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

Reusability

Readability

Modularity

Testing

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 it is called, not when it is specified. When you define a function, you are creating a blueprint for the function. The code in the function is not executed until the function is called.

For example, the following code defines a function called f():

Code snippet

```
def f():
  print("Hello, world!")
```

3. What statement creates a function?

The def statement creates a function in Python. The syntax for defining a function is as follows:

Code snippet

```
def function_name(parameters):
  # Body of the function
```

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

In Python, a function is defined using the def keyword. For example, the following code defines a function called f():

Code snippet

```
def f():
  print("Hello, world!")
```

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

There is only one global scope in a Python program. The global scope is the top-level scope of the program. All variables that are not defined within a function are in the global scope.

There can be any number of local scopes in a Python program. A local scope is created when a function is called. All variables that are defined within a function are in the local scope of the function.

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

When a function call returns, the local variables in the function are destroyed.

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

A return value is the value that a function returns to the calling function when it completes its task

```
def square(x):
   return x * x
```

In this case, the return value of the square function is the square of the input x.

When the function is called, like result = square(5),

the value 25 is returned and stored in the variable result

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 will implicitly return a special value called "None" in Python. "None" is a built-in constant that represents the absence of a value.

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

To make a function variable refer to the global variable, you can use the global keyword. The global keyword tells Python that the variable you are declaring is a global variable, and not a local variable.

10. What is the data type of None?

The data type of None is NoneType. None is a special data type in Python that represents the absence of a value. It is not the same as 0, False, or an empty string.

11. What does the sentence import areallyourpetsnamederic do?

The sentence import areallyourpetsnamederic imports a module named areallyourpetsnamederic. This module does not exist by default, so you will need to create it before you can import it. Once you have created the module, you can add code to it that will be available to your other Python programs.

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

To call the bacon() feature in a spam module after importing spam, you would use the following code:

Code snippet

```
import spam

bacon()
```

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

There are a few things you can do to save a program from crashing if it encounters an error.

Use try-except blocks. A try-except block is a way of handling errors in Python. The basic syntax is:

Code snippet

```
try:

  # Do something that might raise an error

except Exception as e:

  # Handle the error
```

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

The try-except construct in Python is used for handling exceptions or errors that may occur during the execution of a block of code. It allows you to write code that can gracefully handle errors and continue executing the program, rather than abruptly terminating.

```
try:

    # Code that may raise an exception

    result = 10 / 0  # Division by zero

except ZeroDivisionError:

    # Code to handle the specific exception (ZeroDivisionError)

    print("Error: Division by zero occurred!")
```

In this code snippet, the try block contains the code 10 / 0, which attempts to perform division by zero, causing a ZeroDivisionError. The except block specifies the exception type ZeroDivisionError. If the exception occurs, the code within the except block is executed, printing the error message.