

PYTHON LECTURE 24



Today's Agenda



User Defined Functions V

- The map() Function
- The filter() Function
- Using map() and filter() with Lambda Expressions



What Is map() Function?



- As we have mentioned earlier, the advantage of the lambda operator can be seen when it is used in combination with the **map()** function.
- map() is a function which takes two arguments:
 - o r = map(func, iterable)
- The first argument *func* is the **name of a function** and the second argument , *iterable* ,should be a **sequence** (e.g. a list , tuple ,string etc) or anything that can be used with *for* loop.
- map() applies the function func to all the elements of the sequence iterable



What Is map() Function?



• To understand this, let's solve a problem.

• Suppose we want to define a function called **square()** that can accept a number as argument and returns it's square.

• The definition of this function would be:

def square(num):
return num**2



What Is map() Function?



 Now suppose we want to call this function for the following list of numbers:

```
o mynums=[1,2,3,4,5]
```

One way to do this, will be to use a for loop

```
mynums=[1,2,3,4,5]
for x in mynums:
print(square(x))
```



Complete Code



```
def square(num): return num**2
```

```
mynums=[1,2,3,4,5]
for x in mynums:
print(square(x))
```

Output:

1

4

Q

16

25



Using map() Function



Another way to solve the previous problem is to use the map() function.

- The **map()** function will accept 2 arguments from us.
 - The **first** argument will be the **name of the function square**
 - o The **second** argument will be **the list mynums**.
- It will then apply the function **square** on every element of **mynum** and return the corresponding result as **map** object





```
def square(num):
    return num**2

mynums=[1,2,3,4,5]
result=map(square,mynums)
print(result)
Output:
```

<map object at 0x00000000029030F0>

- As we can observe, the return value of map() function is a map object
- To convert it into actual numbers we can pass it to the function list()





def square(num):
 return num**2

mynums=[1,2,3,4,5] my

sqrnum=list(result)

result=map(square,mynums)

print(squrnum)

def square(num):
return num**2

mynums=[1,2,3,4,5]

we can club the 2 lines in 1 line

sqrnum=list(map(square,mynums))

print(sqrnum)

Output:

[1, 4, 9, 16, 25]





To make it even shorter we can directly pass the **list()** function to the function **print()**

def square(num):

return num**2

```
mynums=[1,2,3,4,5]
print(list(map(square,mynums)))
```

Output:

[1, 4, 9, 16, 25]





In case we want to **iterate** over this **list**, then we can use **for loop def square(num)**:

return num**2

```
mynums=[1,2,3,4,5]
for x in map(square,mynums):
    print(x)
```

Output:



Exercise



• Write a function called inspect() that accepts a string as argument and returns the word EVEN if the string is of even length and returns it's first character if the string is of odd length

Now call this function for first 3 month names



Solution



```
def inspect(mystring):
    if len(mystring)%2==0:
        return "EVEN"
    else:
        return mystring[0]

months=["January","February","March"]
print(list(map(inspect,months)))
```

Output:

```
['J', 'EVEN', 'M']
```



What Is filter() Function?



- Like map(), filter() is also a function that is very commonly used in Python.
- The function **filter ()** takes 2 arguments:

filter(function, sequence)

- The **first argument** should be a **function** which must return a **boolean value**
- The **second argument** should be a **sequence** of **items**.
- Now the function filter() applies the function passed as argument to every item of the sequence passed as second argument.
- If the function returned **True** for that item, **filter()** returns that **item** as part of it's return value otherwise the **item** is **not returned**.



What Is filter() Function?



• To understand this, let's solve a problem.

• Suppose we want to define a function called **check_even()** that can accept a **number** as argument and return **True** if it is even, otherwise it should return **False**

• The definition of this function would be:

```
def check_even(num):
    return num%2==0
```



What Is filter() Function?



 Now suppose we have a list of numbers and we want to extract only even numbers from this list

```
o mynums=[1,2,3,4,5,6]
```

One way to do this, will be to use a for loop

```
mynums=[1,2,3,4,5,6]
for x in mynums:
    if check_even(x):
        print(x)
```



Complete Code



```
def check_even(num):
    return num%2==0
```

```
mynums=[1,2,3,4,5,6]
for x in mynums:
  if check_even(x):
    print(x)
```

Output:

2

4

6



Using filter() Function



• Another way to solve the previous problem is to use the **filter()** function .

- The **flter()** function will accept 2 arguments from us.
 - The first argument will be the name of the function check_even
 - The **second** argument will be **the list mynums**.
- It will then apply the function **check_even** on every element of **mynum** and if **check_even** returned **True** for that element then **filter()** will return that element as a part of it's return value otherwise that element will not be returned



Previous Code Using filter()



```
def check_even(num):
    return num%2==0
```

```
mynums=[1,2,3,4,5,6]
print(filter(check_even,mynums))
```

Output:

<filter object at 0x00000000029F3F60>

- As we can observe, the return value of filter() function is a filter object
- To convert it into actual numbers we can pass it to the function list()



Previous Code Using filter()



```
def check_even(num):
    return num%2==0
```

```
mynums=[1,2,3,4,5,6]
print(list(filter(check_even,mynums)))
```

Output:

[2, 4, 6]



Previous Code Using filter()



In case we want to **iterate** over this **list**, then we can use **for loop** as shown below:

```
def check_even(num):
    return num%2==0
```

```
mynums=[1,2,3,4,5,6]
for x in filter(check_even,mynums):
    print(x)
Output:
```

2 4 6



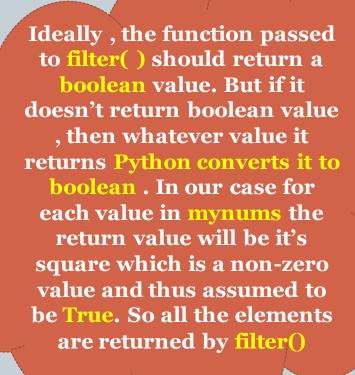


def f1(num):
return num*num

mynums=[1,2,3,4,5] print(list(filter(f1,mynums)))

Output:

[1,2,3,4,5]





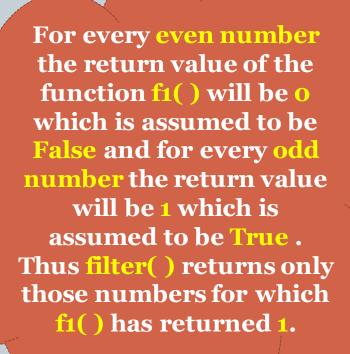


def f1(num):
return num%2

mynums=[1,2,3,4,5] print(list(filter(f1,mynums)))

Output:

[1,3,5]







```
def f1(num):
    print("Hello")
```

mynums=[1,2,3,4,5] print(list(filter(f1,mynums)))

Output:

Hello

Hello

Hello

Hello

Hello

Hello is displayed 5 times because the filter() function has called f1() function 5 times. Now for each value in mynums, since f1() has not returned any value, by default it's return value is assumed to be None which is a representation of False. Thus filter() returned an empty list.





def f1(num):
pass

mynums=[1,2,3,4,5] print(list(filter(f1,mynums)))

Output:

For each value in mynums, since ft() has not returned any value, by default it's return value is assumed to be
None which is a representation of False.
Thus filter() returned an empty list.





def f1():
pass

mynums=[1,2,3,4,5]
print(list(filter(f1,mynums))

The function filter() is trying to call f1() for every value in the list mynums. But since f1() is a non-parametrized function, this call generates

TypeError

Output:

TypeError: f1() takes 0 positional arguments but 1 was given





```
def f1():
    pass

mynums=[]
print(list(filter(f1,mynums)))

Output:
[]
```



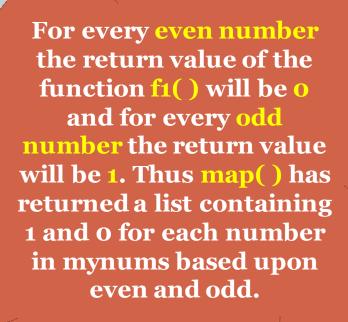


def f1(num):
return num%2

mynums=[1,2,3,4,5] print(list(map(f1,mynums)))

Output:

[1,0,1,0,1]







def f1(num): pass

mynums=[1,2,3,4,5]
print(list(map(f1,mynums))

Since ft() is not returning anything, so it's return value by default is assumed to be None and because map() has internally called ft() 5 times, so the list returned contains None 5 times

Output:

[None,None,None,None]





```
def f1():
    pass

mynums=[]
print(list(map(f1,mynums)))

Output:
[]
```



Using Lambda Expression With map() And filter()



 The best use of Lambda Expression is to use it with map() and filter() functions

 Recall that the keyword lambda creates an anonymous function and returns it's address.



Using Lambda Expression With map() And filter()



• So, we can pass this **lambda expression** as first argument to **map()** and **filter()** functions, since their first argument is the a **function object reference**

• In this way, we wouldn't be required to specially create a separate function using the keyword **def**



Using Lambdas With map()



def square(num): return num**2

To convert the above code using **lambda**, we have to do 2 changes:

- Remove the function square()
- 2. Rewrite this function as **lambda** in place of **first argument** while calling the function **map()**

Following will be the resultant code:

```
mynums=[1,2,3,4,5]
sqrnum=list(map(lambda num: num*num,mynums))
print(sqrnum)
```



Exercise



• Write a lambda expression that accepts a string as argument and returns it's first character

Now use this lambda expression in map() function to work on for first 3 month names



Solution



```
months=["January","February","March"]
print(list(map(lambda mystring: mystring[o],months)))
```

Output:

```
['J', 'F', 'M']
```



Exercise



• Write a lambda expression that accepts a string as argument and returns the word EVEN if the string is of even length and returns it's first character if the string is of odd length

Now use this lambda expression in map() function to work on for first 3 month names



Solution



```
months=["January","February","March"]
print(list(map(lambda mystring: "EVEN" if len(mystring)%2==0 else
mystring[0],months)))
```

Output:

```
['j', 'ÉVEN', 'M']
```



Using Lambdas With filter()



```
def check_even(num):
    return num%2==0
```

```
mynums=[1,2,3,4,5,6]
print(list(filter(check_even,mynums)))
```

To convert the above code using **lambda**, we have to same 2 steps as before.

Following will be the resultant code:

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



Exercise



• Write a lambda expression that accepts a character as argument and returns True if it is a vowel otherwise False

Now ask the user to input his/her name and display only the vowels in the name. In case the name does not contain any vowel display the message No vowels in your name



Solution



```
name=input("Enter your name:")
vowels=list(filter(lambda ch: ch in "aeiou" ,name))
if len(vowels)==0:
    print("No vowels in your name")
else:
    print("Vowels in your name are:",vowels)
```

Output:

```
Enter your name:sachin
Vowels in your name are: ['a', 'i']
```