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

Answer=

Functions are advantageous to have in your programs because they allow for code reuse, modularization, and easier maintenance and debugging. Functions allow for the grouping of related code into a single, reusable block, which can be called from multiple places in the program. This helps to make the code more organized and easier to understand, as well as reducing the likelihood of introducing bugs. Additionally, functions can also make it easier to test and debug the code, as individual functional units can be tested and debugged separately.

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

Answer=

The code in a function runs when the function is called. Functions are defined with a specific block of code, but that code is not executed until the function is called. When the function is called, the code inside the function runs in the order that it is written, and any return value or output is generated. Once the function completes execution, control is returned to the calling code, which can then continue to execute the next instructions.

3. What statement creates a function?

Answer=

The def statement is used to create a function in Python. The def keyword is followed by the name of the function

def my\_function():

print("Hello, World!")

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

Answer=

A function is a block of code that can be reused throughout a program, and it contains one or more statements that are executed when the function is called. A function is defined using the def keyword and has a name, a set of parentheses and a colon.

A function call is the act of calling or executing a function. When a function is called, the code inside the function runs and any return value or output is generated. The function call is done by writing the function name followed by parentheses and any necessary arguments.

def my\_function(name):

print("Hello, " + name + "!")

my\_function("John") # this is a function call

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

Answer=

there is one global scope, but there can be many local scopes, one for each function call.

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

Answer=

when a function call returns, the local scope for that function call is destroyed and all the variables defined within that scope are no longer accessible, unless they are also defined in the global scope and have been modified by the function call.

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

Answer=

a return value is the value that a function returns when it completes execution, it can be used in further calculations or assigned to a variable for later use. It's possible to have a return value in an expression, where the expression is used as the return value of the function.

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

Answer=

if a function does not have a return statement, the return value of a call to that function will be None, which represents the absence of a value or a null value.

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

Answer=

To make a function variable refer to the global variable, you can use the global keyword to indicate that the variable should be accessed from the global scope, which allows the function to modify the global variable.

x = 10 # this variable is in the global scope  
  
def my\_Numb():  
 global x  
 x = x + 1  
 print(x)  
  
my\_Numb()  
print(x) # it will print 11

10. What is the data type of None?

Answer=

In Python, None is often used as the default value for variables that haven't been explicitly initialized with a value, or as the return value for functions that don't return any meaningful value.

None is a special value in Python that represents the absence of a value or a null value, it is not considered to be a data type but rather a special object of the built-in NoneType class.

11. What does the sentence import areallyourpetsnamederic do?

Answer=

It is a nonsensical sentence and does not have any specific meaning or function. It appears to be a random combination of words.

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

Answer=

If I had a bacon() feature in a spam module, I would call it spam.bacon() after importing the spam module. The syntax for calling a function within a module is "module\_name.function\_name()".

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

Answer=

Implement exception handling: Use try-except blocks to catch and handle specific exceptions that may occur in your program. This way, if an error occurs, the program can continue to execute instead of crashing.

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

Answer=

The purpose of the try clause is to define a block of code that may raise an exception. If an exception is raised within the try block, the code in the corresponding except block will be executed.

The purpose of the except clause is to handle exceptions that are raised within the corresponding try block. The code in the except block will be executed if an exception of the specified type is raised within the try block. If no exception is raised, or if an exception of a different type is raised, the code in the except block will not be executed.