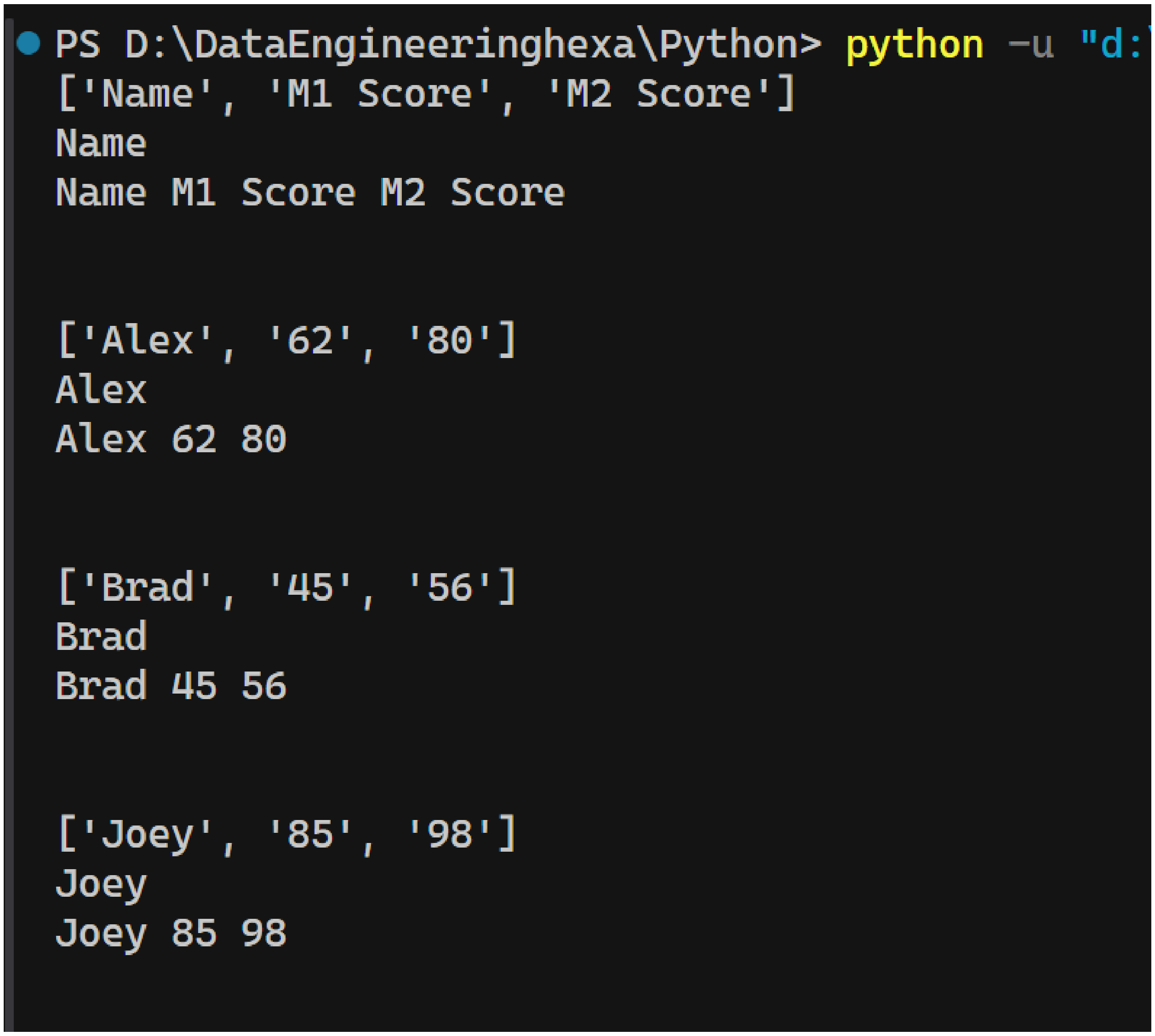
Asmita Porwal
Batch-1
Day-8
30/1/2024
Data engineering

Assignment-8

File IO using Python

```python
#creating csv file
import pandas as pd
header = ['Name', 'M1 Score', 'M2 Score']
data = [['Alex', 62, 80], ['Brad', 45, 56], ['Joey', 85, 98]]
data = pd.DataFrame(data, columns=header)
data.to_csv('Stu_data.csv', index=False)
#reading a CSV file and iterating over lines in the given CSV.
import csv
with open('Stu_data.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")
```

output

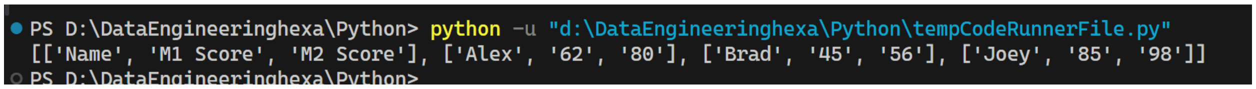


```
PS D:\DataEngineeringhexa\Python> python -u "d:
['Name', 'M1 Score', 'M2 Score']
Name
Name M1 Score M2 Score


['Alex', '62', '80']
Alex
Alex 62 80


['Brad', '45', '56']
Brad
Brad 45 56


['Joey', '85', '98']
Joey
Joey 85 98
```

Read Data from CSV File into Python List

```python
#reading a CSV file and converting the string into the list.
import csv
with open('Stu_data.csv', 'r') as read_obj:
    csv_reader = csv.reader(read_obj)
    # convert string to list
    list_of_csv = list(csv_reader)

    print(list_of_csv)
```

Output

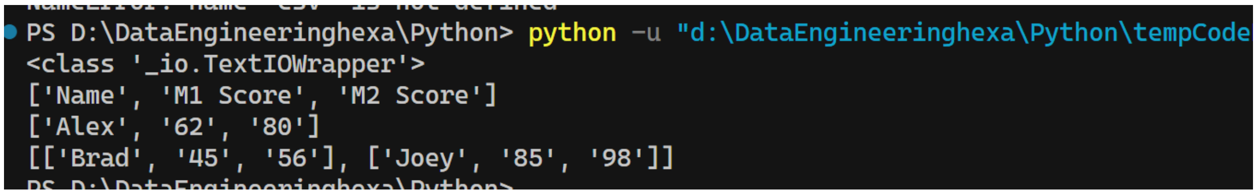


```
PS D:\DataEngineeringhexa\Python> python -u "d:\DataEngineeringhexa\Python\tempCodeRunnerFile.py"
[['Name', 'M1 Score', 'M2 Score'], ['Alex', '62', '80'], ['Brad', '45', '56'], ['Joey', '85', '98']]
PS D:\DataEngineeringhexa\Python>
```

Processing Python Lists

```
import csv
file=open('Stu_data.csv')
print(type(file))
#read csv
csvreader = csv.reader(file)
#empty list header
header=[]
#The .next() method returns the current row and moves to the next row.
#The first time you run next(), it returns the header, and the next time
you run, it returns the first record, and so on.
header=next(csvreader)
print(header)
header=next(csvreader)
print(header)
#Create an empty list called rows and iterate through the csvreader object
and append each row to the rows list.
rows = []
for row in csvreader:
    rows.append(row)
print(rows)
#.close() method is used to close the opened file. Once it is closed, we
cannot perform any operations on it.
file.close()
```

Output



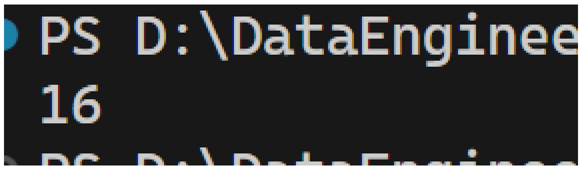
```
NameError: name 'csv' is not defined
PS D:\DataEngineeringhexa\Python> python -u "d:\DataEngineeringhexa\Python\tempCode
<class '_io.TextIOWrapper'>
['Name', 'M1 Score', 'M2 Score']
['Alex', '62', '80']
[['Brad', '45', '56'], ['Joey', '85', '98']]
PS D:\DataEngineeringhexa\Python>
```

Lambda Functions in Python

```
#Lambda Expressions
```

```python
#Lambda expressions are a concise way to create anonymous functions
(functions without a name).

square = lambda x: x**2
print(square(4))
```

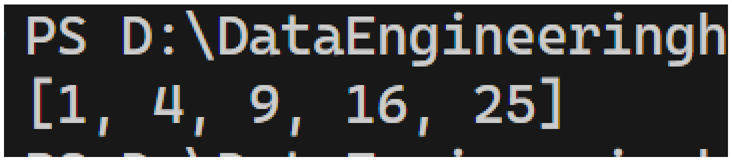
Output



```
PS D:\DataEnginee
16
```

```python
numbers = [1, 2, 3, 4, 5]
squared_numbers = list(map(lambda x: x ** 2, numbers))
print(squared_numbers)
```

Output



```
PS D:\DataEngineeringh
[1, 4, 9, 16, 25]
```

Usage of Lambda Functions

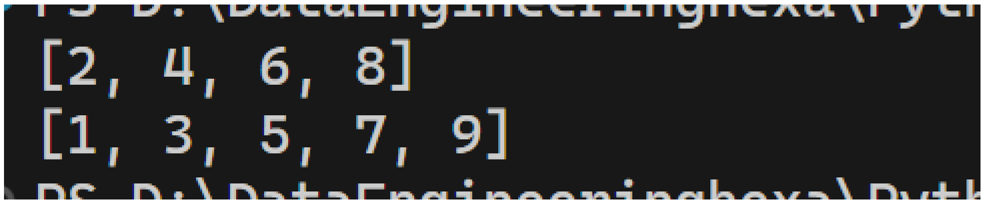
- Lambda functions, also known as anonymous functions or lambda expressions, are a concise way to create small, inline functions in programming languages.
- They are particularly useful in situations where a full function definition is not necessary, and a short, one-time-use function is sufficient.
- Lambda functions are commonly used in functional programming and are supported in various programming languages, including Python, JavaScript, and others.

Filter Data in Python Lists using filter and lambda

```python
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Using filter() with lambda to get even numbers
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
```

```python
print(even_numbers)

# Using filter() with lambda to get odd numbers
odd_numbers = list(filter(lambda x: x % 2 != 0, numbers))
print(odd_numbers)
```

Output



```
PS D:\DataEngineeringhexa\Pyth
[2, 4, 6, 8]
[1, 3, 5, 7, 9]
PS D:\DataEngineeringhexa\Pyth
```

a.Use of Lambda Function in Python

Inline Functions: Lambda functions are often used when a small, temporary function is needed for a short period, and defining a full function using def seems overly verbose.

Functional Programming: Lambda functions play a significant role in functional programming paradigms, where functions are treated as first-class citizens.

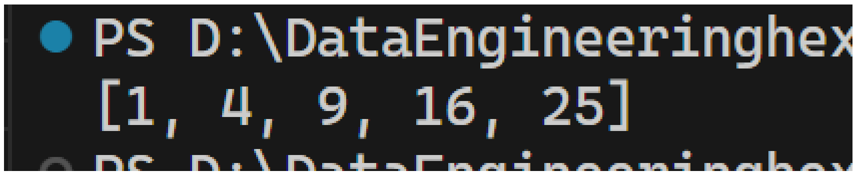
b.Practical Uses of Python lambda function

Sorting: Lambda functions are handy when custom sorting is needed.
Error Handling: Lambda functions can be useful in handling exceptions.

c.Using lambda() Function with map(),filter(),reduce()

```python
numbers = [1, 2, 3, 4, 5]
squared_numbers = list(map(lambda x: x**2, numbers))
print(squared_numbers)
```

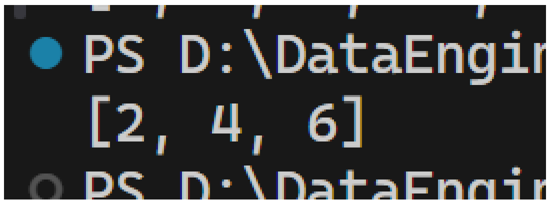
Output



```
PS D:\DataEngineeringhex
  [1, 4, 9, 16, 25]
PS D:\DataEngineeringhex
```

```python
numbers = [1, 2, 3, 4, 5, 6]
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(even_numbers)
```

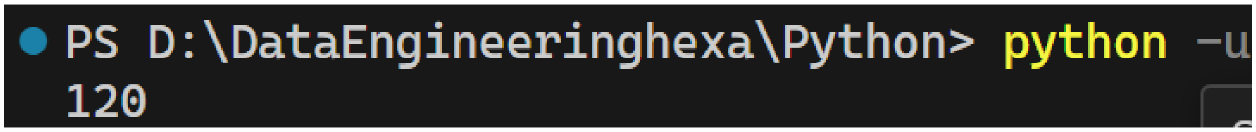
Output



```
PS D:\DataEngi
[2, 4, 6]
PS D:\DataEngi
```

```python
from functools import reduce

numbers = [1, 2, 3, 4, 5]
product = reduce(lambda x, y: x * y, numbers)
print(product)
```

Output



```
PS D:\DataEngineeringhexa\Python> python -u
120
```