PES UNIVERSITY

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Functional Programming-Lambda & Map

Prof. Sindhu R Pai
PCPS Theory Anchor - 2024
Department of Computer Science and Engineering

Functional Programming

- Functional programming is a programming paradigm where programs are constructed by applying and composing functions.
- It is basically a study of computations with functions which uses pure functions, recursive functions, nested functions ,lambda functions etc.
- The ability of functional programming languages to treat functions as values and pass them to functions as parameters make the code more readable and easily understandable.
- They depend only on the input given to them, and the output will be the return value. Their function signature gives all the information about them i.e. their arguments and return type etc.

Functional Programming

- Higher order functions are the functions that take other functions as arguments and they can also return functions. (Ex: Callback Functions)
- They are deterministic in nature .
- They can be understood very easily as they don't change states of any variables.
- Testing and debugging is easier. They use immutable values
- Offers better modularity with a shorter code
- Increased productivity of the developer
- It adopts lazy evaluation which avoids repeated evaluation because the value is evaluated and stored only when it is needed.
- Example: Functional Constructs like Lazy Map & Lists ,tuples, sets etc.

The keyword - lambda

PES UNIVERSITY

Lambda (Anonymous function/Nameless Function/ Throw away function creation)

- Lambda functions are functions without the name.
- Its also called as anonymous functions or throw away functions.
- It can have any number of arguments but can have only one expression.
- The expression is executed and returned (Implicit return)
- Lambda functions are used when function objects are required.
- Lambda functions are used when you need a particular function for a short period of time.

The keyword - lambda

PES UNIVERSITY

Advantages:

- Lambda functions in Python are very useful in defining the in-line functions.
- Most suitable when it is used as a part of another function(especially with map, filter and reduce functions)
- Less Code (Concise code)

The keyword - lambda

PES UNIVERSITY

Syntax: **lambda input arguments: expression / Output**

- **lambda** is a keyword used to declare lambda function.
- **Input arguments:** It can have multiple arguments (if there are more than one arguments, then separate the arguments by a comma).
- **Expression or output:** Expression gets evaluated and results are returned

Examples

PES UNIVERSITY

Example1: To find the square of a number (Comparison between normal and lambda function)

Normal Function	Lambda Function
def square(n): return n*n print(square(4))	square=lambda n : n*n print(square(4)) #Function Call

Examples

PES UNIVERSITY

Example 2: To find the greatest of two numbers using the relational operator.

```
maximum=lambda a,b : a if a>b else b    #max(a,b) can also be used
print("The maximum is ",maximum(6,7))
print(maximum,type(maximum))
```

Output:

```
The maximum is  7
<function <lambda> at 0x00000217A6F93550> <class 'function'>
```

Examples

The power of lambda is better understood when you use them as an anonymous function inside another function.

Example 3

Suppose we have a function definition that takes one argument, and that argument will be multiplied with an unknown number.

```
def myfunc(n):
        return lambda a : a * n

double_N = myfunc(2)

print(double_N(12))
```

Output:24

Examples

We can use the same function definition to make two functions (doube_N and triple_N functions) in the same program.

Example 4

```
def myfunc(n):
    return lambda a : a * n

double_N = myfunc(2)
triple_N = myfunc(3)

print(double_N(15))
print(triple_N(15))
```

Output

```
30
45
```

Map() Function

Python map() function is a built in function which is utilized to apply a function to each element of an iterable (like a list or a tuple) and returns a new iterable object.

Syntax: map(function, iterables)

1. The map() function takes two arguments: a function and iterables.
2. The first argument for the map function should be a callable which takes one argument – could be a free function, function of a type, user defined function, or a lambda function.
3. This function will be applied to every element of the iterable, and creates a map object which is also an iterable.

Map() Function

PES UNIVERSITY

Working:

- The map function causes iteration through the iterable, and applies the callable on each element of the iterable
- All these happen only conceptually as the map function is lazy and the above said operations happen only after the map object is iterated.
- We can force an iteration of the map object by passing the map object as an argument for the list or set constructor.
- We can pass one or more iterable to the map() function.

Map() Function

PES UNIVERSITY

Lazy and Eager objects:

Under lazy object creation, the object is created when it is needed whereas in the case of eager object creation, an object is created as soon it is instantiated. A lazy object representing a function would be quick to create, and populate itself with information when that information is requested. Built-in functions like range(),map(),zip() and open() functions are basically lazy function objects.

Example:

As map object can be iterated over, the computation will be done for only one item in each loop. In contrast, when we use list, which is an eager object, it basically computes all the values at one time.

Lazy evaluation can be a powerful technique for optimizing code and improving performance, but it is only sometimes necessary or appropriate.

Map() Function

PES UNIVERSITY

Choosing between lazy and eager evaluation

- When deciding which evaluation strategy to use for a given programming problem, there is no one-size-fits-all answer. The choice depends on the nature of the problem, the characteristics of the data, the requirements of the solution, and the preferences of the programmer.
- However, some general guidelines can be followed to help make an informed decision. For instance, lazy evaluation should be used when you want to delay or avoid computations that are not needed, save memory by avoiding intermediate results, create and manipulate infinite or recursive data structures, or explore multiple possibilities without committing to one.
- On the other hand, eager evaluation is preferable when you want to perform computations as soon as possible, exploit parallelism or concurrency, or ensure predictable and consistent performance.

Map() Function

PES UNIVERSITY

Example 1 To double all numbers using map and lambda

```
numbers = (1, 2, 3, 4)
result = map(lambda x: x + x, numbers)
print(list(result))
```

Output:

```
<class 'map'>
[2, 4, 6, 8]
```

Map() Function

PES UNIVERSITY

Example 2 – To add two lists using map and lambda

```
num1 = [4, 2, 5]
num2 = [6, 8, 9]

result = map(lambda x, y: x + y, num1, num2)
print(list(result))
```

Output:

```
[10, 10, 14]
```

Map() Function

Example 3 – Function that doubles even numbers present in the list.

```
def double_num(n):
    if n% 2 == 0:
        return n * 2
    else:
        return n

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

# Use map to apply the function to each element in the list
result = list(map(double_num, numbers))
 print("The modified list is ",result)
```

Output: The modified list is [1, 4, 3, 8, 5]

THANK YOU

Department of Computer Science and Engineering

Dr. Shylaja S S, Director, CDSAML & CCBD, PESU
Prof. Sindhu R Pai – sindhurpai@pes.edu
Prof. C N Rajeswari