Python Lambda/Anonymous Function

In Python, a lambda function is a special type of function without the function name. For example,

```
lambda : print('Hello World')
```

Here, we have created a lambda function that prints `'Hello World'`.

Before you learn about lambdas, make sure to know about Python Functions.

Python Lambda Function Declaration

We use the `lambda` keyword instead of `def` to create a lambda function. Here's the syntax to declare the lambda function:

```
lambda argument(s) : expression
```

Here,

- `argument(s)` - any value passed to the lambda function
- `expression` - expression is executed and returned

Let's see an example,

```
greet = lambda : print('Hello World')
```

Here, we have defined a lambda function and assigned it to the variable named *greet*.

To execute this lambda function, we need to call it. Here's how we can call the lambda function

```
# call the lambda
greet()
```

The lambda function above simply prints the text `'Hello World'`.

Note: This lambda function doesn't have any argument.

Example: Python Lambda Function

```
# declare a lambda function
greet = lambda : print('Hello World')

# call lambda function
greet()

# Output: Hello World
```

In the above example, we have defined a lambda function and assigned it to the *greet* variable.

When we call the lambda function, the print() statement inside the lambda function is executed.

Python lambda Function with an Argument

Similar to normal functions, a `lambda` function can also accept arguments. For example,

```
# lambda that accepts one argument
greet_user = lambda name : print('Hey there,', name)

# lambda call
greet_user('Delilah')

# Output: Hey there, Delilah
```

In the above example, we have assigned a lambda function to the *greet_user* variable.

Here, `name` after the `lambda` keyword specifies that the lambda function accepts the argument named `name`.

Notice the call of the lambda function,

```
greet_user('Delilah')
```

Here, we have passed a string value `'Delilah'` to our lambda function.

Finally, the statement inside the lambda function is executed.

Frequently Asked Questions

How to use the lambda function with filter()?

The filter() function in Python takes in a function and an iterable (lists, tuples, and strings) as arguments.

The function is called with all the items in the list, and a new list is returned, which contains items for which the function evaluates to `True`.

Let's see an example,

```
# program to filter out only the even items from a list
my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(filter(lambda x: (x%2 == 0) , my_list))

print(new_list)

# Output: [4, 6, 8, 12]
```

Here, the `filter()` function returns only even numbers from a list.

How to use the lambda function with map()?

The map() function in Python takes in a function and an iterable (lists, tuples, and strings) as arguments.

The function is called with all the items in the list, and a new list is returned, which contains items returned by that function for each item.

Let's see an example,

```
# Program to double each item in a list using map()

my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(map(lambda x: x * 2 , my_list))

print(new_list)

# Output: [2, 10, 8, 12, 16, 22, 6, 24]
```

Here, the `map()` function doubles all the items in a list.

Also Read:

- Python List Comprehension

Table of Contents

- Example: Python lambda Function
- Python lambda Function with an Argument