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

A: **Built-in function:**  Functions that readily comes with Python are called built-in functions. Python provides built-in functions.

**For example:** print(), len(), min(), pop() etc.

**User-defined functionbut:** **Functions** that we define ourselves to do certain specific task are referred as **user**-**defined functions**. The way in which we define and call **functions in Python**.

**For example:**

**def add\_numbers(x,y):**

**sum = x + y**

**return sum**

**num1 = 5**

**num2 = 6**

**print("The sum is", add\_numbers(num1, num2))**

**Output**

**Enter a number: 2.4**

**Enter another number: 6.5**

**The sum is 8.9**

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

A:

* **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.

**Example:**  The following example has a function with one argument (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name

**def my\_function(fname):**

**print(fname + " Refsnes")**

**my\_function("Emil")**

**my\_function("Tobias")**

**my\_function("Linus")**

**Output:**

Emil Refsnes

Tobias Refsnes

Linus Refsnes

* **Positional arguments:** During a function call, values passed through arguments should be in the order of parameters in the function definition. This is called positional arguments.

**Example:**

def add(a,b,c):

return (a+b+c)

print (add(10,20,30))

**Output: 60**

The function call, all arguments are given as positional arguments. Values passed through arguments are passed to parameters by their position. 10 is assigned to a, 20 is assigned to b and 30 is assigned to c.

* **keyword arguments:** During a function call, values passed through arguments don’t need to be in the order of parameters in the function definition. This can be achieved by keyword arguments. But all the keyword arguments should match the parameters in the function definition or Keyword arguments should follow positional arguments only.

**Example:**

**def add(a,b=5,c=10):**

**return (a+b+c)**

**print (add(b=10,c=15,a=20))**

**Output: 45**

Calling the function **add** by giving keyword arguments. All parameters are given as keyword arguments, so there’s no need to maintain the same order.

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

A: **The python return statement is used to return the output from a function. We learned that we can also return a function from another function. Also, expressions are evaluated and then the result is returned from the function.**

**A function can have more than one return statement, but only ever run one based on a condition.**

**Example:**

**def** fun():

str **=** "godforgod"

x **=** 20

**return** str, x;

str, x **=** fun()

print(str)

print(x)

**Output:**

godforgod

20

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

**A: A lambda function is a small anonymous function. A lambda function can take any number of arguments, but can only have one expression.**

**Example:**

Add 10 to argument a, and return the result:

x = lambda a : a + 10

print(x(5))

**Output is 10**

**The lambda functions are different from regular functions like power of lambda is better shown when you use them as an anonymous function inside another function. you have a function definition that takes one argument, and that argument will be multiplied with an unknown number.**

**Example:**

def myfunc(n):

return lambda a : a \* n

mydoubler = myfunc(2)

print(mydoubler(11))

**Output is 22**

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

**A: In Python, the concept of scope is closely related to the concept of the namespace. As you've learned so far, a Python scope determines where in your program a name is visible. Python scopes are implemented as dictionaries that map names to objects. These dictionaries are commonly called namespaces.**

**There are two types of variables: global variables and local variables. The scope of global variables is the entire program whereas the scope of local variable is limited to the function where it is defined.**

## Local variables:

Local variables can only be reached within their scope(like func() above). Like in below program- there are two local variables – x and y.

### Example:

def sum(x,y):

sum = x + y

return sum

print(sum(5, 10))

### Output:

The variables x and y will only work/used inside the function sum() and they don’t exist outside of the function. So trying to use local variable outside their scope, might through NameError. So obviously below line will not work.

File "main.py", line 2

sum = x + y

^

IndentationError: expected an indented block

## Global variables:

A global variable can be used anywhere in the program as its scope is the entire program.

Example:

z = 25

def func():

global z

print(z)

z=20

func()

print(z)

### Output:

25

20

A calling func(), the global variable value is changed for the entire program. Below example shows a combination of local and global variables and function parameters.

def func(x, y):

global a

a = 45

x,y = y,x

b = 33

b = 17

c = 100

print(a,b,x,y)

a,b,x,y = 3,15,3,4

func(9,81)

print (a)

### Output:

45 17 81 9

3

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

**A: In Python, you can return multiple values by simply separating them with commas in the return statement. In Python, comma-separated values are treated as tuples, even without parentheses, unless the syntax requires them.**

**Example:**

def test\_list():

return ['abc', 100]

result = test\_list()

print(result)

print(type(result))

Output:

['abc', 100]

<class 'list'>

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

A:

**Pass by reference**: It is used in some programming languages, where values to the argument of the function are passed by reference which means that the address of the variable is passed and then the operation is done on the value stored at these addresses.

**Example:**

student = {'Jim': 12, 'Anna': 14, 'Preet': 10}

def test(student):

new = {'Sam':20, 'Steve':21}

student.update(new)

print("Inside the function", student)

return

test(student)

print("Outside the function:", student)

**Output:**

Inside the function {'Jim': 12, 'Anna': 14, 'Preet': 10, 'Sam': 20, 'Steve': 21}

Outside the function: {'Jim': 12, 'Anna': 14, 'Preet': 10, 'Sam': 20, 'Steve': 21}

After writing the above code, Once you will print **“student”** then the output will appear. Here, we created a dictionary called **student**, and **test(student)** is the function. Then two more students joined so we created the variable as **“new”** and the **student.update(new)** is used to update the dictionary, also the print will display the output.

## Pass by value:

When we pass something by value then the changes made to the function or copying of the variable are not reflected back to the calling function.

**Example:**

student = {'Jim': 12, 'Anna': 14, 'Preet': 10}

def test(student):

student = {'Sam':20, 'Steve':21}

print("Inside the function", student)

return

test(student)

print("Outside the function:", student)

Output:

Inside the function {'Sam': 20, 'Steve': 21}

Outside the function: {'Jim': 12, 'Anna': 14, 'Preet': 10}

After writing the above code, Once you will print **“student”** then the output will appear. Here, we created a dictionary called **student**, and **test(student)** is the function. Then two more students joined so we created the variable as **“new”** and the print will display the output. We can see that the inside and outside function remains the same.

## Pass by reference vs value:

## In the below example, we can see that all the parameters in the python language are passed by reference. So, if we change what a parameter refers to within a function, the change also reflects back in the calling function.

**Example:**

def marks(list):

list.append([11, 12, 13, 14, 15])

print("Value inside the function: ", list)

return

list = [10,20]

marks(list)

print("Value outside the function: ", list)

Output:

Value inside the function: [10, 20, [11, 12, 13, 14, 15]]

Value outside the function: [10, 20, [11, 12, 13, 14, 15]]

In this output, we can see that we are maintaining the reference of the passed object, and values are appending in the same object. So, you can see the output of the inside function and outside function.

## Arguments pass by reference or value:

## The parameters in the python language are passed by reference. Which mean if we change what parameter refers to within the function, the change also reflect black in the calling function.

**Example:**

teacher = {'Peter':101, 'John':102, 'Suzain':103}

def test(teacher):

new = {'kat':104, 'Satya':105}

teacher.update(new)

print("Inside the function",teacher)

return

test(teacher)

print("Outside the function:",teacher)

**Output:**

Inside the function {'Peter': 101, 'John': 102, 'Suzain': 103, 'kat': 104, 'Satya': 105}

Outside the function: {'Peter': 101, 'John': 102, 'Suzain': 103, 'kat': 104, 'Satya': 105}

After writing the above code, Once you will print **“teacher”** then the output will appear. Here, we created a dictionary called **teacher**, and the **def** **test(teacher)** is the function. Then two more teachers joined so we created the variable as **“new”** and the print will display the output. We can see that the inside and outside function remains the same.

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

A:

**import math**

**print ("The value of log 2 with base 3 is : ", end="")**

**print (math.log(2,3))**

**print ("The e\*\*4 value is : ", end="")**

**print (math.exp(4))**

**print ("The value of 3 to the power 2 is : ", end="")**

**print (math.pow(3,2))**

**print ("The value of square root of 25 : ", end="")**

**print (math.sqrt(25))**

**Output:**

**The value of log 2 with base 3 is : 0.6309297535714574**

**The e\*\*4 value is : 54.598150033144236**

**The value of 3 to the power 2 is : 9.0**

**The value of square root of 25 : 5.0**

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

**A:**

**def format\_name():**

**first\_name = input("Enter the first name : ")**

**last\_name = input("Enter the last name: ")**

**format\_first\_name = first\_name.capitalize()**

**format\_last\_name = last\_name.capitalize()**

**print("Student name is:",format\_first\_name, format\_last\_name)**

**format\_name()**

**Output:**

**Enter the first name : alice**

**Enter the last name: alter**

**Student name is: Alice Alter**