Python Lambda Functions

Python lambda (Anonymous Functions)

- In Python, anonymous function means that a function is without a name.
- This function can have any number of arguments but only one expression, which is evaluated and returned.
- Lambda functions are syntactically restricted to a single expression.

Syntax:

lambda <arguments> : <expression>

Lambda Functions Syntax

lambda arguments : expression

x = **lambda** a:a**2

print(“square = ”,x(4))

---

x = lambda a,b:a*b

print("The total mul = ",x(4,5))

Difference…

```
def volume(l):
    return l**3
  print(volume(5))

g = lambda x,y: x+y**3
print(g(5,6))
```

Example…

```
def table(n):
    return lambd a:a*n;
n = int(input("Enter the number?"))
table = table(n)
for i in range(1,11):
    print(n,"X",i,"=",table(i));
```

Filtering

- The function filter(function, list) offers an elegant way to filter out all the elements of a list, for which the *function* returns True.

- The function filter(f,l) needs a function f as its first argument. f returns a Boolean value, i.e. either True or False.

- This function will be applied to every element of the list *l*. Only if f returns True will the element of the list be included in the result list.

filter(<boolean_condition>,list)

Filtering Example

```
#program to filter out the list which is divisible by 3
Lists = [1,2,3,4,10,123,22]
div = list(filter(lambda x:(x%3 == 0),Lists))
even = list(filter(lambda x:(x%2 == 0),Lists))

print(div)
print(even)
```

Map function

The map() function in Python takes in a function and a list as argument.

The function map(s,l) needs a function f as its first argument. s returns sequence value.

Syntax:

map(<lambda_function>,list)

Map function

```
List = [1,2,3,4,10,123,22]
new_list = list(map(lambda x:x**3,List))
print(new_list)

Lists = [1,2,3,4,5,6,7,8,9,10]
mul = list(map(lambda x:x*5,Lists))
print(mul)
```

Reduce() function

- The reduce() function in Python takes in a function and a list as argument.
- The function is called with a lambda function and a list and a new reduced result is returned.
- This performs a repetitive operation over the pairs of the list.

- reduce(<operation Function>,list)

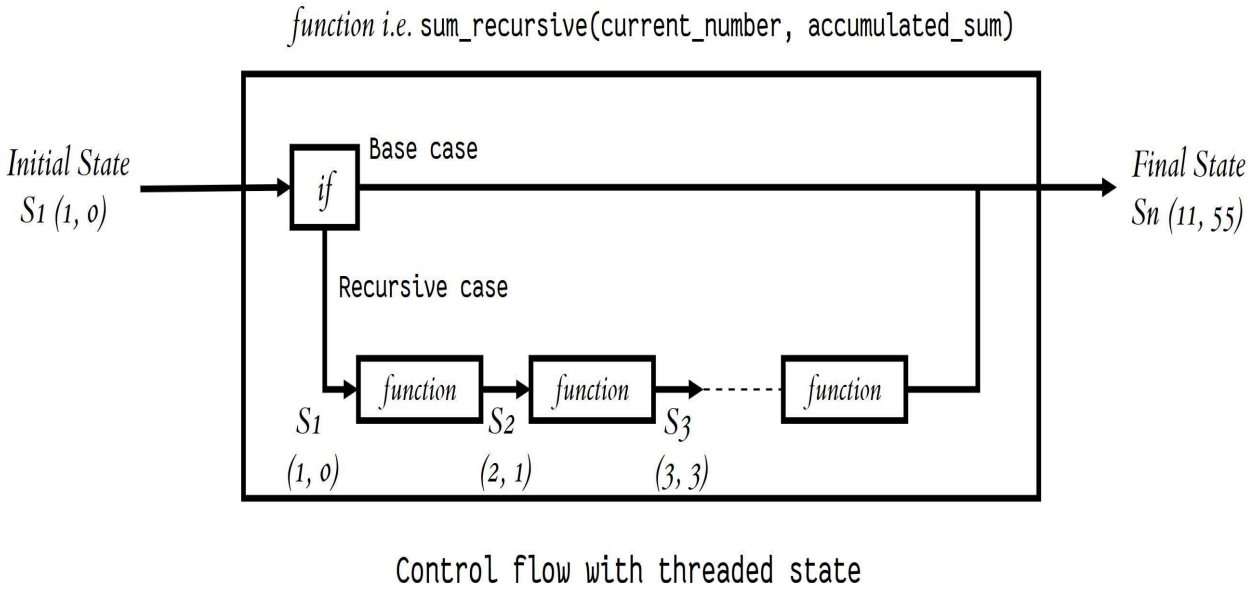
Example…

```
from functools import reduce
lists = [5,5,6,9,3,4,8,8,3]
sum = reduce((lambda x, y: x + y), lists)
print (sum)
-------------------------------------------------
fun = lambda x,y: x if (x > y) else y
reduce(fun, [12,43,32,67,98,23,34])
```

Recursive Function

Recursive Functions in Python

- Recursion is the process of defining something in terms of itself.
- a function calls itself one or more times in its body



Control flow with threaded state

Recursive Functions in Python

```
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)
print(factorial(5))
```

Recursive Functions in Python

```
def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)

fib(10)
```

Recursive Functions in Python

```
def sum_n(n):
    if n== 0:
        return 0
    else:
        return n + sum_n(n-1)
sum_n(5)
```