Assignment-3

1. Advantages of functions:

- **Reusability:** Functions allow you to write code once and use it multiple times within your program or even in other programs. This saves time and reduces redundancy.
- **Modularity:** By breaking down your program into smaller, well-defined functions, you improve code organization and readability. This makes it easier to understand, maintain, and modify your code.
- **Encapsulation:** Functions can encapsulate specific logic, making it easier to control data access and prevent unintended modifications. This promotes better code structure and reduces errors.
- **Testability:** Functions can be tested independently, which simplifies debugging and ensures the correctness of your code.

2. When does function code run?

The code within a function definition is executed only when the function is called. Defining a function simply creates a blueprint for the code to be executed later.

3. Creating a function:

The **def** keyword is used to define a function in Python. Here's the basic structure:

```
def function_name(parameters):
    return value
```

4. Difference between function and function call:

- **Function:** A named block of code that performs a specific task. It's a blueprint for how to execute the code.
- **Function call:** The act of executing the code defined within a function. It involves providing any necessary arguments to the function.

5. Global and Local Scopes:

- **Global scope:** The global scope encompasses the entire Python program. Variables defined outside any function have global scope. There is only one global scope per program.

- **Local scope:** Each function has its own local scope. Variables defined within a function are only accessible within that function. Multiple functions can have their own local scopes with the same variable name, but they won't conflict.

6. Local variables after function return:
When a function call finishes execution (returns), all local variables within that function's scope are destroyed. They are no longer accessible.

7. Return value:
- **Concept:** A function can optionally return a value using the `return` statement. This value is passed back to the place where the function was called.
- **Return value in expressions:** No, a return value cannot be directly used within an expression in Python. However, you can assign the returned value to a variable and then use that variable in an expression.

8. Function without a return statement:
If a function doesn't have a `return` statement, it implicitly returns `None`. `None` is a special data type in Python that signifies the absence of a meaningful value.

9. Making a function variable refer to the global variable:
To access a global variable from within a function, you can use the `global` keyword before declaring the variable inside the function. However, it's generally recommended to avoid using global variables excessively as it can lead to less maintainable code.

10. Data type of None:
`None` is a special data type in Python called `NoneType`. It represents the absence of a value.

11. import areallyourpetsnamederic:
This statement attempts to import a module named areallyourpetsnamederic. If such a module doesn't exist, Python will raise an ImportError. It's a hypothetical example, and you'd typically use a valid module name for import.

12. Calling `bacon()` from `spam` module:

```
import spam

spam.bacon()
```

13. Error handling:

You can use `try-except` blocks to handle potential errors in your program. This prevents your program from crashing and allows you to provide meaningful feedback or take alternative actions.

- **`try` clause:** This block contains the code you expect to execute normally.
- **`except` clause:** This block handles exceptions (errors) that may occur within the `try` block. You can specify the type of exception to handle or use a general `except` clause.

14. Purpose of `try` and `except`:

- **`try` clause:** Attempts to execute the code within its block.
- **`except` clause:** If an error occurs during the execution of the `try` block, control jumps to the first matching `except` clause. You can then handle the error gracefully and continue execution.

By using `try-except` blocks, you can make your code more robust and user-friendly.