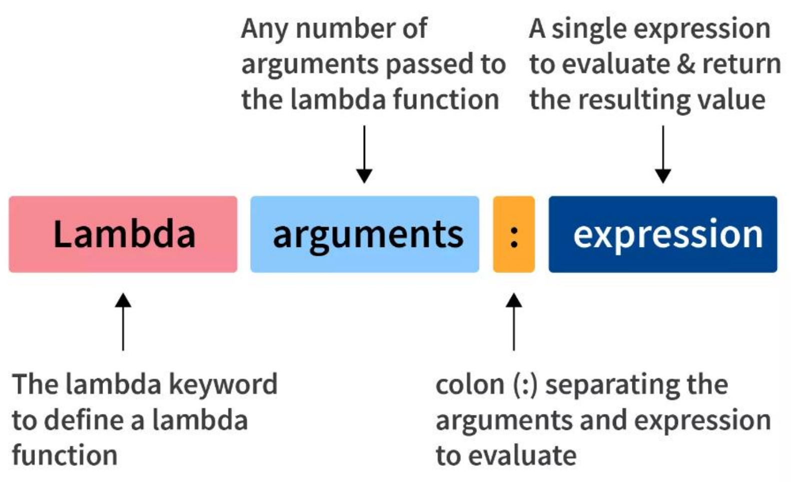
LECTURE TWELEVE

Python Lambda Function (Anonymous Functions)

- In Python, Lambda function is a small anonymous function defined using the lambda keyword.
- Lambda functions, are one-line functions without a name. lambda functions can have many numbers of arguments but only single expression.
- Lambda functions are handy for short, throwaway tasks that are not meant to be reused elsewhere in your code.
- unlike normal functions as follow:

Normal Function	Lambda Function
Regular Function use **def** keyword	Lambda Function use **lambda** keyword
return is required in Regular Function	return is not required in Lambda Function
Return element will be python data-type	Return element will be function object
Execution time is slower	Execution time is faster

Syntax of Python Lambda


Any number of arguments passed to the lambda function
A single expression to evaluate & return the resulting value
Lambda
arguments
:
expression
The lambda keyword to define a lambda function
colon (:) separating the arguments and expression to evaluate

Lambda Function in Python Use Cases

Lambda functions in Python are versatile and can be used in various scenarios. Here are some specific use cases with examples and their outputs:

Lambda function with Single argument

```
square = lambda a : print("Square of given value is ,a*a)
square(9)
```

Output

```
Square of given value is  81
```

Lambda function with Multiple argument

```
multiplication = lambda val1,val2:print("Multiplication is ",val1*val2)
multiplication(5,9)
```

Output

```
Multiplication is  45
```

Lambda function with Conditional statement

```
max_number = lambda x, y: x if x > y else y
print(max_number(5, 7))
print(max_number(10, 3))
```

Output

```
Multiplication is  45
7
10
```

Lambda function with Default Argument

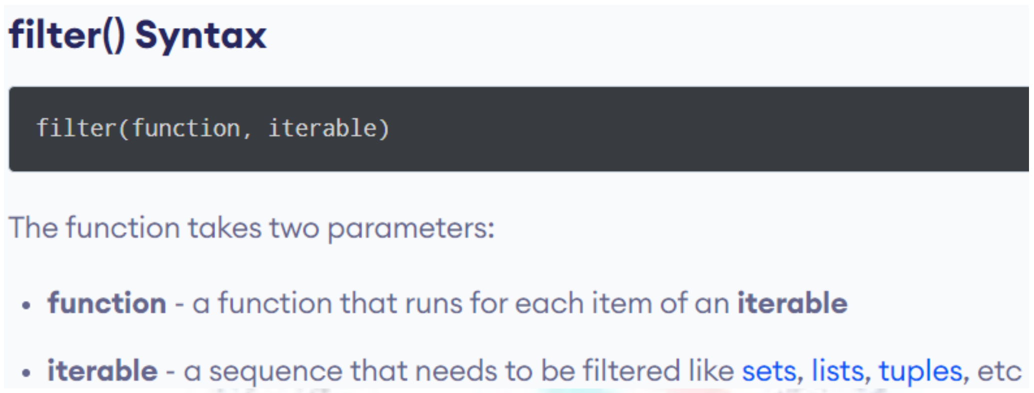
```
tf = lambda a,b,c=5:a+b-c
print(tf(15,10))
print(tf(15,10,15))
```

Output

```
20
10
```

Lambda Function with filter()

- Use a filter function to selects elements from an inerrable based on the result of a function.
- That function is called with every item in the list, and a new list is returned that contains items for which the function evaluates to True.



filter() Syntax

```
filter(function, iterable)
```

The function takes two parameters:

- **function** - a function that runs for each item of an **iterable**
- **iterable** - a sequence that needs to be filtered like sets, lists, tuples, etc

Eample(1): Write a Python program to use the filter function to filter out even numbers where numbers = [1, 2, 3, 4, 5, 6]

```
def check_even(number):
    if number % 2 == 0:
         return True
    else:
         return False
numbers = [1, 2, 3, 4, 5, 6]
even_numbers= list(filter(check_even, numbers))
print(even_numbers)
```

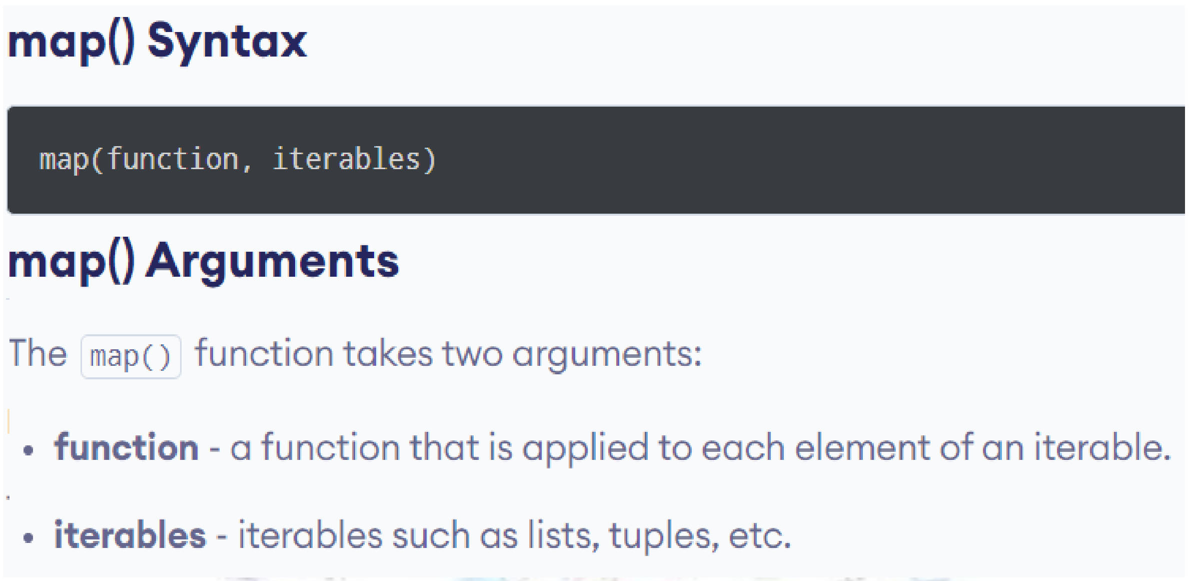
- **Lambda** function used as arguments with **filter()** function in Python takes in a function and a list as arguments.

Example(2): Write a Python program to use the filter function with lambda to filter out even numbers where numbers = [1, 2, 3, 4, 5, 6]

```
numbers = [1, 2, 3, 4, 5, 6]
even_numbers= list(filter(lambda x: x % 2==0, numbers))
print(even_numbers)
```

Map Function

- Use a map function to Returns the specified iterator with the specified function applied to each item.



map() Syntax

```
map(function, iterables)
```

map() Arguments

The `map()` function takes two arguments:

- **function** - a function that is applied to each element of an iterable.
- **iterables** - iterables such as lists, tuples, etc.

Example(3): Write a Python program to use the map function with normal function to double all the numbers in a list, where list = [1, 2, 3, 4]

```
def square(n):
    return n*n
numbers = [1, 2, 3, 4]
result =list(map(square, numbers))
print(result)
```

- **Lambda** function used as arguments with **map ()** function in Python also takes in a function and a list as arguments. That function is called for all the items in that list, and the latest list is returned, that contains items returned by that function.

Example(4): Write a Python program to use the map function with lambda to double all the numbers in a list, where list = [1, 2, 3, 4]

```
numbers = [1, 2, 3, 4, 5, 6]
doubles = list(map(lambda x: x * 2, numbers))
print(doubles)
```

- Multiple Iterables to the Map Function, we can pass multiple sequences to the map functions.

Example(4): Write a Python program to add two list using the map function where list1 = [9, 2, 30, 4] and list2 = [10, 9, 7, 4]

```
list1 = [9, 2, 30, 4]
list2 = [10, 9, 7, 4]
out=list(map(lambda x, y: x + y, list1, list2) )
print(out)
```

- We can use map() function to modify the data type.