What is Lambda Function in Python?

A Lambda Function in Python programming is an anonymous function or a function having no name. It is a small and restricted function having no more than one line. Just like a normal function, a Lambda function can have multiple arguments with one expression.

In Python, lambda expressions (or lambda forms) are utilized to construct anonymous functions. To do so, you will use the lambda keyword (just as you use def to define normal functions). Every anonymous function you define in Python will have 3 essential parts:

1. The lambda keyword.
2. The parameters (or bound variables), and
3. The function body.

A lambda function can have any number of parameters, but the function body can only contain one expression. Moreover, a lambda is written in a single line of code and can also be invoked immediately.

Syntax and Examples

The formal syntax to write a lambda function is as given below:

```
lambda p1,p2:expression
```

However, notice that we do not use brackets around the parameters as we do with regular functions. The last part (expression) is any valid python expression that operates on the parameters you provide to the function.

Example 1

In [1]:
```
add = lambda x,y : x+y
print(add(2,3))
```

```
5
```

Code Explanation

Here, we define a variable that will hold the result returned by the lambda function.

1. The lambda keyword used to define an anonymous function.
2. x and y are the parameters that we pass to the lambda function.
3. This is the body of the function, which adds the 2 parameters we passed. Notice that it is a single expression. You cannot write multiple statements in the body of a lambda function.
4. We call the function and print the returned value.

Example 2

In [2]:
```
# what a lambda returns???

string = "some kind of useless lambda"
print(lambda string : print(string))
```

```
<function <lambda> at 0x000001F16A789310>
```

Code Explanation

1. Here, we define a string that you'll pass as a parameter to the lambda.
2. We declare a lambda that calls a print statement and prints the result.

But why doesn't the program print the string we pass?

This is because the lambda itself returns a function object. In this example, the lambda is not being called by the print function but simply returning the function object and the memory location where it is stored. That's what gets printed at the console.

Example 3

In [3]:
```
# what a lambda returns? #2
x = "some kind of useless lambda"
(lambda x: print(x))(x)

```

```
some kind of useless lambda
```

Code Explanation

- Here is the same string we defined in the previous example.
- In this part, we are defining a lambda and calling it immediately by passing the string as an argument. This is something called an IIFE,

Using lambdas with Python built-ins

Lambda functions provide an elegant and powerful way to perform operations using built-in methods in Python. It is possible because lambdas can be invoked immediately and passed as an argument to these functions.

IIFE in Python Lambda

IIFE stands for immediately invoked function execution. It means that a lambda function is callable as soon as it is defined

lambdas in filter()

The filter function is used to select some particular elements from a sequence of elements. The sequence can be any iterator like lists, sets, tuples, etc.

The elements which will be selected is based on some pre-defined constraint. It takes 2 parameters:

1. A function that defines the filtering constraint
2. A sequence (any iterator like lists, tuples, etc.)

In [4]:
```
num = [45,10,6,24,5,4,90,16]
filter_result = filter(lambda x:x>4,num)
print(list(filter_result))
```

```
[45, 10, 6, 24, 5, 90, 16]
```

Code Explanation:

1. In the first statement, we define a list called num which contains some numbers.
2. Here, we declare a variable called filtered_result, which will store the filtered values returned by the filter() function.
3. A lambda function which runs on each element of the list and returns true if it is greater than 4.
4. Print the result returned by the filter function.

In [5]:
```
name = ['Harshit','Aman','Akash']
list(filter(lambda x:x.startswith("A"), name))

```

Out[5]:
```
['Aman', 'Akash']
```

In the code above, you are using a filter function to return the names that start with the character "A".In the implementation of the filter function, the lambda function lambda x: x.startswith["A"] would return a lambda object, and the filter function would filter out the names that are starting with the character "A".

filter function returning multiple of 3

In [6]:
```
l1 = [3,4,5,6,7,8]
div3 = lambda x:x%3==0

div = filter(div3,l1)
print(list(div))

```

```
[3, 6]
```

write a python program to count the student above 18 age

In [7]:
```
stu_data = {1:["sam",15],2:["deep",20],3:["ram",16],4:["sham",18]}
len(list(filter(lambda x:x[1]>18,stu_data.values())))
```

Out[7]:
```
1
```

In [8]:
```
stu_data = {1:["sam",15],2:["deep",20],3:["ram",16],4:["sham",18]}
print(list(filter(lambda x:x[1]>18,stu_data.values())))
```

```
[['deep', 20]]
```

lambdas in map()

the map function is used to apply a particular operation to every element in a sequence. Like filter(), it also takes 2 parameters:

1. A function that defines the op to perform on the elements
2. One or more sequences

In [9]:
```
num = [45,10,6,24,5,4,90,16]
filter_result = map(lambda x:x*x,num)
print(list(filter_result))
```

```
[2025, 100, 36, 576, 25, 16, 8100, 256]
```

explanation :

- We declare a variable called filtered_result which will store the mapped values
- A lambda function which runs on each element of the list and returns the square of that number.

In [10]:
```
def multi(x):
    return x*2
list_numbers = [1,2,3,4]
sample_map = map(multi, list_numbers)
print(list(sample_map))
```

```
[2, 4, 6, 8]
```

Note: The difference between the previous code and this above-mentioned code is that instead of applying a lambda function, you can use the function object also. multi returns a function object just like the lambda function used earlier.

In [11]:
```
l1 = [2,4,5]
def square(x):
    return x**2

print(list(map(square,l1)))
```

```
[4, 16, 25]
```

lambdas in reduce()

The reduce function, like map(), is used to apply an operation to every element in a sequence. However, it differs from the map in its working. These are the steps followed by the reduce() function to compute an output:

Step 1) Perform the defined operation on the first 2 elements of the sequence.

Step 2) Save this result

Step 3) Perform the operation with the saved result and the next element in the sequence.

Step 4) Repeat until no more elements are left.

It also takes two parameters:

- A function that defines the operation to be performed
- A sequence (any iterator like lists, tuples, etc.)

For example, here is a program that returns the product of all elements in a list:

For example, here is a program that returns the product of all elements in a list.

In [12]:
```
from functools import reduce
```

In [13]:
```
seq = [1,2,3,4,5]
product = reduce(lambda x,y : x*y,seq)
print(product)
```

```
120
```

Code Explanation:

Import reduce from the functools module Here, we define a list called sequences which contains some numbers. We declare a variable called product which will store the reduced value A lambda function that runs on each element of the list. It will return the product of that number as per the previous result. Print the result returned by the reduce function.

In [14]:
```
list_1 = ['harshit','aman']
reduce(lambda x,y: x + y,list_1)
```

Out[14]:
```
'harshitaman'
```

In the code snippet given above, you are importing functools library is being imported to access reduce function. In the implementation of the reduce function, the lambda function x,y : x+y, list_1 is appending two objects remember here if the objects are strings it appends them if the objects are numbers it adds them. Now, the reduce function will convert the list of string into a single list

In [15]:
```
q = reduce(lambda x,y:x*y,range(1,4))
print(q)
```

```
6
```

practice

Using the function Map, count the number of words that start with 'S' in input_list.

In [16]:
```
input_list = ['Santa Cruz','Santa fe','Mumbai','Delhi']
count = sum(map(lambda x: x[0] == 'S', input_list))
print(count)
```

```
2
```

Create a list ‘name’ consisting of the combination of the first name and the second name from list 1 and 2 respectively.

In [17]:
```
data = [['Ankur', 'Avik', 'Kiran', 'Nitin'], ['Narang', 'Sarkar', 'R', 'Sareen']]
name = list(map(' '.join, zip(*data)))
print(name)
```

```
['Ankur Narang', 'Avik Sarkar', 'Kiran R', 'Nitin Sareen']
```

In [18]:
```
# another method

input_list = [['Ankur', 'Avik', 'Kiran', 'Nitin'], ['Narang', 'Sarkar', 'R', 'Sareen']]
first_name = input_list[0]
last_name = input_list[1]

name = list(map(lambda x,y: x+ " " + y, first_name,last_name))

print(list(name))
```

```
['Ankur Narang', 'Avik Sarkar', 'Kiran R', 'Nitin Sareen']
```

Extract a list of numbers that are multiples of 5 from a list of integers named input_list.

In [19]:
```
input_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28,
list_answer = list(filter(lambda x:x%5==0,input_list))
print(list(list_answer))
```

```
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
```

You are given a list of strings such as input_list = ['hdjk', 'salsap', 'sherpa'].Extract a list of names that start with an ‘s’ and end with a ‘p’ (both 's' and 'p' are lowercase) in input_list.

Note: Use the filter() function.

In [20]:
```
input_list = ['hdjk', 'salsap', 'sherpa']
sp = list(filter(lambda x:x[0]=="s" and x[-1]=="p",input_list))
print(sp)
```

```
['salsap']
```

Reduce Function

Description Using the Reduce function, concatenate a list of words in input_list, and print the output as a string.

If input_list = ['I','Love','Python'], the output should be the string 'I Love Python'.

In [21]:
```
input_list = ['All','you','have','to','fear','is','fear','itself']

string1 = str(reduce(lambda x,y: x + " " +y, input_list))

print(string1)
```

```
All you have to fear is fear itself
```

You are given a list of numbers such as input_list = [31, 63, 76, 89]. Find and print the largest number in input_list using the reduce() function.

In [22]:
```
input_list = [31, 63, 76, 89]
answer = reduce(lambda x, y:y if x<y else x ,input_list )

print(answer)

```

```
89
```

In [23]:
```
answer = reduce(lambda a,b : a if a > b else b,input_list)

print(answer)
```

```
89
```

sum of the squares of 2 given numbers

In [24]:
```
square = lambda x,y : x**2 + y**2
```

In [25]:
```
square(3,4)*2
```

Out[25]:
```
50
```

In [26]:
```
# Python code to illustrate cube of a number
# showing difference between def() and lambda().
def cube(y):
    return y*y*y
```

In [27]:
```
cube(3)
```

Out[27]:
```
27
```

In [28]:
```
cube = lambda x:x**3
```

In [29]:
```
cube(3)
```

Out[29]:
```
27
```

Write a lambda function to check a number is even or odd

In [30]:
```
even_or_odd = lambda x:"even" if  x%2==0 else "odd"
```

In [31]:
```
even_or_odd(345)
```

Out[31]: 'odd'

Get the greater of two number

In [32]:
```
greater = lambda x,y : x if x>y else y
```

In [33]:
```
greater(67,4)
```

Out[33]: 67

Square of all the numbers

In [34]:
```
l = [2,4,5]
```

In [35]:
```
list(map(lambda x: x**2,l))
```

Out[35]: [4, 16, 25]

In [36]:
```
# Python code to illustrate
# map() with lambda()
# to get double of a list.
li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]
```

In [37]:
```
list(map(lambda x: x*2 ,li))
```

Out[37]: [10, 14, 44, 194, 108, 124, 154, 46, 146, 122]

In [38]:
```
# Map can take normal function also
L = [1,2,3]
def sqr(x):
    return x*x
```

In [39]:
```
list(map(sqr,  L))
```

Out[39]:
```
[1, 4, 9]
```

Add two Lists using map

In [40]:
```
num1 = [4, 5, 6]
num2 = [5, 6, 7]
```

In [41]:
```
list(map(lambda x,y:x+y,num1,num2))
```

Out[41]:
```
[9, 11, 13]
```

Calculate the cube of the values

In [42]:
```
my_dictionary = {1:['Sam', 15] , 2:['Rob',18], 3:['Kyle', 16], 4:['Cornor',19], 5:['Trump',20]}
```

In [43]:
```
dict(map(lambda x: (x[0],[x[1][0],x[1][1] **3]),my_dictionary.items()))
```

Out[43]:
```
{1: ['Sam', 3375],
 2: ['Rob', 5832],
 3: ['Kyle', 4096],
 4: ['Cornor', 6859],
 5: ['Trump', 8000]}
```

Using with filter()

- filter() - is a built-in function which takes a function and a list as parameters.

- The function given as input will be applied on every value in the list and checks if the value is satisfying the condition and evaluates to Boolean value (True/False). Then the output will be the values from the list which are evaluated to True.

Want to get even numbers

In [44]:
```
l1 = list(range(1,20))
```

In [45]:
```
list(filter(lambda x: x%2==0  ,l1))
```

Out[45]:
```
[2, 4, 6, 8, 10, 12, 14, 16, 18]
```

Write a python program to count the students above age 16

In [46]:
```
students_data = {1:['Sam', 15] , 2:['Rob',18], 3:['Kyle', 16], 4:['Cornor',19], 5:['Trump',20]}
```

In [47]:
```
list(filter(lambda x: x[1] > 16 ,students_data.values()))
```

Out[47]:
```
[['Rob', 18], ['Cornor', 19], ['Trump', 20]]
```

In [48]:
```
list(filter(lambda x: x[1][1] > 16 ,students_data.items()))
```

Out[48]:
```
[(2, ['Rob', 18]), (4, ['Cornor', 19]), (5, ['Trump', 20])]
```

Reduce

In [49]:
```
# Python code to illustrate
# reduce() with lambda()
# to get sum of a list
import functools
from functools import reduce
```

In [50]:
```
li = [5, 8, 10, 20, 50, 100]
```

In [51]:
```
reduce(lambda x,y: x+y,li)
```

Out[51]: 193

In [52]:
```
# initializing list
lis = [ 1 , 3, 5, 6, 2]

# using reduce to compute maximum element from list

reduce(lambda x,y:x if x>y else y, lis)
```

Out[52]: 6

Write a code to find the factorial of a number using reduce function. ¶

In [53]:
```
n = int(input("Enter a number : "))
n_list = range(1,n+1)
```

```
Enter a number : 5
```

In [54]:
```
reduce(lambda x,y: x*y, n_list)
```

Out[54]: 120

In [55]:
```
pairs = [(1, 'a'), (2, 'b'), (3, 'c')]
```

In [56]:
```
reduce(lambda cum, pair: cum+pair[0],pairs, 0)
```

Out[56]: 6