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

Functions are advantageous because:

* **Reusability**: You can write a function once and call it multiple times, avoiding code duplication.
* **Organization**: Functions help organize code into smaller, more manageable pieces, improving readability and maintainability.
* **Abstraction**: Functions allow you to abstract away complex logic, making your code easier to understand and work with.
* **Modularity**: Functions can break down complex problems into smaller, focused tasks, making the program more modular.

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

The code in a function **runs when the function is called**. Defining a function only specifies what it does; it doesn't execute until you invoke it.

**3. What statement creates a function?**

The def statement is used to create (define) a function in Python.  
Example:

python

Copy

def my\_function():

print("Hello, world!")

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

* **Function**: A function is a block of code that performs a specific task when called. It is defined once using the def keyword.
* **Function call**: A function call is when you invoke a function to run the code inside it. For example, my\_function() is a function call that invokes the code in my\_function.

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

* **Global scope**: There is **one global scope** in a Python program, which refers to the top-level environment where variables are defined that are accessible throughout the program.
* **Local scopes**: There can be multiple local scopes in a program, typically created when functions or methods are defined. Each function has its own local scope.

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

When a function call returns, the variables in its local scope are **destroyed**. They only exist for the duration of the function call. After the function finishes execution, the local variables are no longer accessible.

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

* **Return value**: A return value is the value that a function sends back to the caller when it finishes execution. The return statement is used to specify this value. Example:

python

Copy

def add(x, y):

return x + y

result = add(2, 3) # result will be 5

* **Return value in an expression**: Yes, a function's return value can be used in an expression. For example:

python

Copy

def add(x, y):

return x + y

result = add(2, 3) \* 10 # The return value of add(2, 3) is used in an expression

**8. 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, it implicitly returns **None**. Example:

python

Copy

def no\_return():

print("This function has no return")

result = no\_return() # result will be None

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

To make a function refer to a global variable, you can use the global keyword inside the function. This allows the function to modify the global variable. Example:

python

Copy

x = 10

def change\_global():

global x

x = 20

change\_global()

print(x) # x is now 20

**10. What is the data type of None?**

The data type of None is **NoneType**. It is a special constant in Python used to represent the absence of a value or a null value.

**11. What does the sentence import areallyourpetsnamederic do?**

The sentence import areallyourpetsnamederic attempts to import a module named areallyourpetsnamederic. If the module with that name exists in the Python environment, Python will load it and allow you to use its functions, variables, and classes. If the module doesn't exist, it will raise an **ImportError**.

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

After importing the spam module, you would call the bacon() function using the syntax:

python

Copy

import spam

spam.bacon()

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

To save a program from crashing due to an error, you can use **exception handling** with try and except blocks. This allows you to handle errors gracefully instead of letting the program crash. Example:

python

Copy

try:

# Code that might raise an error

x = 1 / 0

except ZeroDivisionError:

print("You can't divide by zero!")

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

* **try clause**: The try block contains code that might raise an exception. If an exception occurs, Python will jump to the except block to handle the error.
* **except clause**: The except block contains the code to handle the exception. If an error occurs in the try block, Python will execute the code in the except block instead of crashing the program.

Example:

python

Copy

try:

x = 10 / 0 # This will raise a ZeroDivisionError

except ZeroDivisionError:

print("Cannot divide by zero!")