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

* In Python, a built-in function is a pre-defined function i.e., the user does not have to create the function it is always ready to use. For e.g.: sum(), max(), min(), etc.

A user-defined function is not already defined in the interpreter and has to be created by the user. For e.g.: sumOfProduct(n), factorial(n), fibonacci(n), etc.

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

* Arguments are specified after the function name, inside the parentheses. We can add as many arguments as we want, separating them with commas.

Positional arguments must be included in the correct order. Keyword arguments are included with a keyword and equals sign.

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

* The Python return statement is a special statement that you can use inside a function or method to send the function's result back to the caller. No, a function cannot have multiple return statements because the function ends after the first return statement.

For e.g.:

def foo(n):

n += 1

return n

return n + 100

print(foo(1))

#Output is 2. The second return statement is ignored.

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

* A lambda function is an anonymous function (because it doesn't require a standard function declaration using the def keyword). A lambda function is different from regular functions because it is concise, it is restricted to a single expression and it doesn’t have a specific name associated with it.

For e.g.:

greater = lambda x, y: x if x > y else y

print(greater(30, 40)) #Gives output as 40 which is the greater of the two.

1. 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 visibility or accessibility of variables within a specific part of a program.

|  |  |
| --- | --- |
| Local Scope | Global Scope |
| Local scope refers to variables that are defined inside a specific function or block of code. | Global scope refers to variables that are defined outside of any function or block of code. |
| Variables defined within a function are only accessible within that function. They are said to have local scope. | Global variables are accessible from anywhere in the program, including within functions. |
| Local variables are created when the function is called and are destroyed when the function completes its execution. | Global variables are created when they are defined and remain in memory until the program terminates. |

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

* We can use the “return” statement in a Python function to return multiple values by using commas. For e.g.: return a, b, c #a, b, c are some variables in the function

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

* In Python, when you pass an object as an argument to a function, what gets passed is not the value of the object itself, but a reference to the object. This means that the function receives a copy of the reference to the object, allowing it to access and modify the underlying object. However, the reference itself is passed by value.

1. Create a function that can intake integer or decimal value and do following operations:
   1. Logarithmic function (log x)
   2. Exponential function (exp(x))
   3. Power function with base 2 (2x)
   4. Square root

* The code for the above problem is as follows:

import math

def operations(n):

log = math.log(n) #Calculates logarithmic function

exp = math.exp(n) #Calculates exponential function

pow = math.pow(2, n) #Calculates power function with base 2

sqrt = math.sqrt(n) #Calculates square root

return log, exp, pow, sqrt

# Example usage

print(op(1.4))

#Output is as follows:

#(0.3364722366212129, 4.0551999668446745, 2.6390158215457884, 1.1832159566199232)

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

* The code for the above problem is as follows:

def get\_first\_last\_name(full\_name):

names = full\_name.split()

first\_name = names[0]

last\_name = names[-1]

return first\_name, last\_name

# Example usage

full\_name = "John Doe"

first\_name, last\_name = get\_first\_last\_name(full\_name)

print("First name is: ", first\_name)

print("Last name is: ", last\_name)