Map
&
Filter

Python map()

Python map() applies a function on all the items of an iterator given as input. An iterator, for example, can be a list, a tuple, a set, a dictionary, a string, and it returns an iterable map object. Python map() is a built-in function.

Syntax:

```
map(function, iterator1,iterator2 ...iteratorN)
```

Parameters

Here are two important

- function: A mandatory function to be given to map, that will be applied to all the items available in the iterator.
- iterator: An iterable compulsory object. It can be a list, a tuple, etc. You can pass multiple iterator objects to map() function.

How map() function works?

The map() function takes two inputs as a function and an iterable object. The function that is given to map() is a normal function, and it will iterate over all the values present in the iterable object given.

For example, consider you have a list of numbers, and you want to find the square of each of the numbers.

The get the output, we need the function that will return the square of the given number. The function will be as follows:

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

The list of items that we want to find the square is as follows:

```
my_list = [2,3,4,5,6,7,8,9]
```

Now let us use map() python built-in function to get the square of each of the items in my_list.

The final code is as follows:

```
def square(n):
    return n*n
my_list = [2,3,4,5,6,7,8,9]
updated_list = map(square, my_list)
print(updated_list)
print(list(updated_list))
```

Output:

```
<map object at 0x0000002C59601748>
[4, 9, 16, 25, 36, 49, 64, 81]
```

The output of the map() function, as seen in the output, is a map object displayed on the screen as <map object at 0x0000002C59601748>.

You will have to iterate the output from the map using a for-loop or using list() method to get the final output. I have used list() in the code that displays the values inside the list given.

Using map() with a string as an iterator

We can also make use of map() on a string. In Python, a string acts like an array so we can easily use it inside the map().

In the example, we have a function myMapFunc() that takes care of converting the given string to uppercase. The function myMapFunc () is given to map() function.The map function will take care of converting the string given to uppercase by passing the string to myMapFunc().

```
def myMapFunc(s):
    return s.upper()
my_str = "welcome to guru99 tutorials!"
updated_list = map(myMapFunc, my_str)
print(updated_list)
for i in updated_list:
    print(i, end="")
```

Using map() with Lambda function

In Python, lambda expressions are utilized to construct anonymous functions. You will have to use the lambda keyword just as you use def to define normal functions.

So in the example, we are going to use the lambda function inside the map(). The lambda function will multiply each value in the list with 10.

Example:

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

Output:

```
<map object at 0x000000BD18B11898>
[20, 30, 40, 50, 60, 70, 80, 90]
```

Using Multiple Iterators inside map() function

Example 1: Passing two list iterators to map()

You can send more than one iterator i.e. a list, a tuple, etc. all at the same time to the map() function.

For example, if you want to add two lists. The same can be done using the map() function. We are going to make use of two lists my_list1 and my_list2.

In the example below, the first item in the my_list1 is added to the first item of my_list2. The function myMapFunc() takes in items of my_list1 and my_list2 and returns the sum of both.

Here is the working example of adding two given lists using map() function.

```
def myMapFunc(list1, list2):
    return list1+list2

my_list1 = [2,3,4,5,6,7,8,9]
my_list2 = [4,8,12,16,20,24,28]

updated_list = map(myMapFunc, my_list1,my_list2)
print(updated_list)
print(list(updated_list))
```

Summary

- Python map() is a built-in function that applies a function on all the items of an iterator given as input. An iterator, for example, can be a list, a tuple, a string, etc. and it returns an iterable map object.
- The map() function is going to apply the given function on all the items inside the iterator and return an iterable map object i.e a tuple, a list, etc.
- Python map() function is a built-in function and can also be used with other built-in functions available in Python.

The **filter() function** facilitates a functional approach to Python programming.
It takes as an argument a *function* and an *iterable* and
applies the passed function to each element of the iterable.

Once this is done, it returns an *iterable*.
The filter() function is similar to a for-loop in Python but is much fast as a built-in function.

Syntax

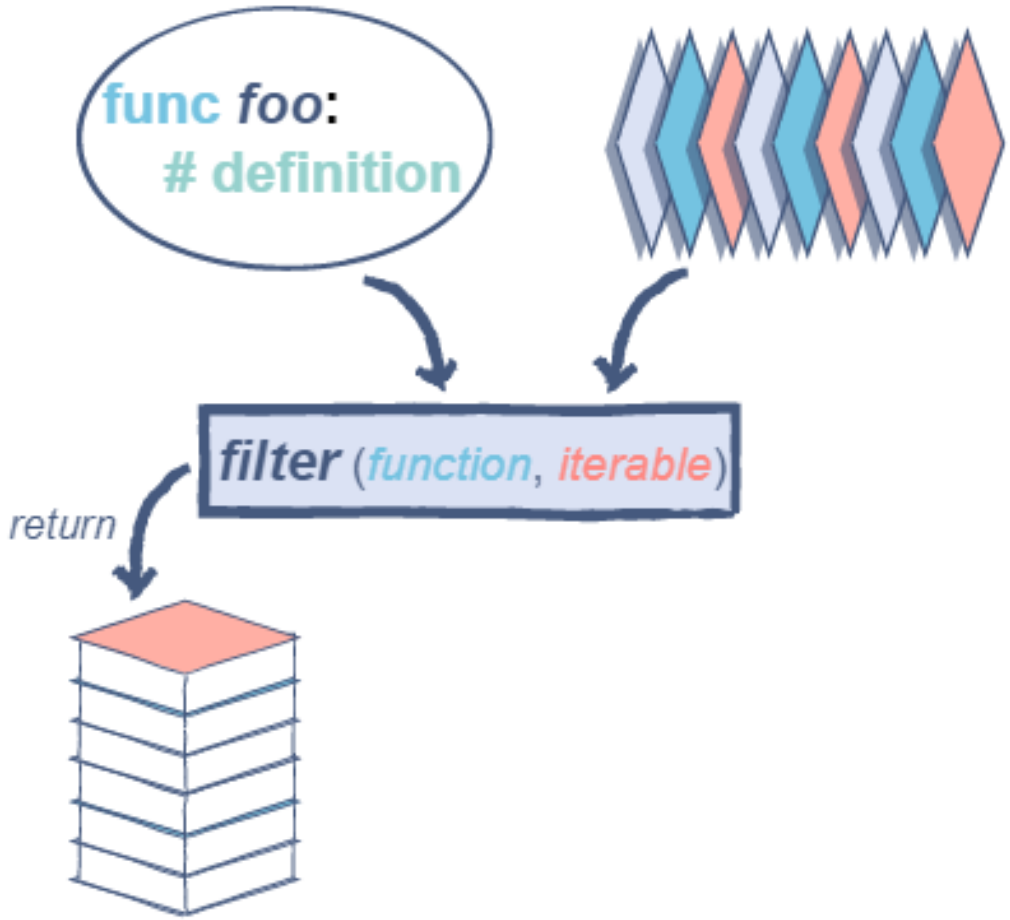
Here is the function signature for the `filter()` function in Python:

```
# function signature for the filter() method
returned_iterable = filter(function, iterable)
```

As described above, the `filter()` function takes the following two arguments as input:

- `function`: A valid, pre-defined function. This is a lambda function in most cases.
- `iterable`: This is an iterable object (e.g. list, tuple, dictionary).

Returned: An iterator to the *filtered* items' iterable object.



Examples

Let’s take a look at a couple of examples of how the filter() method works:

1. Using a lambda function

```
1  myList = [10, 25, 17, 9, 30, -5]
2  # Returns the elements which are multiples of 5
3  myList2 = list(filter(lambda n : n%5 == 0, myList))
4  print(myList2)
```

2. Using a pre-defined function

```
1  # Returns the elements which are multiples of 5
2  def multipleOf5(n):
3      if(n % 5 == 0):
4          return n
5  myList = [10, 25, 17, 9, 30, -5]
6  myList2 = list(filter(multipleOf5, myList))
7  print(myList2)
```

```python
# create function that returns True if the number passed in is even and False if it is not even
def is_even(num):
  return num % 2 == 0

# original list of numbers
list_of_nums = [1,2,3,4,5,6]

# new list that will contain only the even numbers from list_of_nums
list_of_even_nums = list(filter(is_even, list_of_nums))

print(list_of_even_nums) # output would be [2,4,6]
```

```python
letters = ['a', 'b', 'd', 'e', 'i', 'j', 'o']

# a function that returns True if letter is vowel
def filter_vowels(letter):
    vowels = ['a', 'e', 'i', 'o', 'u']
    return True if letter in vowels else False

filtered_vowels = filter(filter_vowels, letters)

# converting to tuple
vowels = tuple(filtered_vowels)
print(vowels)
```