

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:

```
1 str1 = 'HexaforHexa'
2
3 upper = lambda string: string.upper()
4 print(upper(str1))
```

Run lamda x

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_numeric**' calls the lambda function, and the num is passed as a parameter to perform operations.

```
6 format_numeric = lambda num: f"{num:e}" if isinstance(num, int) else f"{num:.2f}"
7
8 print("Int formatting:", format_numeric(1000000))
9 print("float formatting:", format_numeric(999999.789541235))
```

Run lamda x

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:

The screenshot shows a code editor with a file explorer on the left containing files like `pyvenv.cfg`, `arguments.py`, `classes.py`, `csv.py`, `csv1.py`, `dataabstraction.py`, `datatypes.py`, `DictionariesDictionaryMethods.p`, `encapsulation.py`, `filehandling.py`, `firstprogram.py`, `functions.py`, and `ifelseconditions.py`. The main editor displays the following code:

```
1 usage
13 def cube(y):
14     return y * y * y
15
16
17 lambda_cube = lambda y: y * y * y
18 print("Using function defined with 'def' keyword, cube:", cube(5))
19 print("Using lambda function, cube:", lambda_cube(5))
```

The Run console at the bottom shows the output:

```
Using function defined with 'def' keyword, cube: 125
Using lambda function, cube: 125
Process finished with exit code 0
```

The status bar at the bottom indicates the file is `pythonProject > lamda.py`, with settings for 19:54, CRLF, UTF-8, 4 spaces, and Python 3.10.

→Python Lambda Function with List Comprehension:

The screenshot shows the same code editor with the following code:

```
28
29
30 is_even_list = [lambda arg=x: arg * 10 for x in range(1, 5)]
31 for item in is_even_list:
32     print(item())
33
34
```

The Run console shows the output:

```
10
20
30
40
Process finished with exit code 0
```

→Python Lambda Function with if-else:

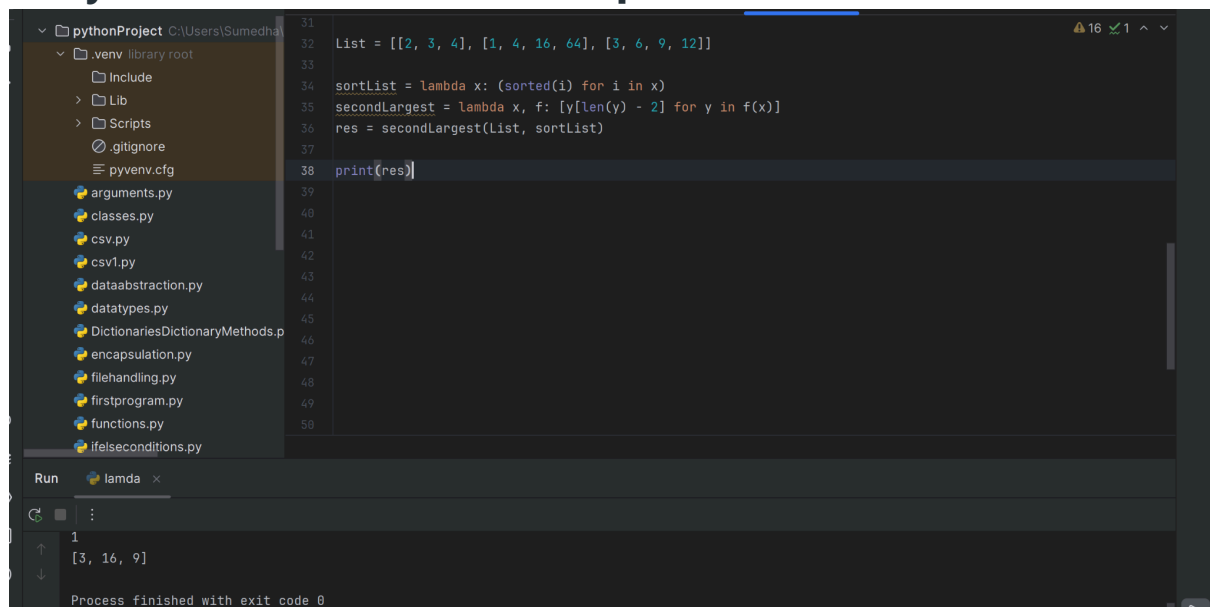
The screenshot shows the same code editor with the following code:

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

The Run console shows the output:

```
2
1
Process finished with exit code 0
```

→Python Lambda with Multiple Statements:



```
31 List = [[2, 3, 4], [1, 4, 16, 64], [3, 6, 9, 12]]
32
33
34 sortList = lambda x: (sorted(i) for i in x)
35 secondLargest = lambda x, f: [y[len(y) - 2] for y in f(x)]
36 res = secondLargest(List, sortList)
37
38 print(res)
39
40
41
42
43
44
45
46
47
48
49
50
```

Run lamda x

```
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.

```
pythonProject C:\Users\Sumedha
├── .venv library root
│   ├── Include
│   ├── Lib
│   └── Scripts
├── .gitignore
├── pyvenv.cfg
├── arguments.py
├── classes.py
├── csv.py
├── csv1.py
├── dataabstraction.py
├── datatypes.py
├── DictionariesDictionaryMethods.p
├── encapsulation.py
├── filehandling.py
├── firstprogram.py
├── functions.py
├── ifelseconditions.py
└── inheritance.py
```

```
39
40
41 li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]
42
43 final_list = list(filter(lambda x: (x % 2 != 0), li))
44
45 print(final_list)
46
47
48
49
50
51
52
53
54
55
56
57
58
59
```

Run lamda x

[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

```
pythonProject C:\Users\Sumedha
├── .venv library root
│   ├── Include
│   ├── Lib
│   └── Scripts
├── .gitignore
├── pyvenv.cfg
├── arguments.py
├── classes.py
├── csv.py
├── csv1.py
├── dataabstraction.py
├── datatypes.py
├── DictionariesDictionaryMethods.p
├── encapsulation.py
├── filehandling.py
├── firstprogram.py
├── functions.py
├── ifelseconditions.py
└── inheritance.py
```

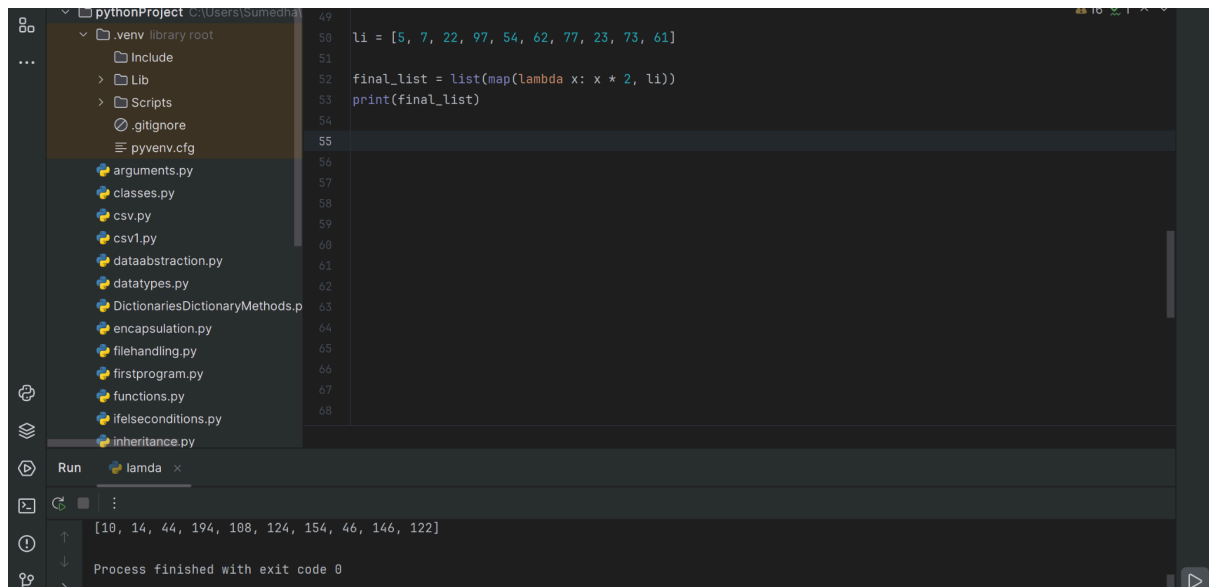
```
45 ages = [13, 90, 17, 59, 21, 60, 5]
46 adults = list(filter(lambda age: age > 18, ages))
47
48 print(adults)
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
```

Run lamda x

[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.



```
49
50 li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]
51
52 final_list = list(map(lambda x: x * 2, li))
53 print(final_list)
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
```

Run lamda x

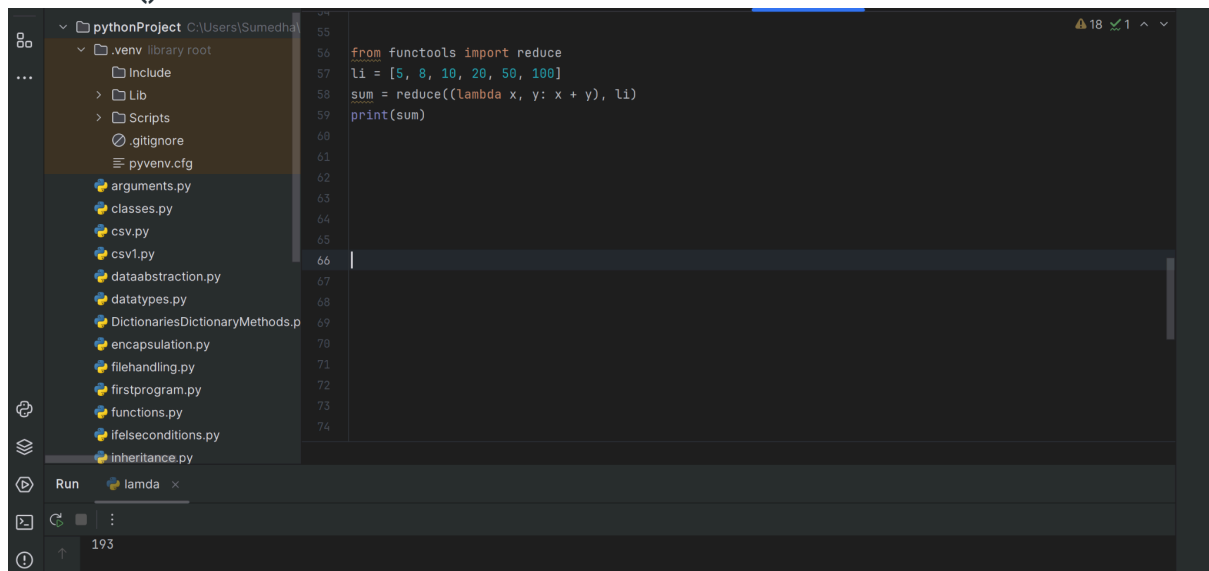
[10, 14, 44, 194, 108, 124, 154, 46, 146, 122]

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 functools module.

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



```
55
56 from functools import reduce
57 li = [5, 8, 10, 20, 50, 100]
58 sum = reduce(lambda x, y: x + y, li)
59 print(sum)
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
```

Run lamda x

193

