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

Answer

- Functions allow you to define a block of code that can be executed repeatedly.
- Once a function is defined, it can be called from different parts of your program as many times as needed, avoiding code duplication
- Functions provide a way to break down a complex program into smaller, manageable parts
- By using functions, you can give meaningful names to blocks of code, making it easier for yourself and others to understand the purpose and functionality of each part.
- Functions help organize the code into logical units, making it easier to navigate and manage.
- You can group related code together within a function, promoting better code organization and reducing the chance of errors.

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

Answer

The code in a function runs when the function is called.

3. What statement creates a function?

Answer

def function_name(argument)

- *def:* It is the keyword used to indicate the start of a function definition.
- *function_name:* This is the name you choose to give to your function.
- *arguments*:These are optional parameters that you can define within the parentheses.

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

Answer

- **Function:** A function is a named block of code that defines a specific behavior or task.It encapsulates a set of instructions that perform a particular task or produce a specific result. Functions are defined using the def statement in Python and can take optional arguments and return values.
- **Function Call:** A function call is the action of invoking or executing a defined function at a specific point in the program.t involves using the function's name followed by parentheses `()` to indicate that you want to execute the code within the function.

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

Answer

- **Global Scope**: The global scope is the top-level scope in a Python program. It is accessible throughout the entire program and any variable defined in the global scope can be accessed from anywhere within the program. Variables defined in the global scope are often referred to as global variables. The global scope is created when the program starts and remains in existence until the program terminates.
- **Local Scopes**: Local scopes are created whenever a function or a code block is executed. These scopes are limited to the specific function or code block where they are created. Variables defined within a local scope are called local variables and are only accessible within that specific scope. Once the execution of the function or code block completes, the local scope is destroyed, and the local variables cease to exist.

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

Answer

When a function call returns in Python, the local scope associated with that function is destroyed, and the variables defined within that local scope cease to exist.

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

Answer

The concept of a return value refers to the value that a function can send back to the caller once it has completed its execution. When a function reaches a return statement, it terminates the function's execution and provides the specified value as the return value.In Python, a function can have a return value by using the return statement followed by an expression or a value.

```
def add_numbers(a, b):

return a + b

result = add_numbers(3, 4)  # Function call with return value

print(result)  # Output: 7
```

In this example, the **add_numbers()** function takes two arguments **a** and **b**. It adds these two values together using the + operator and returns the sum using the **return** statement. When the function is called with arguments **3** and **4**, the return value **7** is assigned to the variable **result**. Finally, the value of **result** is printed, resulting in the output **7**.

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 is **None**.

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

Answer

In Python, if you want to make a function variable refer to the global variable of the same name, you can use the **global** keyword within the function. The **global** keyword allows you to indicate that a variable within the function should refer to the global variable rather than creating a new local variable.

10. What is the data type of None?

Answer

The data type of **None** is actually its own unique type called **`NoneType`**

11. What does the sentence import areallyourpetsnamederic do?

Answer

If you attempt to run the statement "import areallyourpetsnamederic" in Python, it would result in a **ModuleNotFoundError**, indicating that the module named "areallyourpetsnamederic" does not exist.

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

Answer

After importing the **spam** module, you can access the **bacon()** feature by using the syntax **spam.bacon()**. This syntax allows you to call the **bacon()** function that is defined within the spam module.

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

Answer

Wrap the potentially error-raising code within a **try-except** block. The **try block** contains the code that might raise an exception, and the **except block** specifies the actions to take if an exception occurs.

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

Answer

The purpose of the **try** clause in a try-except block is to enclose the code that might raise an exception. The code within the **try** block is executed, and if an exception occurs during its execution, the control is transferred to the corresponding **except** block.

The purpose of the **except** clause is to define the actions to be taken when a specific exception or a set of exceptions occurs within the corresponding **try** block. The **except** block specifies the exception type(s) to catch, and if an exception of that type is raised, the code within the corresponding **except** block is executed.