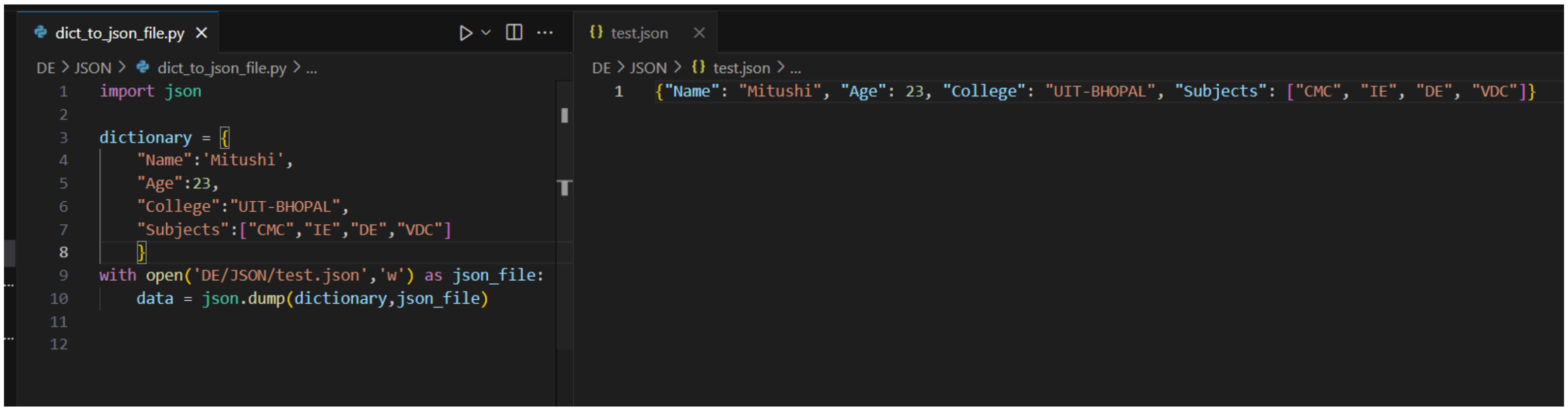
DAY-10 (Python Assignment - 4)
Mitushi Vishwakarma

- **Writing JSON to a file in Python :** Using file handling, Open a file in ‘w’ mode and then using dump() from json module will convert python dictionary into json string and it will get saved in file.



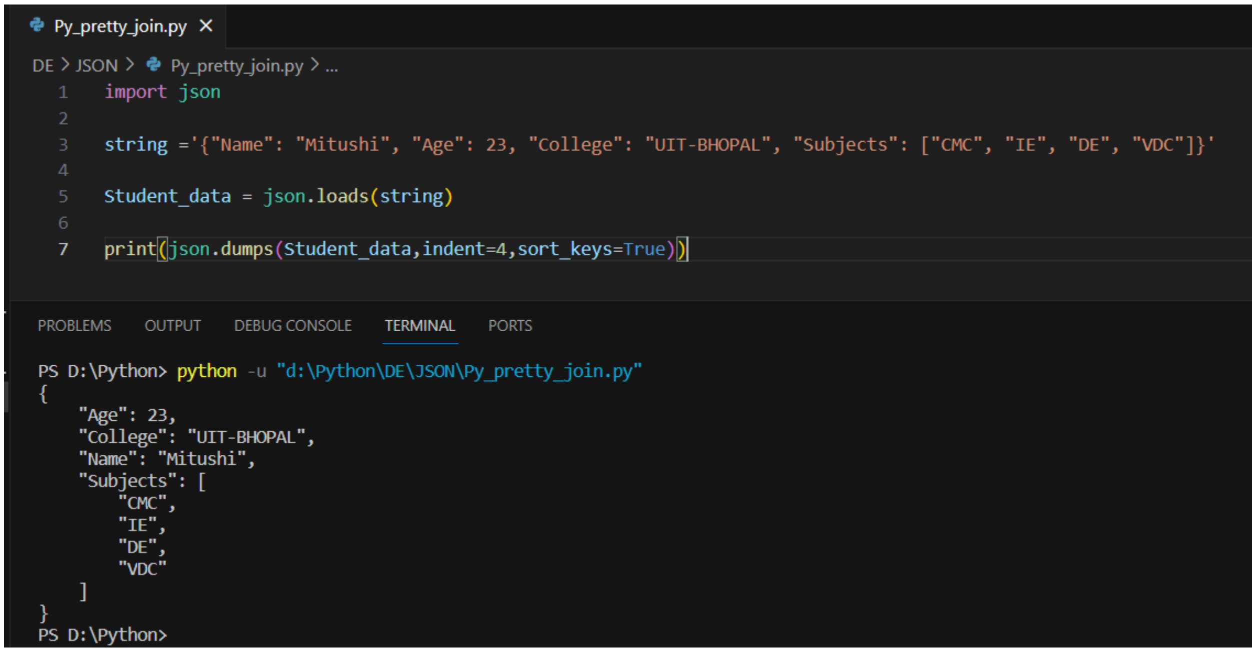
dict_to_json_file.py

```python
import json

dictionary = {
    "Name":'Mitushi',
    "Age":23,
    "College":"UIT-BHOPAL",
    "Subjects":["CMC","IE","DE","VDC"]
    }
with open('DE/JSON/test.json','w') as json_file:
    data = json.dump(dictionary,json_file)
```

test.json

```
{"Name": "Mitushi", "Age": 23, "College": "UIT-BHOPAL", "Subjects": ["CMC", "IE", "DE", "VDC"]}
```

- **Python Pretty Print JSON :** To make data more readable we can use pretty printing in json.dumps() by passing arguments in ‘indent’ and ‘sort_keys’.

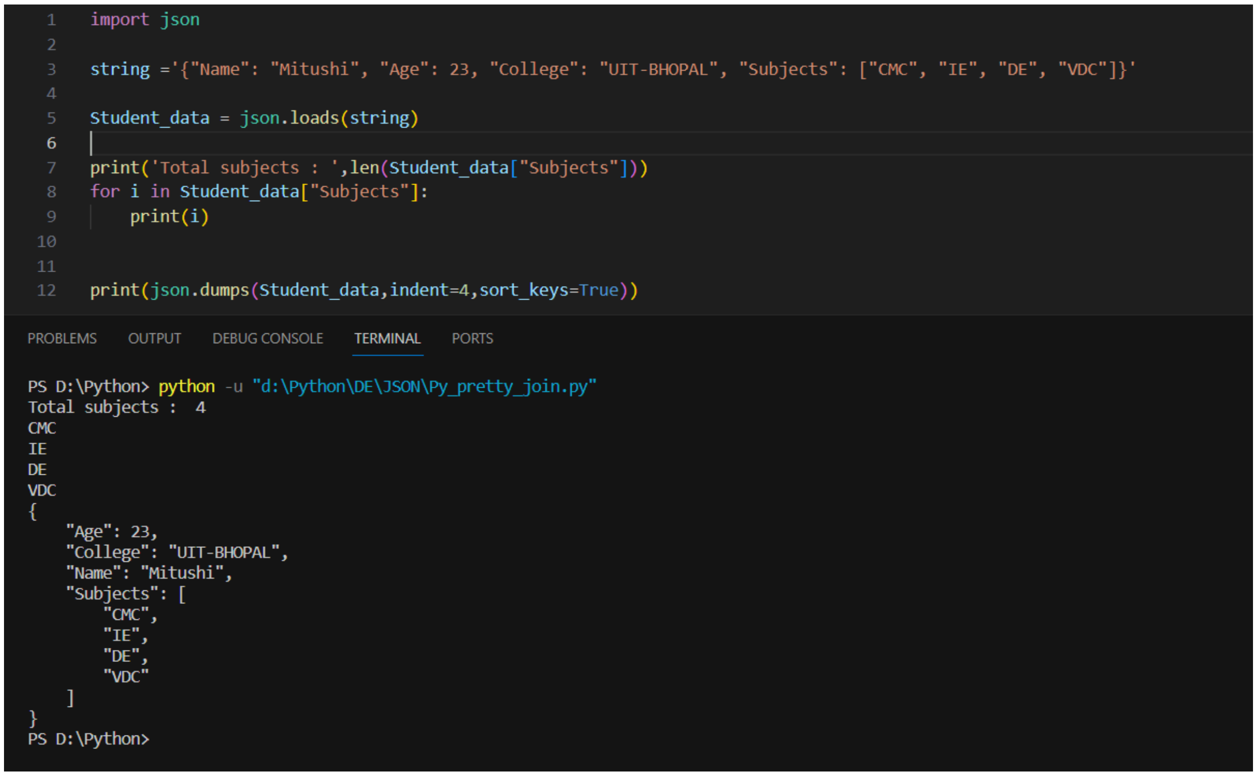


Py_pretty_join.py

```python
import json

string ='{"Name": "Mitushi", "Age": 23, "College": "UIT-BHOPAL", "Subjects": ["CMC", "IE", "DE", "VDC"]}'

Student_data = json.loads(string)

print(json.dumps(Student_data,indent=4,sort_keys=True))
```

```
PS D:\Python> python -u "d:\Python\DE\JSON\Py_pretty_join.py"
{
    "Age": 23,
    "College": "UIT-BHOPAL",
    "Name": "Mitushi",
    "Subjects": [
        "CMC",
        "IE",
        "DE",
        "VDC"
    ]
}
PS D:\Python>
```

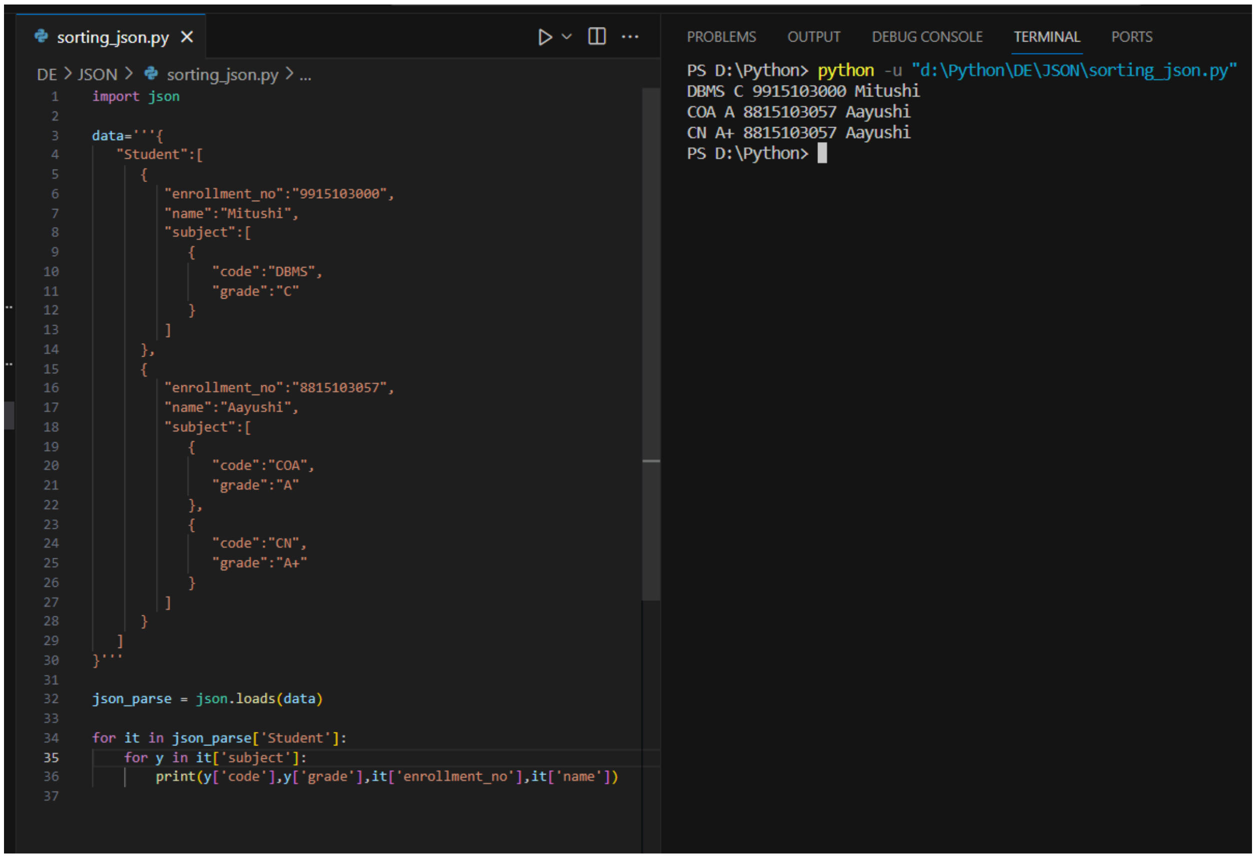
- **Extract Details from Complex JSON Arrays using Python :**
  Here we are extracting the array values of Subjects Key in json string.



```python
import json

string ='{"Name": "Mitushi", "Age": 23, "College": "UIT-BHOPAL", "Subjects": ["CMC", "IE", "DE", "VDC"]}'

Student_data = json.loads(string)

print('Total subjects : ',len(Student_data["Subjects"]))
for i in Student_data["Subjects"]:
    print(i)


print(json.dumps(Student_data,indent=4,sort_keys=True))
```

```
PS D:\Python> python -u "d:\Python\DE\JSON\Py_pretty_join.py"
Total subjects :  4
CMC
IE
DE
VDC
{
    "Age": 23,
    "College": "UIT-BHOPAL",
    "Name": "Mitushi",
    "Subjects": [
        "CMC",
        "IE",
        "DE",
        "VDC"
    ]
}
PS D:\Python>
```

- **Sort Data in JSON Arrays using Python :** We can sort data in json using custom sorting or using sort() function with pandas module.

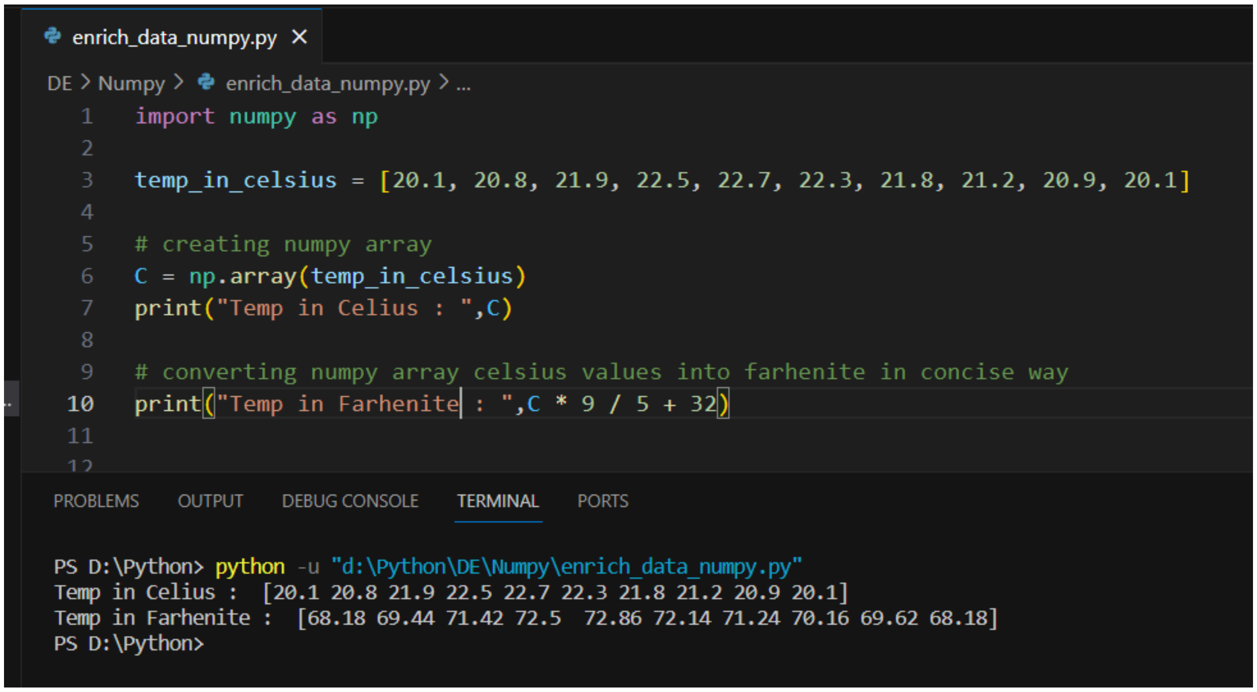


sorting_json.py
DE > JSON > sorting_json.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```python
import json

data='''{
    "Student":[
        {
            "enrollment_no":"9915103000",
            "name":"Mitushi",
            "subject":[
                {
                    "code":"DBMS",
                    "grade":"C"
                }
            ]
        },
        {
            "enrollment_no":"8815103057",
            "name":"Aayushi",
            "subject":[
                {
                    "code":"COA",
                    "grade":"A"
                },
                {
                    "code":"CN",
                    "grade":"A+"
                }
            ]
        }
    ]
}'''

json_parse = json.loads(data)

for it in json_parse['Student']:
    for y in it['subject']:
        print(y['code'],y['grade'],it['enrollment_no'],it['name'])

```

```
PS D:\Python> python -u "d:\Python\DE\JSON\sorting_json.py"
DBMS C 9915103000 Mitushi
COA A 8815103057 Aayushi
CN A+ 8815103057 Aayushi
PS D:\Python>
```

- **Enriching Data using Numpy :** Using numpy array, it is very easy and concise to perform a collective operation on all the values all together by using numpy array variable.


enrich_data_numpy.py
DE > Numpy > enrich_data_numpy.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```python
import numpy as np

temp_in_celsius = [20.1, 20.8, 21.9, 22.5, 22.7, 22.3, 21.8, 21.2, 20.9, 20.1]

# creating numpy array
C = np.array(temp_in_celsius)
print("Temp in Celius : ",C)

# converting numpy array celsius values into farhenite in concise way
print("Temp in Farhenite : ",C * 9 / 5 + 32)
```

```
PS D:\Python> python -u "d:\Python\DE\Numpy\enrich_data_numpy.py"
Temp in Celius :  [20.1 20.8 21.9 22.5 22.7 22.3 21.8 21.2 20.9 20.1]
Temp in Farhenite :  [68.18 69.44 71.42 72.5  72.86 72.14 71.24 70.16 69.62 68.18]
PS D:\Python>
```

• Creation of Arrays with Evenly Spaced Values

Using Arange() : The advantage of numpy.arange() over the normal in-built range() function is that it allows us to generate sequences of numbers that are not integers.

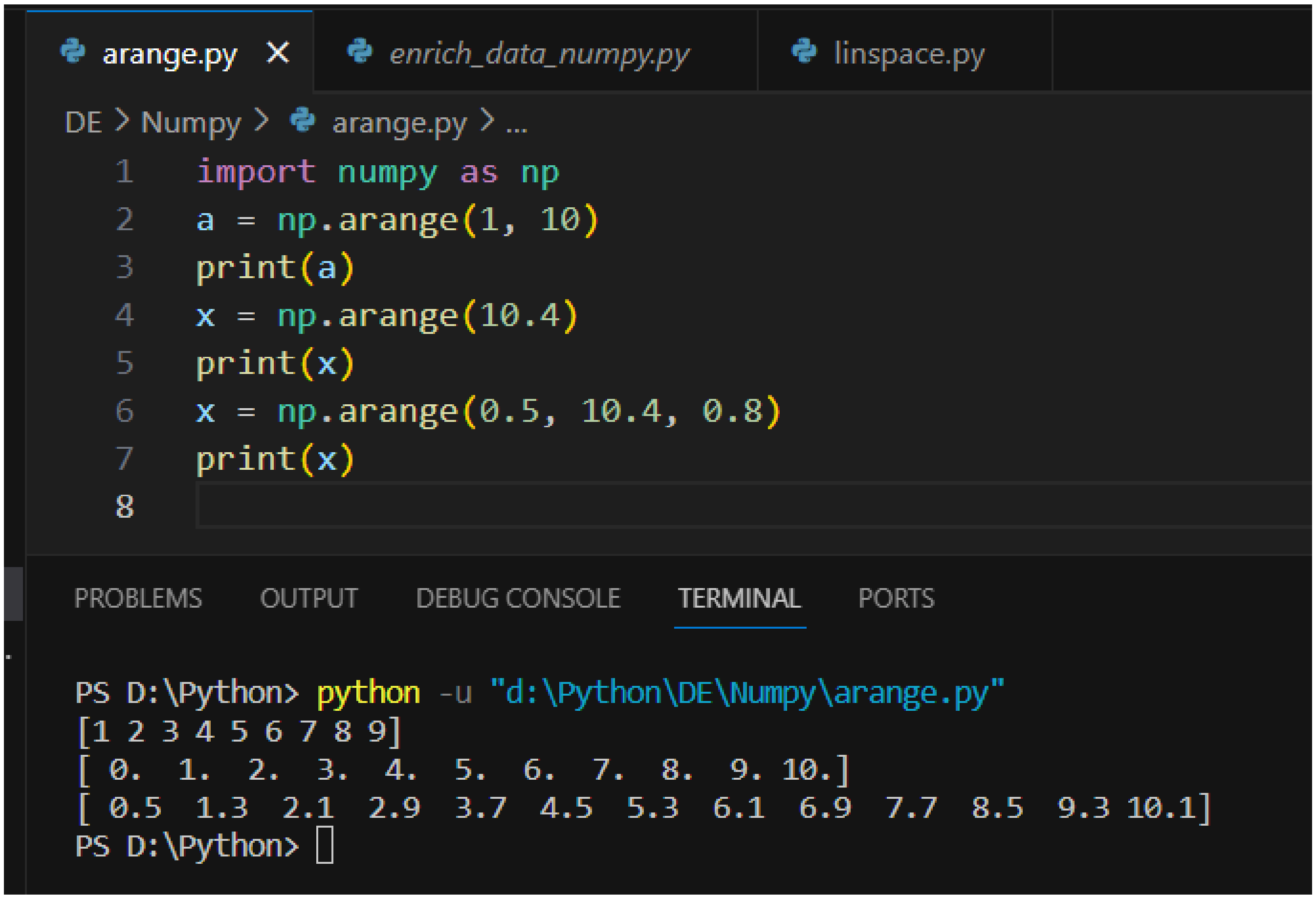
start : [optional] start of interval range. By default start = 0
stop : end of interval range
step : [optional] step size of interval. By default step size = 1,
dtype : type of output array

Using linspace() : The syntax for linspace is given below :

numpy.linspace(start, stop, num=50, endpoint=True, retstep=False)

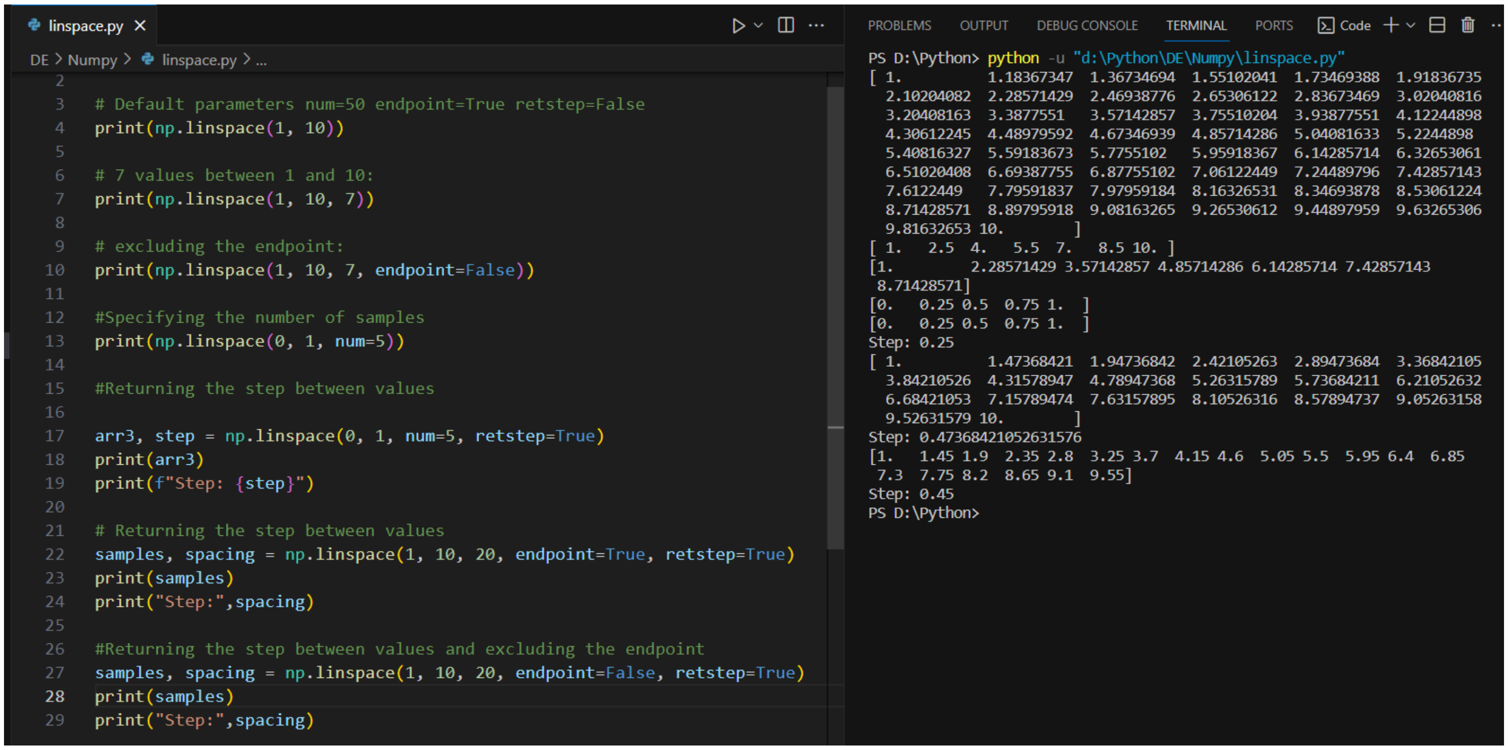
- start: The starting value of the sequence.
- stop: The end value of the sequence.
- num: Optional. The number of evenly spaced samples to generate. The default is 50.
- endpoint: Optional. If True (default), stop is the last value in the range. If False, the range does not include stop.

- retstep: Optional. If True, return the step between values as well.



```python
import numpy as np
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)
```

```
PS D:\Python> python -u "d:\Python\DE\Numpy\arange.py"
[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]
PS D:\Python>
```



```python

# Default parameters num=50 endpoint=True retstep=False
print(np.linspace(1, 10))

# 7 values between 1 and 10:
print(np.linspace(1, 10, 7))

# excluding the endpoint:
print(np.linspace(1, 10, 7, endpoint=False))

#Specifying the number of samples
print(np.linspace(0, 1, num=5))

#Returning the step between values

arr3, step = np.linspace(0, 1, num=5, retstep=True)
print(arr3)
print(f"Step: {step}")

# Returning the step between values
samples, spacing = np.linspace(1, 10, 20, endpoint=True, retstep=True)
print(samples)
print("Step:",spacing)

#Returning the step between values and excluding the endpoint
samples, spacing = np.linspace(1, 10, 20, endpoint=False, retstep=True)
print(samples)
print("Step:",spacing)
```

```
PS D:\Python> python -u "d:\Python\DE\Numpy\linspace.py"
[ 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.   0.25 0.5  0.75 1.  ]
[0.   0.25 0.5  0.75 1.  ]
Step: 0.25
[ 1.          1.47368421  1.94736842  2.42105263  2.89473684  3.36842105
  3.84210526  4.31578947  4.78947368  5.26315789  5.73684211  6.21052632
  6.68421053  7.15789474  7.63157895  8.10526316  8.57894737  9.05263158
  9.52631579 10.        ]
Step: 0.47368421052631576
[1.   1.45 1.9  2.35 2.8  3.25 3.7  4.15 4.6  5.05 5.5  5.95 6.4  6.85
 7.3  7.75 8.2  8.65 9.1  9.55]
Step: 0.45
PS D:\Python>
```

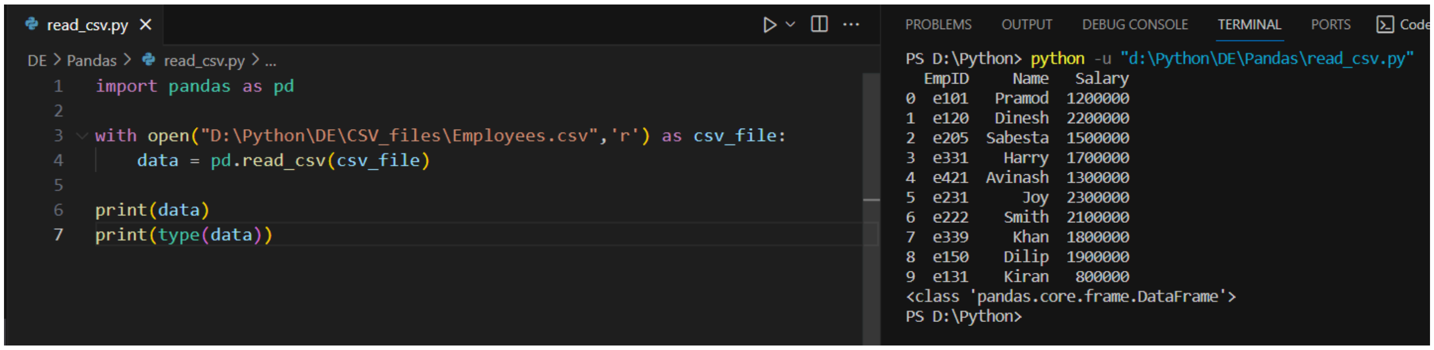
Data Processing using Pandas Dataframe APIs

- **Reading CSV Data using Pandas**

There are three methods to read CSV files using pandas :

METHOD 1 : using read_csv()

We use read_csv() method to read csv data. First we open file using file handling and then use the read_csv() to store the data into a variable whose type is dataframe.

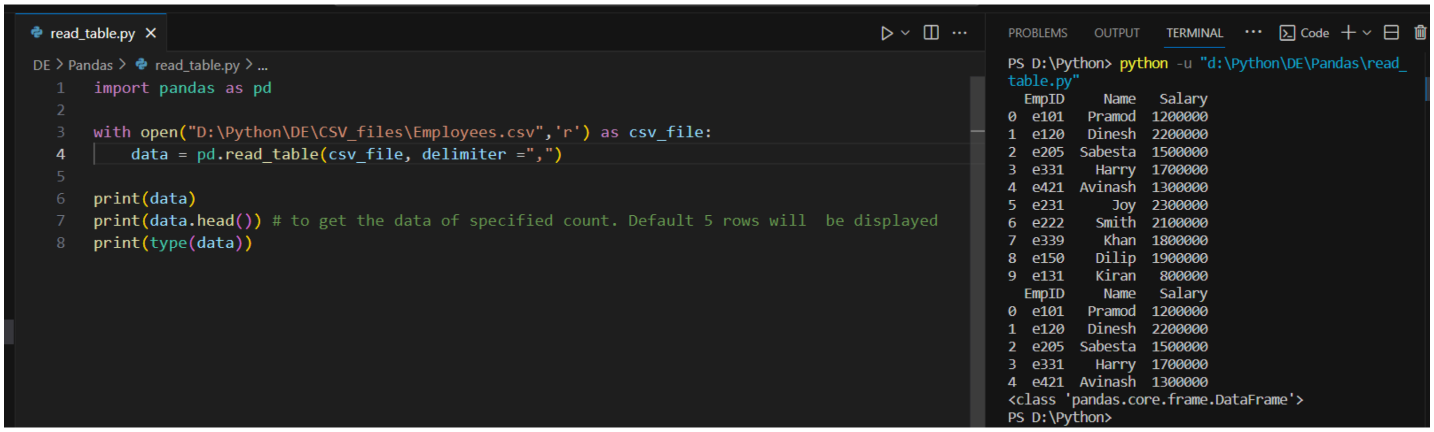

read_csv.py
DE > Pandas > read_csv.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```python
import pandas as pd

with open("D:\Python\DE\CSV_files\Employees.csv",'r') as csv_file:
    data = pd.read_csv(csv_file)

print(data)
print(type(data))
```

```
PS D:\Python> python -u "d:\Python\DE\Pandas\read_csv.py"
  EmpID     Name   Salary
0  e101   Pramod  1200000
1  e120   Dinesh  2200000
2  e205  Sabesta  1500000
3  e331    Harry  1700000
4  e421  Avinash  1300000
5  e231      Joy  2300000
6  e222    Smith  2100000
7  e339     Khan  1800000
8  e150    Dilip  1900000
9  e131    Kiran   800000
<class 'pandas.core.frame.DataFrame'>
PS D:\Python>
```

METHOD 2 : using read_table()

We can use read_table() with the attribute delimiter that defines how we are separating the data.

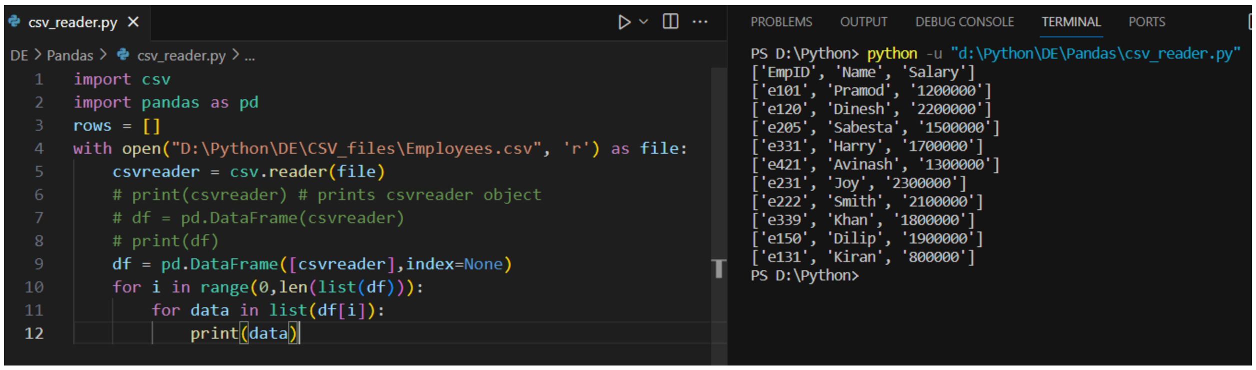


```python
import pandas as pd

with open("D:\Python\DE\CSV_files\Employees.csv",'r') as csv_file:
    data = pd.read_table(csv_file, delimiter =",")

print(data)
print(data.head()) # to get the data of specified count. Default 5 rows will  be displayed
print(type(data))
```

```
PS D:\Python> python -u "d:\Python\DE\Pandas\read_
table.py"
  EmpID     Name   Salary
0  e101   Pramod  1200000
1  e120   Dinesh  2200000
2  e205  Sabesta  1500000
3  e331    Harry  1700000
4  e421  Avinash  1300000
5  e231      Joy  2300000
6  e222    Smith  2100000
7  e339     Khan  1800000
8  e150    Dilip  1900000
9  e131    Kiran   800000
  EmpID     Name   Salary
0  e101   Pramod  1200000
1  e120   Dinesh  2200000
2  e205  Sabesta  1500000
3  e331    Harry  1700000
4  e421  Avinash  1300000
<class 'pandas.core.frame.DataFrame'>
PS D:\Python>
```

METHOD 3 : using csv.reader()

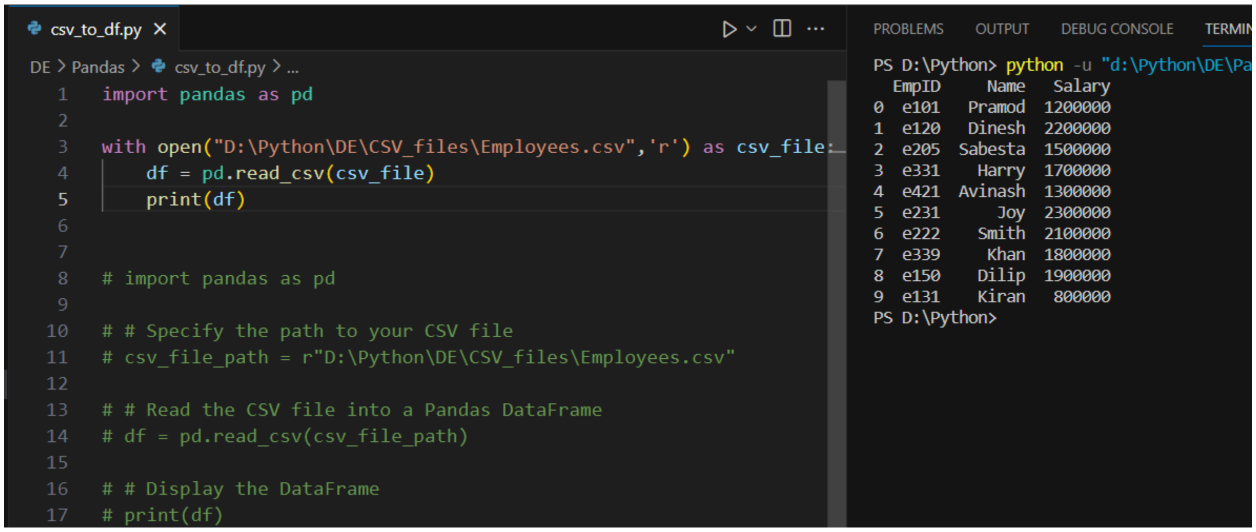
First creating csv.reader object using csv module then passing the csv.reader object as list in dataframe() then iterating through the dataframe.



```python
import csv
import pandas as pd
rows = []
with open("D:\Python\DE\CSV_files\Employees.csv", 'r') as file:
    csvreader = csv.reader(file)
    # print(csvreader) # prints csvreader object
    # df = pd.DataFrame(csvreader)
    # print(df)
    df = pd.DataFrame([csvreader],index=None)
    for i in range(0,len(list(df))):
        for data in list(df[i]):
            print(data)
```

```
PS D:\Python> python -u "d:\Python\DE\Pandas\csv_reader.py"
['EmpID', 'Name', 'Salary']
['e101', 'Pramod', '1200000']
['e120', 'Dinesh', '2200000']
['e205', 'Sabesta', '1500000']
['e331', 'Harry', '1700000']
['e421', 'Avinash', '1300000']
['e231', 'Joy', '2300000']
['e222', 'Smith', '2100000']
['e339', 'Khan', '1800000']
['e150', 'Dilip', '1900000']
['e131', 'Kiran', '800000']
PS D:\Python>
```

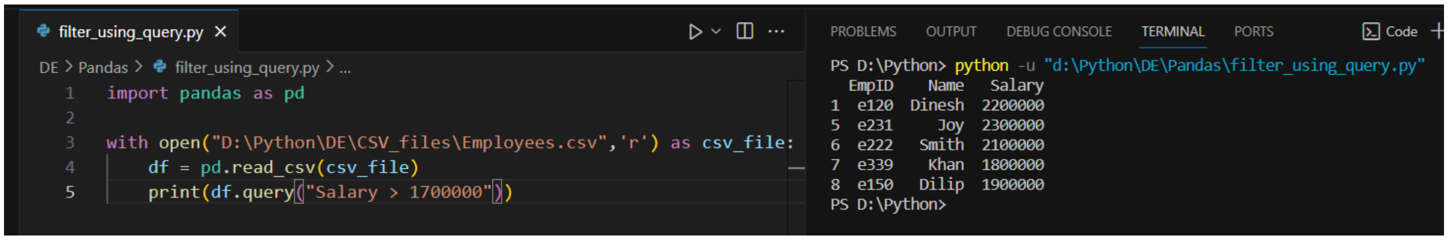
- **Read Data from CSV Files to Pandas Dataframes**
  Reading csv files using read_csv which stores the data in dataframe.



```python
import pandas as pd

with open("D:\Python\DE\CSV_files\Employees.csv",'r') as csv_file:
    df = pd.read_csv(csv_file)
    print(df)


# import pandas as pd

# # Specify the path to your CSV file
# csv_file_path = r"D:\Python\DE\CSV_files\Employees.csv"

# # Read the CSV file into a Pandas DataFrame
# df = pd.read_csv(csv_file_path)

# # Display the DataFrame
# print(df)
```

```
PS D:\Python> python -u "d:\Python\DE\Pa
  EmpID     Name   Salary
0  e101   Pramod  1200000
1  e120   Dinesh  2200000
2  e205  Sabesta  1500000
3  e331    Harry  1700000
4  e421  Avinash  1300000
5  e231      Joy  2300000
6  e222    Smith  2100000
7  e339     Khan  1800000
8  e150    Dilip  1900000
9  e131    Kiran   800000
PS D:\Python>
```

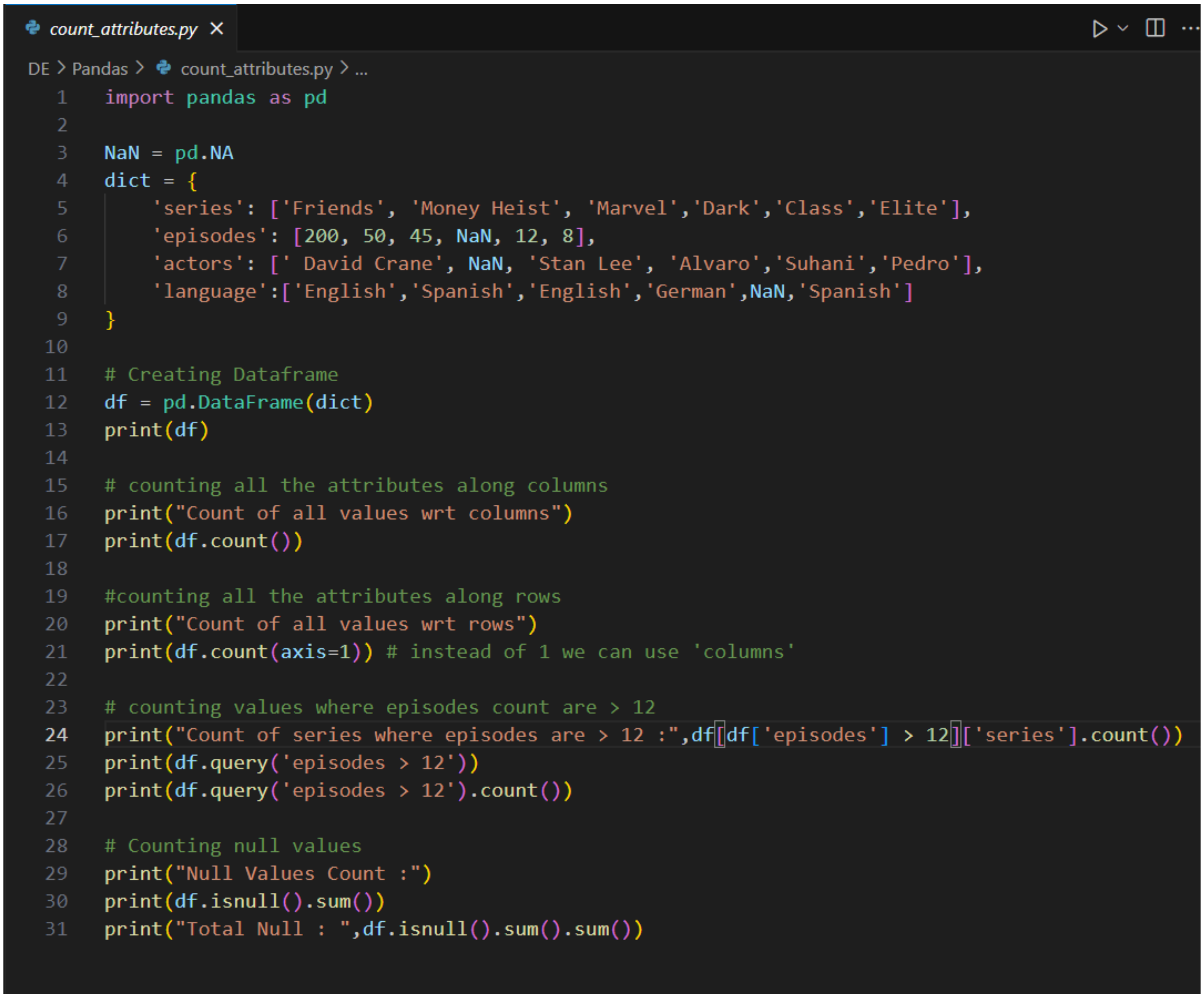
- **Filter Data in Pandas Dataframe using query**
  Using query() to filter the data in Dataframe where a condition which returns boolean expression is specified in the query().



```python
import pandas as pd

with open("D:\Python\DE\CSV_files\Employees.csv",'r') as csv_file:
    df = pd.read_csv(csv_file)
    print(df.query("Salary > 1700000"))
```

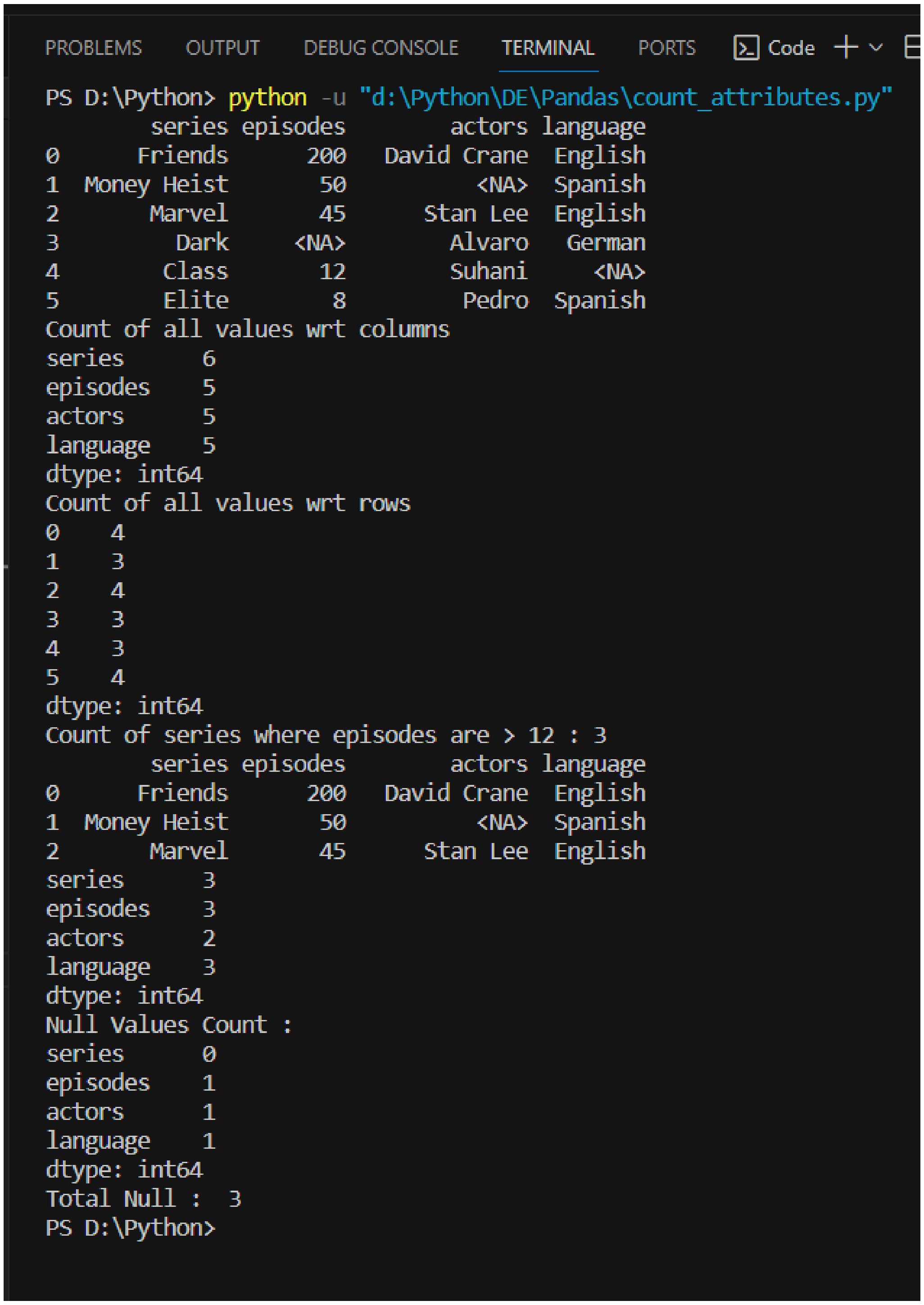
```
PS D:\Python> python -u "d:\Python\DE\Pandas\filter_using_query.py"
  EmpID    Name   Salary
1  e120  Dinesh  2200000
5  e231     Joy  2300000
6  e222   Smith  2100000
7  e339    Khan  1800000
8  e150   Dilip  1900000
PS D:\Python>
```

- **Get Count by Episodes,Null using Pandas Dataframe APIs**
  Counting of values along rows or columns can be done using DataFrame.count(*axis=0*, *level=None*, *numeric_only=False*).
  We can also count filtered data using query() to filter the data then counting.


count_attributes.py ×

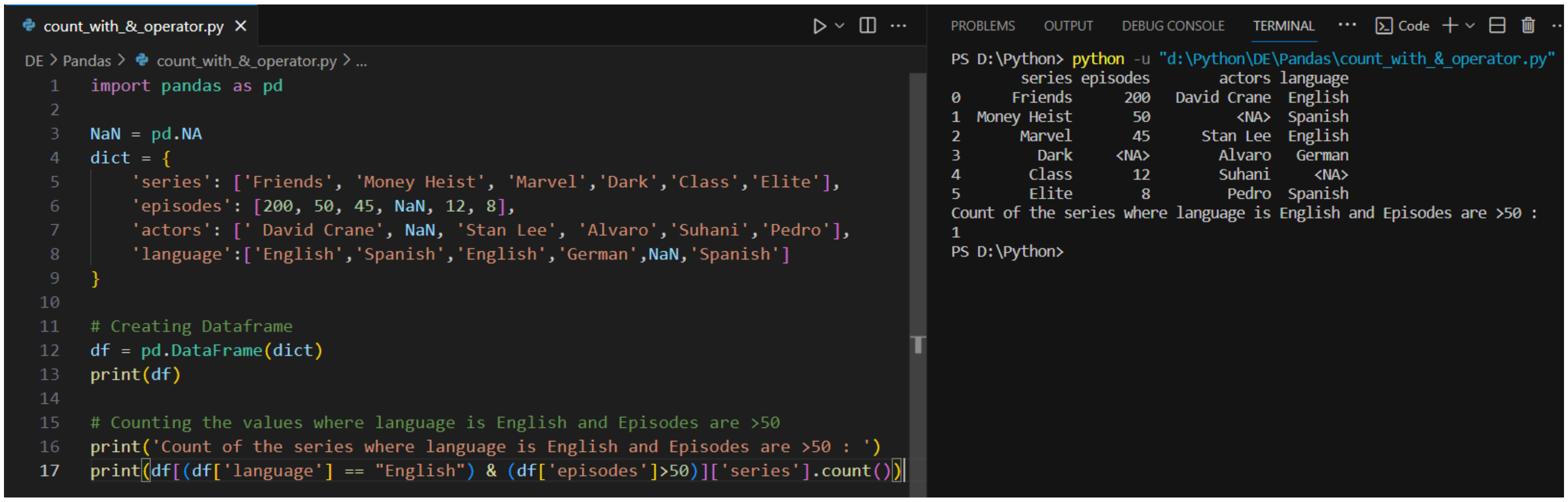
DE > Pandas > count_attributes.py > ...

```python
import pandas as pd

NaN = pd.NA
dict = {
    'series': ['Friends', 'Money Heist', 'Marvel','Dark','Class','Elite'],
    'episodes': [200, 50, 45, NaN, 12, 8],
    'actors': [' David Crane', NaN, 'Stan Lee', 'Alvaro','Suhani','Pedro'],
    'language':['English','Spanish','English','German',NaN,'Spanish']
}

# Creating Dataframe
df = pd.DataFrame(dict)
print(df)

# counting all the attributes along columns
print("Count of all values wrt columns")
print(df.count())

#counting all the attributes along rows
print("Count of all values wrt rows")
print(df.count(axis=1)) # instead of 1 we can use 'columns'

# counting values where episodes count are > 12
print("Count of series where episodes are > 12 :",df[df['episodes'] > 12]['series'].count())
print(df.query('episodes > 12'))
print(df.query('episodes > 12').count())

# Counting null values
print("Null Values Count :")
print(df.isnull().sum())
print("Total Null : ",df.isnull().sum().sum())
```



```
PS D:\Python> python -u "d:\Python\DE\Pandas\count_attributes.py"
        series episodes       actors language
0      Friends      200  David Crane  English
1  Money Heist       50         <NA>  Spanish
2       Marvel       45     Stan Lee  English
3         Dark     <NA>       Alvaro   German
4        Class       12       Suhani     <NA>
5        Elite        8        Pedro  Spanish
Count of all values wrt columns
series      6
episodes    5
actors      5
language    5
dtype: int64
Count of all values wrt rows
0    4
1    3
2    4
3    3
4    3
5    4
dtype: int64
Count of series where episodes are > 12 : 3
        series episodes       actors language
0      Friends      200  David Crane  English
1  Money Heist       50         <NA>  Spanish
2       Marvel       45     Stan Lee  English
series      3
episodes    3
actors      2
language    3
dtype: int64
Null Values Count :
series      0
episodes    1
actors      1
language    1
dtype: int64
Total Null :  3
PS D:\Python>
```

- **Get count by Episodes and Language using Pandas Dataframe APIs**



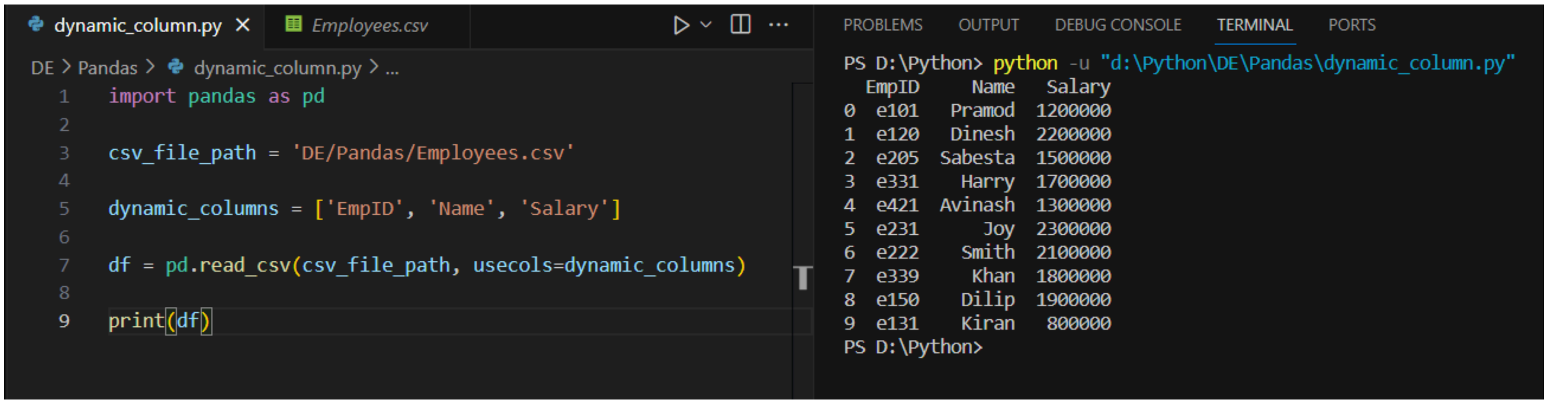
count_with_&_operator.py

DE > Pandas > count_with_&_operator.py > ...

```python
import pandas as pd

NaN = pd.NA
dict = {
    'series': ['Friends', 'Money Heist', 'Marvel','Dark','Class','Elite'],
    'episodes': [200, 50, 45, NaN, 12, 8],
    'actors': [' David Crane', NaN, 'Stan Lee', 'Alvaro','Suhani','Pedro'],
    'language':['English','Spanish','English','German',NaN,'Spanish']
}

# Creating Dataframe
df = pd.DataFrame(dict)
print(df)

# Counting the values where language is English and Episodes are >50
print('Count of the series where language is English and Episodes are >50 : ')
print(df[(df['language'] == "English") & (df['episodes']>50)]['series'].count())
```

```
PS D:\Python> python -u "d:\Python\DE\Pandas\count_with_&_operator.py"
        series episodes       actors language
0      Friends      200  David Crane  English
1  Money Heist       50         <NA>  Spanish
2       Marvel       45     Stan Lee  English
3         Dark     <NA>       Alvaro   German
4        Class       12       Suhani     <NA>
5        Elite        8        Pedro  Spanish
Count of the series where language is English and Episodes are >50 :
1
PS D:\Python>
```

- **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



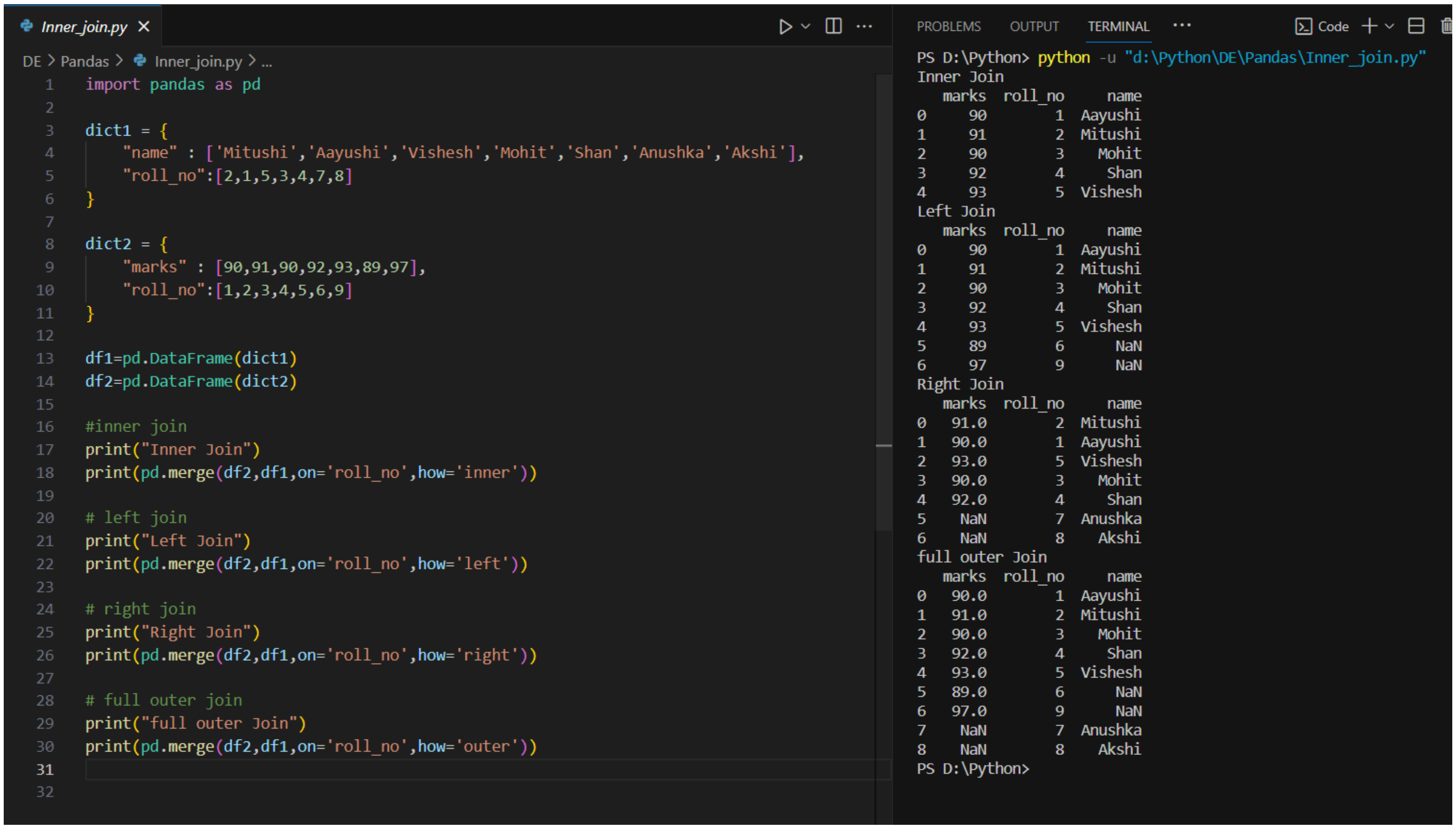
dynamic_column.py   Employees.csv

DE > Pandas > dynamic_column.py > ...

```python
import pandas as pd

csv_file_path = 'DE/Pandas/Employees.csv'

dynamic_columns = ['EmpID', 'Name', 'Salary']

df = pd.read_csv(csv_file_path, usecols=dynamic_columns)

print(df)
```

```
PS D:\Python> python -u "d:\Python\DE\Pandas\dynamic_column.py"
  EmpID     Name   Salary
0  e101   Pramod  1200000
1  e120   Dinesh  2200000
2  e205  Sabesta  1500000
3  e331    Harry  1700000
4  e421  Avinash  1300000
5  e231      Joy  2300000
6  e222    Smith  2100000
7  e339     Khan  1800000
8  e150    Dilip  1900000
9  e131    Kiran   800000
PS D:\Python>
```

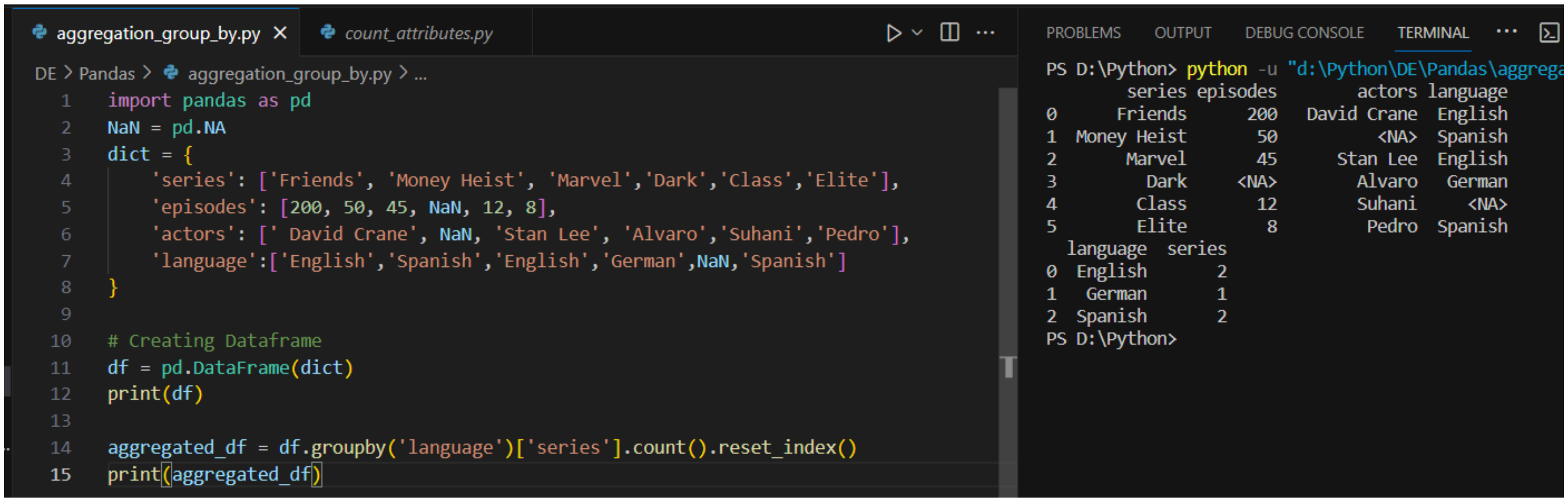
- **Performing Inner Join between Pandas Dataframes :** Pandas similar to SQL allows us to perform joins on pandas dataframes.



```python
import pandas as pd

dict1 = {
    "name" : ['Mitushi','Aayushi','Vishesh','Mohit','Shan','Anushka','Akshi'],
    "roll_no":[2,1,5,3,4,7,8]
}

dict2 = {
    "marks" : [90,91,90,92,93,89,97],
    "roll_no":[1,2,3,4,5,6,9]
}

df1=pd.DataFrame(dict1)
df2=pd.DataFrame(dict2)

#inner join
print("Inner Join")
print(pd.merge(df2,df1,on='roll_no',how='inner'))

# left join
print("Left Join")
print(pd.merge(df2,df1,on='roll_no',how='left'))

# right join
print("Right Join")
print(pd.merge(df2,df1,on='roll_no',how='right'))

# full outer join
print("full outer Join")
print(pd.merge(df2,df1,on='roll_no',how='outer'))
```

```
PS D:\Python> python -u "d:\Python\DE\Pandas\Inner_join.py"
Inner Join
   marks  roll_no     name
0     90        1  Aayushi
1     91        2  Mitushi
2     90        3    Mohit
3     92        4     Shan
4     93        5  Vishesh
Left Join
   marks  roll_no     name
0     90        1  Aayushi
1     91        2  Mitushi
2     90        3    Mohit
3     92        4     Shan
4     93        5  Vishesh
5     89        6      NaN
6     97        9      NaN
Right Join
   marks  roll_no     name
0   91.0        2  Mitushi
1   90.0        1  Aayushi
2   93.0        5  Vishesh
3   90.0        3    Mohit
4   92.0        4     Shan
5    NaN        7  Anushka
6    NaN        8    Akshi
full outer Join
   marks  roll_no     name
0   90.0        1  Aayushi
1   91.0        2  Mitushi
2   90.0        3    Mohit
3   92.0        4     Shan
4   93.0        5  Vishesh
5   89.0        6      NaN
6   97.0        9      NaN
7    NaN        7  Anushka
8    NaN        8    Akshi
PS D:\Python>
```

- **Perform Aggregations on Join results**



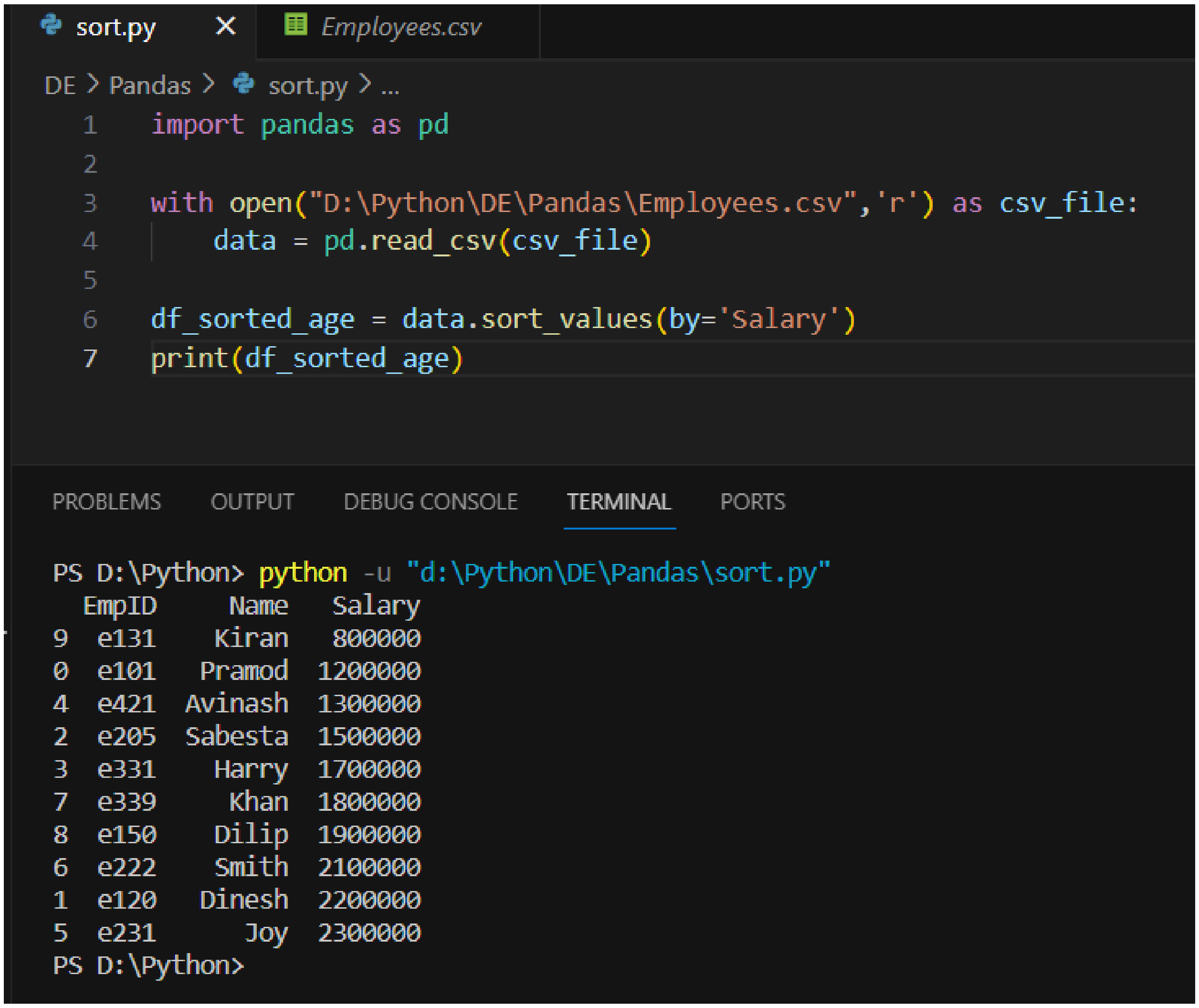
```python
import pandas as pd
NaN = pd.NA
dict = {
    'series': ['Friends', 'Money Heist', 'Marvel','Dark','Class','Elite'],
    'episodes': [200, 50, 45, NaN, 12, 8],
    'actors': [' David Crane', NaN, 'Stan Lee', 'Alvaro','Suhani','Pedro'],
    'language':['English','Spanish','English','German',NaN,'Spanish']
}

# Creating Dataframe
df = pd.DataFrame(dict)
print(df)

aggregated_df = df.groupby('language')['series'].count().reset_index()
print(aggregated_df)
```

```
PS D:\Python> python -u "d:\Python\DE\Pandas\aggrega
        series episodes       actors language
0      Friends      200  David Crane  English
1  Money Heist       50         <NA>  Spanish
2       Marvel       45     Stan Lee  English
3         Dark     <NA>       Alvaro   German
4        Class       12       Suhani     <NA>
5        Elite        8        Pedro  Spanish
  language  series
0  English       2
1   German       1
2  Spanish       2
PS D:\Python>
```

- Grouping data on the basis of language and using count() to get the grouped data count.

- **Sort Data in Pandas Dataframes**

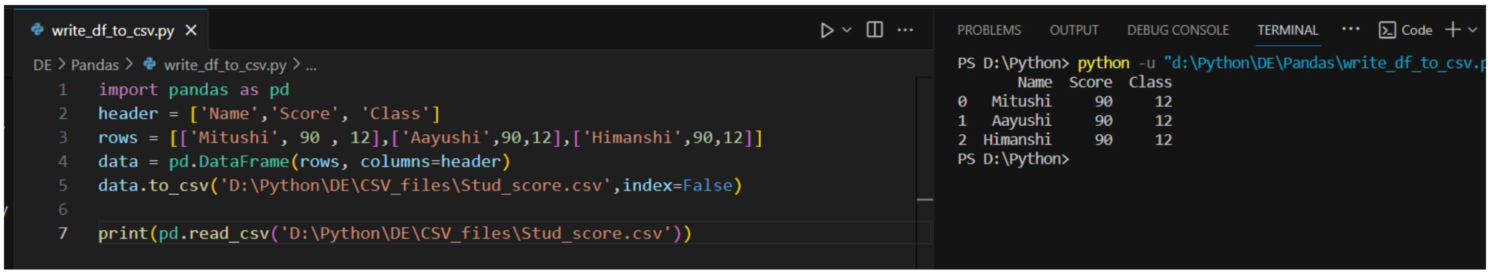
Sorting data using sort() method and giving the column name to the ‘by’ attribute.



```python
import pandas as pd

with open("D:\Python\DE\Pandas\Employees.csv",'r') as csv_file:
    data = pd.read_csv(csv_file)

df_sorted_age = data.sort_values(by='Salary')
print(df_sorted_age)
```

```
PS D:\Python> python -u "d:\Python\DE\Pandas\sort.py"
  EmpID     Name   Salary
9  e131    Kiran   800000
0  e101   Pramod  1200000
4  e421  Avinash  1300000
2  e205  Sabesta  1500000
3  e331    Harry  1700000
7  e339     Khan  1800000
8  e150    Dilip  1900000
6  e222    Smith  2100000
1  e120   Dinesh  2200000
5  e231      Joy  2300000
PS D:\Python>
```

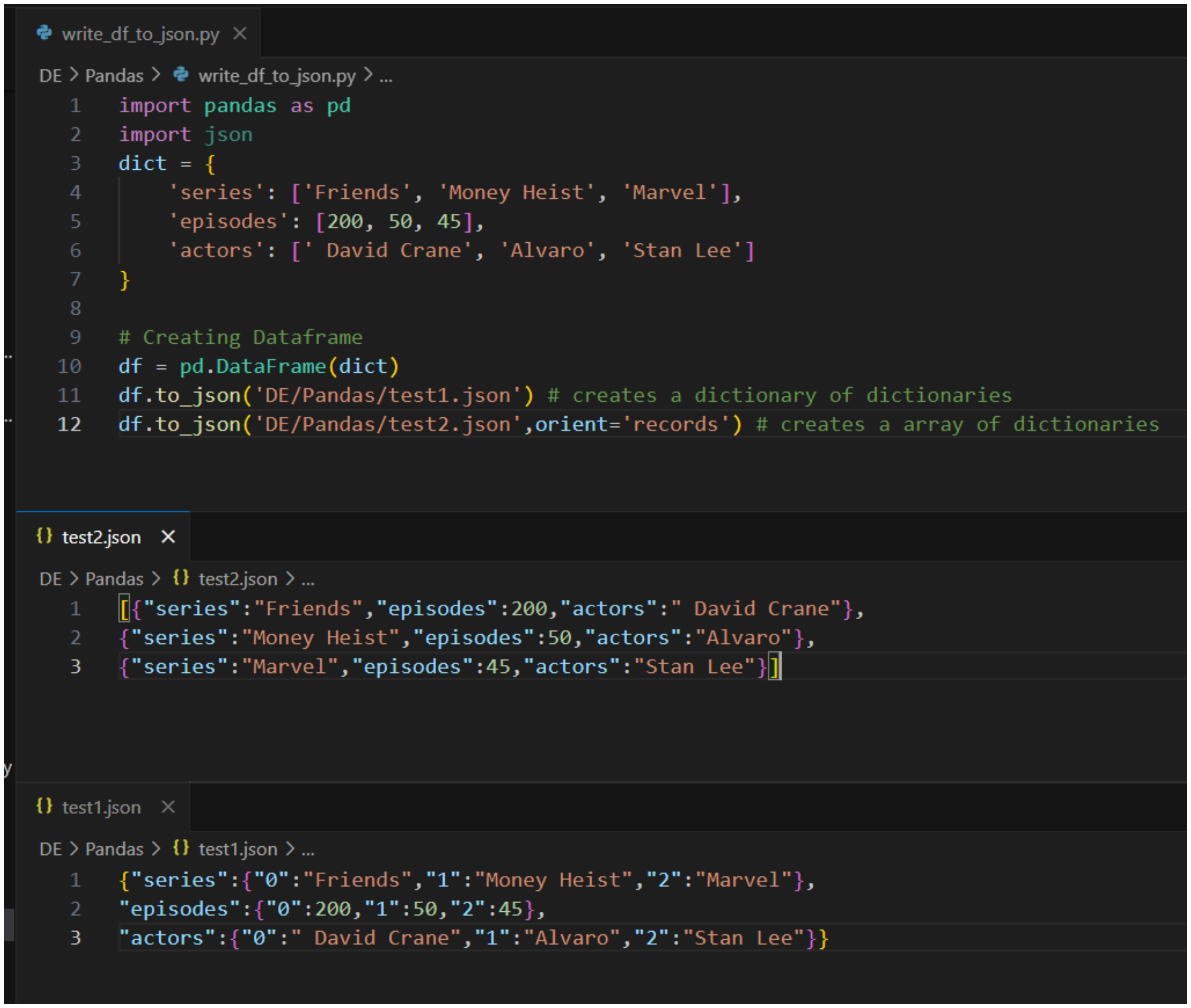
- **Writing Pandas Dataframes to Files**
  We can write pandas dataframe to different types of file such as excel,sql and csv file.



```python
import pandas as pd
header = ['Name','Score', 'Class']
rows = [['Mitushi', 90 , 12],['Aayushi',90,12],['Himanshi',90,12]]
data = pd.DataFrame(rows, columns=header)
data.to_csv('D:\Python\DE\CSV_files\Stud_score.csv',index=False)

print(pd.read_csv('D:\Python\DE\CSV_files\Stud_score.csv'))
```

```
PS D:\Python> python -u "d:\Python\DE\Pandas\write_df_to_csv.p
       Name  Score  Class
0   Mitushi     90     12
1   Aayushi     90     12
2  Himanshi     90     12
PS D:\Python>
```

- **Write Pandas Dataframes to JSON Files**
  The to_json() method can be used to write pandas dataframe into json files. The attribute orient='records' will store data in the form of arrays of dictionaries.



write_df_to_json.py

DE > Pandas > write_df_to_json.py > ...

```python
import pandas as pd
import json
dict = {
    'series': ['Friends', 'Money Heist', 'Marvel'],
    'episodes': [200, 50, 45],
    'actors': [' David Crane', 'Alvaro', 'Stan Lee']
}

# Creating Dataframe
df = pd.DataFrame(dict)
df.to_json('DE/Pandas/test1.json') # creates a dictionary of dictionaries
df.to_json('DE/Pandas/test2.json',orient='records') # creates a array of dictionaries
```

test2.json

DE > Pandas > test2.json > ...

```json
[{"series":"Friends","episodes":200,"actors":" David Crane"},
{"series":"Money Heist","episodes":50,"actors":"Alvaro"},
{"series":"Marvel","episodes":45,"actors":"Stan Lee"}]
```

test1.json

DE > Pandas > test1.json > ...

```json
{"series":{"0":"Friends","1":"Money Heist","2":"Marvel"},
"episodes":{"0":200,"1":50,"2":45},
"actors":{"0":" David Crane","1":"Alvaro","2":"Stan Lee"}}
```