Assignment 1 (10th June)

1. Difference between Built-in Functions and User Defined Functions.

**Answer :**

Built-in functions are predefined functions provided by Python itself, whereas User Defined functions are programmer specific functions based on the requirements.

Programmer can have their own specified name for user defined functions defined by def keyword followed by parenthesis (), parameters and colon (:).

Ex: **User Defined Function**

def add\_num(num1, num2):

sum = num1 + num2

return sum

result = add\_num(10, 15)

print("The sum of two numbers is : ", result)

Output : The sum of two numbers is : 25

Ex: **Built in Function**

List1 = [1, 2, 3, 4, 56, 87, 93648]

Result = max(List1)

print(“The max element of given element is : ”, Result)

The max element of given element is : 93648

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

**Answer:**

Positional arguments can be passed based on its position or order of the argument.

Keyword arguments can be passed irrespective of the order as it considers the name of the argument passed as a parameter. Keyword arguments are passed to a function using the name of the parameter and the corresponding value.

Ex : **Positional Argument**

def introduction(name, age, role):

print(f"Hello, I'm {name} and my age is {age}. I'm a {role}")

introduction("Dhoni", 42, "Wicketkeeper Batsman")

Output : Hello, I’m Dhoni and my age is 42. I’m a wicketkeeper Batsman

Ex : **Keyword Argument**

def introduction(name, age, role):

print(f"Hello, I'm {name} and my age is {age}. I'm a {role}")

introduction(age = 42, role = "Wicketkeeper Batsman", name = "Dhoni")

Output : Hello, I’m Dhoni and my age is 42. I’m a wicketkeeper Batsman

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

**Answer :**

The return statement in a function is used to specify the value of the output. When a return statement is encountered, the function execution halts, and the specified value(s) are returned. It is also used to exit the function prematurely if needed.

A function can have multiple return statements. Once a return statement is executed, the function immediately exits, and no further code is executed.

Ex : def marks(score):

if score >= 90:

return "A"

elif score >= 80:

return "B"

elif score >= 70:

return "C"

else:

return "D"

result = marks(90)

print(result)

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

**Answer :**

Lambda functions small, single-line functions in Python that are defined without a name. Keyword **lambda** is used to define the function and it is used for simple, one-time operations.

* Lambda functions are compact and simple and followed by the arguments, a colon, and the expression to be done.
* They don't require the use of the **def** keyword or a **return** statement.
* Lambda functions are namelessness as it doesn’t have a specific name assigned to them.
* They are not designed to contain multiple statements or complex logic.

Ex:

add = lambda x, y : x+y

print (add(6,4))

**Output :** 10

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

**Answer:**

Scope refers to the region or portion of the code where a variable or a name is visible and can be accessed.

* **Local Scope:** Local scope refers to the variables that are defined within a function. These variables are only accessible within the function itself and are not visible outside of it. They are created when the function is called and destroyed when the function execution completes

Ex:

def calculate\_sum(a, b):

    result\_1 = a + b

return result\_1

print(calculate\_sum(3, 5))  # Output: 8

print(result\_1)

**Output:** NameError: name 'result\_1' is not defined

* **Global Scope:** Global scope refers to the variables that are defined outside of any function. These variables can be accessed from anywhere within the module, including inside functions. Can be used with **global** keyword.

Ex:

global\_var = 10

def modify\_global():

global global\_var

global\_var = 20

print(global\_var)  # Output: 10

modify\_global()

print(global\_var)

**Output:**

**10**

**20**

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

**Answer :** In Python, the **return** statement can be used to return multiple values from a function like tuples, lists, or dictionaries.

**Return as Tuple:**

def get\_name\_and\_age():

name = "Alice"

age = 25

return name, age

result = get\_name\_and\_age()

print(result)

**Output :** (‘Alice’, 25)

**Return as Dictionary:**

def get\_person\_details():

    details = {

        "name": "Alice",

        "age": 25,

        "city": "New York"

    }

    return details

result = get\_person\_details()

print(result)

**Output:** {"name": "Alice", "age": 25, "city": "New York"}

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

**Answer:**

* **Pass by Value:** In pass by value, a copy of the value of a variable is passed to the function. Any modifications made to the function parameter inside the function do not affect the original variable outside the function.

Ex:

def increment(num):

num = num + 1

x = 10

increment(x)

print(x)

**Output: 10 (**Here it does not modify the x value by the function**)**

* **Pass by reference:** In pass by reference, the reference or memory address of a variable is passed to the function. Any modifications made to the function parameter inside the function directly affect the original variable outside the function.

Ex:

def increment(arr):

arr[0] = arr[0] + 1

my\_list = [10, 20, 30]

increment(my\_list)

print(my\_list)

**Output:** [11, 20, 30]

1. 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

**Answer:**

import math

def math\_operations():

number = int(input(“Enter your number : ”))

result = {}

result[‘Logrithm’] = math.log(number)

result[‘Exponential’] = math.exp(number)

result[‘Power’] = 2 \*\* number

result[‘Square root’] = math.sqrt(number)

operations\_math = math\_operations()

print(operations\_math)

**Output:**

Enter your number : 10

{'Logrithm': 2.302585092994046, 'Exponential': 22026.465794806718, 'Power Function': 1024, 'Square Root': 3.1622776601683795}

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

**Answer :**

def player\_name():

fullname = input("Enter the player fullname : ")

name = fullname.split()

first\_name = name[0]

last\_name = name[-1]

return first\_name, last\_name

first , last = player\_name()

print(f"First name is ", first)

print(f"Last name is ", last)

**Output:**

Enter the player fullname : Virat Kohli

First name is Virat

Last name is Kohli