10th June – Python (Functions Assignment)-1

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

Ans- Built-in functions are functions that are provided for us by the standard includes. User-defined functions are those that you write yourself. Third-party functions are those that are written for you, but that are not provided by the standard includes.

A built-in function is any function, operator or statement that is provided as part of the standard language, and is common to all developers regardless of platform. For instance, if(), while(), for(), return() and switch() are all examples of built-in functions.

A user-defined function is any function you write yourself.

Your IDE may also include third-party functions, some of which are common to all developers, such as string.h for string handling functions. These may be considered built-in because they are common to all developers, however they are strictly user-defined functions which must be #included in your programs to make use of them.

Example of a built-in function:

```
# Example of a built-in function
print("Hello World!")
Output:
Hello World!
```

Example of a user-defined function:

```
# Example of a user-defined function
def my_function():
    print("Hello from my function!")

my_function()
Output:
Hello from my function!
```

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

Ans- In Python, we can pass arguments to a function in two ways: positional arguments and keyword arguments.

Positional arguments are passed to a function in the order they are defined in the function signature. For example, if we have a function that takes two positional arguments, we must pass two values to the function in the order they are defined.

Keyword arguments are passed to a function using their names. This allows us to pass arguments in any order and also makes your code more readable and self-explanatory.

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

Ans- The purpose of the return statement in a function is to return a value to the caller. When a return statement is executed, the function stops executing and returns the value of its expression (if any) to the caller.

A function can have multiple return statements. When any of them is executed, the function terminates.

Example:

```
def my_function(x):
  if x > 0:
    return "Positive"
  elif x < 0:
    return "Negative"
  else:
    return "Zero"

print(my_function(1))
print(my_function(-1))
print(my_function(0))

Output:
Positive
Negative
Zero
```

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

Ans- Lambda functions are anonymous functions that are defined without a name. They are also known as inline functions. They are used when we need a small function for a short period of time. Lambda functions are defined using the keyword lambda. They can take any number of arguments but can only have one expression. The expression is evaluated and returned when the function is called.

Regular functions are defined using the keyword def. They can have any number of arguments and any number of expressions. Regular functions can be called by their name anywhere in the program.

lambda functions are anonymous functions that are used when we need a small function for a short period of time while regular functions are named functions that can be called anywhere in the program.

Uses of lambda functions: -------

- To use as a key function for sorting, filtering, or reducing data.
- To perform advanced array operations with lambda helper functions.
- To apply a simple calculation or transformation to each element in a list or an array with the map function.
- To summarize or multiply multiple arguments and return the result.

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

Ans- A scope is a region of the program where a particular variable is accessible. The scope of a variable determines the portion of the program where we can access that variable. In Python, there are two types of scopes:

1. **Local scope:** A variable that is defined inside a function body has a local scope. It can only be accessed within that function.
2. **Global scope:** A variable that is defined outside a function has a global scope. It can be accessed by any function in the program.

The difference between local and global scope is that local variables can only be accessed within the function where they are defined, while global variables can be accessed by any function in the program.

Example:

```
x = 10

def my_func():
    y = 5
    print(x) # This will work because x is defined in the global scope
    print(y) # This will work because y is defined in the local scope

my_func()
output:
10
5
```

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

Ans- You can return multiple values by simply separating them with commas in the return statement.

For example, we can define a function that returns a string and an integer as follows:

```
def test():
    return 'abc', 100, True

result = test()
print(result) # Output: ('abc', 100, True)

x, y, z = test()
print(x)
print(y)
print(z)
```

Output:

```
('abc', 100, True)
abc
100
True
```

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

Ans- In Python, everything is passed by value. However, the values are references. Every value in Python is a reference (pointer) to an object. Objects cannot be values. Assignment always copies the value (which is a pointer); two such pointers can thus point to the same object.

And if, when you pass an argument to a function in Python, you are actually passing the reference of the object to the function. This means that if you modify the object inside the function, it will also be modified outside the function.

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 (2 x)

d. Square root

Ans-

```
import math


def operations(x):
    print("Logarithmic function (log x):", math.log(x))
    print("Exponential function (exp(x)):", math.exp(x))
    print("Power function with base 2 (2^x):", pow(2,x))
    print("Square root:", math.sqrt(x))
```

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

Ans-

```
def split_name(fullname):


    first_name, last_name = fullname.strip().split()
    return first_name, last_name
```