Lambda

By: Ravishankar Chavare

Lambda

Lambda function used to create a small anonymous function without name. Lambda is a powerful python function with some limited capabilities . Lambda function is also referred to as an anonymous function . Lambda function can accept any number of elements as input but only one expression is evaluated and returned.

Why Use Lambda Function?

Lambda Function is anonymous .Anonymous means we don't need to give a name to lambda function when we are writing single line function prefer lambda function instead of using normal functions. To write a normal function we need a **def** keyword to define a function and return keyword is used to **return** something from a function . Lambda function requires a **lambda** keyword to define a lambda function and don't need any keyword for return values from lambda functions.

Syntax

For single Input

```
lambda <argument>: <Calculation or Evaluation>
```

For Multiple Input

```
lambda <argument1,argument2,argument3..>: <Calculation or Evaluation>
```

Example

```
lambda a : a+10
```

Above function accepts 1 variable and returns it after adding 10 into it.

Example Of Addition of Two Variable

```
lambda a,b : a+b
```

Above function accepts 2 variables a and b and return addition of a and b.

Example Single Inputs

Using Normal Function

```
# Create a Function To Add 10 To Input Number
def Add(a):
  return a+10

result = Add(5)
print(result)
```

Using Lambda Function

```
# Create a Lambda Function
Add = lambda a : a+10
result = Add(5)
print(result)
```

Output of Above Both Programs

```
15
```

Example Multiple Inputs

Using Normal Function

```
# Create a Function to accept two number and returns addition
def Add(a,b):
  return a+b

result = Add(5,6)
print(result)
```

Using Lambda Function

```
Add = lambda a,b : a+b
result = Add(5,6)
print(result)
```

Output of Above Both Programs

```
11
```

Lambda Function can be used with Following python built-in functions

- map
- filter
- Reduce

Note : Lambda Function has some limited capabilities such as only one expression can be evaluated .

map

Map

map() is a python built-in function used to apply a specific function on every element of an iterable object such as list, tuple, strings etc . map() returns a mapped object after applying function to every element from iterable object . map() can be used with a normal function or lambda function.

Syntax

```
map(func, *iterables)
```

Where

func : a normal a function or a Lambda function.

Iterbales : Any iterable Object such as list, string, tuple etc..

Examples

We are going to practice here multiple Examples using **map()** function with the help of normal user defined function and a lambda function .

Add 5 To every list Element (normal function)

```
# Function to add 5 to given input element
def Add(a):
  return a+5

# A Sample list
lst = [1,2,3,4]
# Apply Add Function to lst with map()
result = map(Add,lst)
print(f" Result of Mapped Function :{result}")
# Display original List
print(f" Original List : {lst}")
# Type cast map object to list
final_result = list(result)
print(f" After Applying Map Function(Add) :
{final_result}")
```

Add 5 To every list Element (lambda function)

```
# A Sample list
lst = [1,2,3,4]
# Apply Add Function to lst with map()
result = map(lambda a : a+5,lst)
print(f" Result of Mapped Function :{result}")
# Display original List
print(f" Original List : {lst}")
# Type cast map object to list
final_result = list(result)
print(f" After Applying Map Function(Add) :
{final_result}")
```

Output of above Examples

```
Result of Mapped Function :<map object at 0x7ffaeb8714a8>
Original List : [1, 2, 3, 4]
After Applying Map Function(Add) : [6, 7, 8, 9]
```

Convert a String Numbers List to int Numbers

```
# A Sample strings numbers list
lst = ['1','2','3','4']

# Convert it into int
result = map(int,lst)


# Display original List
print(f" Original List : {lst}")
# Type cast map object to list

final_result = list(result)
print(f" Int List(After Conversion) : {final_result}")
```

Output

```
Original List : ['1', '2', '3', '4']
Int List(After Conversion) : [1, 2, 3, 4]
```

Original List (Datatype)

```
type(lst[0])    :    <class 'str'>
```

Converted List (Datatype)

```
type(final_result[0])    :    <class 'int'>
```

Addition of Two numbers List

```
# Define Two Lists
lst1 = [1,2,3,4]
lst2 = [5,6,7,8]

# Use of Multiple Parameter as input
Addition = map(lambda a,b:a+b,lst1,lst2)


print(f" List 1 : {lst1} \n List 2 : {lst2}")

# Convert mapped object to list
final_result = list(Addition)
print(f" Addition of List1 and List2 : {final_result}")
```

Output

```
List 1 : [1, 2, 3, 4]
List 2 : [5, 6, 7, 8]
Addition of List1 and List2 : [6, 8, 10, 12]
```

Addition of list1 and list2

list1	1	2	3	4
list2	5	6	7	8
Addition	6	8	10	12

Filter

Filter

filter() is a python built-in method used to filter out records from an iterable object based on condition evaluated from a Function . filter() function is applicable to every element of an iterable object. filter() methods added the element to the final list only if applied function returns **True** Otherwise that element is excluded .

Syntax

```
filter(function or None, iterable)
```

Where

function : Any user defined normal function or lambda function.

iterable : any iterable Object such as list, tuple, strings etc...

Examples

```
# Define a  List
lst1 = [1,2,3,4,5,6,7,8]

# Filter out odd Numbers
even_numbers = filter(lambda a:a%2,lst1)

print(f" List 1 : {lst1}")
# Convert mapped object to list
final_result = list(even_numbers)
print(f"Filtered List(Odd) : {final_result}")
```

Output

```
List 1 : [1, 2, 3, 4, 5, 6, 7, 8]
Filtered List(Odd) : [1, 3, 5, 7]
```

Filter Uppercase letters

```
# Define a  string
text = "PyThOn"

# Filter out Capital Letters
even_numbers = filter(lambda a:a.isupper(),text)

print(f" text : {text}")
# Convert mapped object to list
final_result = list(even_numbers)
print(f"Uppercase Letters : {final_result}")
```

Output

```
text : PyThOn
Uppercase Letters : ['P', 'T', 'O']
```

Filter Even numbers from given input list

```
 # Define a  List
lst1 = [1,2,3,4,5,6,7,8]

# Filter out Even Numbers
even_numbers = filter(lambda a:a%2==0,lst1)
```

Reduce

Reduce

reduce() function is not a part of python built-in function you need to import it from **functools**. In python 3 most of the developers ignore the use of reduce() method. reduce() is a function used to reduces a iterable object to a single value by combining element via a supplied function.

Syntax

```
from functools import reduce
reduce(function, sequence[, initial])
```

Where

function : Any lambda or normal user defined function.

Sequence : any sequence iterable list

Example

Consider

Lst1 = [1,2,3,4,5]

Output Should be

15

Reduce Calculate the Sequence if reduce(lambda a,b :a+b,Lst1)

Res = ((((1+2)+3)+4)+5)

Res= 15

Example

```
# Import reduce from functools
from functools import reduce

Lst1 = [1,2,3,4,5]

result = reduce(lambda a,b :a+b,Lst1)

print(f"Addition of All Elements in Lst1 is : {result}")
```

Output

```
Addition of All Elements in Lst1 is : 15
```

Exercise

1. Write a Program using lambda to calculate square of given user input radius .

2. Write a Program using lambda to multiply three elements

3. Write a Program For Addition of Following Lists

```
Lst1 = [0,1,2]
Lst2 = ["3","4","5"]
```

4. Write a Program to find out numbers from range(1,121) divisible by 3

5. Write a Program to calculate sum of odd numbers (1,100)

6. Write a program to convert List element to int

```
Lst = ["3","4","5"]
```

Note: Submit solution on Following address

To everyone who has taken time out of their day to read it.

Created by : Ravishankar Chavare
LinkedIn : https://in.linkedin.com/in/ravishankar-chavare-84474a102
Mobile : 983485612