1. Why are functions advantageous to have in your programs?

```
Ans :- Reusability, Modularity, Abstraction, Code Organisation and readability,
            Testing &    Debugging.
```

Reusability:- Functions allows you to write a block of code that can be used multiple times throughout your program. You can encapsulate it in a function and call the function whenever needed. Modularity:- Function helps in breaking down a complex program into smaller manageable modules. Each function can perform specific task, making the code more organized and easier to understand. Abstraction:- Abstraction level enhances the code clarity and simplifies overall programming. Testing & Debugging: By isolating few blocks from the regular program, we can test the reliability and correctness. Code Organisation and readability: Function improves the code organised by grouping related code together thus achieving the better readability of code.

1. When does the code in a function run: when it's specified or when it's called?

Ans: Code written in function run when it is being called.

1. What statement creates a function?

Ans:- ‘def’ statement is used to create a function. The Syntax for defining a function is as follows: def fun_name(parameters):

function block

code block

1. What is the difference between a function and a function call?

Ans:- Function defines a block of code which is ready for execution of a specific task and when function call occurs then the execution of that particular function will start by giving appropriate results. In other words, function is a blue print and function call is a real execution.

1. How many global scopes are there in a Python program? How many local scopes?

Ans:- In Python programming there can be one global scope and multiple local scopes. Global Scope: The global scope refers to the outermost level of the program, where variables, functions and classes can be defined. Any variable or object defined in the global scope is accessible throughout the program. Local Scope: are created when functions are executed. The scope is limited within given function or loop so it has limited lifespan.

1. What happens to variables in a local scope when the function call returns?

Ans: When the function call returns in Python, the local variables within the local scope cease to exist.

In [14]:
```
def my_function():
    x=10
    y=20
    result=x+y
    return result

# function call
sum_result=my_function()
print(sum_result)
```

```
print(x)   # here x variable is not available for printing hence throws error.
# After the function call local variables (x,y,result) are no longer accessible
```

```
30
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_2320\829669792.py in <module>
      8 sum_result=my_function()
      9 print(sum_result)
---> 10 print(x)
     11 # After the function call local variables (x,y,result) are no longer accessible

NameError: name 'x' is not defined
```

1. What is the concept of a return value? Is it possible to have a return value in an expression?

Ans:- The concept of the return value in a programming refers to the value that a function can optionally send back to the code that called it. The return value can be of any datatype such as numbers, strings, list, dictionaries, or even custom objects.

1. If a function does not have a return statement, what is the return value of a call to that function?

If a function does not have a return statement, or if the return statement is omitted, the function will automatically return a special value called None.

In [15]:
```
def name(name):
    print(f'Hello, {name}')

result = name ('Surya')
print(result)  # Output: None
```

```
Hello, Surya
None
```

1. How do you make a function variable refer to the global variable?

In Python if we want a function variable to refer as global variable, we can use global key word. it allows us to indicate that a variable inside the function should refer to the global variable with the same name

In [21]:
```
x = 10  # Global variable

def  global_var():
    global x  # Declare 'x' as global within the function
    x += 5   # Modify the global 'x' variable

print(f'Global Variable:{x}')  # Output: 10
global_var()
print(f'Local variable:{x}')  # Output: 15
```

```
Global Variable:10
Local variable:15
```

10. What is the data type of None?

In [22]:
```
type(None)
```

Out[22]:
```
NoneType
```

type of None is NoneType

1. What does the sentence import areallyourpetsnamederic do?

it would result in a ModuleNotFoundError.

1. If you had a bacon() feature in a spam module, what would you call it after importing spam?

import spam()

spam.bacon()

1. What can you do to save a programme from crashing if it encounters an error?

To prevent a programme from crashing we can use error handling techniques such as exceptional handling.

in Python, we can use a combination of try, except,else,finally blocks to implement exception handling.

Try: code that may raise an error except: code to handle specific exceptional type

else: code to execute if no exception occurred finally: Code that always executes, regardless of whether an exception occurred or not

In [23]:
```
try:
    result = 10 / 0  # Attempting to divide by zero
except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
```

```
Error: Division by zero is not allowed.
```

with the above program, instead of crashing the except block will print an error : Division by zero is not allowed

1. What is the purpose of the try clause? What is the purpose of the except clause?

Try: code that may raise an error except: code to handle specific exceptional type

In [ ]:
```

```