ANONYMOUS FUNCTIONS / LAMBDAS

- Function without a name.
- Keyword **lambda**.
- Also known as **Lambda Functions.**
- No more than one line.
- Can have multiple arguments with one expression.
- Return the result implicitly.

Parts of Lambda Functions

- The lambda Keyword
- The parameters
- The function body

Normal Function:

```
def square(x):
    return x*x
```

Syntax

lambda argument_list : expression

Example:

```
lambda  x:x*x

add= lambda x,y : x+y
print(add(3,4))  #Function Call
```

Use of Lambda Function in Python

- When we require a nameless function for a short period of time.

IIFE in Python Lambda

- Immediately invoked function execution (IIFA).
- It is callable as soon as it is defined.
- In IDLE , **(lambda x:x+x)(2)** – get the result immediately.

Using Lambdas with filter () function

- To filter out the elements of a sequence.
- Takes in a function and a list as an argument.
- **Syntax: filter (function, sequence)**
- It is useful to filter out the elements of a sequence depending on the result of the function.

Example:

```
#Program to filter out only even items from a list using lambda
list1=[1,4,6,7,2,34,56,12,17,3,55]
new_list=list(filter(lambda x : ((x%2)==0),list1))
print(new_list)

sequence=[10,40,6,7,2,4,11,1,0]
filtered_result=filter(lambda x : x>4,sequence)
print(list(filtered_result))
```

Using Lambdas with map () Function

- Changes the elements
- The map() function is similar to filter() function but it acts on each element of the sequence and perhaps changes the elements.
- **Syntax: map (function, sequence)**

Example:

```
#To double each element in a list using map
list1=[1,4,3,8,12,32,7]
new_list=list(map(lambda x:x*2,list1))
print(new_list)
```

Using Lambdas with reduce() function

- Reduces the sequence of elements to a single value by processing the elements according to a function supplied.
- **Syntax: reduce (function, sequence)**

Working

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

Example:

```
#calculate the products of a list
from functools import *
lst=[1,2,3,4,5]
result=reduce(lambda x,y:x*y,lst)
print(result)

#To calculate the sum of m=numbers from 1 to 50
from functools import reduce
sum=reduce(lambda a,b:a+b,range(1,51))
print(sum)
```

Why use Lambdas Functions?

- Compact syntax
- Functional Programming
- Allows you to provide a function as a parameter to another function.

When should you not use Lambdas Functions?

- You should never write complicated lambda functions in a production environment.
- Very difficult for the coder to decrypt it.

LAMBDAS VS. REGULAR FUNCTIONS

Lambdas	Functions
Lambda x : x+x	Def (x): Return x+x
One expression	Multiple expression
Do not have a name	Must have a name and signature
Do not contain return statement	Functions which need to return value should include a return statement.