The filter method and lambda functions in Python

Often times when working with iterable data structures such as arrays, lists, and tuples, we have to perform filtering operations over them. Filtering refers to taking out the values that match a particular condition. One can perform filtering by iterating over the iterable data structure value and value using a `for` loop or a `while` loop and checking if the element satisfies the condition or not.

Since this is a very common task, the Python programming language has an in-built method, `filter()`, for it. This method needs a function to perform filtration. Generally, `lambda` methods are considered for this job since they are straightforward to write. In this article, we will learn how to use the `filter()` method and `lambda` functions together in Python.

`filter()` method and `lambda` functions in Python

The `filter` method is an in-built method in Python programming language that is used to filter elements from an iterable that meet a certain condition. The condition is represented by a function, and the values for which the function returns `True` is considered in the result.

The `filter()` accepts two parameters; a function to use for filtering that returns a boolean value and an iterable object such as a `list` or a `tuple`.

The result of the `filter()` method is an iterator or `filter object`. We have to convert the iterator to a `list` or a `tuple` using the `list()` method or the `tuple()` method, respectively.

The `lambda` functions inline and anonymous functions in Python. These functions can be used along with the `filter` method.

The `lambda` functions have the following syntax.

```
lambda <parameters> : <expression>
```

Parameters should be comma-separated, and the expression should return a boolean result (`True` or `False`).

Let us understand how to use the two together using a simple example. Refer to the following Python code for the same.

```
array = [11, 23, 13, 4, 15, 66, 7, 8, 99, 10]
new_array = list(filter(lambda x : x <= 20, array))
print("Old Array:", array)
print("New Array:", new_array)
```

Output:

```
Old Array: [11, 23, 13, 4, 15, 66, 7, 8, 99, 10]
New Array: [11, 13, 4, 15, 7, 8, 10]
```

The above Python code filters all the values from the list of integer, `array`, that are less than or equal to `20`. Each value of the list is passed to the `lambda` function, and if it returns `True`, the value is added to the result; otherwise, not. Once the result is obtained in the form of an iterator, it is converted to a list using the in-built `list()` method. Lastly, both the arrays are printed to the console.

Examples

Following are some examples to understand the usage of the `filter()` method and the `lambda` functions together.

Filter all the even numbers from a list of integers

```
array = [1, 2, 3, 4, 5, 66, 77, 88, 99, 100]
new_array = list(filter(lambda x : x % 2 == 0, array))
print("Old Array:", array)
print("New Array:", new_array)
```

Output:

```
Old Array: [1, 2, 3, 4, 5, 66, 77, 88, 99, 100]
New Array: [2, 4, 66, 88, 100]
```

Filter all the odd numbers from a list of integers

```
array = [1, 2, 3, 4, 5, 66, 77, 88, 99, 100]
new_array = list(filter(lambda x : x % 2 == 1, array))
print("Old Array:", array)
print("New Array:", new_array)
```

Output:

```
Old Array: [1, 2, 3, 4, 5, 66, 77, 88, 99, 100]
New Array: [1, 3, 5, 77, 99]
```

Filter all strings of length 5 from a list of strings

```
array = ["hello", "python", "world", "walking", "sleep", "shelter", "food", ]
new_array = list(filter(lambda x : len(x) == 5, array))
print("Old Array:", array)
print("New Array:", new_array)
```

Output:

```
Old Array: ['hello', 'python', 'world', 'walking', 'sleep', 'shelter', 'food']
New Array: ['hello', 'world', 'sleep']
```

Filter all the numbers in the range `[10, 20]` from a list of integers

```
array = [11, 23, 13, 4, 15, 66, 7, 8, 99, 10]
new_array = list(filter(lambda x : 10 <= x <= 20, array))
print("Old Array:", array)
print("New Array:", new_array)
```

Output:

```
Old Array: [11, 23, 13, 4, 15, 66, 7, 8, 99, 10]
New Array: [11, 13, 15, 10]
```