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

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

3. What statement creates a function?

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

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

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

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

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

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

10. What is the data type of None?

11. What does the sentence import areallyourpetsnamederic do?

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

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

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

**ANSWERS**

1. Functions are advantageous to have in your programs for several reasons:

- Reusability: You can define a function once and use it multiple times in your code, reducing redundancy and making your code more maintainable.

- Modularity: Functions allow you to break down a complex problem into smaller, manageable pieces, making your code easier to understand and debug.

- Encapsulation: Functions encapsulate a set of instructions, which can be helpful for abstracting complex operations and isolating their logic from the rest of the code.

- Readability: Using functions with descriptive names can make your code more readable and self-documenting.

2. The code in a function runs when the function is called, not when it's specified or defined. Functions are executed when they are invoked by their name in the code.

3. The `def` statement is used to create a function in Python. Here's the basic syntax:

```python

def function\_name(parameters):

# Function body

# Code to perform some task

```

4. A function is a named block of code that performs a specific task when called. A function call, on the other hand, is the action of invoking a function to execute its code. In simpler terms, a function is like a recipe, and a function call is when you actually cook the dish using that recipe.

5. In a Python program, there is one global scope, which is the outermost scope, and multiple local scopes. The number of local scopes depends on how many functions or code blocks (e.g., within loops or conditional statements) are defined in your program.

6. When a function call returns, the variables in its local scope are destroyed (with some exceptions, like closures). This means that the variables defined within the function are no longer accessible after the function has completed its execution.

7. A return value is the value that a function can send back to the caller. It allows functions to communicate results or data to the calling code. Yes, it is possible to have a return value in an expression, and you can use the returned value directly in other parts of your code.

8. If a function does not have a `return` statement, or if it has a `return` statement without a value, the function will return `None` by default. `None` represents the absence of a value in Python.

9. To make a function variable refer to a global variable, you can use the `global` keyword within the function. For example:

```python

global\_variable = 10

def my\_function():

global global\_variable

global\_variable = 20

my\_function()

print(global\_variable) # This will print 20

```

10. The data type of `None` in Python is `NoneType`.

11. The sentence `import areallyourpetsnamederic` is not a valid Python import statement. It seems to be a humorous or fictional module name. In Python, you would use the `import` statement to import external modules or libraries for use in your code.

12. If you had a `bacon()` function in a `spam` module, you could call it after importing `spam` like this:

```python

import spam

spam.bacon()

```

13. To save a program from crashing when it encounters an error, you can use exception handling. You can wrap the potentially problematic code in a `try` block and provide one or more `except` blocks to handle specific types of exceptions that may occur.

14. The `try` clause in exception handling is used to enclose the code that might raise an exception. Its purpose is to attempt the code within it and monitor for any exceptions. The `except` clause(s) are used to specify how to handle specific types of exceptions if they occur. If an exception is raised inside the `try` block, the corresponding `except` block is executed to handle the exception gracefully.