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

Functions allow code reusability, improve modularity, reduce redundancy, and make programs easier to debug and maintain.

Example:

```
def greet():
    print('Hello!')

# Call the function
greet()
```

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

The code in a function runs only when it is explicitly called.

Example:

```
def add(a, b):
    return a + b

# Call the function
result = add(3, 5)  # Function runs here.
```

3. What statement creates a function?

The 'def' statement creates a function.

Example:

```
def my_function():
    print('This is a function.')
```

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

A function is a block of code defined using 'def', while a function call executes it.

Example:

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

result = square(4)  # Function is called here.
```

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, but local scopes are created whenever a function is called.

Example:

```
x = 10  # Global scope
def local_example():
    y = 5  # Local scope
```

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

Variables in a local scope are destroyed when the function call completes.

Example:

```
def temporary_scope():
    temp = 42  # Exists only during this function call
```

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 output a function sends back after execution. Yes, return values can be used in expressions.

Example:

```
def multiply(a, b):
    return a * b

result = multiply(3, 4) + 2
```

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:

```
def no_return():
    print('No return value')

result = no_return()  # result is None
```

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

Use the 'global' keyword to refer to a global variable inside a function.

Example:

```
x = 5
def modify_global():
    global x
    x = 10
```

10. What is the data type of None?

The data type of None is NoneType.

Example:

```
result = None
print(type(result))  # Outputs: <class 'NoneType'>
```

11. What does the sentence import areallyourpetsnamederic do?

The statement imports a module named 'areallyourpetsnamederic', if it exists.

Example:

```
import areallyourpetsnamederic
```

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

Call the bacon() function from the spam module as follows:

Example:

```
import spam
spam.bacon()
```

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

Use try and except blocks to handle errors and prevent crashes.

Example:

```
try:
    result = 10 / 0
```

```
except ZeroDivisionError:
    print('Cannot divide by zero.')
```

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

The try clause contains the code that might raise an exception, and the except clause handles it.

Example:

```
try:
    print(10 / 0)
except ZeroDivisionError:
    print('Division by zero error.')
```