Q1. Permissible use of multiple import statements for the same module:

- Yes, it is allowed. The goal might be to use different names or aliases for the module in different parts of the code. It could be beneficial for code readability.

Q2. Characteristics of a module:

- Encapsulation of code into a file or set of files, promoting code organization and reusability.

Q3. Avoiding circular importing:

- Restructure the code to minimize mutual dependencies.

- Use the `importlib` module to import modules dynamically within functions or methods.

Q4. Purpose of \_ \_all\_ \_ in Python:

- It is a list of names defined in a module, indicating which names should be imported when using `from module import \*`. It helps control what gets imported to avoid importing unwanted names.

Q5. Useful situations for referring to \_ \_name\_ \_ or '\_ \_main\_ \_':

- Checking if the module is being run as the main program (`if \_\_name\_\_ == "\_\_main\_\_":`). It allows code to be executed only when the script is run directly, not when imported as a module.

Q6. Benefits of attaching a program counter to the RPN interpreter:

- Allows tracking and controlling the execution state of the RPN script, facilitating debugging and providing insights into the interpreter's behaviour.

Q7. Minimum expressions or statements for a basic RPN interpreter:

- Stack manipulation commands (push, pop).

- Arithmetic and logical operations.

- Conditional statements (if-else).

- Looping constructs (for, while).

- I/O operations for user interaction.

- Function definition and invocation.