**Batch: P1 – 2 Roll No.: 16014022050**

**Experiment / assignment / tutorial No.: 5**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE**: Recursion and Lambda Function |

**AIM:** To implement recursion function and lambda function

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Expected OUTCOME of Experiment:**

**CO2:** Use different Decision-making statements and Functions in Python.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Theory:**

## Python Functions:

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.

## Creating a Function

## In Python a function is defined using the def keyword:

Example: def my\_function():  
   print("Hello from a function")

## Arguments

## Information can be passed into functions as arguments.

## Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

## Parameters or Arguments?

## The terms parameter and argument can be used for the same thing: information that is passed into a function. From a function's perspective:

## A parameter is the variable listed inside the parentheses in the function definition.

## An argument is the value that is sent to the function when it is called.

## Number of Arguments

## By default, a function must be called with the correct number of arguments i.e., if your function expects 2 arguments; you have to call the function with 2 arguments, not more, and not less.

## Keyword Arguments

You can also send arguments with the key = value syntax.

This way the order of the arguments does not matter.

## Arbitrary Keyword Arguments, \*\*kwargs

If you do not know how many keyword arguments will be passed into your function, add two asterisks: \*\* before the parameter name in the function definition.

This way the function will receive a dictionary of arguments, and can access the items accordingly

## Default Parameter Value

The following example shows how to use a default parameter value.

If we call the function without argument, it uses the default value:

## Passing a List as an Argument

## You can send any data types of argument to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function.

## Return Values

To let a function, return a value, use the return statement.

## 1.10 The pass Statement

Function definitions cannot be empty, but if you for some reason have a function definition with no content, put in the pass statement to avoid getting an error.

## Recursion:

## Python also accepts function recursion, which means a defined function can call itself. Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result. The developer should be very careful with recursion as it can be quite easy to slip into writing a function which never terminates, or one that uses excess amounts of memory or processor power. However, when written correctly recursion can be a very efficient and mathematically-elegant approach to programming.

## To a new programmer it can take some time to work out how exactly this works, best way to find out is by testing and modifying it.

1. **Lambda function:**

A lambda function is a small anonymous function.

A lambda function can take any number of arguments, but can only have one expression. Syntax of Lambda Function is given below

*lambda*arguments:expression

Lambda functions can take any number of arguments:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Problem Definition:**

1. In below table input variable, python code and output column is given. You have to complete blank cell in every row.

|  |  |  |
| --- | --- | --- |
| **S.No** | **Python Code** | **Output** |
| 1. | def my\_function(fname, lname):   print(fname + " " + lname)  my\_function("Amit", "Kumar") | Amit Kumar |
| 2. | def my\_function(fname, lname):   print(fname + " " + lname)  my\_function("Emil") | my\_function("Emil")  TypeError: my\_function() missing 1 required positional argument: 'lname' |
| 3. | def my\_function(\*kids):  print("The youngest child is " + kids[2])  my\_function("Emil", "Tobias", "Linus") | The youngest child is Linus |
| 4. | def my\_function(college3, college2, college1):      print("The Best college is " + college3)  my\_function('KJSCE',' ',' ') | The Best college is KJSCE |
| 5. | def my\_function(**country= "Norway"**):   print("I am from " + country)  my\_function("Sweden") my\_function("India") my\_function() my\_function("Brazil") | I am from Sweden  I am from India  I am from Norway  I am from Brazil |
| 6. | def tri\_recursion(k):  if(k > 0):  result = k + tri\_recursion(k - 1)  print(result)  else:  result = 0  return result  print("Recursion Example Results")  tri\_recursion(6) | Recursion Example Results  1  3  6  10  15  21 |
| 7. | print((lambda x: x\*2) (9)) | 18 |
| 8. | twice = lambda x: x\*2  print(twice(9)) | 18 |

1. Write a Python program using a recursive function that takes a string as input from the user and displays whether the string is Palindrome or not.
2. Write a Python program to separate out even and odd numbers from the list entered by user by using Lambda function

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Books/ Journals/ Websites referred:**

1. Reema Thareja, *Python Programming: Using Problem Solving Approach*, Oxford University Press, First Edition 2017, India
2. Sheetal Taneja and Naveen Kumar, *Python Programming: A modular Approach*, Pearson India, Second Edition 2018, India

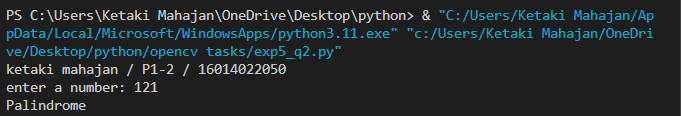
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Implementation details:**

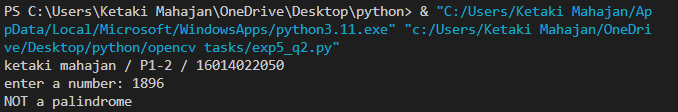
* 1. print("ketaki mahajan / P1-2 / 16014022050")
  2. def PalinNum(str):
  3. if len(str) <= 1:
  4. return True
  5. elif str[0] == str[-1]:
  6. return PalinNum(str[1:-1])
  7. else:
  8. return False
  9. palin\_input = input("enter a number: ")
  10. if PalinNum(palin\_input):
  11. print("palindrome")
  12. else:
  13. print("NOT a palindrome")
  14. print("ketaki mahajan / P1-2 / 16014022050")
  15. input\_list = list(map(int,input().split()))
  16. even\_arr = []
  17. odd\_arr = []
  18. odd = lambda y: y%2 == 1
  19. odd\_arr = filter(odd, input\_list)
  20. print("Odd numbers: ", list(odd\_arr))
  21. even = lambda y: y%2 == 0
  22. even\_arr = filter(even, input\_list)
  23. print("EVen Numbers: ", list(even\_arr))

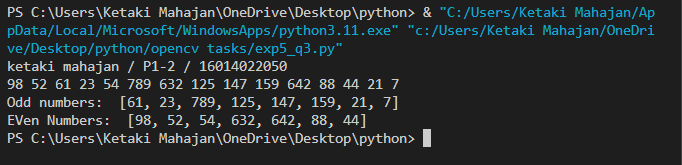
**Output(s):**

testing for palindrome number –



testing for **INVALID** palindrome number –



****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Conclusion:**

In this experiment, the recursion function and lambda function were implemented successfully.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Post Lab Questions:**

1. **Write a python program to calculate factorial using recursion.**

print("ketaki mahajan / P1-2 / 16014022050")

def factorial(x):

   if x == 1:

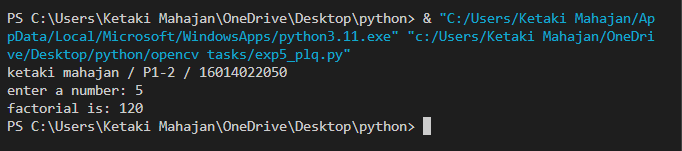
       return 1

   else:

       return x \* factorial(x-1)

fact = int(input("enter a number: "))

print("factorial is:", factorial(fact))



1. **What are the common functional programming methods that use lambdas?**

Lambda expressions are used to create anonymous functions, or functions without a name. They are useful when we need to create a function that will only need to be used once (a throw-away function) and can be written in one line.

The three functions, which provide a functional programming style within the object-oriented python language, are the map(), filter(), and reduce() functions.

* The ***map function*** provides us a way to apply a function to each element in an iterable object, such as lists, strings, tuples, etc…

Thus, the map function takes in two arguments: the function we want to apply, and the iterable object we want to apply it to.

**map(function, iterable)**

* The ***filter function*** will “filter out” an iterable object based on a specified condition. This condition will be decided by the function that we pass in.

**filter(function, iterable)**

The filter function takes in two arguments: the function that checks for a specific condition and the iterable we want to apply it to (such as a list).

The filter function takes each element from our list and passes it in to the function given. If the function argument returns True, the filter function will add that value to the filter object. If the function returns False, then that element will not be added to our filter object

* The ***reduce function*** is taking an iterable, such as a list, and reduces it down to a single cumulative value.

The reduce function can take in three arguments, two of which are required.

The two required arguments are: a function (that itself takes in two arguments), and an iterable (such as a list). The third argument, which is an initializer, is optional.

**reduce(function, iterable[, initializer])**

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**