PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Prof. C N Rajeswari, Prof. Sindhu R Pai
Department of Computer Science
and Engineering

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Unit - 4: Functional and Object Oriented programming

Functional Programming-Lambda & Map

Prof. C.N.Rajeswari , Prof. Sindhu R Pai

Department of Computer Science and Engineering

Functional Programming

PES UNIVERSITY

- 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.

Lambda functions

PES UNIVERSITY

Lambda function(Anonymous function/Nameless Function)

- 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.

Lambda Functions:

PES UNIVERSITY

Advantages of Lambda functions:

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)

Lambda Function:

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

Lambda Functions

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

Lambda Functions

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'>
```

Lambda Functions

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

Lambda Functions

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

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

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

Team Python - 2022

Department of Computer Science and Engineering

Contact Email ID's:

sindhurpai@pes.edu

cnrajeswari@pes.edu

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Prof. C N Rajeswari, Prof. Sindhu R Pai
Department of Computer Science
and Engineering

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Unit - 4: Functional and Object Oriented programming

Filter and Reduce Functions

Prof. C.N.Rajeswari , Prof. Sindhu R Pai

Department of Computer Science and Engineering

Filter() Function

Filter function filters the given sequence with the help of a function that tests each element in the sequence to be true or not.

There are cases where we want to remove a few elements of the input iterable. Then we make use of the filter function.

Syntax: filter(*function*, *sequence/iterable*)
function: function that tests if each element of a sequence is true or not.

sequence: The sequence which needs to be filtered, it can be sets, lists, tuples, or containers of any iterators.

Returns: an iterator that is already filtered.

Filter() Function

The filter function has the following characteristics.

- Input : an iterable of some number of elements (say n)
- Output: a lazy iterable of 0 to n elements (between 0 and n)
- - Elements in the output: apply the callback on each element of the iterable – if the function returns true, then the input element is selected otherwise the input element is rejected.

Filter() function

PES UNIVERSITY

Example 1 Program to list the marks which are greater than 70

```
marks = [55, 78, 90, 87, 65, 45]

def myFunc(m):
  if m <70 :
    return False
  else:
    return True

Distinction = list(filter(myFunc, marks))
print("Students with marks greater than 70 are",Distinction)
```

Output:

Students with marks greater than 70 are [78, 90, 87]

Filter() Function

PES UNIVERSITY

Example 2 – Function that filters vowels

```
def fun(char):
    letters = ['a', 'e', 'i', 'o', 'u']
    if (char in letters):
        return True
    else:
        return False

characters = ['h', 'e', 'l', 'l', 'o', 'w', 'o', 'r','l','d']
 vowels = list(filter(fun, characters))
print('The filtered letters are:',vowels)
```

Output

```
The filtered letters are: ['e', 'o', 'o']
```

Filter() Functions

Example 3 –Use of lambda function to filter out the odd and even numbers from a list.

```
sequence = [0,1, 1, 2, 3, 5, 8,19]
# result contains list of odd numbers
result = filter(lambda x: x % 2 != 0, sequence)
print("The odd number list is",list(result))

# result contains list of even numbers
result = filter(lambda x: x % 2 == 0, sequence)
print("The even number list is",list(result))
```

Output:

```
The odd number list is [1, 1, 3, 5, 19]
The even number list is [0, 2, 8]
```

Filter() Function

PES UNIVERSITY

Example 4: Function to check whether a number is a multiple of 3 using Lambda function in filter function

```
#def is_multiple_of_3(num):
    return num % 3 == 0

# Create a list of numbers to filter
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
result = list(filter(lambda x: is_multiple_of_3(x), numbers))
print("The list of multiples of 3 is", result)
```

Output:

The list of multiples of 3 is [3, 6, 9]

Filter() Function

Example 5 : To pickup all words whose length exceeds 5 using map,filter and Lambda function.

```
Names=[ 'Ram', 'Tejas', 'Aditya', 'Ravi', 'Dinesh', 'Raghu' ]
 #finds all names whose length exceeds 5 and converts them to uppercase
print(list(map(str.upper, filter(lambda name : len(name) > 5, Names))))
```

Output:

```
['ADITYA', 'DINESH']
```

reduce() Function

reduce() is a function that applies a given function to the elements of an iterable, reducing them to a single value. This function is defined in "**functools**" module.

Syntax

functools.reduce(function, iterable[, initializer])

- The **function argument** is a function that takes two arguments and returns a single value. The first argument is the accumulated value, and the second argument is the current value from the iterable.
- The **iterable** argument is the sequence of values to be reduced.
- The optional initializer argument is used to provide an initial value. If no initializer is specified, the first element of the iterable is used as the initial value.

reduce() Function

PES UNIVERSITY

Working of reduce function:

- At first step, first two elements of the sequence are picked and the result is obtained.
- The same function is applied to the previously attained result and the number just succeeding the second element and the result is again stored.
- This process continues till no more elements are left in the container.
- There will be n – 1 calls if no initializer is specified.(n is the number of elements in the input iterable)
- The final result is returned as a single value.

reduce() Function

Example 1.To find the factorial of a number using reduce() function

```
import functools
n = 5
print("The factorial is ",functools.reduce(int.__mul__ , range(1, n + 1)))
```

Output:

```
The factorial is  120
```

Example 2. To find the sum of first 10 numbers using reduce() function

```
import functools
print(functools.reduce(int.__add__, range(10)))
```

output

```
The sum of first 10 numbers is  55
```

reduce() Function

Example 3.To find the product of numbers with 100 as the initial value.

```
def product(x, y):
        print("product : ", x, y)
        return x * y
print("The product with 100 as initial value
is",functools.reduce(product, [11, 22, 33, 44], 100))
```

Output:

```
product :  100 11
product :  1100 22
product :  24200 33
product :  798600 44
The product with 100 as initial value is 35138400
```

reduce()Function

PES UNIVERSITY

Example4: To find the sum and maximum temperature

```
import functools
temperature = [22.5, 24.6, 26, 32, 27.5]
 # using reduce to compute sum of temperature
sum=functools.reduce(lambda a, b: a+b, temperature)
print("The average temperature is ", sum/5)
 # using reduce to compute maximum temperature in the list
print("The maximum temperature is : ", end="")
print(functools.reduce(lambda a, b: a if a > b else b,
temperature))
```

Output:

```
 The average temperature is  26.52
The maximum temperature is : 32
```

THANK YOU

Team Python - 2022

Department of Computer Science and Engineering

Contact Email ID's:

sindhurpai@pes.edu

cnrajeswari@pes.edu

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Prof. C N Rajeswari, Prof. Sindhu R Pai
Department of Computer Science
and Engineering

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Unit - 4: Functional and Object Oriented programming

Zip function and practice programs

Prof. C.N.Rajeswari , Prof. Sindhu R Pai

Department of Computer Science and Engineering

Zip() function

- Python zip() method takes iterable containers and returns a single iterator object, having mapped values from all the containers.
- It is used to map the similar index of multiple containers so that they can be used just using a single entity.
- The function zip is used to associate the corresponding elements of two or more iterables into a single lazy iterable of tuples.
- It does not have any callback function.

Syntax : zip(*iterators)

Zip() function

PES UNIVERSITY

Working

- The zip() function is used to combine two or more lists (or any other iterables) into a single iterable.
- Elements from corresponding positions are paired together.
- The resulting iterable contains tuples, where the first element from each list is paired together, the second element from each list is paired together, and so on.

Zip() function

Example 1 Consider the code below which pairs the two lists:

```
m=[1,2,3]
n=[4,5,6]
l_new=[]
for i in range(len(m)):
        l_new.append((m[i],n[i]))
print(l_new)
```

The above can be done very easily and with less code using zip. We can observe the same output.

```
print(list(zip(m,n)))
```

Output

```
[(1, 4), (2, 5), (3, 6)]
[(1, 4), (2, 5), (3, 6)]
```

Zip() function

Example 2: Combining two iterables (tuples) into a single iterable

```
name = [ "Sudha", "Suma", "Sara", "Asha" ]
roll_no = [ 404, 112, 393, 223 ]

# using zip() to map values
mapped = zip(name, roll_no)

print(set(mapped))
```

Output:

```
{('Sudha', 404), ('Suma', 112), ('Sara', 393), ('Asha', 223)}
```

Zip() function

Example 3: Combining two iterables (lists) into a single list

```
a = [1, 2, 3, 4, 5]

b = list(map(lambda x : x * x * x, a))

print(a)

print(b)

print(list(zip(a, b)))
```

Output:

```
[1, 2, 3, 4, 5]
[1, 8, 27, 64, 125]
[(1, 1), (2, 8), (3, 27), (4, 64), (5, 125)]
```

Zip() Function

Example 4: Zipping list and tuple with unequal size

```
#Define lists for 'persons',and a tuple for 'ages'
persons = ["Baskar", "Monica", "Riya", "Madhav", "John",
"Prashanth"]
ages = (35, 26, 28, 14)

#lists along with the 'ages' tuple
zipped_result = zip(persons,ages)

print("Zipped result as a list:")
for i in list(zipped_result):
  print(i)
```

Zip() Function

Example 5: Zipping list with unequal size.(Two ways)

```
lis1 = [1,2,3]
lis2 = [4,5,6,7]
print(list(zip(lis1,lis2)))
```

The same can be acheived using the directory comprehension as:

```
print({k:v for k,v in (zip(lis1,lis2))})
```

Output

```
[(1, 4), (2, 5), (3, 6)]
{1: 4, 2: 5, 3: 6}
```

Practice programs

1.Given list of circle areas all in five decimal places, round each element in the list up to its position decimal places, meaning round up the first element in the list to one decimal place, the second element to two decimal places, the third element to three decimal places and so on.

2.Given
list= [1,2,3,4,5]
Chars= [‘a’, ’b’, ’c’, ’d’, ’e’]
Output the following.

('a', 1), ('b', 2), ('c', 3), ('d', 4), ('e', 5)]

3.The following are the scores of chemistry exam . Filter out those who passed with scores >75. Use an appropriate function

Practice programs

4.You are given a list of fruits names. Using lambda and map function print the list of fruit names starting with ‘A’
Lst=[‘Orange’, ’Apple’ , ’ Mango’, ‘ Apricot’]
Output=[‘Apple’, ‘Apricot’]

5.Using reduce function find the sum of the digits of the digits of the given number

For ex: n = '1729’
output= summation of 1,7,2,9=19

6. Using the min and reduce function find the smallest number in a given list of 10 numbers.

THANK YOU

Team Python - 2022

Department of Computer Science and Engineering

Contact Email ID's:

sindhurpai@pes.edu

cnrajeswari@pes.edu

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Prof. C N Rajeswari, Prof. Sindhu R Pai
Department of Computer Science
and Engineering

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Unit - 4: Functional and Object Oriented programming

List Comprehension

Prof. C.N.Rajeswari , Prof. Sindhu R Pai

Department of Computer Science and Engineering

List comprehension

PES UNIVERSITY

List comprehension

- List comprehension is a concise way of defining and creating a list.
- It is used to create functionality within a single line of code.
- Return value is always a new list obtained by evaluating the expression in the context of for and if clauses which follows it.
- List comprehension is faster in processing than a list using for loop.

List Comprehension

PES UNIVERSITY

Syntax:

list = [expression for <variable> in <iterable> [if condition]]

This is equivalent to:

```
for variable in iterable :
        if condition:
                expression
```

List Comprehension

Consider the code :

```
m=[1,2,3]
n=[4,5,6]
new_li=[]
for x in m:
        for y in n:
                new_li.append(x+y)
print(new_li)
```

The same can be done with a single line of code using list comprehension.

```
print([x+y for x in m for y in n])
```

Output:

```
5, 6, 7, 6, 7, 8, 7, 8, 9]
[5, 6, 7, 6, 7, 8, 7, 8, 9]
```

List comprehension

Consider an example where we want to calculate a list with the cube of the first 10 natural numbers:

Method 1: (without using list comprehension)

```
cube=[]
for i in range (0,11):
        cube.append(i**3)
print("result = ",cube)
```

Method 2: (using list comprehension)

```
cube = [x**3 for x in range (11)]
print("result = ", cube)
```

Output: (Both code gives the same output)

```
result = [0, 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

List comprehension

Example 2: To print the last letter of every word in the list.

```
Words = ["hi","how","are","you"]
items = [ word[-1] for word in Words ]
print (items)
```

Output:

```
['i', 'w', 'e', 'u']
```

List comprehension

Example 3: To print the common numbers in two lists.

```
list1 = [1, 2, 3, 4]
list2 = [2, 3, 4, 5]

common_num = [a for a in list1 for b in list2 if a == b]
print(common_num)
```

Output:

```
[2, 3, 4]
```

List comprehension

Example 4: To print each city with its length.

```
# list of strings and its length
city = [ (x, len(x)) for x in ['Bangalore', 'Delhi',’Bombay']]
print(city)
```

Output:

```
[('Bangalore', 9), ('Delhi', 5), (‘Bombay', 6)]
```

List comprehension

PES UNIVERSITY

Example 5: To print the squares of first 10 odd numbers.

```
list1=[x*x  for  x  in  range(1,11)]
list2=[x  for  x in list1 if  x%2 != 0]
 print(list2)
```

Output:

```
[1, 9, 25, 49, 81]
```

List comprehension –Practice programs

PES UNIVERSITY

1. Using list comprehension find the transpose of a given matrix.

```
matrix = [[1, 2], [3,4], [5,6], [7,8]]
transpose = [[row[i] for row in matrix] for i in range(2)]
print (transpose)
```

Output:

```
[[1, 3, 5, 7], [2, 4, 6, 8]]
```

List comprehension –Practice programs

2. Using list comprehension print all the numbers which are divisible by both
   2 and 5 in the range of 0 to 100

```
num_list = [y for y in range(100) if y % 2 == 0 if y % 5 == 0]    #Use of nested if
print(num_list)
```

Output:

```
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
```

List comprehension –Practice programs

PES UNIVERSITY

3.Using list comprehension find all of the numbers that have a 6 in them.

```
nums=input("Enter the number")
List_with_6 = [num for num in nums if  "6"  in  str(num)]
print(List_with_6)
```

Output:

```
Enter the number76548
['6']
```

List comprehension –Practice programs

4.Using list comprehension count the number of spaces in a string

```
string=input("Enter the string:  ")
len_spaces = len([char for char in string if char == " "])
print("The length of the spaces is::",len_spaces)
```

Output:

```
Enter the string:  This is a test string
The length of the spaces is:: 4
```

List comprehension –Practice programs

5.Using list comprehension, remove all of the vowels present in the input string.

```
string=input("Enter the string::")
str_no_vowel = "".join([char for char in string.lower() if char not in ["a","e","i","o","u"]])
print("The  string without vowel::",str_no_vowel)
```

Output:

```
Enter the string:: All welcome
The  string without vowel::  ll wlcm
```

THANK YOU

Team Python - 2022

Department of Computer Science and Engineering

Contact Email ID's:

sindhurpai@pes.edu

cnrajeswari@pes.edu