Python Programming

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

Academic Year : 2020-2021

FUNCTIONS - 5

Agenda:

1. Recursive Functions

2. Anonymous or Lambda Functions

5.RECURSIVE FUNCTIONS

Recursive Functions

A function that calls itself is known as Recursive Function.

Eg:

factorial(3) =3*factorial(2)

=3*2*factorial(1)

=3*2*1*factorial(0)

=3*2*1*1 =6

factorial(n)= n*factorial(n-1)

The main advantages of recursive functions are:

1. We can reduce length of the code and improves readability.

2. We can solve complex problems very easily. For example, Towers of Hanoi, Ackerman's Problem etc.,

Q 1. Write a Python Function to find factorial of given number with recursion.

```
def factorial(n):
    if n==0:
        result=1
    else:
        result=n*factorial(n-1)
    return result
print("Factorial of 0 is :",factorial(0))
print("Factorial of 4 is :",factorial(4))
print("Factorial of 5 is :",factorial(5))
print("Factorial of 40 is :",factorial(40))
```

```
Factorial of 0 is : 1
Factorial of 4 is : 24
Factorial of 5 is : 120
Factorial of 40 is : 815915283247897734345611269596115894272000000000
```

6. ANONYMOUS FUNCTIONS

Anonymous Functions

- Sometimes we can declare a function without any name, such type of nameless functions are called anonymous functions or lambda functions.
- The main purpose of anonymous function is just for instant use(i.e., for one time usage).

Normal Function:

- ✓ We can define by using **def** keyword.

```
def squareIt(n):
    return n*n
```

lambda Function:

- ✓ We can define by using lambda keyword

```
lambda n:n*n
```

Syntax of lambda Function:

lambda argument_list : expression

Note:

- By using Lambda Functions we can write very concise code so that readability of the program will be improved.

Q 1. Write a program to create a lambda function to find square of given number.

```
s=lambda n:n*n
print("The Square of 4 is :",s(4))
print("The Square of 5 is :",s(5))
```

```
The Square of 4 is : 16
The Square of 5 is : 25
```

Q 2. Write a program to create a Lambda function to find sum of 2 given numbers.

```
s=lambda a,b:a+b
print("The Sum of 10,20 is:",s(10,20))
print("The Sum of 100,200 is:",s(100,200))
```

```
The Sum of 10,20 is: 30
The Sum of 100,200 is: 300
```

Q 3.Write a program to create a Lambda Function to find biggest of given values.

```
s=lambda a,b:a if a>b else b
print("The Biggest of 10,20 is:",s(10,20))
print("The Biggest of 100,200 is:",s(100,200))
```

```
The Biggest of 10,20 is: 20
The Biggest of 100,200 is: 200
```

Note:

- Lambda Function internally returns expression value and we are not required to write return statement explicitly.
- Sometimes we can pass a function as argument to another function. In such cases lambda functions are best choice.
- We can use lambda functions very commonly with filter(),map() and reduce() functions, because these functions expect function as argument.

1. filter() function:

- We can use filter() function to filter values from the given sequence based on some condition.
- For example, we have 20 numbers and if we want to retrieve only even numbers from them.

Syntax:

filter(function, sequence)

Where,

- **function argument** is responsible to perform conditional check.
- **sequence** can be list or tuple or string.

Q 1. Program to filter only even numbers from the list by using filter() function.

Without lambda Function:

```
def isEven(x):
    if x%2==0:
        return True
    else:
        return False
l=[0,5,10,15,20,25,30]
l1=list(filter(isEven,l))
print(l1)
```

```
[0, 10, 20, 30]
```

With lambda Function:

```
l=[0,5,10,15,20,25,30]
l1=list(filter(lambda x:x%2==0,l))
print(l1)
l2=list(filter(lambda x:x%2!=0,l))
print(l2)
```

Output:

```
[0, 10, 20, 30]
[5, 15, 25]
```

2. map() function:

- For every element present in the given sequence, apply some functionality and generate new element with the required modification. For this requirement we should go for **map()** function.

Syntax:

```
map( function, sequence)
```

- The function can be applied on each element of sequence and generates new sequence.

Eg 1: For every element present in the list perform double and generate new list of doubles.

Without lambda

```
l=[1,2,3,4,5]
def doubleIt(x):
    return 2*x
l1=list(map( doubleIt, l))
print(l1)
```

Output:

[2, 4, 6, 8, 10]

With lambda

```
l=[1,2,3,4,5]
l1=list(map(lambda x:2*x,l))
print(l1)
```

Output:

```
[2, 4, 6, 8, 10]
```

Eg 2: Find square of given numbers using map() function.

```
l=[1,2,3,4,5]
l1=list(map(lambda x:x*x,l))
print(l1)
```

Output:

[1, 4, 9, 16, 25]

- We can apply map() function on multiple lists also. But make sure all list should have same length.

Syntax:

```
map(lambda x,y:x*y,l1,l2))
```

x is from l1 and y is from l2

Eg:

```
l1=[1,2,3,4]
l2=[2,3,4,5]
l3=list(map(lambda x,y:x*y,l1,l2))
print(l3)          ➔ [2, 6, 12, 20]
```

Eg:

```
l1=[1,2,3,4,5,6,7]          # The extra elements will be ignored
l2=[2,3,4,5]
l3=list(map(lambda x,y:x*y,l1,l2))
print(l3)          ➔ [2, 6, 12, 20]
```

3. reduce() function:

- reduce() function reduces sequence of elements into a single element by applying the specified function.

Syntax:

reduce(function, sequence)

Note :

- reduce() function present in **functools module** and hence we should write import statement.

Eg 1:

```
from functools import *
l=[10,20,30,40,50]
result=reduce(lambda x,y:x+y,l)
print(result)
```

➔ 150

Eg 2:

```
from functools import *
l=sum([10,20,30,40,50])
# result=reduce(lambda x,y:x*y,l)
print(l)
```

➔150

Eg 3:

```
from functools import *
l=[10,20,30,40,50]
result=reduce(lambda x,y:x*y,l)
print(result)
```

➔12000000

Eg 4:

```
from functools import *
result=reduce(lambda x,y:x+y,range(1,101))
print(result)
```

➔ 5050

Note:

- In Python every thing is treated as object. (Except keywords).
- Even functions also internally treated as objects only.

Eg:

```
def f1():
    print("Hello")
print(f1)
print(id(f1))
print(id(print))        # print is also an object
print(id(id))           # id is also an object
```

```
<function f1 at 0x000001697E535598>
1552602584472
1552519043040
1552519042104
```

Any question?

If you try to practice programs yourself, then you will learn many things automatically

Spend few minutes and then enjoy the study

Thank You