Q1. Yes, it is permissible to use several import statements to import the same module. The goal might be to use different names for the module in different parts of the code. For example, you might import the `math` module using the name `m` in one part of the code and `mathematics` in another part of the code.

Q2. Modules are files containing Python definitions and statements. They allow code to be reused, organised and shared. One characteristic of a module is that it can be imported and used in other programs or modules using the `import` statement.

Q3. To avoid circular importing, you can refactor the code to eliminate the circular dependency. This can be done by moving the shared functionality to a third module that can be imported by both modules, or by reorganising the code so that the import dependency is one way instead of two ways.

Q4. The `\_\_all\_\_` variable is used to define the public API of a module. It lists the names of the objects that are intended to be used by clients of the module. When a client uses the `from module import \*` syntax, only the names listed in the `\_\_all\_\_` variable will be imported.

Q5. The `\_\_name\_\_` attribute is useful in situations where a module can be run as either the main program or imported as a module in another program. The value of `\_\_name\_\_` is set to `'\_\_main\_\_'` when the module is run as the main program, and to the name of the module when it is imported. This allows the module to perform different actions depending on whether it is run as the main program or imported.

Q6. Attaching a program counter to the RPN interpreter application can provide several benefits, such as allowing for the implementation of conditional statements and loops. It can also allow for better error reporting and debugging by providing information about the current position in the script.

Q7. The minimum expressions or statements needed to make a basic programming language like RPN primitive but complete are:

- Arithmetic operations (addition, subtraction, multiplication, division)

- Conditional statements (if-else)

- Looping constructs (for, while)

- Variables and assignments

- Input/output statements (print, input)

- Functions or subroutines