Name: Chandana Ramesh Galgali
Batch: P6-1 Roll No.: 16010422234
Experiment / ~~assignment~~ / ~~tutorial~~ No. 05
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.

Resource Needed: Python IDE

Theory:

1. 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.

1.1 Creating a Function

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

Example:
```
def my_function():
    print("Hello from a function")
```

1.2 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.

1.3 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.

1.4 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.

1.5 Keyword Arguments

You can also send arguments with the key = value syntax.
This way the order of the arguments does not matter.

1.6 Arbitrary Keyword Arguments, **kwargs

If you do not know how many keyword arguments will be passed into your function, add two asterisk: ** before the parameter name in the function definition.
This way the function will receive a dictionary of arguments, and can access the items accordingly

1.7 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:

1.8 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.

1.9 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.

2. 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 exactly how this works, the best way to find out is by testing and modifying it.

3. 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 the below table input variable, python code and output column is given. You have to complete the 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","SK SOMAIYA","VJTI")	The Best college is KJSCE
5.	def my_function(country= "Norway"): print("I am from " + country) my_function("Sweden") my_function("India")	I am from Sweden I am from India

	my_function() my_function("Brazil")	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

2. 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.

3. 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:

2.

```
def isPalindrome(s):
    s=s.lower()
    l=len(s)
    if l<2:
        return True
    elif s[0]==s[l-1]:
        return isPalindrome(s[1:l-1])
    else:
        return False
s=input("Enter a string: ")
ans=isPalindrome(s)
if ans:
    print("The string is a palindrome")
else:
    print("The string is not a palindrome")
```

3.

```
numbers=[]
i=int(input("Enter the number of values in the list : "))
for j in range (0,i):
    ele=int(input(f"Enter value for index {j} : "))
    numbers.append(ele)
    j=j+1
even = filter(lambda n: n % 2 == 0, numbers)
odd = filter(lambda n: n % 2 == 1, numbers)
print("The even numbers are: ",list(even))
print("The odd numbers are: ",list(odd))
```

Output(s):

2.

```
Enter a string: 1221
The string is a palindrome
```

```
Enter a string: chandana
The string is not a palindrome
```

3.

```
Enter the number of values in the list : 4
Enter value for index 0 : 2
Enter value for index 1 : 8
Enter value for index 2 : 7
Enter value for index 3 : 9
The even numbers are:  [2, 8]
The odd numbers are:  [7, 9]
```

Conclusion:

The use of python functions such as recursion and lambda functions is understood.

Post Lab Descriptive Questions

1. Write a python program to calculate factorial using recursion.
   Ans:

```
n=int(input("Enter the number: "))
def factorial(n):
    if n==0:
        return 1
    else:
        return n*factorial(n-1)
result=factorial(n)
print("The factorial of the entered number is: ",result)
```

```
Enter the number: 5
The factorial of the entered number is:  120
```

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

   Ans: Functional programming is a programming paradigm that focuses on writing code using functions that do not change state or mutate data. Python supports functional programming and provides many built-in methods that work with functions and use lambdas. Here are some of the common functional programming methods in Python that use lambdas:

   - map: The map function applies a given function to each element of an iterable and returns a new iterable. It takes two arguments: the function to apply and the iterable to apply it to. Lambdas are commonly used with map to provide a quick and easy way to create simple, one-time use functions.
   - filter: The filter function returns an iterable containing the elements from the input iterable for which the given function returns True. It takes two arguments:

the function to apply and the iterable to apply it to. Lambdas are often used with filters to create simple filtering functions.

- reduce: The reduce function applies a binary function to the elements of an iterable in a cumulative way, reducing the iterable to a single value. It takes two arguments: the function to apply and the iterable to apply it to. Lambdas can be used with reduce to create custom binary functions.
- sorted: The sorted function returns a sorted list of the elements in an iterable. It takes two optional arguments: the key argument specifies a function to apply to each element before sorting, and the reverse argument specifies whether to sort in descending order. Lambdas are often used with the key argument to provide custom sorting functions.

These are just a few of the common functional programming methods in Python that use lambdas. There are many other built-in methods and libraries that support functional programming concepts in Python, such as itertools, functools, and more.

Date: _____ **Signature of faculty in-charge**