Assignment 8

1.In Python, what is the difference between a built-in function and a user-defined function? Provide an example of each.

Pre-defined or built-in function	User-defined function
• **Pre-defined or built-in function is a function which is already defined in a Python Programming itself.**	• **User-defined function is a function which is defined by the user to reduce the usage of code again and again.**
• **print('Total built-in functions in python are:',len(dir(__builtins__)))** **input(), len(),max() are some of the built-in functions** **My_list = [1,2,3]** **my_list = [1,2,4]** **print('The maximum number in the list are',max(my_list))** **Output:** **Total built-in functions in python are: 157** **The maximum number in the list are: 4**	• **def printText():** **print('Hii Ineuron')** **printText()** **Output:** **Hii Ineuron**

2. How can you pass arguments to a function in Python? Explain the difference between positional arguments and keyword arguments.

The arguments can be passed to a function in two ways.

- **Positional arguments**
- **Keyword arguments**

Positional arguments	Keyword arguments
Positional arguments are passed to a function based on the position or order.	**Keyword arguments are passed with parameter name explicitly defined.**
Values of order matters	**Value of order doesn't matter.**
def printText(name,age): **print(f'Hii,{name} age is {age}')** **name = input("enter the name: ")** **age = int(input("Enter the age: ")** **printText(name,age)**	**def printText(name,age):** **print(f'Hii,{name} age is {age}')** **name = input("enter the name: ")** **age = int(input("Enter the age: "))** **printText(name=name,age=age)**

Output: **enter the name: surya** **Enter the age: 23** **Hii,surya age is 23**	**Output:** **enter the name: surya** **Enter the age: 23** **Hii,surya age is 23**

3.What is the purpose of the return statement in a function? Can a function have multiple return statements? Explain with an example.

The purpose of return statement is to specify the value should be evaluated and return when it is called. When a return statement is in function it immediately exits the function and returns the value.

Yes, function can have multiple return statements.

Example:

Snippet:

```
score =int(input('Enter the score'))
def calGrade(score):
   if score >=90:
      return 'A'
   elif score >= 80:
      return 'B'
   elif score >= 70:
      return 'C'
   elif score >=60:
      return 'D'
   elif score >=50:
      return 'E'
   else:
      return 'F’
grades = calGrade(score)
print(grades)
```

Output:

Enter the score: 50
E

4. What are lambda functions in Python? How are they different from regular functions? Provide an example where a lambda function can be useful.

Lambda functions are known as anonymous functions or single-expression functions.

Lambda functions	Regular functions
• **They are defined using lambda keyword followed by function arguments and colon, all in single line**	• **They are defined using the def keyword, followed by the function name, parentheses for arguments, a colon, and an indented block of code.**
• **They are single expression function to perform single task**	• **They can perform multiple statement which can perform complex task.**

add = lambda val1,val2: val1+val2
val1 = int(input("Enter the value1: "))
val2 = int(input("Enter the value2: "))
print('The sum of two numbers are ',add(val1,val2))
	def add(val1,val2):
	  sumOfNumbers = val1+val2
	  return sumOfNumbers
	val1 = int(input("Enter the value 1"))
	val2 = int(input("Enter the value 2"))
	addition = add(val1,val2)
	print(f'Sum of two numbers are \ {addition}')

5. How does the concept of "scope" apply to functions in Python? Explain the difference between local scope and global scope.

In python, Scope refers to the region or context. The concept of scope applies to functions in two ways.

Local scope	Global Scope
• **Variables that are defined inside the function or class are known as local scope which can be accessed only within the function or class.**	• **Variables that are defined outside the function or class are known as global scope which can be accessed with in and outside the function or class using global keyword along with variable name**
• **Snippet:** **def printName():** ** name = input('Enter the name')** ** print(f'Hi {name}!')** **printName()**	• **Snippet:** **name = input("Enter the name")** **def printName():** ** global name** ** print(f'Hi {name}! - global ')** **printName()**
Output: Enter the namesurya Hi surya! -local	**Output:** Enter the namesurya Hi surya! -global

6. How can you use the "return" statement in a Python function to return multiple values?

Explanation:

In python, in return statement we can use multiple statements by

```
def printUserDetails(name,age,email,phone_number,address):
    return name,age,email,phone_number,address
name = input("Enter the name: ")
age = int(input("Enter the age: "))
email = input('enter the email')
phone_number = int(input("enter the mobile number"))
address = input('Enter the address')
user = printUserDetails(name,age,email,phone_number,address)
print(user)
```

7. What is the difference between the "pass by value" and "pass by reference" concepts when it comes to function arguments in Python?

Pass by value:

- **Copy of the value of variable passed to the function.**
- **Any changes of the variable inside the function will not affect the values of original variable.**
- **The original variable remains after the function call.**

Snippet:

```
def pass_by_val(value):
        print("Before updating the value inside the function:",value)
        value+=10
        print("After updating the value inside the function:",value)
print('----------pass by value----------')
number = int(input("enter the value:"))
print('Before the function call:',number)
pass_by_val(number)
print('After the function call:',number)
print('----------------------------------------')
```

Output:

----------pass by value----------

enter the value:10

Before the function call: 10

Before updating the value inside the function: 10

After updating the value inside the function: 20

After the function call: 10

Pass by Reference

- **Original value is passed to the function's aruguments.**
- **Any changes of the variable inside the function will affect the values of original variable.**
- **The original variable value will change.**

Snippet:

```
def pass_by_ref(my_list):
        print("Before updating the value inside the function:",my_list)
        var = int(input())
        my_list.append(var)
        print("After updating the value inside the function:",my_list)
print('----------pass by reference----------')
my_list= [1,2,3]
print('Before the function call',my_list)
pass_by_ref(my_list)
print('After the function call',my_list)
print('----------')
```

Output:

```
----------pass by reference----------
Before the function call [1, 2, 3]
Before updating the value inside the function: [1, 2, 3]
5
After updating the value inside the function: [1, 2, 3, 5]
After the function call [1, 2, 3, 5]
----------
```

8. Create a function that can intake integer or decimal value and do following operations:

a. Logarithmic function (log x)

b. Exponential function (exp(x))

c. Power function with base 2 (2x)

d. Square root

```
import math
def perform_operations(value):
        log = math.log(value)
        exp = math.exp(value)
        power = math.pow(2,value)
        sqrt = math.sqrt(value)
        return log,exp,power,sqrt
value = int(input("Enter the value: "))
logarithmic,exponential,power,squareroot = perform_operations(value)
print(f"Logarithmic function: {logarithmic} , Exponential function: {exponential}, power function: {power}, square root: {squareroot}")
```

Output:

```
Enter the value: 5
Logarithmic function: 1.6094379124341003 , Exponential function: 148.4131591025766, power function: 32.0, square root: 2.23606797749979
```

9. Create a function that takes a full name as an argument and returns first name and last name.

Snippet:

```
def full_name(fullname):
        split_name = fullname.index(' ')
        first_name = fullname[:split_name]
        last_name = fullname[split_name+1:]
        return first_name, last_name
fullname = input("enter the full name: ")
first,last = full_name(fullname)
```

```
print(f'First Name:{first} Last Name:{last}')
```

Output:

```
enter the full name: surya prakash
First Name: Surya Last Name: Prakash
```