Data Engineering Batch 1 **Day 8: 30/01/2024**

Name: Pradip Bochare

File IO using Python

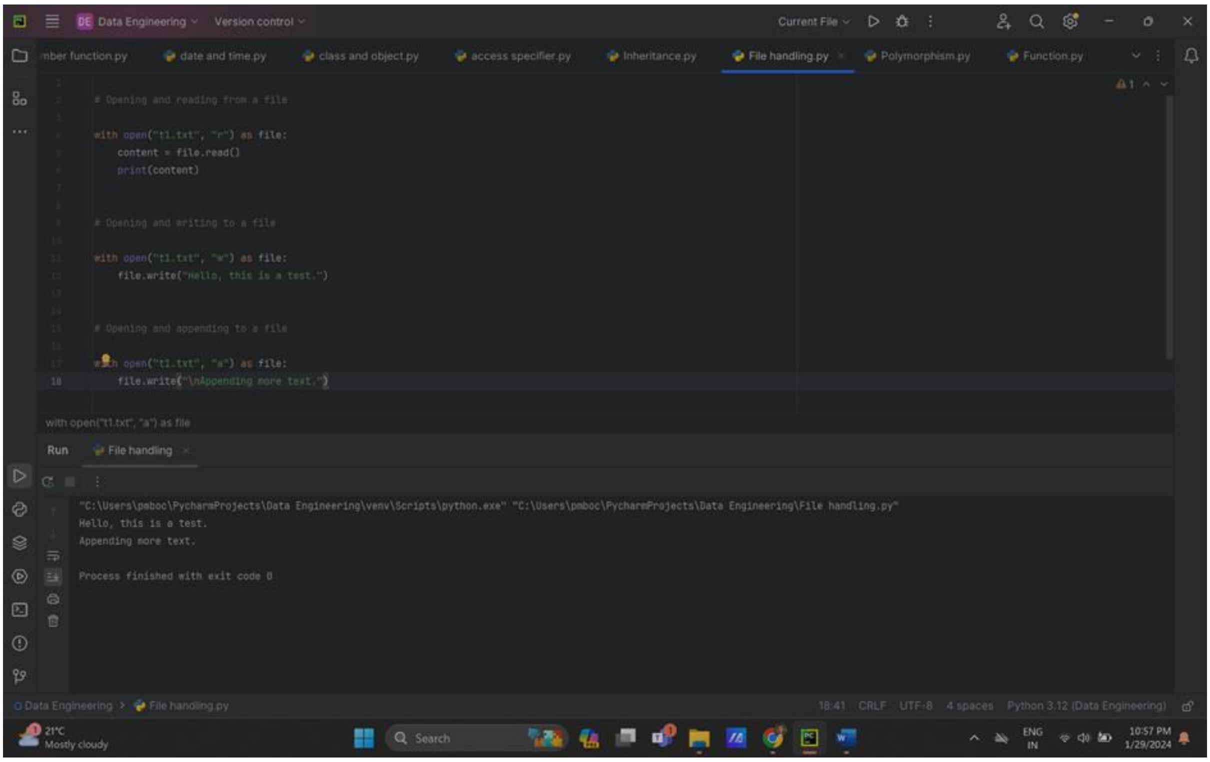
- File Input/Output (I/O) in Python is a fundamental aspect of working with files.
- File Modes:

  'r': Read (default mode).

  'w': Write (creates a new file or truncates an existing file).

  'a': Append (opens the file for writing, but appends to the end).

  'b': Binary mode (e.g., 'rb' or 'wb' for reading/writing binary data).



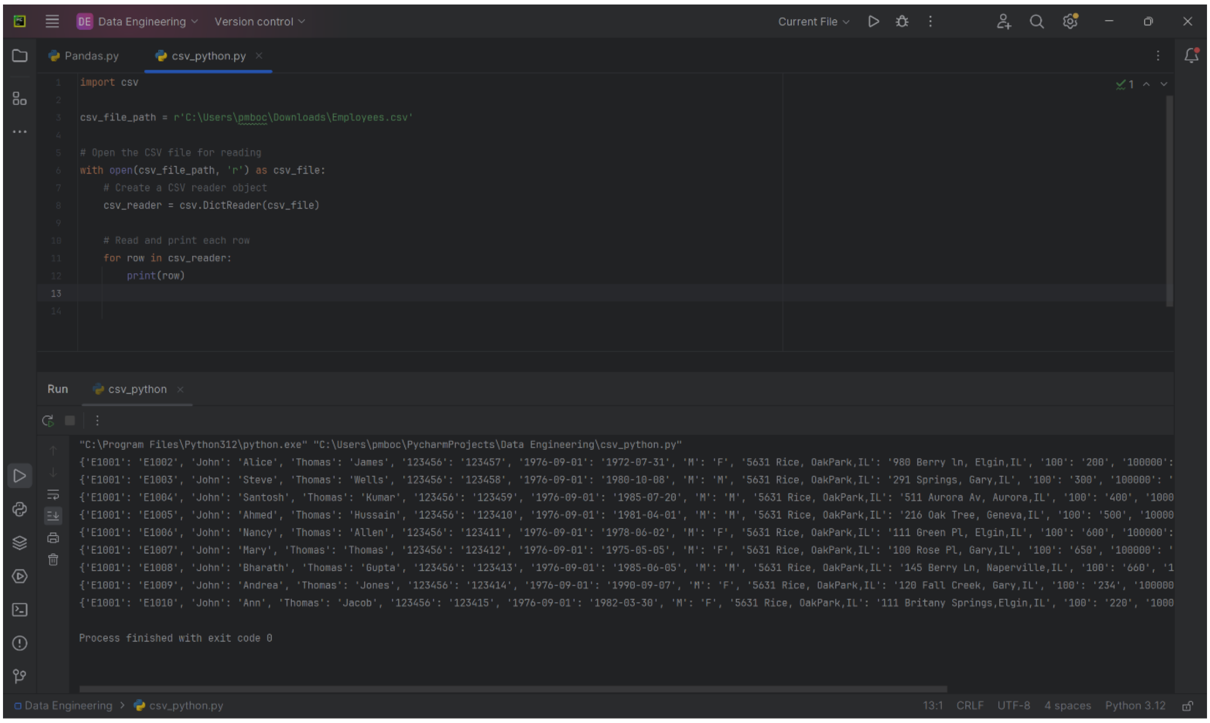
```python

# Opening and reading from a file

with open("t1.txt", "r") as file:
    content = file.read()
    print(content)


# Opening and writing to a file

with open("t1.txt", "w") as file:
    file.write("Hello, this is a test.")


# Opening and appending to a file

with open("t1.txt", "a") as file:
    file.write("\nAppending more text.")
```

with open("t1.txt", "a") as file

```
"C:\Users\pmboc\PycharmProjects\Data Engineering\venv\Scripts\python.exe" "C:\Users\pmboc\PycharmProjects\Data Engineering\File handling.py"
Hello, this is a test.
Appending more text.

Process finished with exit code 0
```

Read Data from CSV File:

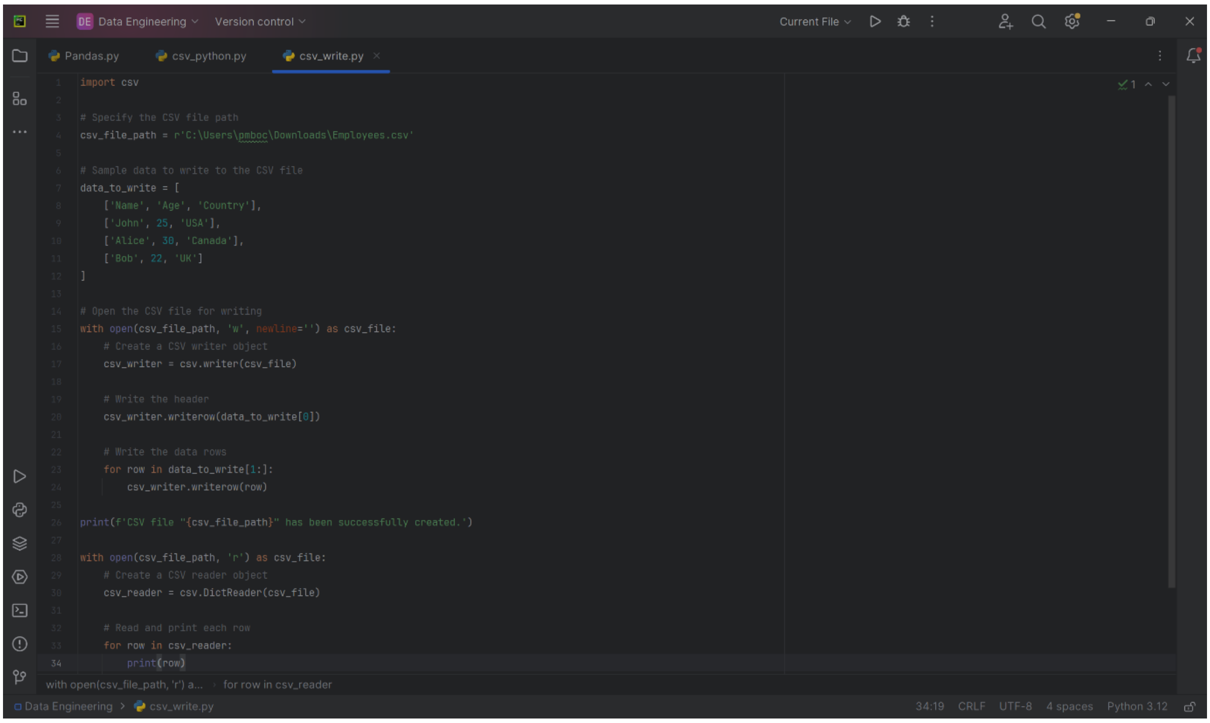
Dict is a hash table of keys and values structured in Python. The dict() method is used to create a dictionary object from either a specified set or iterables of keys and values. The csv module. DictReader is used to read CSV files.



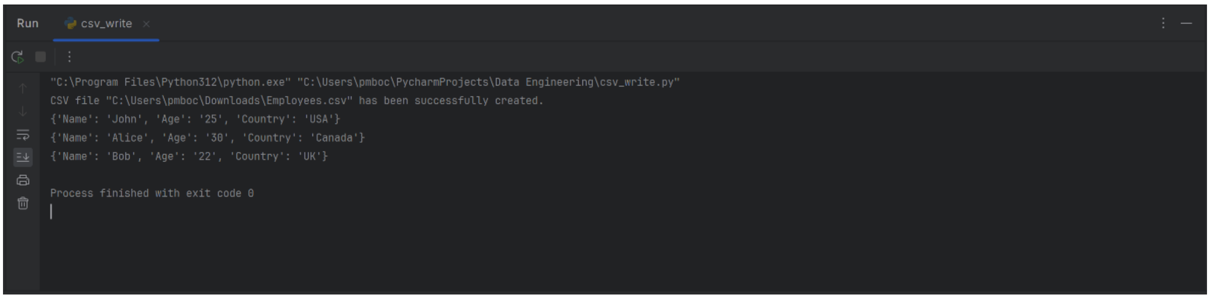
```python
import csv

csv_file_path = r'C:\Users\pmboc\Downloads\Employees.csv'

# Open the CSV file for reading
with open(csv_file_path, 'r') as csv_file:
    # Create a CSV reader object
    csv_reader = csv.DictReader(csv_file)

    # Read and print each row
    for row in csv_reader:
        print(row)
```

```
"C:\Program Files\Python312\python.exe" "C:\Users\pmboc\PycharmProjects\Data Engineering\csv_python.py"
{'E1001': 'E1002', 'John': 'Alice', 'Thomas': 'James', '123456': '123457', '1976-09-01': '1972-07-31', 'M': 'F', '5631 Rice, OakPark,IL': '980 Berry ln, Elgin,IL', '100': '200', '100000':
{'E1001': 'E1003', 'John': 'Steve', 'Thomas': 'Wells', '123456': '123458', '1976-09-01': '1980-10-08', 'M': 'M', '5631 Rice, OakPark,IL': '291 Springs, Gary,IL', '100': '300', '100000': '
{'E1001': 'E1004', 'John': 'Santosh', 'Thomas': 'Kumar', '123456': '123459', '1976-09-01': '1985-07-20', 'M': 'M', '5631 Rice, OakPark,IL': '511 Aurora Av, Aurora,IL', '100': '400', '1000
{'E1001': 'E1005', 'John': 'Ahmed', 'Thomas': 'Hussain', '123456': '123410', '1976-09-01': '1981-04-01', 'M': 'M', '5631 Rice, OakPark,IL': '216 Oak Tree, Geneva,IL', '100': '500', '10000
{'E1001': 'E1006', 'John': 'Nancy', 'Thomas': 'Allen', '123456': '123411', '1976-09-01': '1978-06-02', 'M': 'F', '5631 Rice, OakPark,IL': '111 Green Pl, Elgin,IL', '100': '600', '100000':
{'E1001': 'E1007', 'John': 'Mary', 'Thomas': 'Thomas', '123456': '123412', '1976-09-01': '1975-05-05', 'M': 'F', '5631 Rice, OakPark,IL': '100 Rose Pl, Gary,IL', '100': '650', '100000': '
{'E1001': 'E1008', 'John': 'Bharath', 'Thomas': 'Gupta', '123456': '123413', '1976-09-01': '1985-06-05', 'M': 'M', '5631 Rice, OakPark,IL': '145 Berry Ln, Naperville,IL', '100': '660', '1
{'E1001': 'E1009', 'John': 'Andrea', 'Thomas': 'Jones', '123456': '123414', '1976-09-01': '1990-09-07', 'M': 'F', '5631 Rice, OakPark,IL': '120 Fall Creek, Gary,IL', '100': '234', '100000
{'E1001': 'E1010', 'John': 'Ann', 'Thomas': 'Jacob', '123456': '123415', '1976-09-01': '1982-03-30', 'M': 'F', '5631 Rice, OakPark,IL': '111 Britany Springs,Elgin,IL', '100': '220', '1000

Process finished with exit code 0
```

Write CSV file Using csv.writer

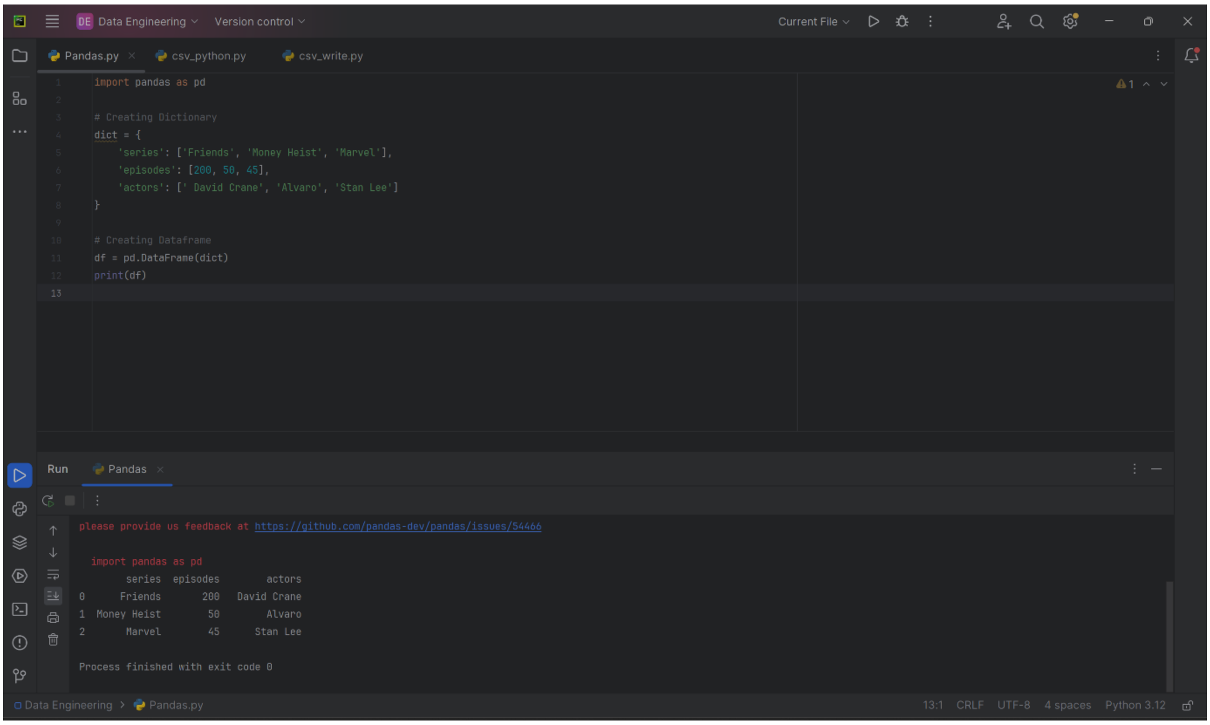


```python
import csv

# Specify the CSV file path
csv_file_path = r'C:\Users\pmboc\Downloads\Employees.csv'

# Sample data to write to the CSV file
data_to_write = [
    ['Name', 'Age', 'Country'],
    ['John', 25, 'USA'],
    ['Alice', 30, 'Canada'],
    ['Bob', 22, 'UK']
]

# Open the CSV file for writing
with open(csv_file_path, 'w', newline='') as csv_file:
    # Create a CSV writer object
    csv_writer = csv.writer(csv_file)

    # Write the header
    csv_writer.writerow(data_to_write[0])

    # Write the data rows
    for row in data_to_write[1:]:
        csv_writer.writerow(row)

print(f'CSV file "{csv_file_path}" has been successfully created.')

with open(csv_file_path, 'r') as csv_file:
    # Create a CSV reader object
    csv_reader = csv.DictReader(csv_file)

    # Read and print each row
    for row in csv_reader:
        print(row)
```



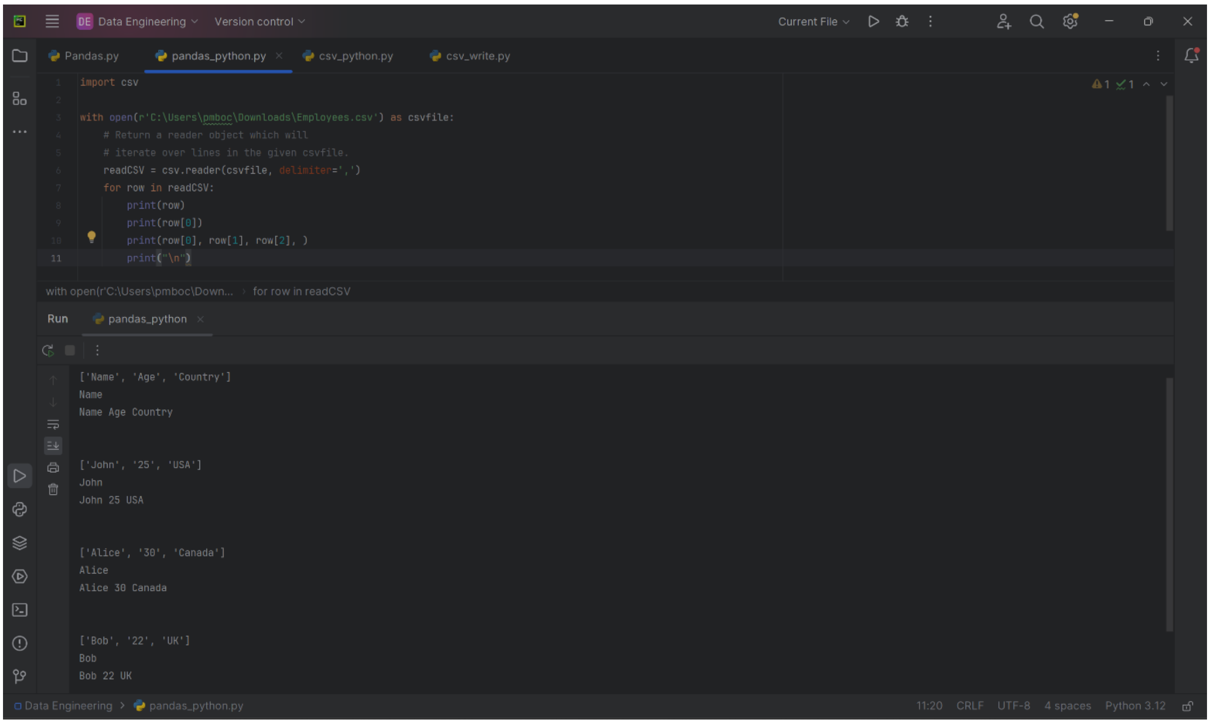
```
"C:\Program Files\Python312\python.exe" "C:\Users\pmboc\PycharmProjects\Data Engineering\csv_write.py"
CSV file "C:\Users\pmboc\Downloads\Employees.csv" has been successfully created.
{'Name': 'John', 'Age': '25', 'Country': 'USA'}
{'Name': 'Alice', 'Age': '30', 'Country': 'Canada'}
{'Name': 'Bob', 'Age': '22', 'Country': 'UK'}

Process finished with exit code 0
```

Pandas



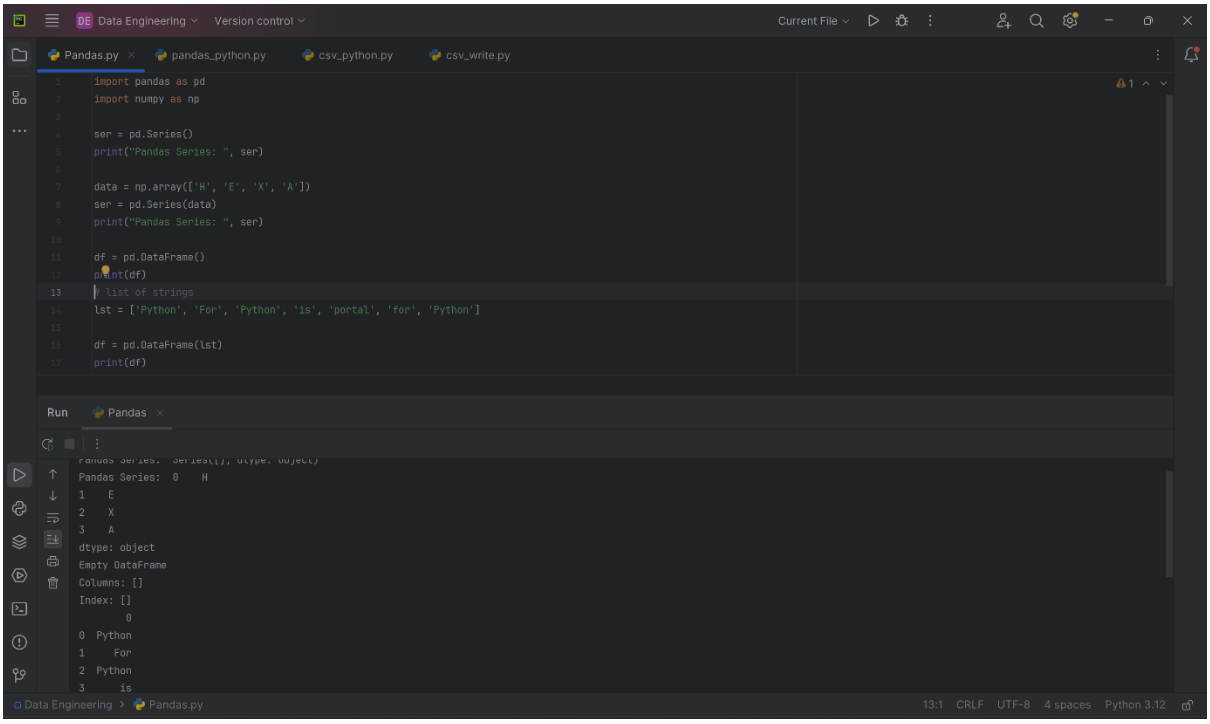
```python
import pandas as pd

# Creating Dictionary
dict = {
    'series': ['Friends', 'Money Heist', 'Marvel'],
    'episodes': [200, 50, 45],
    'actors': [' David Crane', 'Alvaro', 'Stan Lee']
}

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

```
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
        series  episodes       actors
0      Friends       200   David Crane
1  Money Heist        50       Alvaro
2       Marvel        45     Stan Lee

Process finished with exit code 0
```



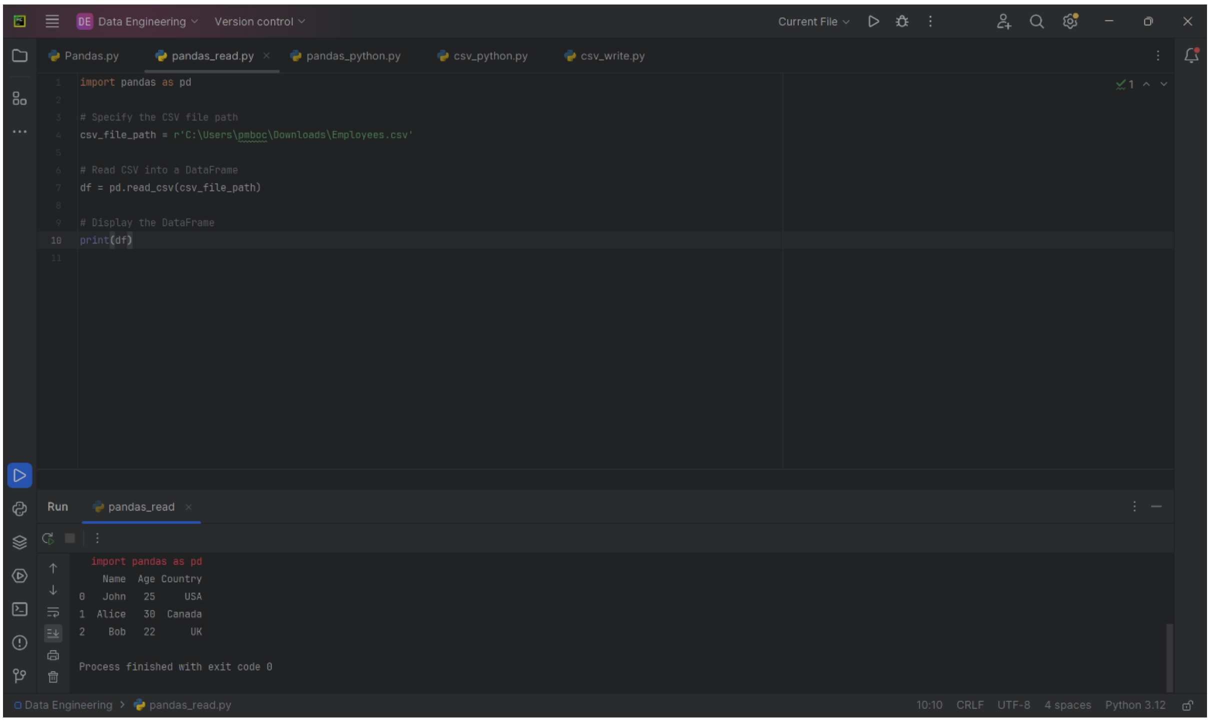
```python
import csv

with open(r'C:\Users\pmboc\Downloads\Employees.csv') as csvfile:
    # Return a reader object which will
    # iterate over lines in the given csvfile.
    readCSV = csv.reader(csvfile, delimiter=',')
    for row in readCSV:
        print(row)
        print(row[0])
        print(row[0], row[1], row[2], )
        print("\n")
```

```
['Name', 'Age', 'Country']
Name
Name Age Country


['John', '25', 'USA']
John
John 25 USA


['Alice', '30', 'Canada']
Alice
Alice 30 Canada


['Bob', '22', 'UK']
Bob
Bob 22 UK
```



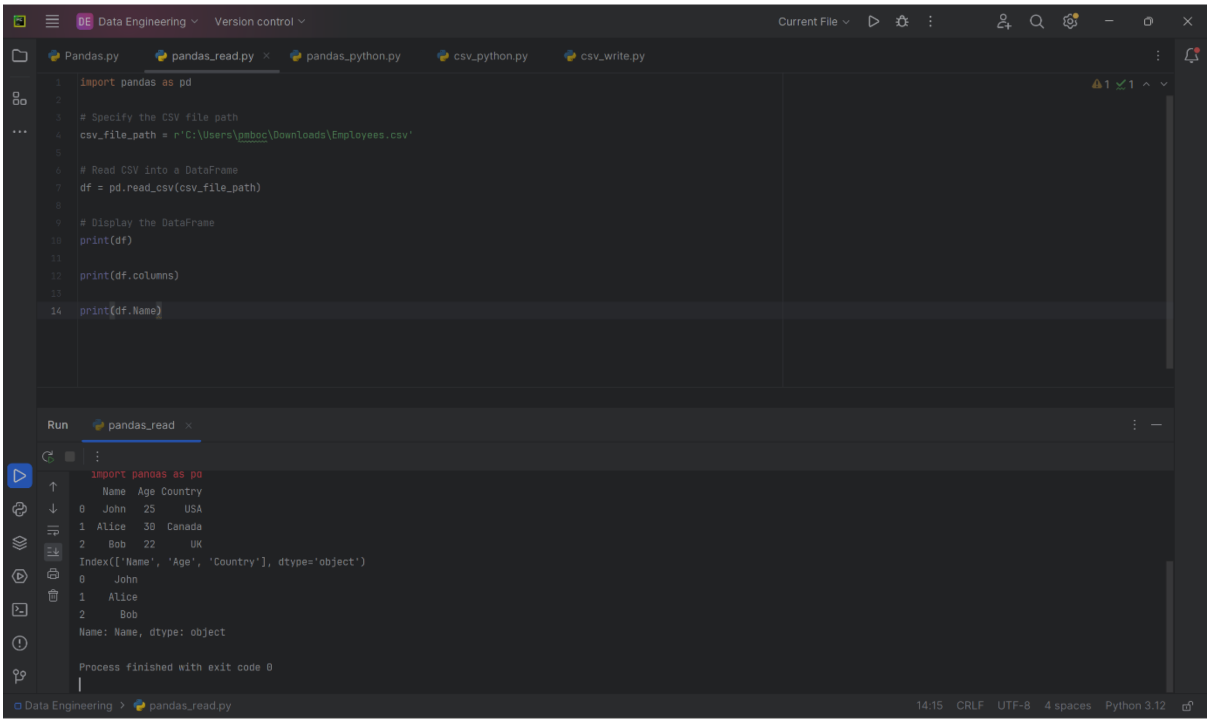
```python
import pandas as pd
import numpy as np

ser = pd.Series()
print("Pandas Series: ", ser)

data = np.array(['H', 'E', 'X', 'A'])
ser = pd.Series(data)
print("Pandas Series: ", ser)

df = pd.DataFrame()
print(df)
# list of strings
lst = ['Python', 'For', 'Python', 'is', 'portal', 'for', 'Python']

df = pd.DataFrame(lst)
print(df)
```

```
Pandas Series:  Series([], dtype: object)
Pandas Series:  0    H
1    E
2    X
3    A
dtype: object
Empty DataFrame
Columns: []
Index: []
        0
0  Python
1     For
2  Python
3      is
```



```python
import pandas as pd

# Specify the CSV file path
csv_file_path = r'C:\Users\pmboc\Downloads\Employees.csv'

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

# Display the DataFrame
print(df)
```

```
 import pandas as pd
    Name  Age Country
0   John   25     USA
1  Alice   30  Canada
2    Bob   22      UK

Process finished with exit code 0
```



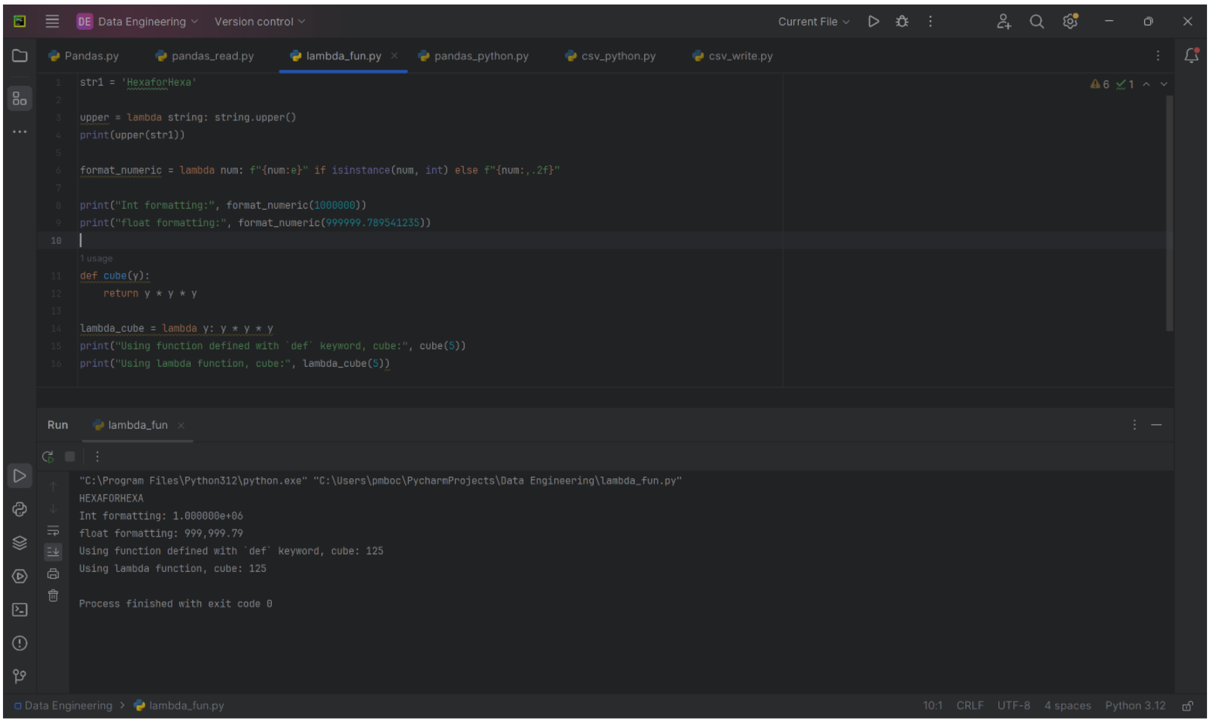
```python
import pandas as pd

# Specify the CSV file path
csv_file_path = r'C:\Users\pmboc\Downloads\Employees.csv'

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

# Display the DataFrame
print(df)

print(df.columns)

print(df.Name)
```

```
 import pandas as pd
    Name  Age Country
0   John   25     USA
1  Alice   30  Canada
2    Bob   22      UK
Index(['Name', 'Age', 'Country'], dtype='object')
0     John
1    Alice
2      Bob
Name: Name, dtype: object

Process finished with exit code 0
```

Python Lambda Function

Syntax: lambda arguments : expression

- This function can have any number of arguments but only one expression, which is evaluated and returned.
- One is free to use lambda functions wherever function objects are required.
- You need to keep in your knowledge that lambda functions are syntactically restricted to a single expression.
- It has various uses in particular fields of programming, besides other types of expressions in functions.



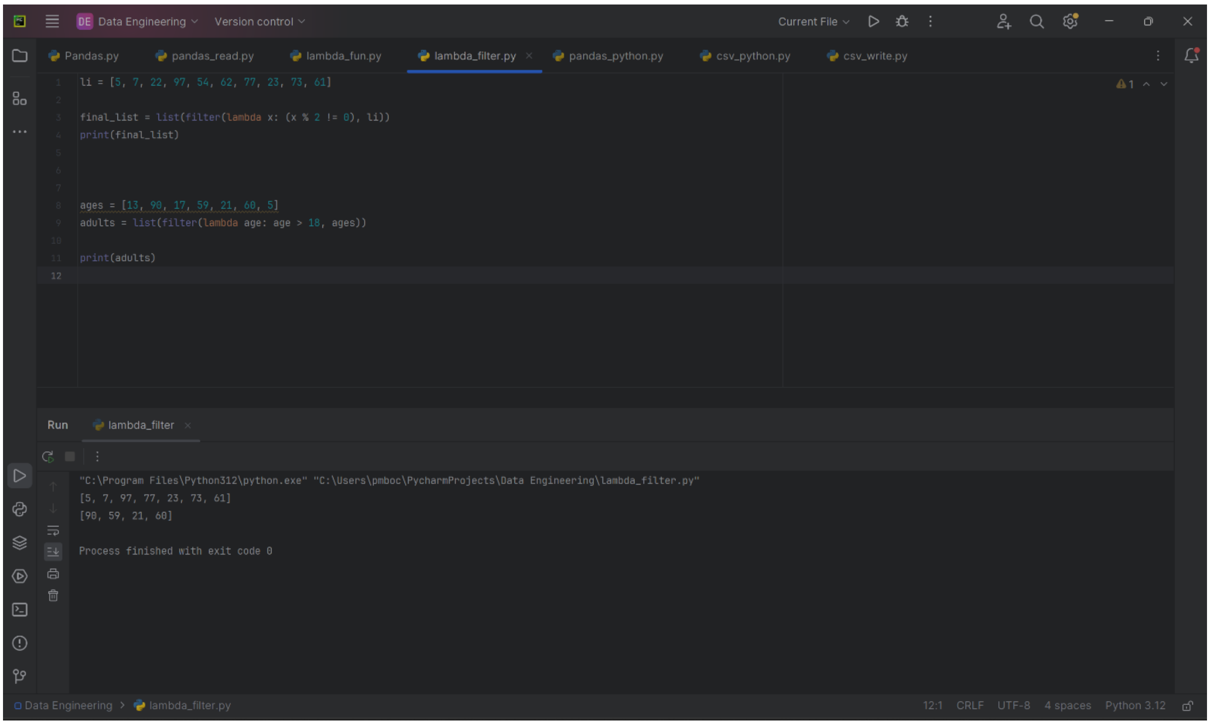
```python
str1 = 'HexaforHexa'

upper = lambda string: string.upper()
print(upper(str1))

format_numeric = lambda num: f"{num:e}" if isinstance(num, int) else f"{num:,.2f}"

print("Int formatting:", format_numeric(1000000))
print("float formatting:", format_numeric(999999.789541235))

1 usage
def cube(y):
    return y * y * y

lambda_cube = lambda y: y * y * y
print("Using function defined with `def` keyword, cube:", cube(5))
print("Using lambda function, cube:", lambda_cube(5))
```

```
"C:\Program Files\Python312\python.exe" "C:\Users\pmboc\PycharmProjects\Data Engineering\lambda_fun.py"
HEXAFORHEXA
Int formatting: 1.000000e+06
float formatting: 999,999.79
Using function defined with `def` keyword, cube: 125
Using lambda function, cube: 125

Process finished with exit code 0
```

With lambda function	**Without lambda function**
Supports single-line sometimes statements that return some value.	Supports any number of lines inside a function block
Good for performing short operations/data manipulations.	Good for any cases that require multiple lines of code.
Using the lambda function can sometime reduce the readability of code.	We can use comments and function descriptions for easy readability.

Using lambda () Function with filter ()

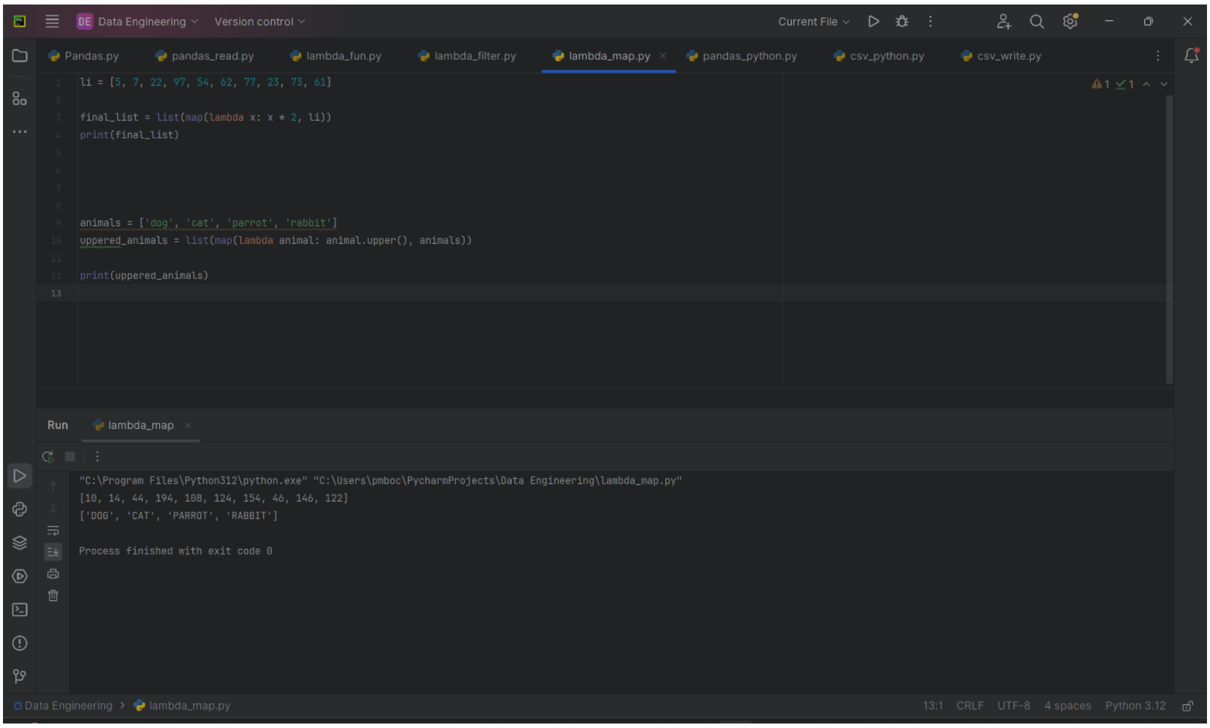
The filter () function in Python takes in a function and a list as arguments. This offers an elegant way to filter out all the elements of a sequence “sequence”, for which the function returns True.



```python
li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]

final_list = list(filter(lambda x: (x % 2 != 0), li))
print(final_list)



ages = [13, 90, 17, 59, 21, 60, 5]
adults = list(filter(lambda age: age > 18, ages))

print(adults)
```

```
"C:\Program Files\Python312\python.exe" "C:\Users\pmboc\PycharmProjects\Data Engineering\lambda_filter.py"
[5, 7, 97, 77, 23, 73, 61]
[90, 59, 21, 60]

Process finished with exit code 0
```

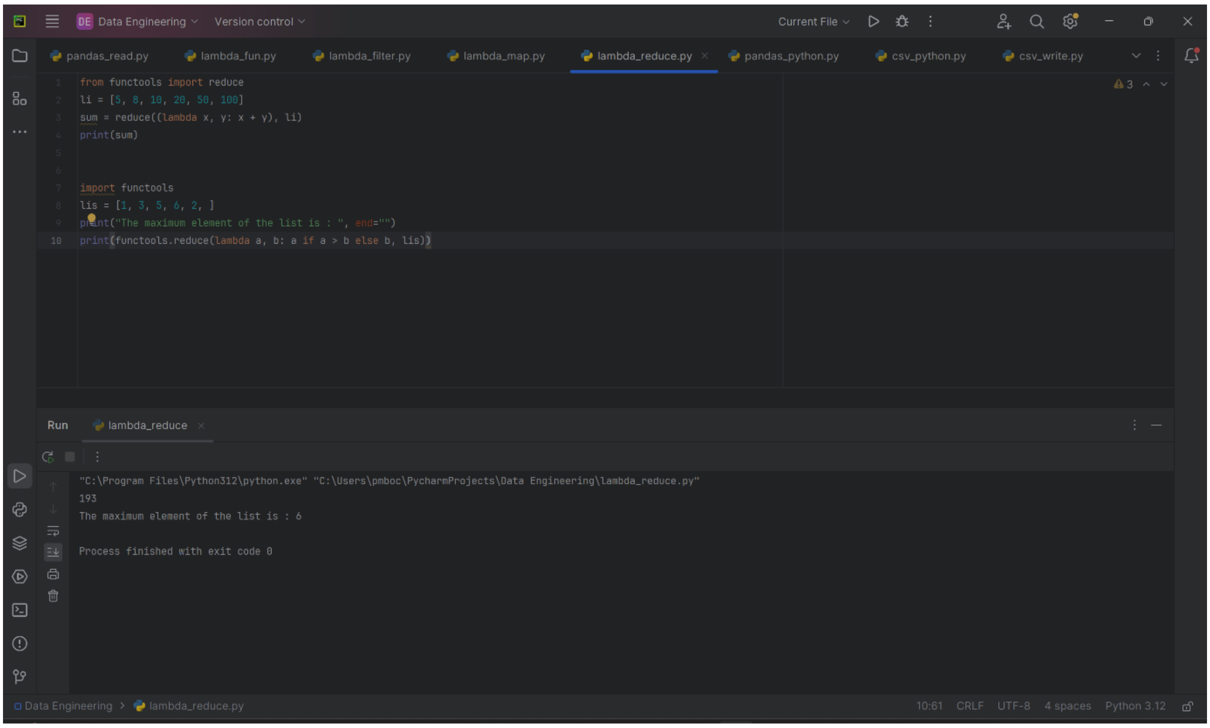
Using lambda() Function with map()



```python
li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]

final_list = list(map(lambda x: x * 2, li))
print(final_list)




animals = ['dog', 'cat', 'parrot', 'rabbit']
uppered_animals = list(map(lambda animal: animal.upper(), animals))

print(uppered_animals)
```

```
"C:\Program Files\Python312\python.exe" "C:\Users\pmboc\PycharmProjects\Data Engineering\lambda_map.py"
[10, 14, 44, 194, 108, 124, 154, 46, 146, 122]
['DOG', 'CAT', 'PARROT', 'RABBIT']

Process finished with exit code 0
```

Using lambda () Function with reduce ()



```python
from functools import reduce
li = [5, 8, 10, 20, 50, 100]
sum = reduce((lambda x, y: x + y), li)
print(sum)


import functools
lis = [1, 3, 5, 6, 2, ]
print("The maximum element of the list is : ", end="")
print(functools.reduce(lambda a, b: a if a > b else b, lis))
```

```
"C:\Program Files\Python312\python.exe" "C:\Users\pmboc\PycharmProjects\Data Engineering\lambda_reduce.py"
193
The maximum element of the list is : 6

Process finished with exit code 0
```