLOWER                         ST                             NaN  34.0444 -118.2628
2       200320258  11/11/2020 12:00:00 AM  11/04/2020 12:00:00 AM      1700  ...   1400 W  37TH                           ST                             NaN  34.0210 -118.3002
3       200907217  05/10/2023 12:00:00 AM  03/10/2020 12:00:00 AM      2037  ...  14000    RIVERSIDE                      DR                             NaN  34.1576 -118.4387
```



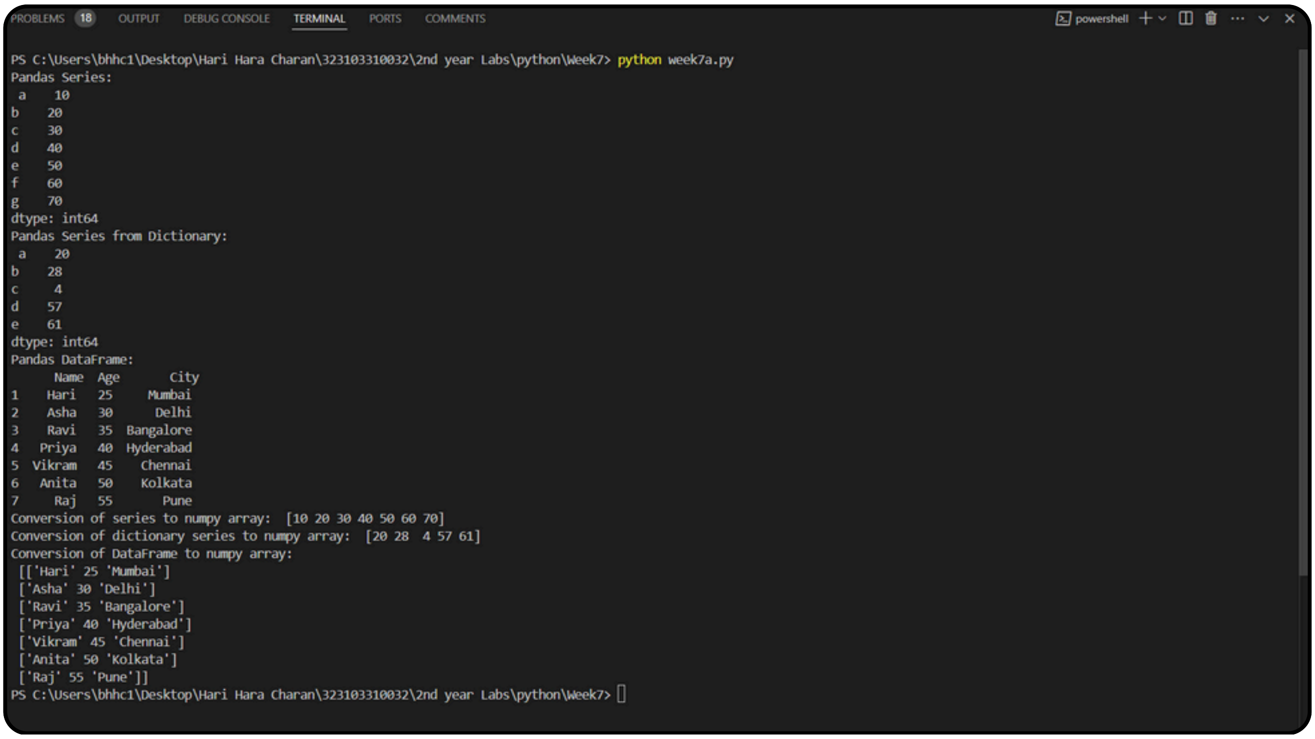
```
*****FROM JSON FILE******
<bound method DataFrame.info of      sepalLength  sepalWidth  petalLength  petalWidth    species
0            5.1         3.5          1.4         0.2     setosa
1            4.9         3.0          1.4         0.2     setosa
2            4.7         3.2          1.3         0.2     setosa
3            4.6         3.1          1.5         0.2     setosa
4            5.0         3.6          1.4         0.2     setosa
..           ...         ...          ...         ...        ...
145          6.7         3.0          5.2         2.3  virginica
146          6.3         2.5          5.0         1.9  virginica
147          6.5         3.0          5.2         2.0  virginica
148          6.2         3.4          5.4         2.3  virginica
149          5.9         3.0          5.1         1.8  virginica

[150 rows x 5 columns]>
       sepalLength  sepalWidth  petalLength  petalWidth
count   150.000000  150.000000   150.000000  150.000000
mean      5.843333    3.057333     3.758000    1.199333
std       0.828066    0.435866     1.765298    0.762238
min       4.300000    2.000000     1.000000    0.100000
25%       5.100000    2.800000     1.600000    0.300000
50%       5.800000    3.000000     4.350000    1.300000
std       0.828066    0.435866     1.765298    0.762238
min       4.300000    2.000000     1.000000    0.100000
25%       5.100000    2.800000     1.600000    0.300000
std       0.828066    0.435866     1.765298    0.762238
min       4.300000    2.000000     1.000000    0.100000
std       0.828066    0.435866     1.765298    0.762238
std       0.828066    0.435866     1.765298    0.762238
std       0.828066    0.435866     1.765298    0.762238
std       0.828066    0.435866     1.765298    0.762238
std       0.828066    0.435866     1.765298    0.762238
std       0.828066    0.435866     1.765298    0.762238
min       4.300000    2.000000     1.000000    0.100000
25%       5.100000    2.800000     1.600000    0.300000
50%       5.800000    3.000000     4.350000    1.300000
std       0.828066    0.435866     1.765298    0.762238
min       4.300000    2.000000     1.000000    0.100000
25%       5.100000    2.800000     1.600000    0.300000
50%       5.800000    3.000000     4.350000    1.300000
min       4.300000    2.000000     1.000000    0.100000
25%       5.100000    2.800000     1.600000    0.300000
50%       5.800000    3.000000     4.350000    1.300000
25%       5.100000    2.800000     1.600000    0.300000
50%       5.800000    3.000000     4.350000    1.300000
50%       5.800000    3.000000     4.350000    1.300000
75%       6.400000    3.300000     5.100000    1.800000
max       7.900000    4.400000     6.900000    2.500000
   sepalLength  sepalWidth  petalLength  petalWidth species
0          5.1         3.5          1.4         0.2  setosa
1          4.9         3.0          1.4         0.2  setosa
2          4.7         3.2          1.3         0.2  setosa
3          4.6         3.1          1.5         0.2  setosa
```

Week7
Pandas Library(Selection):

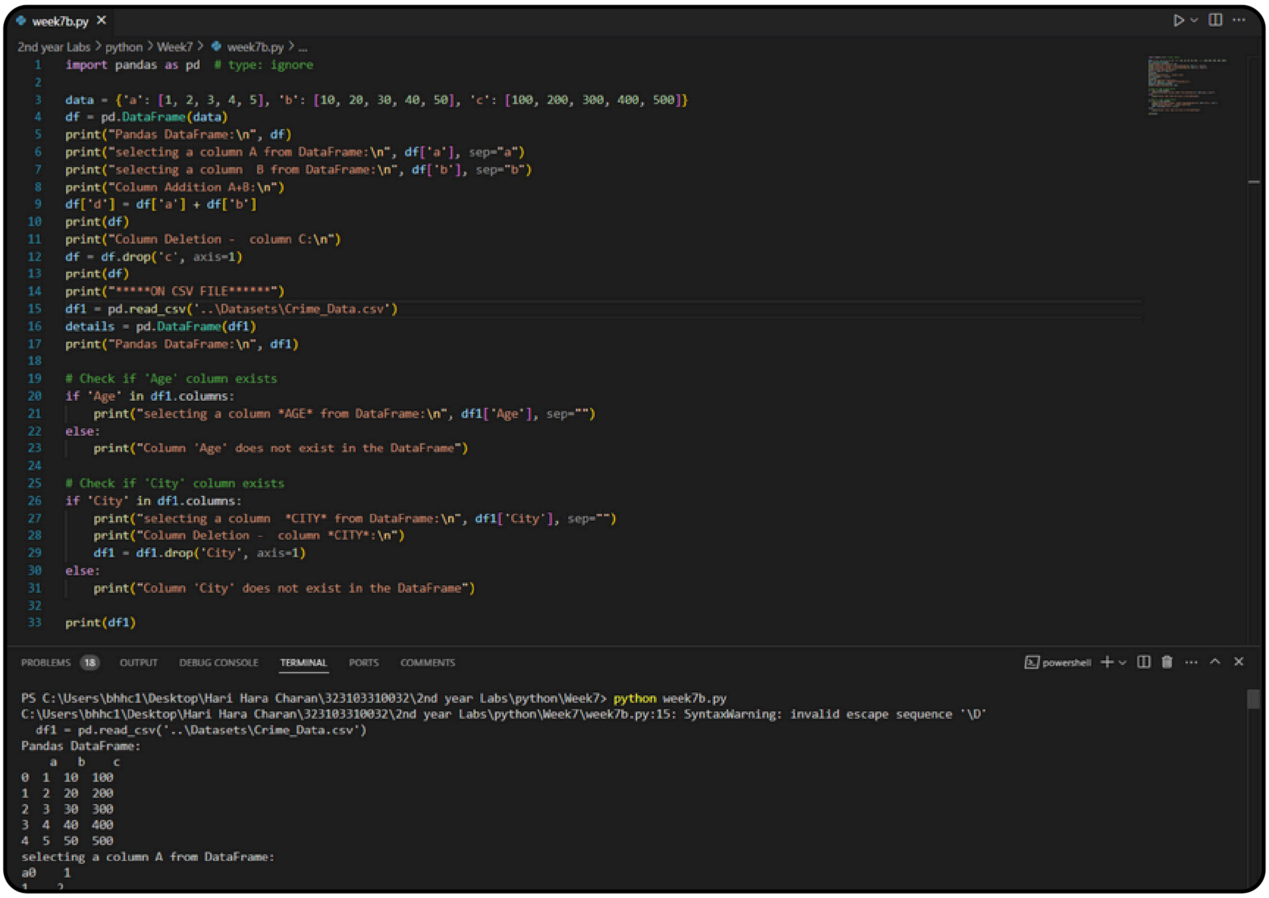


```python
import pandas as pd

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
data = [10, 20, 30, 40, 50, 60, 70]
series = pd.Series(data, index=labels)
print("Pandas Series:\n", series)
dataDict = {'a': 20, 'b': 28, 'c': 4, 'd': 57, 'e': 61}
seriesFromDist = pd.Series(dataDict)
print("Pandas Series from Dictionary:\n", seriesFromDist)
data = {
    'Name': ['Hari', 'Asha', 'Ravi', 'Priya', 'Vikram', 'Anita', 'Raj'],
    'Age': [25, 30, 35, 40, 45, 50, 55],
    'City': ['Mumbai', 'Delhi', 'Bangalore', 'Hyderabad', 'Chennai', 'Kolkata', 'Pune']
}
df = pd.DataFrame(data, index=[1, 2, 3, 4, 5, 6, 7])
print("Pandas DataFrame:\n", df)
print("Conversion of series to numpy array: ", series.to_numpy())
print("Conversion of dictionary series to numpy array: ", seriesFromDist.to_numpy())
print("Conversion of DataFrame to numpy array:\n", df.to_numpy())
```

```
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week7> python week7a.py
Pandas Series:
 a    10
b    20
c    30
d    40
e    50
f    60
g    70
dtype: int64
Pandas Series from Dictionary:
 a    20
b    28
c     4
d    57
e    61
dtype: int64
Pandas DataFrame:
```

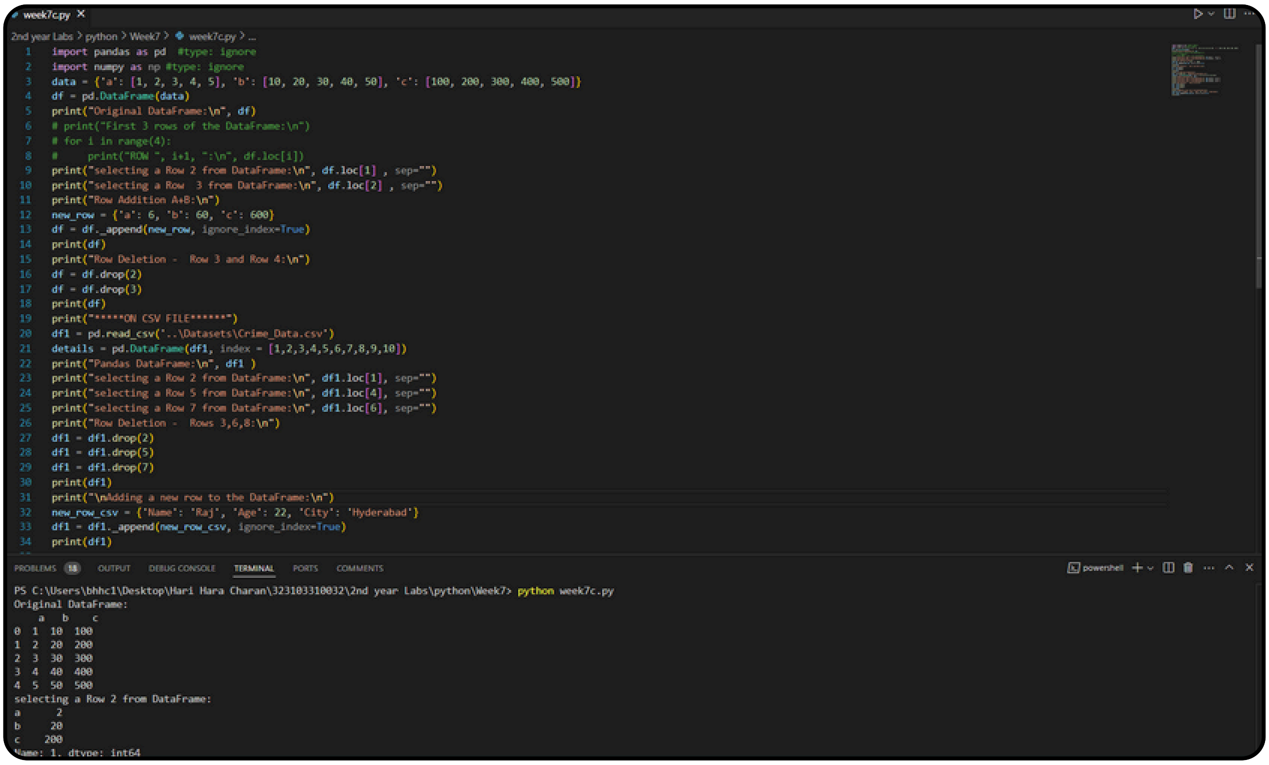


```
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week7> python week7a.py
Pandas Series:
 a    10
b    20
c    30
d    40
e    50
f    60
g    70
dtype: int64
Pandas Series from Dictionary:
 a    20
b    28
c     4
d    57
e    61
dtype: int64
Pandas DataFrame:
      Name  Age       City
1    Hari   25     Mumbai
2    Asha   30      Delhi
3    Ravi   35  Bangalore
4   Priya   40  Hyderabad
5  Vikram   45    Chennai
6   Anita   50    Kolkata
7     Raj   55       Pune
Conversion of series to numpy array:  [10 20 30 40 50 60 70]
Conversion of dictionary series to numpy array:  [20 28  4 57 61]
Conversion of DataFrame to numpy array:
 [['Hari' 25 'Mumbai']
 ['Asha' 30 'Delhi']
 ['Ravi' 35 'Bangalore']
 ['Priya' 40 'Hyderabad']
 ['Vikram' 45 'Chennai']
 ['Anita' 50 'Kolkata']
 ['Raj' 55 'Pune']]
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week7> 
```



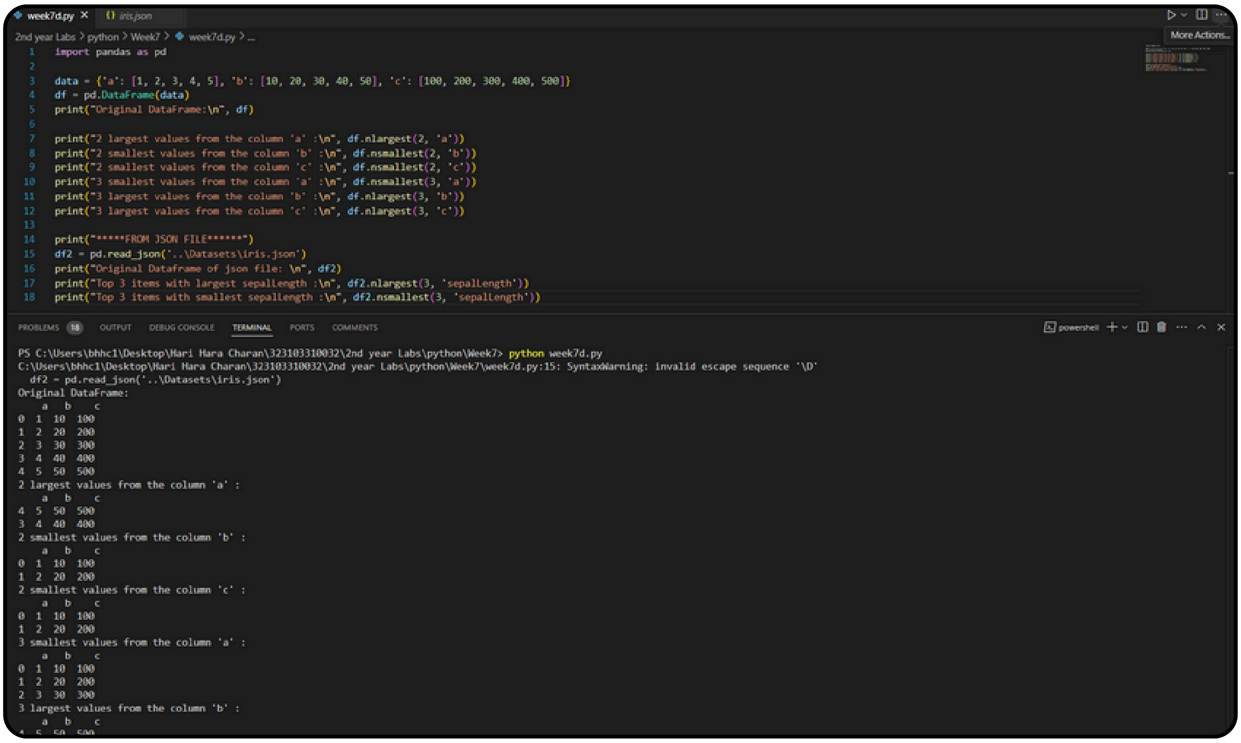
```python
import pandas as pd  # type: ignore

data = {'a': [1, 2, 3, 4, 5], 'b': [10, 20, 30, 40, 50], 'c': [100, 200, 300, 400, 500]}
df = pd.DataFrame(data)
print("Pandas DataFrame:\n", df)
print("selecting a column A from DataFrame:\n", df['a'], sep="a")
print("selecting a column  B from DataFrame:\n", df['b'], sep="b")
print("Column Addition A+B:\n")
df['d'] = df['a'] + df['b']
print(df)
print("Column Deletion -  column C:\n")
df = df.drop('c', axis=1)
print(df)
print("*****ON CSV FILE******")
df1 = pd.read_csv('..\Datasets\Crime_Data.csv')
details = pd.DataFrame(df1)
print("Pandas DataFrame:\n", df1)

# Check if 'Age' column exists
if 'Age' in df1.columns:
    print("selecting a column *AGE* from DataFrame:\n", df1['Age'], sep="")
else:
    print("Column 'Age' does not exist in the DataFrame")

# Check if 'City' column exists
if 'City' in df1.columns:
    print("selecting a column  *CITY* from DataFrame:\n", df1['City'], sep="")
    print("Column Deletion -  column *CITY*:\n")
    df1 = df1.drop('City', axis=1)
else:
    print("Column 'City' does not exist in the DataFrame")

print(df1)
```

```
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week7> python week7b.py
C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week7\week7b.py:15: SyntaxWarning: invalid escape sequence '\D'
  df1 = pd.read_csv('..\Datasets\Crime_Data.csv')
Pandas DataFrame:
   a   b    c
0  1  10  100
1  2  20  200
2  3  30  300
3  4  40  400
4  5  50  500
selecting a column A from DataFrame:
a0    1
1     2
```



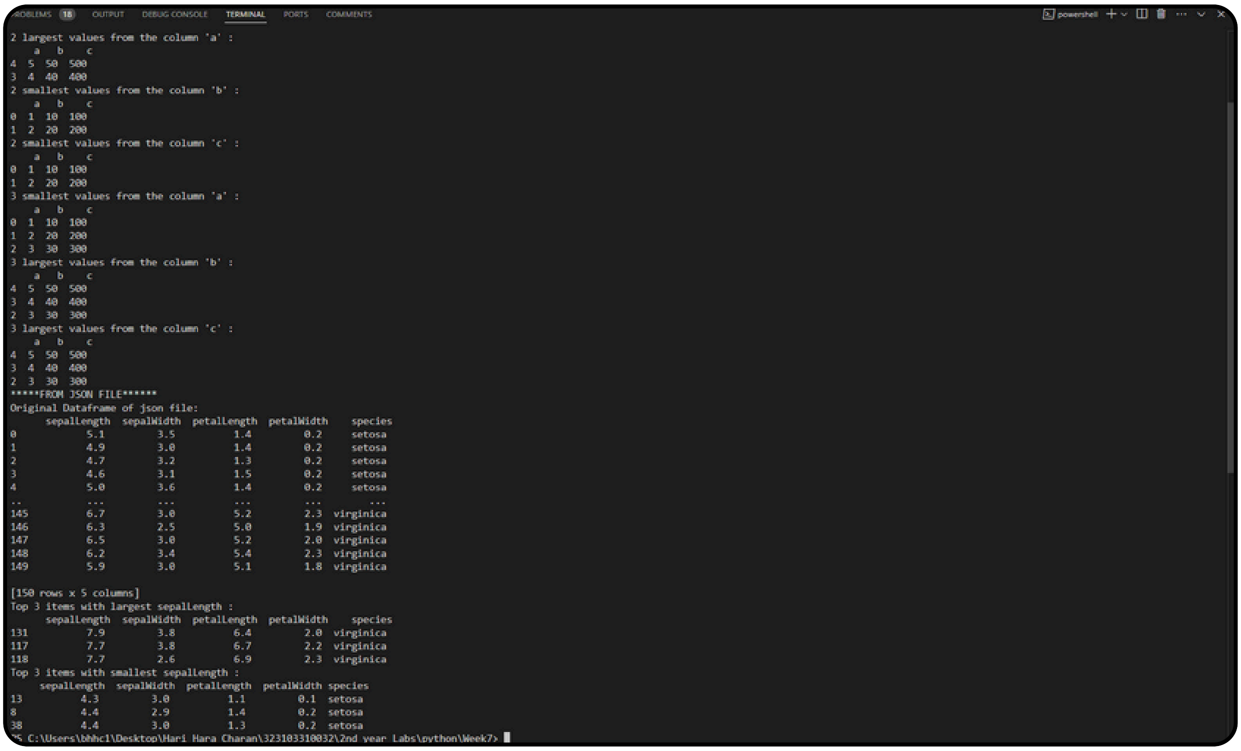
```python
import pandas as pd  #type: ignore
import numpy as np #type: ignore
data = {'a': [1, 2, 3, 4, 5], 'b': [10, 20, 30, 40, 50], 'c': [100, 200, 300, 400, 500]}
df = pd.DataFrame(data)
print("Original DataFrame:\n", df)
# print("First 3 rows of the DataFrame:\n")
# for i in range(4):
#     print("ROW ", i+1, ":\n", df.loc[i])
print("selecting a Row 2 from DataFrame:\n", df.loc[1] , sep="")
print("selecting a Row  3 from DataFrame:\n", df.loc[2] , sep="")
print("Row Addition A+B:\n")
new_row = {'a': 6, 'b': 60, 'c': 600}
df = df._append(new_row, ignore_index=True)
print(df)
print("Row Deletion -  Row 3 and Row 4:\n")
df = df.drop(2)
df = df.drop(3)
print(df)
print("*****ON CSV FILE******")
df1 = pd.read_csv('..\Datasets\Crime_Data.csv')
details = pd.DataFrame(df1, index = [1,2,3,4,5,6,7,8,9,10])
print("Pandas DataFrame:\n", df1 )
print("selecting a Row 2 from DataFrame:\n", df1.loc[1], sep="")
print("selecting a Row 5 from DataFrame:\n", df1.loc[4], sep="")
print("selecting a Row 7 from DataFrame:\n", df1.loc[6], sep="")
print("Row Deletion -  Rows 3,6,8:\n")
df1 = df1.drop(2)
df1 = df1.drop(5)
df1 = df1.drop(7)
print(df1)
print("\nAdding a new row to the DataFrame:\n")
new_row_csv = {'Name': 'Raj', 'Age': 22, 'City': 'Hyderabad'}
df1 = df1._append(new_row_csv, ignore_index=True)
print(df1)
```

```
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week7> python week7c.py
Original DataFrame:
   a   b    c
0  1  10  100
1  2  20  200
2  3  30  300
3  4  40  400
4  5  50  500
selecting a Row 2 from DataFrame:
a      2
b     20
c    200
Name: 1, dtype: int64
```



```python
import pandas as pd

data = {'a': [1, 2, 3, 4, 5], 'b': [10, 20, 30, 40, 50], 'c': [100, 200, 300, 400, 500]}
df = pd.DataFrame(data)
print("Original DataFrame:\n", df)

print("2 largest values from the column 'a' :\n", df.nlargest(2, 'a'))
print("2 smallest values from the column 'b' :\n", df.nsmallest(2, 'b'))
print("2 smallest values from the column 'c' :\n", df.nsmallest(2, 'c'))
print("3 smallest values from the column 'a' :\n", df.nsmallest(3, 'a'))
print("3 largest values from the column 'b' :\n", df.nlargest(3, 'b'))
print("3 largest values from the column 'c' :\n", df.nlargest(3, 'c'))

print("*****FROM JSON FILE******")
df2 = pd.read_json('..\Datasets\iris.json')
print("Original Dataframe of json file: \n", df2)
print("Top 3 items with largest sepalLength :\n", df2.nlargest(3, 'sepalLength'))
print("Top 3 items with smallest sepalLength :\n", df2.nsmallest(3, 'sepalLength'))
```

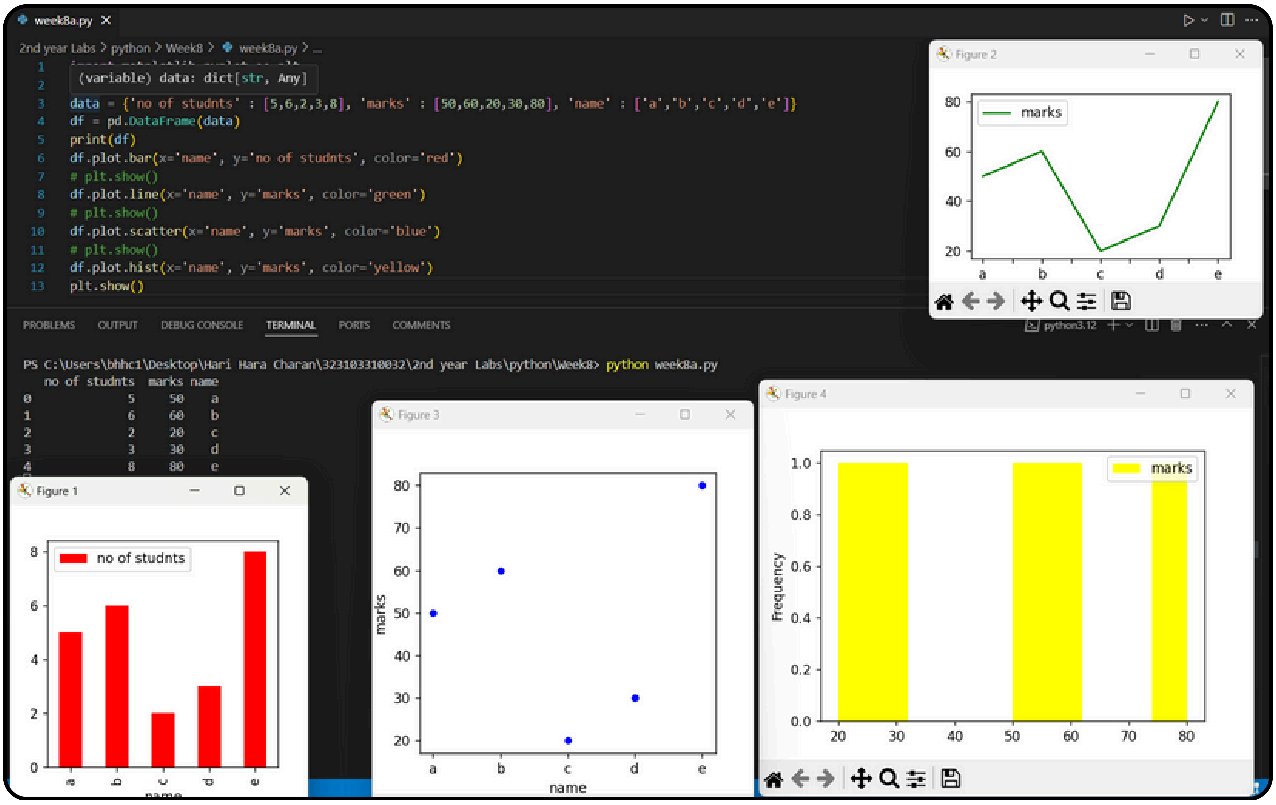
```
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week7> python week7d.py
C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week7\week7d.py:15: SyntaxWarning: invalid escape sequence '\D'
  df2 = pd.read_json('..\Datasets\iris.json')
Original DataFrame:
    a   b    c
0  1  10  100
1  2  20  200
2  3  30  300
3  4  40  400
4  5  50  500
2 largest values from the column 'a' :
    a   b    c
4  5  50  500
3  4  40  400
2 smallest values from the column 'b' :
    a   b    c
0  1  10  100
1  2  20  200
2 smallest values from the column 'c' :
    a   b    c
0  1  10  100
1  2  20  200
3 smallest values from the column 'a' :
    a   b    c
0  1  10  100
1  2  20  200
2  3  30  300
3 largest values from the column 'b' :
    a   b    c
4  5  50  500
```



```
2 largest values from the column 'a' :
    a   b    c
4  5  50  500
3  4  40  400
2 smallest values from the column 'b' :
    a   b    c
0  1  10  100
1  2  20  200
2 smallest values from the column 'c' :
    a   b    c
0  1  10  100
1  2  20  200
3 smallest values from the column 'a' :
    a   b    c
0  1  10  100
1  2  20  200
2  3  30  300
3 largest values from the column 'b' :
    a   b    c
4  5  50  500
3  4  40  400
2  3  30  300
3 largest values from the column 'c' :
    a   b    c
4  5  50  500
3  4  40  400
2  3  30  300
*****FROM JSON FILE******
Original Dataframe of json file:
      sepalLength  sepalWidth  petalLength  petalWidth    species
0            5.1         3.5          1.4         0.2     setosa
1            4.9         3.0          1.4         0.2     setosa
2            4.7         3.2          1.3         0.2     setosa
3            4.6         3.1          1.5         0.2     setosa
4            5.0         3.6          1.4         0.2     setosa
..           ...         ...          ...         ...        ...
145          6.7         3.0          5.2         2.3  virginica
146          6.3         2.5          5.0         1.9  virginica
147          6.5         3.0          5.2         2.0  virginica
148          6.2         3.4          5.4         2.3  virginica
149          5.9         3.0          5.1         1.8  virginica

[150 rows x 5 columns]
Top 3 items with largest sepalLength :
      sepalLength  sepalWidth  petalLength  petalWidth    species
131          7.9         3.8          6.4         2.0  virginica
117          7.7         3.8          6.7         2.2  virginica
118          7.7         2.6          6.9         2.3  virginica
Top 3 items with smallest sepalLength :
     sepalLength  sepalWidth  petalLength  petalWidth species
13           4.3         3.0          1.1         0.1  setosa
8            4.4         2.9          1.4         0.2  setosa
38           4.4         3.0          1.3         0.2  setosa
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week7>
```

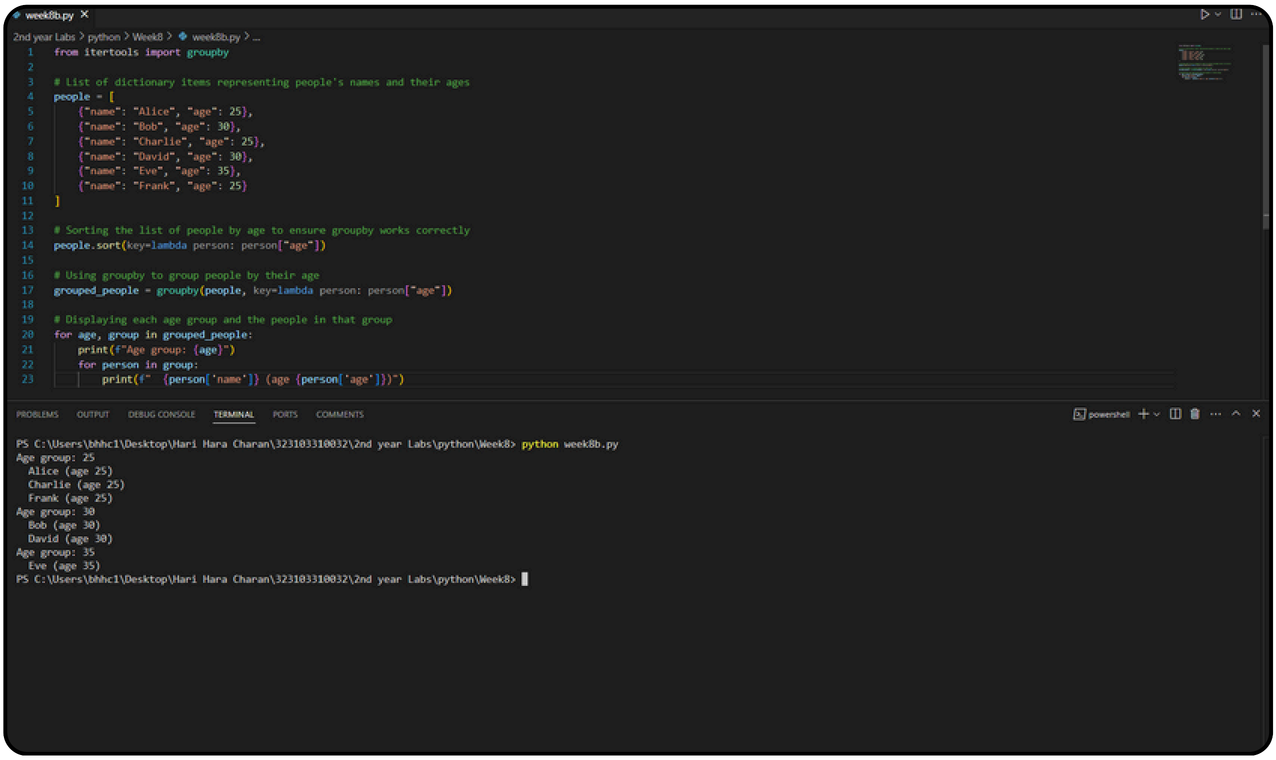
Week8

Pandas Library(Visualization):



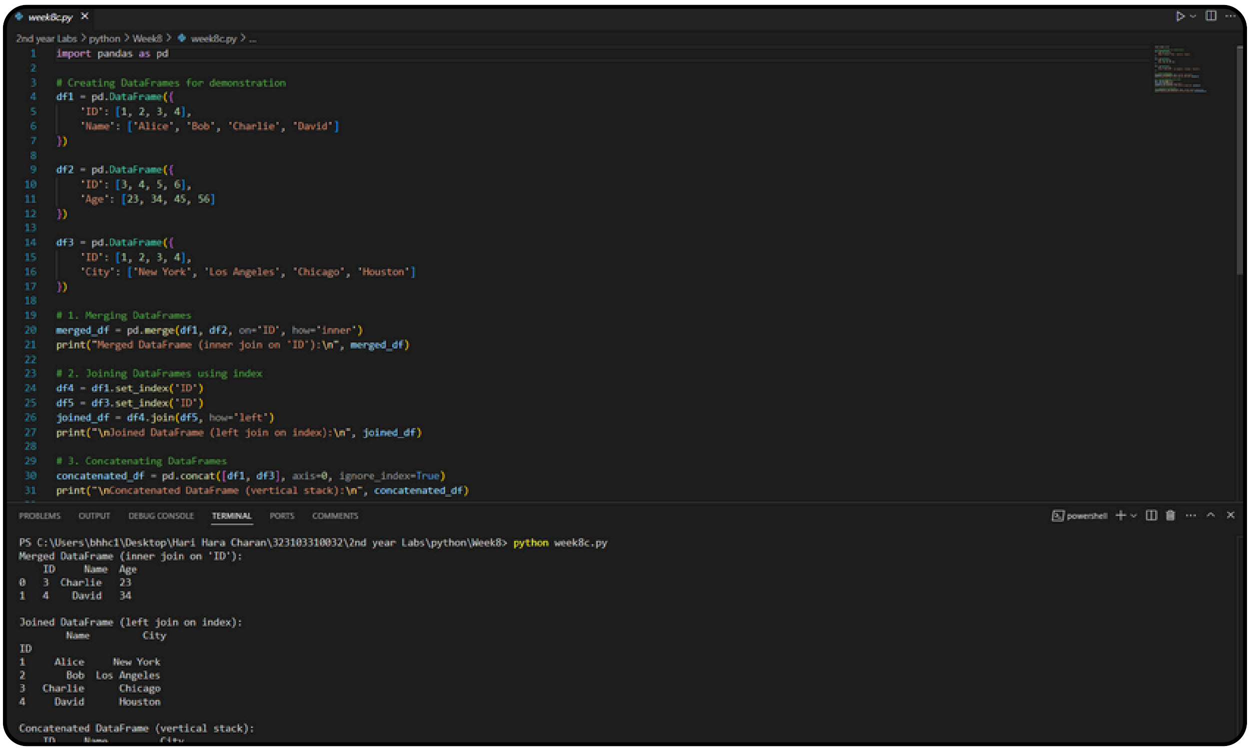
```
data = {'no of studnts' : [5,6,2,3,8], 'marks' : [50,60,20,30,80], 'name' : ['a','b','c','d','e']}
df = pd.DataFrame(data)
print(df)
df.plot.bar(x='name', y='no of studnts', color='red')
# plt.show()
df.plot.line(x='name', y='marks', color='green')
# plt.show()
df.plot.scatter(x='name', y='marks', color='blue')
# plt.show()
df.plot.hist(x='name', y='marks', color='yellow')
plt.show()
```

```
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week8> python week8a.py
   no of studnts  marks name
0              5     50    a
1              6     60    b
2              2     20    c
3              3     30    d
4              8     80    e
```



```
from itertools import groupby

# List of dictionary items representing people's names and their ages
people = [
    {"name": "Alice", "age": 25},
    {"name": "Bob", "age": 30},
    {"name": "Charlie", "age": 25},
    {"name": "David", "age": 30},
    {"name": "Eve", "age": 35},
    {"name": "Frank", "age": 25}
]

# Sorting the list of people by age to ensure groupby works correctly
people.sort(key=lambda person: person["age"])

# Using groupby to group people by their age
grouped_people = groupby(people, key=lambda person: person["age"])

# Displaying each age group and the people in that group
for age, group in grouped_people:
    print(f"Age group: {age}")
    for person in group:
        print(f"  {person['name']} (age {person['age']})")
```

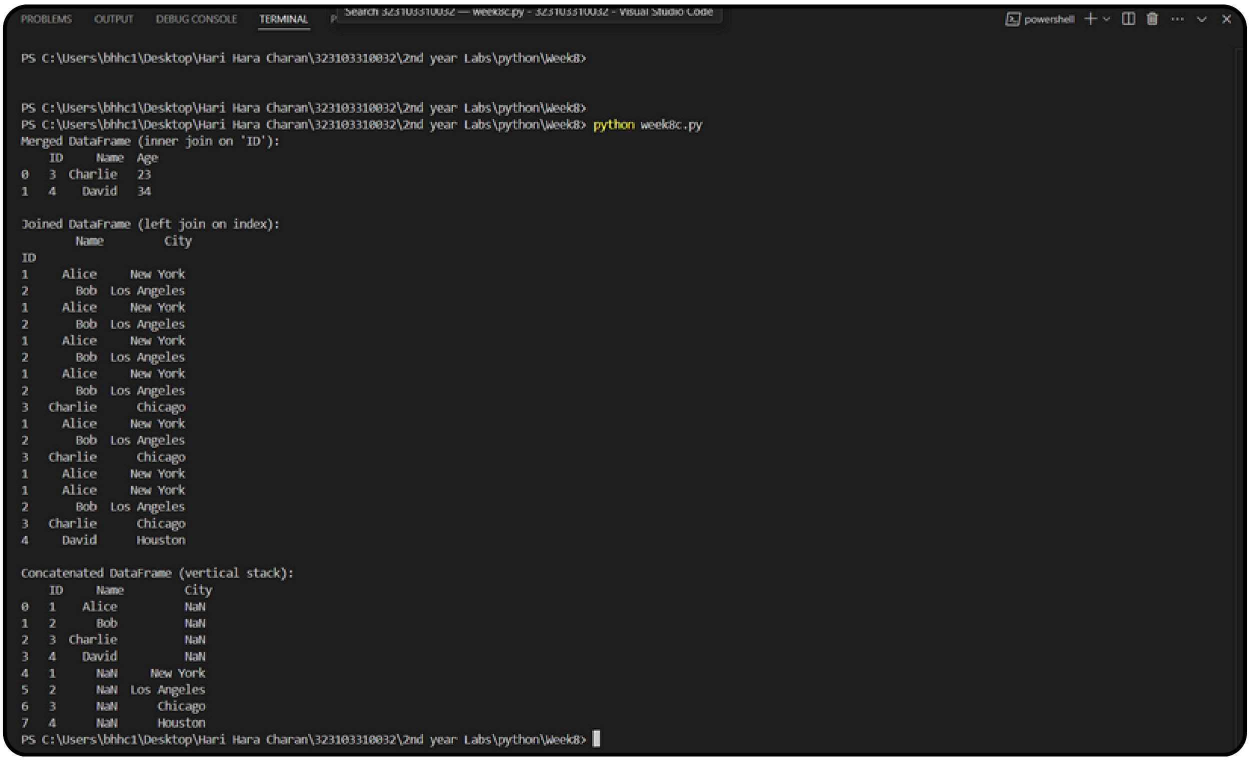
```
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week8> python week8b.py
Age group: 25
  Alice (age 25)
  Charlie (age 25)
  Frank (age 25)
Age group: 30
  Bob (age 30)
  David (age 30)
Age group: 35
  Eve (age 35)
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week8> 
```



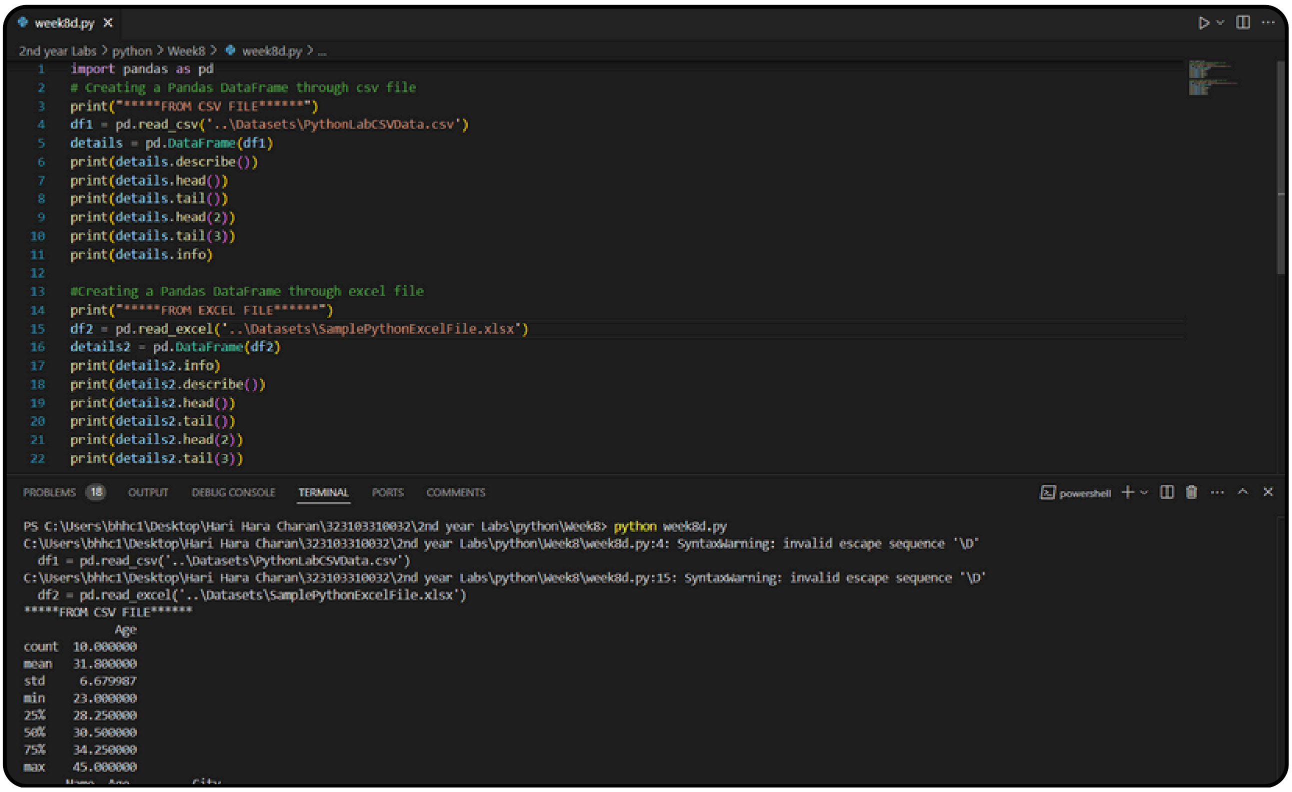
```python
import pandas as pd

# Creating DataFrames for demonstration
df1 = pd.DataFrame({
    'ID': [1, 2, 3, 4],
    'Name': ['Alice', 'Bob', 'Charlie', 'David']
})

df2 = pd.DataFrame({
    'ID': [3, 4, 5, 6],
    'Age': [23, 34, 45, 56]
})

df3 = pd.DataFrame({
    'ID': [1, 2, 3, 4],
    'City': ['New York', 'Los Angeles', 'Chicago', 'Houston']
})

# 1. Merging DataFrames
merged_df = pd.merge(df1, df2, on='ID', how='inner')
print("Merged DataFrame (inner join on 'ID'):\n", merged_df)

# 2. Joining DataFrames using index
df4 = df1.set_index('ID')
df5 = df3.set_index('ID')
joined_df = df4.join(df5, how='left')
print("\nJoined DataFrame (left join on index):\n", joined_df)

# 3. Concatenating DataFrames
concatenated_df = pd.concat([df1, df3], axis=0, ignore_index=True)
print("\nConcatenated DataFrame (vertical stack):\n", concatenated_df)
```

```
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week8> python week8c.py
Merged DataFrame (inner join on 'ID'):
   ID     Name  Age
0   3  Charlie   23
1   4    David   34

Joined DataFrame (left join on index):
       Name         City
ID
1     Alice     New York
2       Bob  Los Angeles
3   Charlie      Chicago
4     David      Houston

Concatenated DataFrame (vertical stack):
   ID     Name         City
```

Output:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Search 323103310032 — week8c.py - 323103310032 - Visual Studio Code
powershell

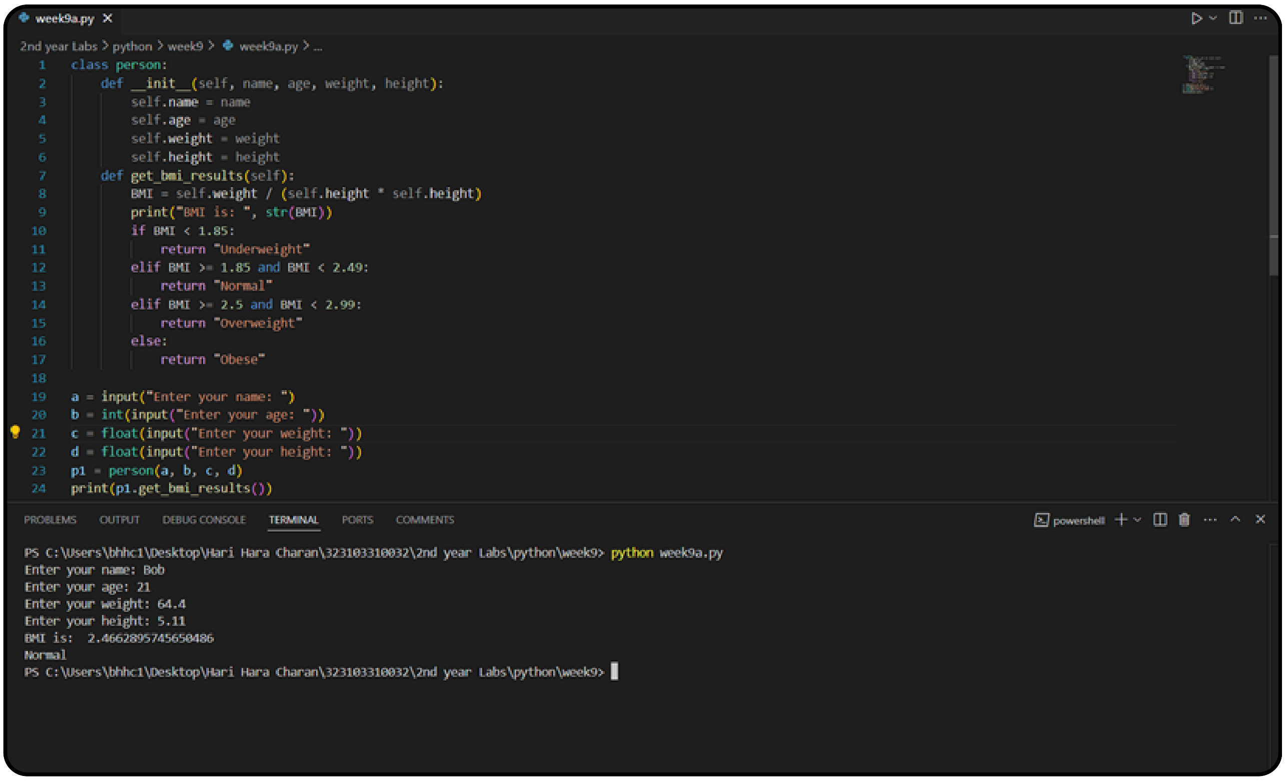
```
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week8>


PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week8>
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week8> python week8c.py
Merged DataFrame (inner join on 'ID'):
   ID     Name  Age
0   3  Charlie   23
1   4    David   34

Joined DataFrame (left join on index):
       Name         City
ID
1     Alice     New York
2       Bob  Los Angeles
1     Alice     New York
2       Bob  Los Angeles
1     Alice     New York
2       Bob  Los Angeles
1     Alice     New York
2       Bob  Los Angeles
3   Charlie      Chicago
1     Alice     New York
2       Bob  Los Angeles
3   Charlie      Chicago
1     Alice     New York
1     Alice     New York
2       Bob  Los Angeles
3   Charlie      Chicago
4     David      Houston

Concatenated DataFrame (vertical stack):
   ID     Name         City
0   1    Alice          NaN
1   2      Bob          NaN
2   3  Charlie          NaN
3   4    David          NaN
4   1      NaN     New York
5   2      NaN  Los Angeles
6   3      NaN      Chicago
7   4      NaN      Houston
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week8> 
```



```python
import pandas as pd
# Creating a Pandas DataFrame through csv file
print("*****FROM CSV FILE******")
df1 = pd.read_csv('..\Datasets\PythonLabCSVData.csv')
details = pd.DataFrame(df1)
print(details.describe())
print(details.head())
print(details.tail())
print(details.head(2))
print(details.tail(3))
print(details.info)

#Creating a Pandas DataFrame through excel file
print("*****FROM EXCEL FILE******")
df2 = pd.read_excel('..\Datasets\SamplePythonExcelFile.xlsx')
details2 = pd.DataFrame(df2)
print(details2.info)
print(details2.describe())
print(details2.head())
print(details2.tail())
print(details2.head(2))
print(details2.tail(3))
```

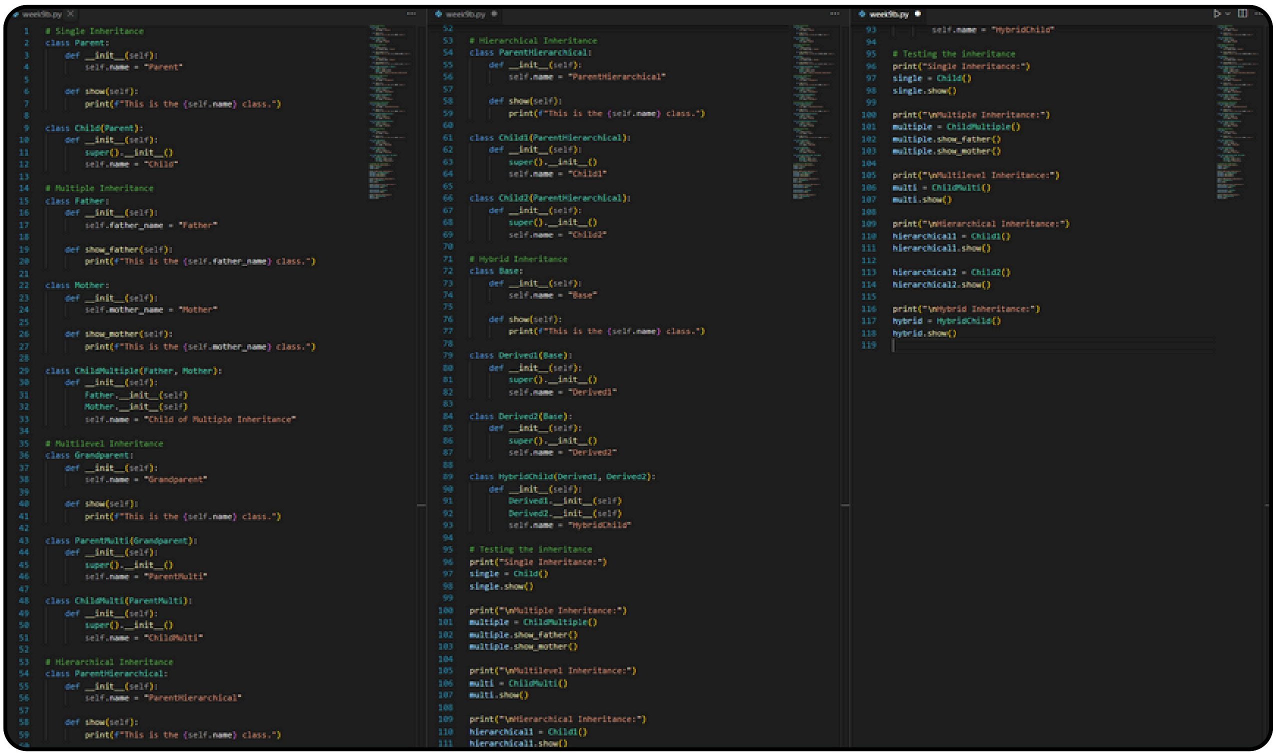
```
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week8> python week8d.py
C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week8\week8d.py:4: SyntaxWarning: invalid escape sequence '\D'
  df1 = pd.read_csv('..\Datasets\PythonLabCSVData.csv')
C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\Week8\week8d.py:15: SyntaxWarning: invalid escape sequence '\D'
  df2 = pd.read_excel('..\Datasets\SamplePythonExcelFile.xlsx')
*****FROM CSV FILE******
             Age
count  10.000000
mean   31.800000
std     6.679987
min    23.000000
25%    28.250000
50%    30.500000
75%    34.250000
max    45.000000
      Name  Age          City
```

Week9
Object Oriented Programming: basic



```python
class person:
    def __init__(self, name, age, weight, height):
        self.name = name
        self.age = age
        self.weight = weight
        self.height = height
    def get_bmi_results(self):
        BMI = self.weight / (self.height * self.height)
        print("BMI is: ", str(BMI))
        if BMI < 1.85:
            return "Underweight"
        elif BMI >= 1.85 and BMI < 2.49:
            return "Normal"
        elif BMI >= 2.5 and BMI < 2.99:
            return "Overweight"
        else:
            return "Obese"

a = input("Enter your name: ")
b = int(input("Enter your age: "))
c = float(input("Enter your weight: "))
d = float(input("Enter your height: "))
p1 = person(a, b, c, d)
print(p1.get_bmi_results())
```

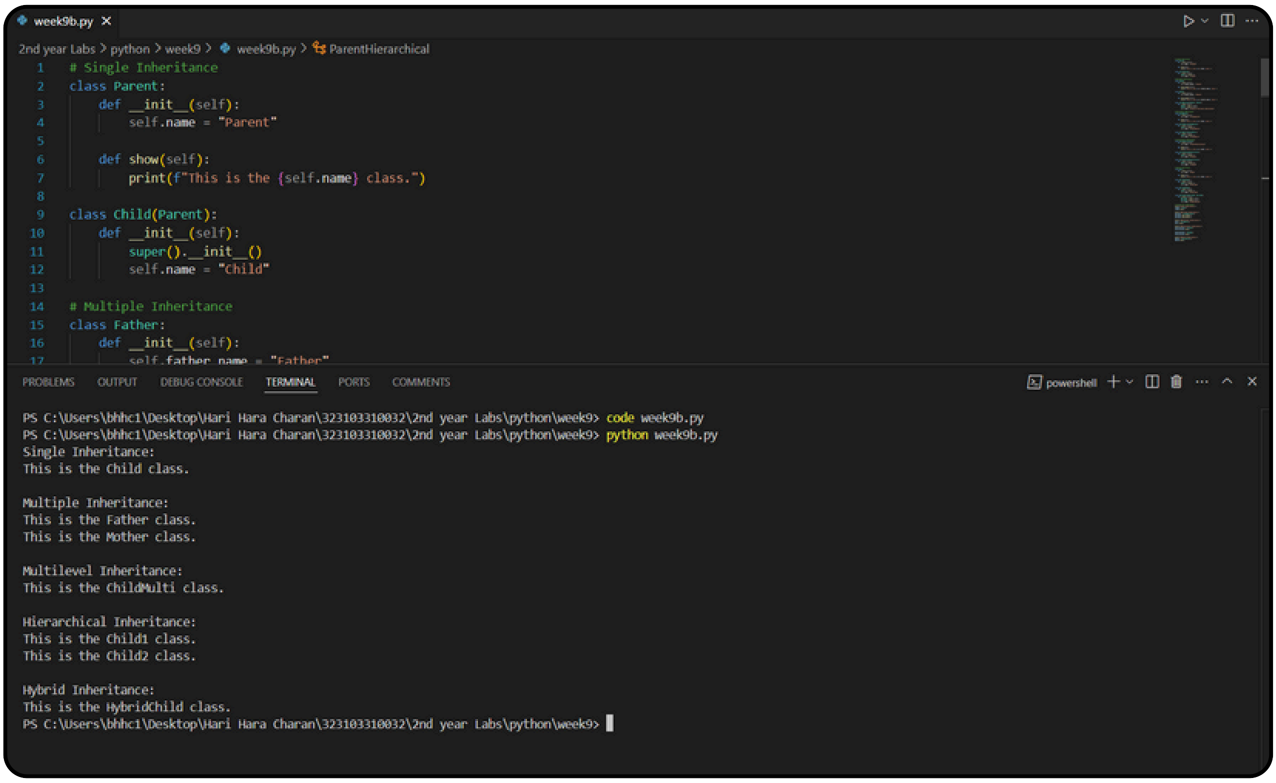
```
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\week9> python week9a.py
Enter your name: Bob
Enter your age: 21
Enter your weight: 64.4
Enter your height: 5.11
BMI is:  2.4662895745650486
Normal
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\week9> 
```



```python
# Single Inheritance
class Parent:
    def __init__(self):
        self.name = "Parent"

    def show(self):
        print(f"This is the {self.name} class.")

class Child(Parent):
    def __init__(self):
        super().__init__()
        self.name = "Child"

# Multiple Inheritance
class Father:
    def __init__(self):
        self.father_name = "Father"

    def show_father(self):
        print(f"This is the {self.father_name} class.")

class Mother:
    def __init__(self):
        self.mother_name = "Mother"

    def show_mother(self):
        print(f"This is the {self.mother_name} class.")

class ChildMultiple(Father, Mother):
    def __init__(self):
        Father.__init__(self)
        Mother.__init__(self)
        self.name = "Child of Multiple Inheritance"

# Multilevel Inheritance
class Grandparent:
    def __init__(self):
        self.name = "Grandparent"

    def show(self):
        print(f"This is the {self.name} class.")

class ParentMulti(Grandparent):
    def __init__(self):
        super().__init__()
        self.name = "ParentMulti"

class ChildMulti(ParentMulti):
    def __init__(self):
        super().__init__()
        self.name = "ChildMulti"

# Hierarchical Inheritance
class ParentHierarchical:
    def __init__(self):
        self.name = "ParentHierarchical"

    def show(self):
        print(f"This is the {self.name} class.")
```

```python

# Hierarchical Inheritance
class ParentHierarchical:
    def __init__(self):
        self.name = "ParentHierarchical"

    def show(self):
        print(f"This is the {self.name} class.")

class Child1(ParentHierarchical):
    def __init__(self):
        super().__init__()
        self.name = "Child1"

class Child2(ParentHierarchical):
    def __init__(self):
        super().__init__()
        self.name = "Child2"

# Hybrid Inheritance
class Base:
    def __init__(self):
        self.name = "Base"

    def show(self):
        print(f"This is the {self.name} class.")

class Derived1(Base):
    def __init__(self):
        super().__init__()
        self.name = "Derived1"

class Derived2(Base):
    def __init__(self):
        super().__init__()
        self.name = "Derived2"

class HybridChild(Derived1, Derived2):
    def __init__(self):
        Derived1.__init__(self)
        Derived2.__init__(self)
        self.name = "HybridChild"

# Testing the inheritance
print("Single Inheritance:")
single = Child()
single.show()

print("\nMultiple Inheritance:")
multiple = ChildMultiple()
multiple.show_father()
multiple.show_mother()

print("\nMultilevel Inheritance:")
multi = ChildMulti()
multi.show()

print("\nHierarchical Inheritance:")
hierarchical1 = Child1()
hierarchical1.show()
```

```python
        self.name = "HybridChild"

# Testing the inheritance
print("Single Inheritance:")
single = Child()
single.show()

print("\nMultiple Inheritance:")
multiple = ChildMultiple()
multiple.show_father()
multiple.show_mother()

print("\nMultilevel Inheritance:")
multi = ChildMulti()
multi.show()

print("\nHierarchical Inheritance:")
hierarchical1 = Child1()
hierarchical1.show()

hierarchical2 = Child2()
hierarchical2.show()

print("\nHybrid Inheritance:")
hybrid = HybridChild()
hybrid.show()

```


week9b.py ×
2nd year Labs > python > week9 > week9b.py > ParentHierarchical

```python
# Single Inheritance
class Parent:
    def __init__(self):
        self.name = "Parent"

    def show(self):
        print(f"This is the {self.name} class.")

class Child(Parent):
    def __init__(self):
        super().__init__()
        self.name = "Child"

# Multiple Inheritance
class Father:
    def __init__(self):
        self.father_name = "Father"
```

```
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\week9> code week9b.py
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\week9> python week9b.py
Single Inheritance:
This is the Child class.

Multiple Inheritance:
This is the Father class.
This is the Mother class.

Multilevel Inheritance:
This is the ChildMulti class.

Hierarchical Inheritance:
This is the Child1 class.
This is the Child2 class.

Hybrid Inheritance:
This is the HybridChild class.
PS C:\Users\bhhc1\Desktop\Hari Hara Charan\323103310032\2nd year Labs\python\week9> 
```