Day 10 Python Assignment no 4

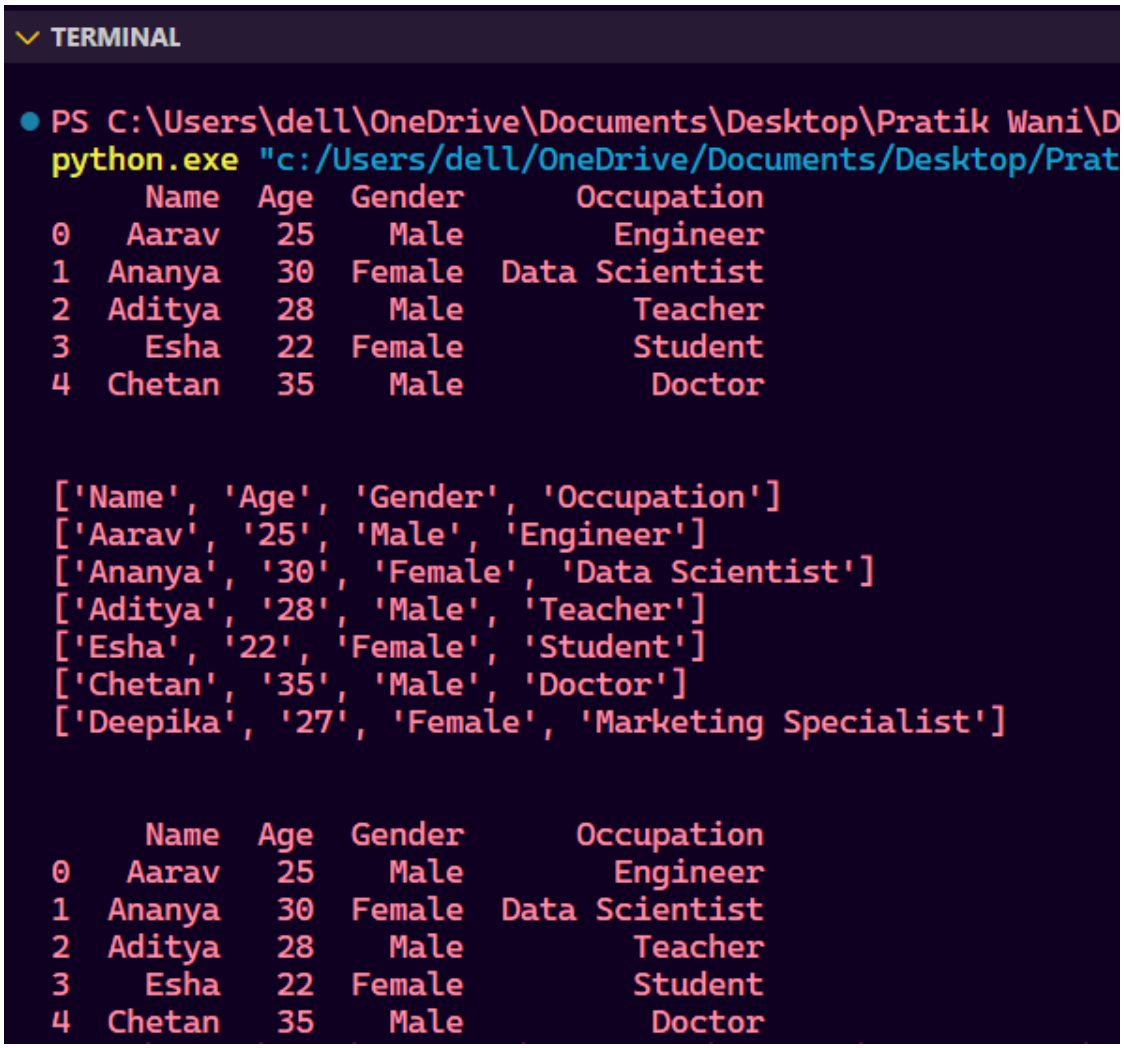
Pratik Wani

Pandas for Data Processing:

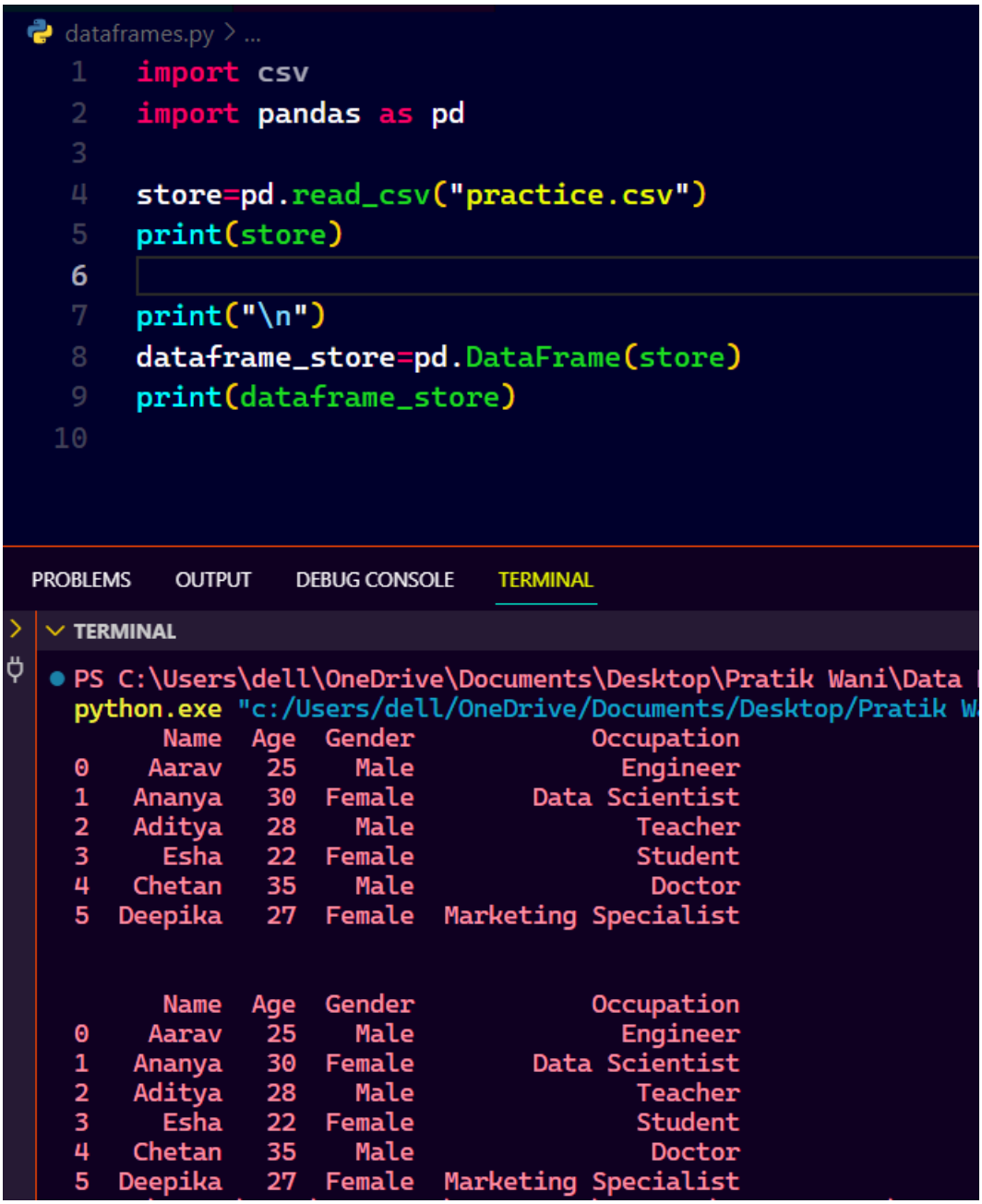
- Reading CSV Data using Pandas:
  - Three ways to read CSV data:
    - Read_table()
    - Using CSV module
    - Read_csv()

Read_csv.py > ...

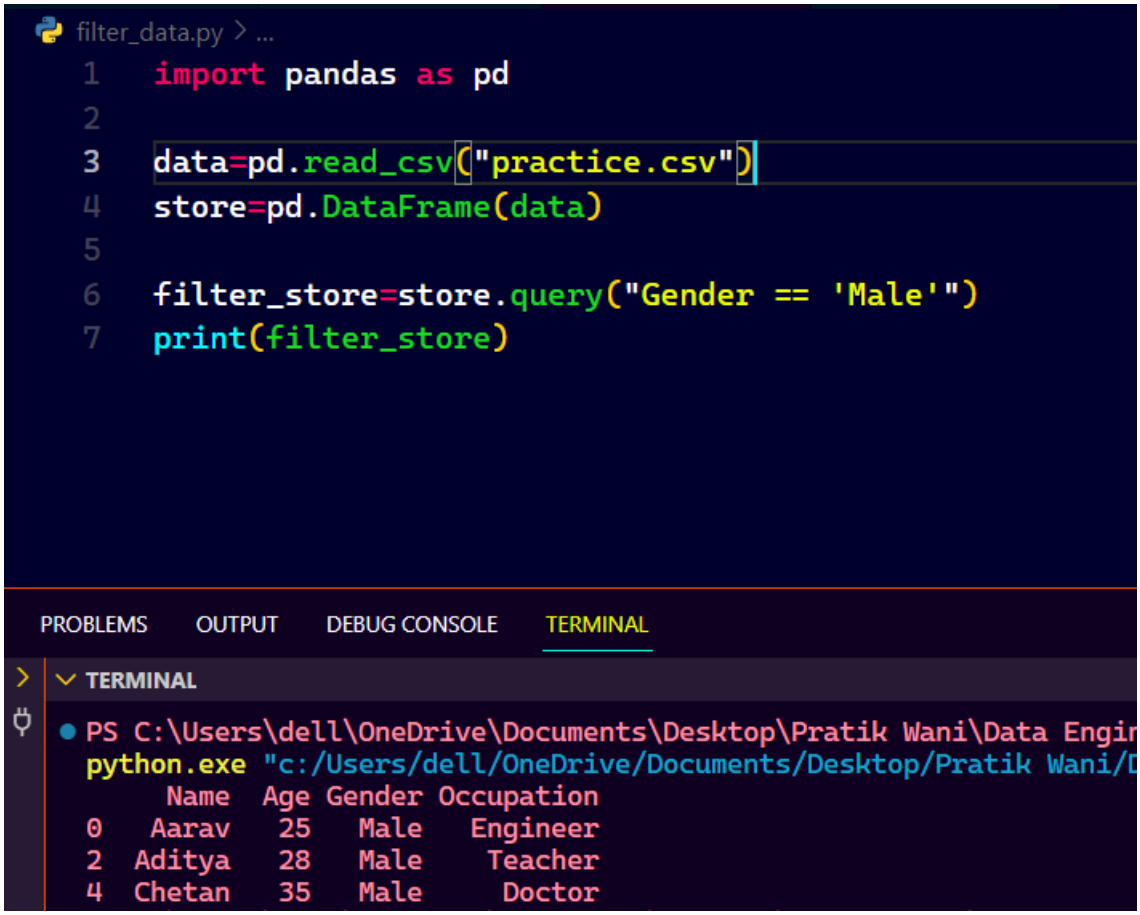
```python
#using read_table

import pandas as pd
file=pd.read_table("practice.csv",delimiter =",")

print(file.head())
print("\n")

#using csv module
import csv

with open("practice.csv") as file:
    store=csv.reader(file)
    data=pd.DataFrame([store],index=None)

    for i in range(0,len(list(data))):
        for val in list(data[i]):
            print(val)

print("\n")
#using read_csv

import pandas as pd
data=pd.read_csv("practice.csv")

print(data.head())
```


TERMINAL
PS C:\Users\dell\OneDrive\Documents\Desktop\Pratik Wani\D
python.exe "c:/Users/dell/OneDrive/Documents/Desktop/Prat
Name Age Gender Occupation
0 Aarav 25 Male Engineer
1 Ananya 30 Female Data Scientist
2 Aditya 28 Male Teacher
3 Esha 22 Female Student
4 Chetan 35 Male Doctor
['Name', 'Age', 'Gender', 'Occupation']
['Aarav', '25', 'Male', 'Engineer']
['Ananya', '30', 'Female', 'Data Scientist']
['Aditya', '28', 'Male', 'Teacher']
['Esha', '22', 'Female', 'Student']
['Chetan', '35', 'Male', 'Doctor']
['Deepika', '27', 'Female', 'Marketing Specialist']
Name Age Gender Occupation
0 Aarav 25 Male Engineer
1 Ananya 30 Female Data Scientist
2 Aditya 28 Male Teacher
3 Esha 22 Female Student
4 Chetan 35 Male Doctor

- Read Data from CSV Files to Pandas Dataframes:
    - To read the data from csv file to pandas dataframes we need to use read_csv method
    - And then using dataframes method we can convert it in dataframes



- Filter Data in Pandas Dataframe using query:
    - We can filter the data in pandas dataframes based on different criteria by using dataframe.query method



- Get Count by Gender using Pandas Dataframe APIs:
    - We can count the values in Pandas dataframe
    - First, we will create a data frame, and then we will count the values of different attributes.
    - Parameters:
        - Axis
        - Level
        - Numeric_only

count.py > ...

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

store=pd.read_csv("practice.csv")
data=pd.DataFrame(store)

print(data.count(),"\n")
print(data.count(axis=1),"\n")
print("Count of males:\n", (data.query("Gender == 'Male'")).count(),"\n")
```

TERMINAL

```
PS C:\Users\dell\OneDrive\Document
python.exe "c:/Users/dell/OneDrive
Name          6
Age           6
Gender        6
Occupation    6
dtype: int64

0    4
1    4
2    4
3    4
4    4
5    4
dtype: int64

Count of males:
 Name          3
Age           3
Gender        3
Occupation    3
dtype: int64
```

- Get Count by Age and Gender using Pandas Dataframe APIs:

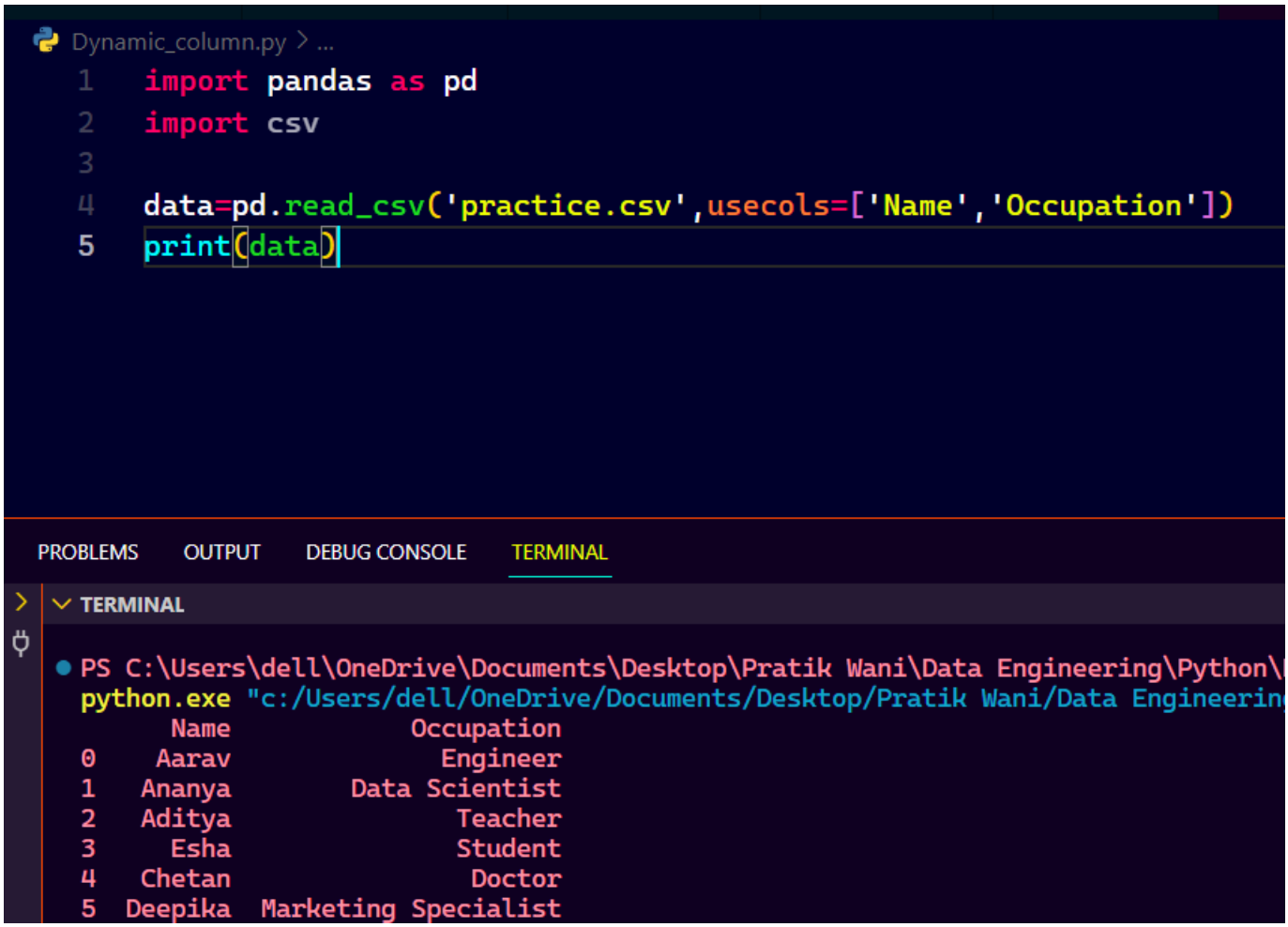
count.py > ...

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

store=pd.read_csv("practice.csv")
data=pd.DataFrame(store)


print("Count of males:\n", (data.query("Gender == 'Male' and Age>25")).count(),"\n")
```

TERMINAL

```
PS C:\Users\dell\OneDrive\Documents\Desktop\Pratik Wani\Data En
python.exe "c:/Users/dell/OneDrive/Documents/Desktop/Pratik Wan
Count of males:
 Name          2
Age            2
Gender         2
Occupation     2
dtype: int64
```

- Create Dataframes using dynamic column list on CSV Data:
    - We can get the particular column with the use of extra parameter usecols in read_csv method.
    - We just need to pass the list of column names which we want to extract from the csv file



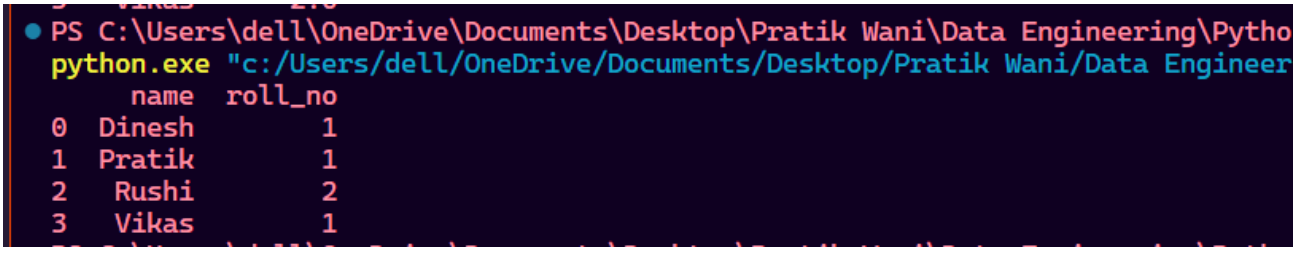
- Performing Inner Join between Pandas Dataframes:
    - Inner join is the most common type of join.
    - It returns a Dataframe with only those rows that have common characteristics.
    - This is similar to the intersection of two sets.

```python
import pandas as pd


data1= {'roll_no': [1, 2, 10, 12],
    'name': ['Pratik', 'Vikas', 'Rushi', 'Dinesh']}

dataframe1= pd.DataFrame(data1)


data2= {'roll_no': [1, 2, 10, 8],
    'course': ['Math', 'Science', 'History', 'English']}

dataframe2= pd.DataFrame(data2)


result= pd.merge(dataframe1, dataframe2, on='roll_no', how='inner')
print(result)
```

```
PS C:\Users\dell\OneDrive\Documents\Desktop\Pratik Wani\Data Engineering\Python\Day_10_P
python.exe "c:/Users/dell/OneDrive/Documents/Desktop/Pratik Wani/Data Engineering/Python
   roll_no    name   course
0        1  Pratik     Math
1        2   Vikas  Science
2       10   Rushi  History
PS C:\Users\dell\OneDrive\Documents\Desktop\Pratik Wani\Data Engineering\Python\Day_10_P
```

- Perform Aggregations on Join results:
    - Here I performed the count aggregate function on the merged dataframe

```python
import pandas as pd

data1= {'roll_no': [1, 2, 10, 12, 14],
    'name': ['Pratik', 'Vikas', 'Rushi', 'Dinesh','Rushi']}

dataframe1= pd.DataFrame(data1)



data2= {'roll_no': [1, 2, 10, 12, 14],
    'course': ['Math', 'Science', 'History', 'Science', 'Science']}

dataframe2= pd.DataFrame(data2)


result= pd.merge(dataframe1, dataframe2, on='roll_no', how='inner')
# print(result)


aggregated_result= result.groupby('name')['roll_no'].count().reset_index()
print(aggregated_result)
```



```
PS C:\Users\dell\OneDrive\Documents\Desktop\Pratik Wani\Data Engineering\Pytho
python.exe "c:/Users/dell/OneDrive/Documents/Desktop/Pratik Wani/Data Engineer
     name  roll_no
0  Dinesh        1
1  Pratik        1
2   Rushi        2
3   Vikas        1
```

- Sort Data in Pandas Dataframes:

```python
import pandas as pd

data=pd.read_csv("practice.csv")
dataframe=pd.DataFrame(data)
print(dataframe,"\n")

sorted_df=dataframe.sort_values(by='Name')
print("Ascending order:")
print(sorted_df,"\n")

sorted_df_desc=dataframe.sort_values(by='Name',ascending=False)


print("Descending order:")
print(sorted_df_desc)
```

```
TERMINAL
PS C:\Users\dell\OneDrive\Documents\Desktop\Pratik Wa
python.exe "c:/Users/dell/OneDrive/Documents/Desktop/
      Name  Age  Gender            Occupation
0    Aarav   25    Male              Engineer
1   Ananya   30  Female        Data Scientist
2   Aditya   28    Male               Teacher
3     Esha   22  Female               Student
4   Chetan   35    Male                Doctor
5  Deepika   27  Female  Marketing Specialist

Ascending order:
      Name  Age  Gender            Occupation
0    Aarav   25    Male              Engineer
2   Aditya   28    Male               Teacher
1   Ananya   30  Female        Data Scientist
4   Chetan   35    Male                Doctor
5  Deepika   27  Female  Marketing Specialist
3     Esha   22  Female               Student

Descending order:
      Name  Age  Gender            Occupation
3     Esha   22  Female               Student
5  Deepika   27  Female  Marketing Specialist
4   Chetan   35    Male                Doctor
1   Ananya   30  Female        Data Scientist
2   Aditya   28    Male               Teacher
0    Aarav   25    Male              Engineer
```

- Writing Pandas Dataframes to Files:
    - First we will convert the python dictionary into pandas dataframe
    - Then we will load the into the file by giving name if file not exist then it will be created
    - This is done using dataframe.to_csv function.to_csv function

write_csv.py > ...

```python
import pandas as pd

header=['Name','Age','Salary']
data=[['Pratik',21,'50000'],['Vikas',23,100000],['Rushi',22,120000]]
df=pd.DataFrame(data,columns=header)
print(df)


df.to_csv("student_salary_info.csv",index=False)

print("\n**********\n")
print(pd.read_csv('student_salary_info.csv'))
print("\n**********\n")
```

student_salary_info.csv

```
Name,Age,Salary
Pratik,21,50000
Vikas,23,100000
Rushi,22,120000
```

- Write Pandas Dataframes to JSON Files:
    - First we will convert the python dictionary into pandas dataframe
    - Then we will load the into the file by giving name if file not exist then it will be created
    - This is done using dataframe.to_json function.

dataframe_to_json.py > ...

```python
import pandas as pd

data={
    "name": ["Pratik","Vikas","Rushi"],
    "age": [21,23,25],
    "city": ["nashik","pune","nagpur"]
    }
df=pd.DataFrame(data)
print(df)


df.to_json("student_salary.json",orient='records')

print("\n**********\n")

```

student_salary.json > {} 2

```json
[
{"name":"Pratik","age":21,"city":"nashik"},
{"name":"Vikas","age":23,"city":"pune"},
{"name":"Rushi","age":25,"city":"nagpur"}
]
```

- Enriching Data using Numpy & Pandas:
    - There are functions provided by Numpy to create arrays with evenly spaced values within a given interval.
    - One is ‘arange' which is use to print sequence with a given distance and the other one 'linspace' needs the number of elements and creates the distance automatically.

numpy_practice.py > ...

```python
import numpy as np

cvalues = [20.1, 20.8, 21.9, 22.5, 22.7, 22.3, 21.8, 21.2, 20.9, 20.1]
C=np.array(cvalues)
print(C)

print(C*9/5+32)
print(C)
print(type(C))

a=np.arange(1,10)
print(a)

x=np.arange(10.4)
print(x)

x = np.arange(0.5, 10.4, 0.8)
print(x)

print(np.linspace(1,10))
print(np.linspace(1,10,7))
print(np.linspace(1,10,7,endpoint=False))

samples,spacing=np.linspace(1, 10, retstep=True)
print(spacing)

samples, spacing=np.linspace(1, 10, 20, endpoint=True, retstep=True)
print(spacing)

samples, spacing=np.linspace(1, 10, 20, endpoint=False, retstep=True)
print(spacing)
```

```
PS C:\Users\dell\OneDrive\Documents\Desktop\Pratik Wani\Data Engineering\Python\Day_10_Python_Assg_4> &
python.exe "c:/Users/dell/OneDrive/Documents/Desktop/Pratik Wani/Data Engineering/Python/Day_10_Python_A
[20.1 20.8 21.9 22.5 22.7 22.3 21.8 21.2 20.9 20.1]
[68.18 69.44 71.42 72.5  72.86 72.14 71.24 70.16 69.62 68.18]
[20.1 20.8 21.9 22.5 22.7 22.3 21.8 21.2 20.9 20.1]
<class 'numpy.ndarray'>
[1 2 3 4 5 6 7 8 9]
[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
[ 0.5  1.3  2.1  2.9  3.7  4.5  5.3  6.1  6.9  7.7  8.5  9.3 10.1]
[ 1.          1.18367347  1.36734694  1.55102041  1.73469388  1.91836735
  2.10204082  2.28571429  2.46938776  2.65306122  2.83673469  3.02040816
  3.20408163  3.3877551   3.57142857  3.75510204  3.93877551  4.12244898
  4.30612245  4.48979592  4.67346939  4.85714286  5.04081633  5.2244898
  5.40816327  5.59183673  5.7755102   5.95918367  6.14285714  6.32653061
  6.51020408  6.69387755  6.87755102  7.06122449  7.24489796  7.42857143
  7.6122449   7.79591837  7.97959184  8.16326531  8.34693878  8.53061224
  8.71428571  8.89795918  9.08163265  9.26530612  9.44897959  9.63265306
  9.81632653 10.        ]
[ 1.   2.5  4.   5.5  7.   8.5 10. ]
[1.         2.28571429 3.57142857 4.85714286 6.14285714 7.42857143
 8.71428571]
0.1836734693877551
0.47368421052631576
0.45
```