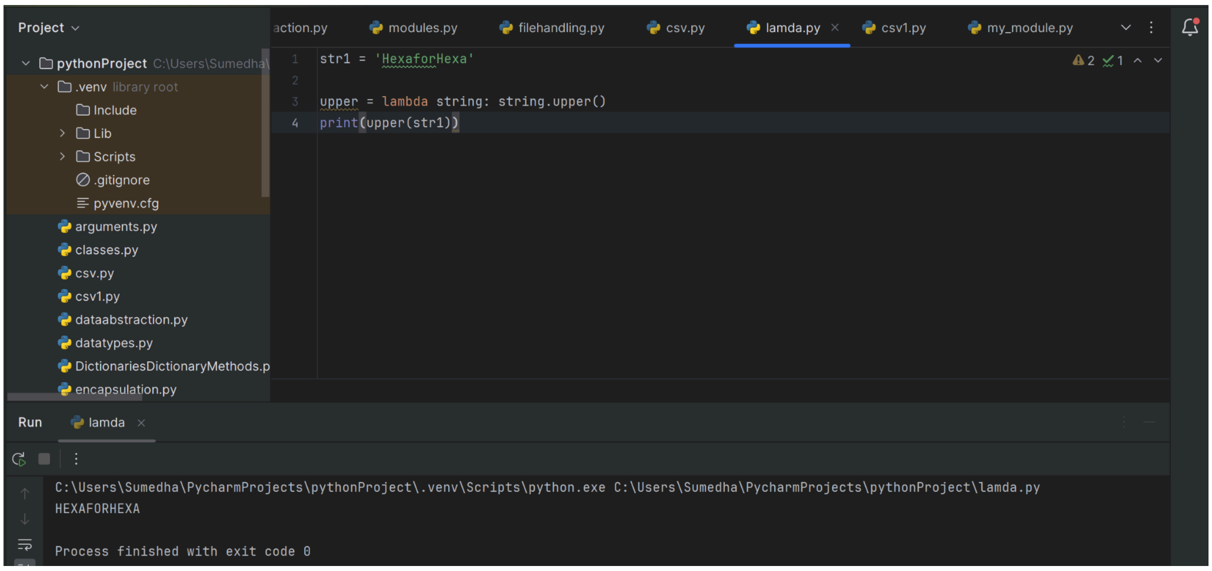
NAME : AKULA SHARATH CHANDRA
BATCH : DATA ENGINEERING
DATE : 30-01-24
TOPIC : Lambda Functions in Python
Usage of Lambda Functions
Filter Data in Python Lists using filter and lambda
a.Use of Lambda Function in Python
b.Practical Uses of Python lambda function
c.Using lambda() Function with map(),filter(),reduce()

1)LAMBDA FUNCTIONS: Lambda Functions are anonymous functions means that the function is without a name.
—>**Syntax:** lambda arguments : expression
EXAMPLE:

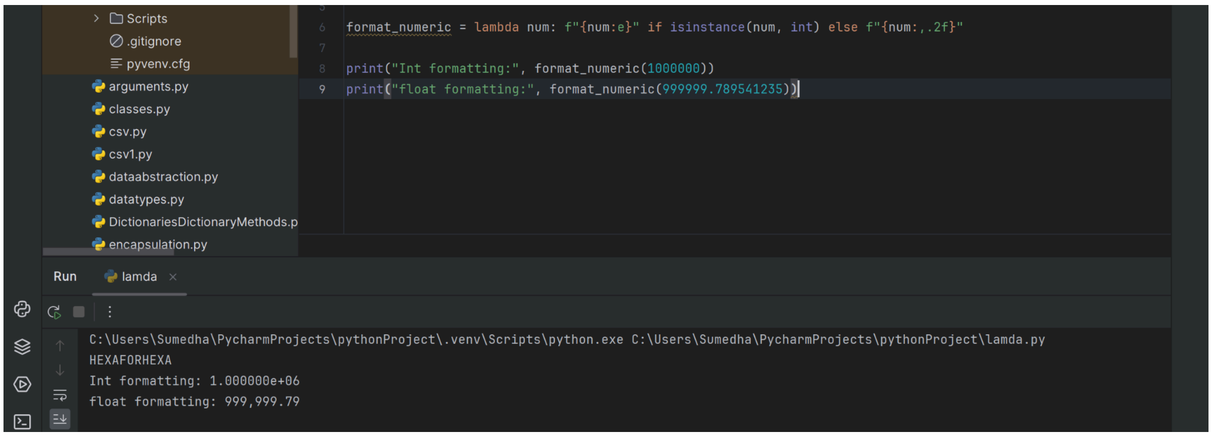


```python
str1 = 'HexaforHexa'

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

```
C:\Users\Sumedha\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\Sumedha\PycharmProjects\pythonProject\lamda.py
HEXAFORHEXA

Process finished with exit code 0
```

—>Use of Lambda Function in Python:

the **‘format_numric’** calls the lambda function, and the num is passed as a parameter to perform operations.



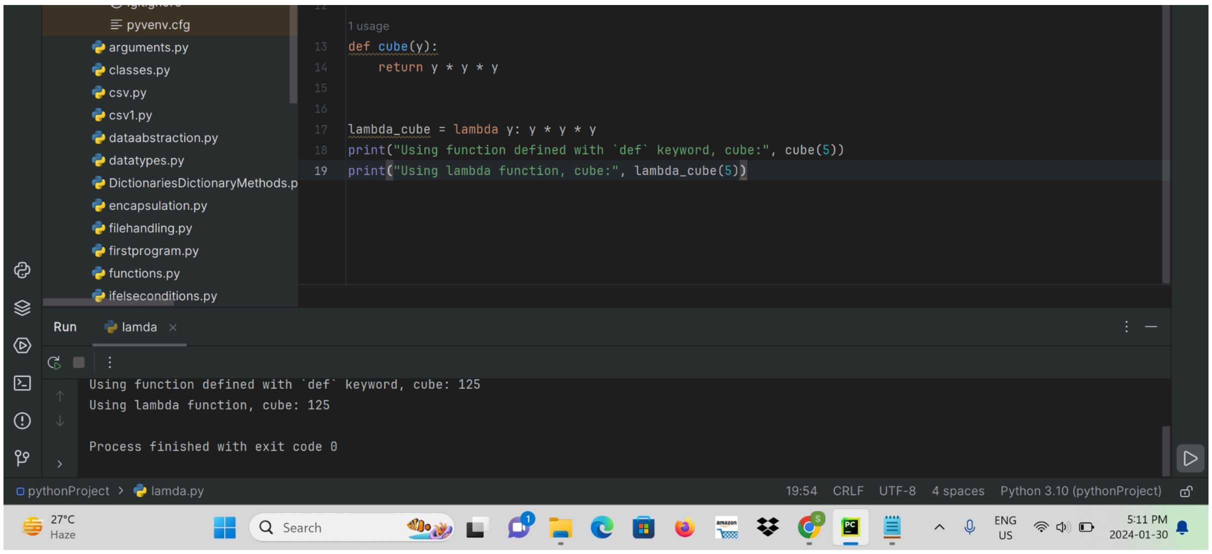
```python
format_numeric = lambda num: f"{num:e}" if isinstance(num, int) else f"{num:,.2f}"

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

```
C:\Users\Sumedha\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\Sumedha\PycharmProjects\pythonProject\lamda.py
HEXAFORHEXA
Int formatting: 1.000000e+06
float formatting: 999,999.79
```

→Difference Between Lambda functions and def defined function:

The code defines a cube function using both the **‘def'** keyword and a lambda function. It calculates the cube of a given number (5 in this case) using both approaches and prints the results. The output is 125 for both the **‘def'** and lambda functions, demonstrating that they achieve the same cube calculation.

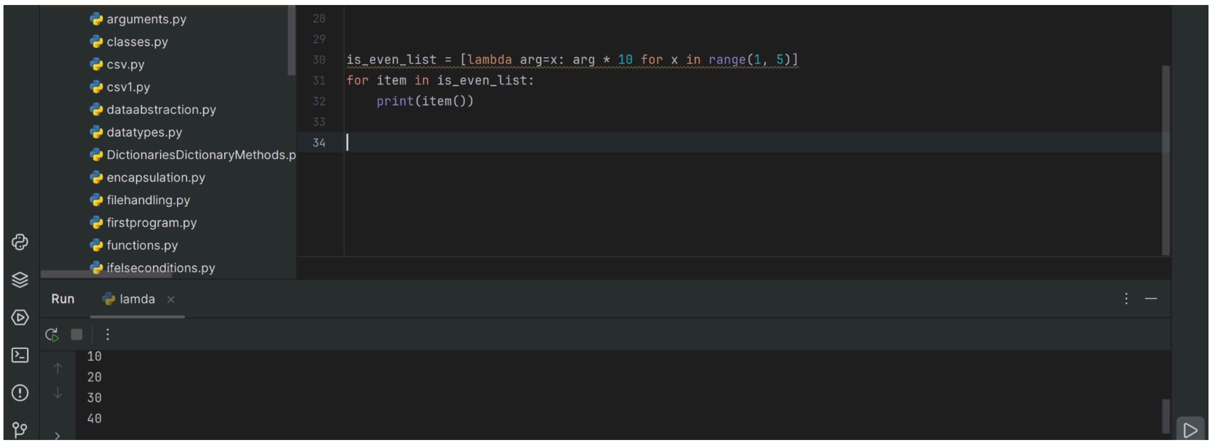
Example:



```
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))
```

```
Using function defined with `def` keyword, cube: 125
Using lambda function, cube: 125

Process finished with exit code 0
```

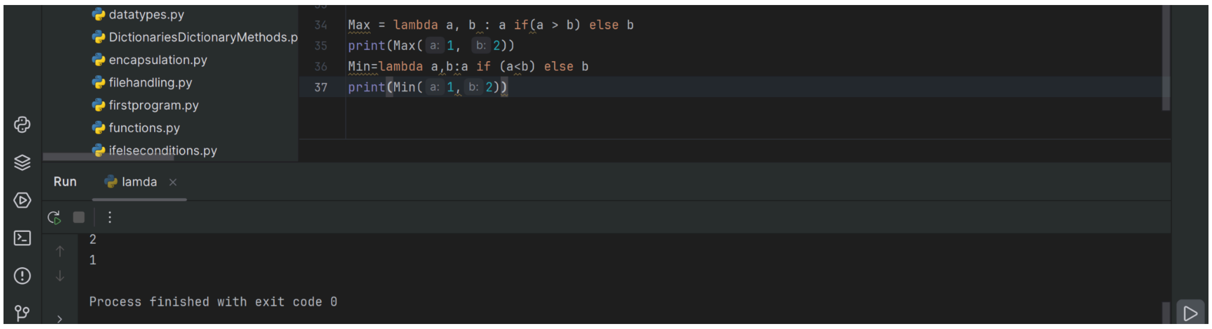
—>Python Lambda Function with List Comprehension:



```
is_even_list = [lambda arg=x: arg * 10 for x in range(1, 5)]
for item in is_even_list:
    print(item())
```

```
10
20
30
40
```

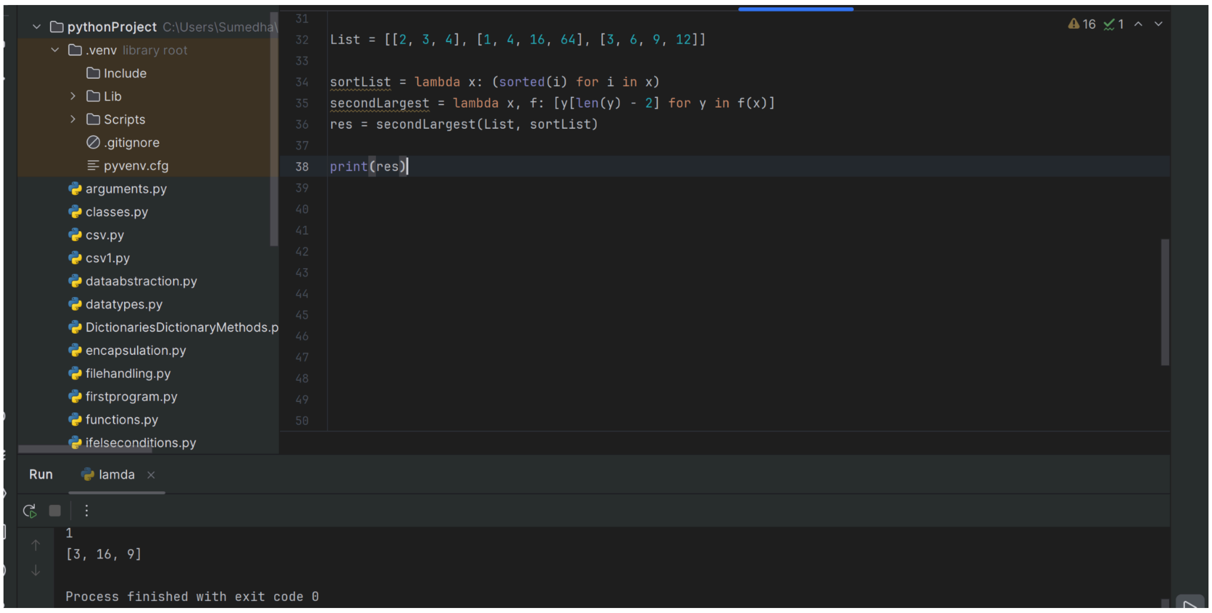
→Python Lambda Function with if-else:



```
Max = lambda a, b : a if(a > b) else b
print(Max( a: 1,  b: 2))
Min=lambda a,b:a if (a<b) else b
print(Min( a: 1, b: 2))
```

```
2
1

Process finished with exit code 0
```

→Python Lambda with Multiple Statements:



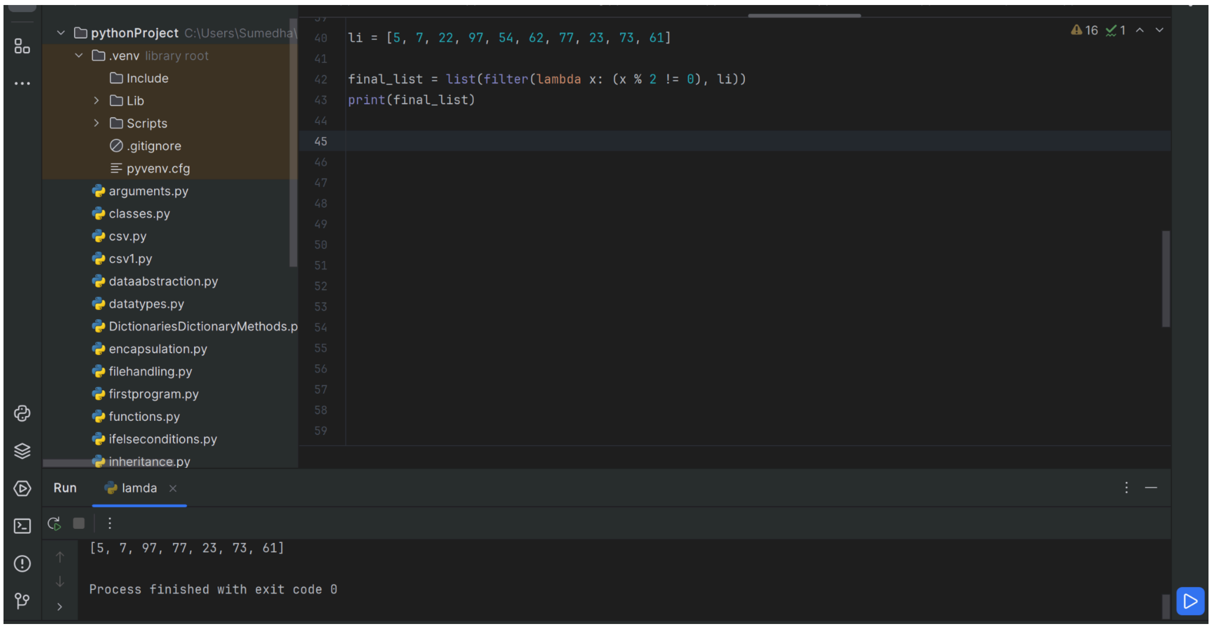
```python
List = [[2, 3, 4], [1, 4, 16, 64], [3, 6, 9, 12]]

sortList = lambda x: (sorted(i) for i in x)
secondLargest = lambda x, f: [y[len(y) - 2] for y in f(x)]
res = secondLargest(List, sortList)

print(res)
```

```
1
[3, 16, 9]

Process finished with exit code 0
```

—>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.

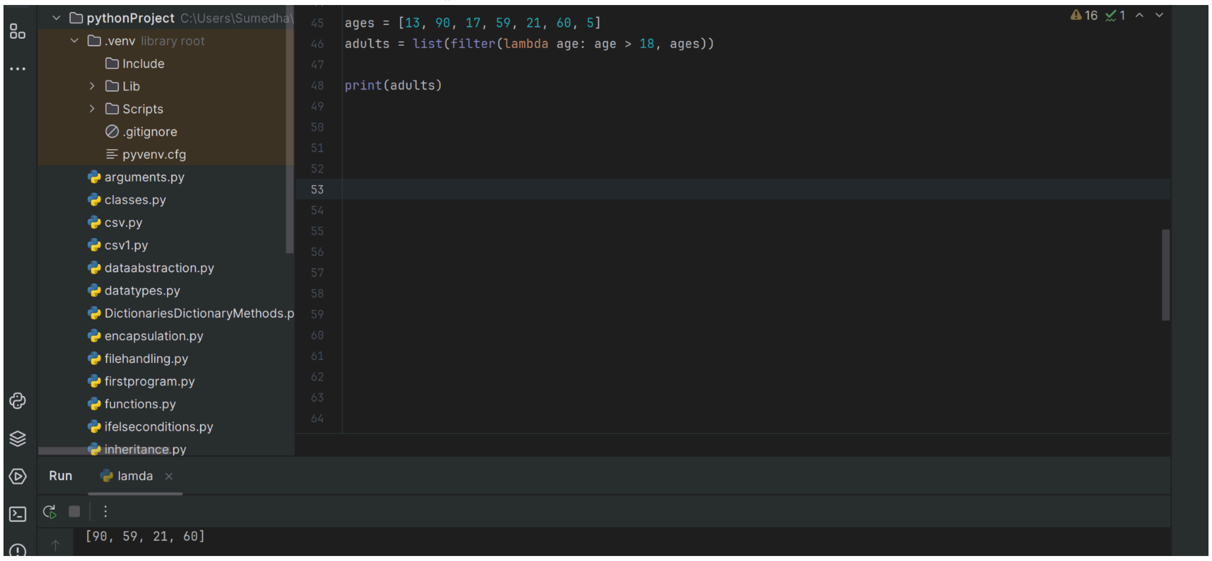
example:Filter out all odd numbers using filter() and lambda function.



```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)
```

```
[5, 7, 97, 77, 23, 73, 61]

Process finished with exit code 0
```

example2:Filter all people having age more than 18, using lambda and filter() function

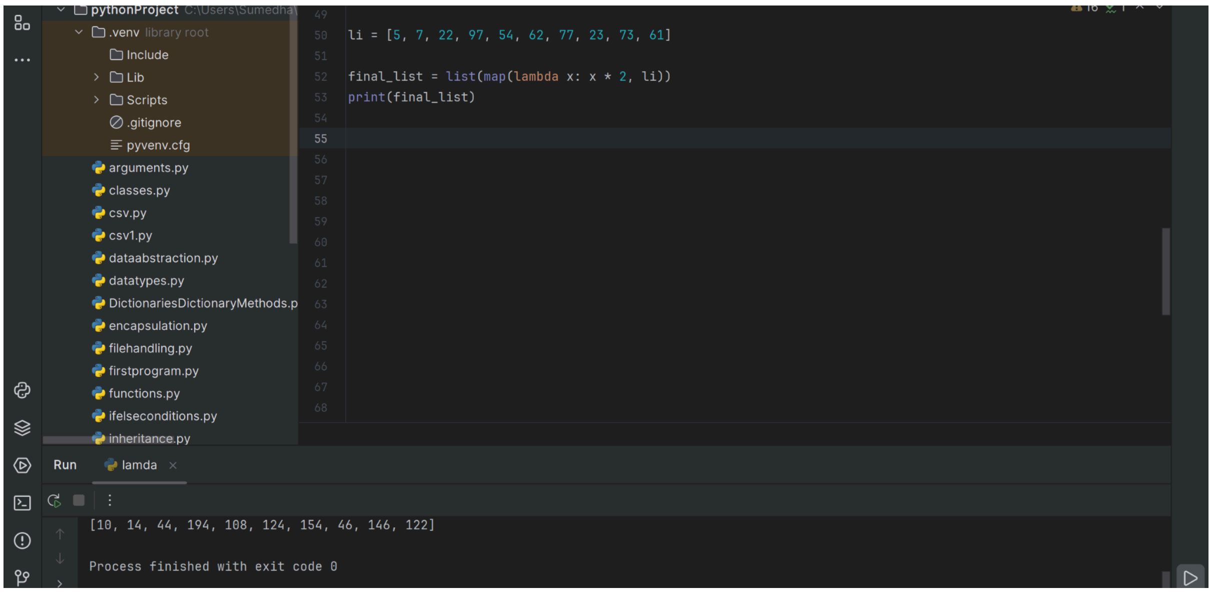


```python
ages = [13, 90, 17, 59, 21, 60, 5]
adults = list(filter(lambda age: age > 18, ages))

print(adults)
```

```
[90, 59, 21, 60]
```

—>Using lambda() Function with map():

The map() function in Python takes in a function and a list as an argument. The function is called with a lambda function and a list and a new list is returned which contains all the lambda-modified items returned by that function for each item.



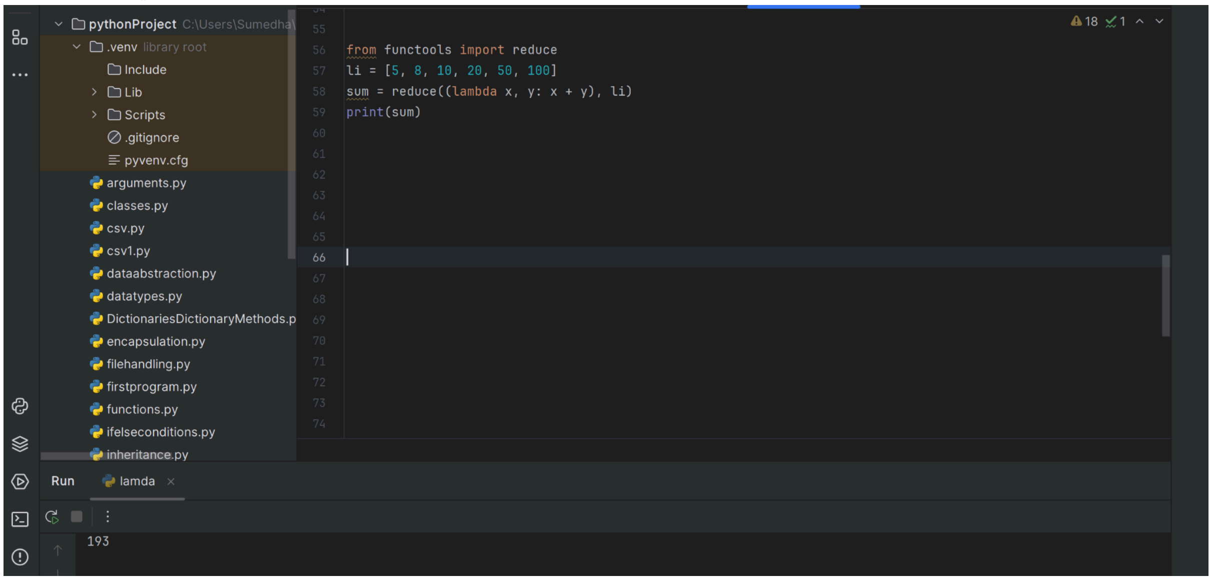
```
li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]

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

```
[10, 14, 44, 194, 108, 124, 154, 46, 146, 122]

Process finished with exit code 0
```

—>Using lambda() Function with reduce():

The reduce() function in Python takes in a function and a list as an argument. The function is called with a lambda function and an iterable and a new reduced result is returned.The reduce() function belongs to the funtools module.

example:A sum of all elements in a list using lambda and reduce() function.



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

```
193
```