1. Why are functions advantageous to have in your programs?  
  
**Answer:** Functions are advantageous to have in programs for several reasons:

1. Code reusability: Functions allow you to reuse code by calling the function multiple times in your program. This makes your code more modular and easier to maintain. You don't have to write the same code over and over again in different parts of your program.

2. Encapsulation: Functions allow you to encapsulate complex logic into a single, easy-to-understand unit. This makes your code more readable and easier to understand.

3. Abstraction: Functions allow you to abstract away the implementation details of a piece of code. This means that you can focus on what the code does, rather than how it does it.

4. Testing: Functions are easier to test than large blocks of code. You can test a function in isolation to ensure that it works correctly before integrating it into your program.

5. Maintenance: Functions make it easier to maintain your code. If you need to change the behavior of a specific piece of code, you can modify the function rather than having to modify the entire program.

Overall, using functions can make your code more modular, easier to read, and easier to maintain, which can save you time and effort in the long run.

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

**Answer:** The code in a function runs when the function is called, not when it is specified.

3. What statement creates a function?  
  
**Answer:** In Python, you can create a function using the def statement.

4. What is the difference between a function and a function call?  
  
**Answer:** A function is a block of code that performs a specific task. It is defined using the def statement in Python, and can take input parameters and return output values.

A function call, on the other hand, is a statement that tells Python to execute the code inside a function.

5. How many global scopes are there in a Python program? How many local scopes?  
  
**Answer:** In Python, there is only one global scope per program. This global scope is created when the program starts running and contains all the variables and functions that are defined at the top level of the program.

On the other hand, local scopes are created whenever a function is called. Each function call creates its own local scope, which contains the variables and functions defined within that function. These local scopes are destroyed when the function call completes, and any variables defined within the function are removed from memory.

6. What happens to variables in a local scope when the function call returns?  
  
**Answer:** When a function call returns, the local scope created by that function is destroyed, and all the variables defined within that scope are removed from memory. These variables are no longer accessible to the main program or any other functions that may be called after the function that defined them has returned.

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 is an important one in programming. In Python, when a function is called, it can optionally return a value to the caller. The return value is the result of the function's computation, and can be any type of object, including numbers, strings, lists, dictionaries, or even other functions.

To return a value from a function in Python, you use the return statement followed by the value you want to return.  
  
It is also possible to use a return value as part of an expression. For example:

def add\_numbers(a, b):

return a + b

result = add\_numbers(2, 3) \* 4

print(result) # Output: 20

In this example, the add\_numbers function returns the sum of its two arguments a and b. The return value of the function (5) is then multiplied by 4 in the expression add\_numbers(2, 3) \* 4, resulting in the value 20, which is stored in the result variable. The print statement then outputs the value of result to the console.

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 in Python does not have a return statement, the function will still execute any code contained within it, but it will not return any value to the caller. When a function without a return statement is called, the return value of the function call will be None.

9. How do you make a function variable refer to the global variable?  
  
**Answer:** In Python, you can use the `global` keyword to make a function variable refer to a global variable with the same name. This is useful when you need to modify a global variable from within a function.

x = 42 # global variable

def my\_function():

global x # use the "global" keyword to indicate that we want to use the global variable "x"

x = 10 # modify the global variable "x" within the function

print("x inside function:", x)

my\_function()

print("x outside function:", x) # Output: 10

```

In this example, we define a global variable `x` with the value `42`. We then define a function `my\_function` that modifies the global variable `x` by using the `global` keyword. The function sets the value of `x` to `10`, and then prints its value. When the function is called, it modifies the global variable `x` and sets its value to `10`.

After the function call, we print the value of `x` again, which now has the value of `10` that was set inside the function. Without using the `global` keyword inside the function, `x` would have remained a local variable within the function and would not have affected the value of the global variable `x` outside the function.

10. What is the data type of None?

**Answer:** In Python, None is a special constant value that represents the absence of a value. It is commonly used to indicate that a variable or function has no value or has not been initialized.

None is considered to be its own data type, which is called NoneType. It is a built-in type in Python, and is the only instance of the NoneType class.

11. What does the sentence import areallyourpetsnamederic do?

**Answer:** The sentence import areallyourpetsnamederic is a syntactically valid Python statement, but it does not have any built-in functionality in Python.

In this case, areallyourpetsnamederic is not a built-in Python module or a commonly used third-party module, so attempting to import it will likely result in a ModuleNotFoundError indicating that the module cannot be found.

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

**Answer:** If you have a function called bacon() in a module called spam, you can access the function after importing the spam module using dot notation.

13. What can you do to save a programme from crashing if it encounters an error?  
  
**Answer:** There are several things you can do to save a program from crashing if it encounters an error:

1. Use exception handling: You can use try-except blocks to catch and handle exceptions that occur during program execution. By catching exceptions and handling them gracefully, you can prevent your program from crashing.

2. Use input validation: If your program accepts user input, you can validate the input to ensure that it meets certain criteria before processing it. This can prevent your program from crashing due to invalid input. For example, you can check that user input is within a certain range, or that it is of the correct data type.

3. Use defensive programming techniques: You can use defensive programming techniques to write code that is more resilient to errors. This can include things like checking that variables have valid values before using them, avoiding hard-coded values that can cause errors, and using default values or fallbacks when appropriate.

4. Log errors: You can log errors that occur during program execution to a file or a database, rather than crashing the program. This can help you identify and fix issues without interrupting program execution.

By implementing these techniques, you can help ensure that your program is more resilient to errors and less likely to crash.

14. What is the purpose of the try clause? What is the purpose of the except clause?  
  
**Answer:** The try and except clauses are used together in Python to implement exception handling.

The purpose of the try clause is to enclose a block of code that may raise an exception. The try block is executed normally, and if an exception occurs during the execution of the block, the normal flow of execution is interrupted, and the program jumps to the except block.

The purpose of the except clause is to define how to handle the exception that was raised in the try block. The except block specifies the type of exception to catch, and the code to execute when that exception is caught. If an exception of the specified type is raised in the try block, the program jumps to the except block and executes the code specified in the block. If no exception is raised, the except block is skipped.