Q1. Is it permissible to use several import statements to import the same module? What would the
goal be? Can you think of a situation where it would be beneficial?

Q2. What are some of a module's characteristics? (Name at least one.)

Q3. Circular importing, such as when two modules import each other, can lead to dependencies and
bugs that aren't visible. How can you go about creating a program that avoids mutual importing?

Q4. Why is _ _all_ _ in Python?

Q5. In what situation is it useful to refer to the _ _name_ _ attribute or the string '_ _main_ _'?

Q6. What are some of the benefits of attaching a program counter to the RPN interpreter application, which interprets an RPN script line by line?

Q7. What are the minimum expressions or statements (or both) that you'd need to render a basic
programming language like RPN primitive but complete— that is, capable of carrying out any computerised task theoretically possible?

A1. Yes, it is permissible to use several import statements to import the same module in Python. The goal might be to provide different names for the same module or to import only specific components of the module in different parts of the program. One situation where it might be beneficial is when dealing with large codebases where different parts of the program require different subsets of a module.

A2. Some characteristics of a module in Python are:

- A module is a file containing Python definitions and statements.
- The module name is the same as the filename without the .py extension.
- A module can define functions, classes, and variables.
- A module can be imported and used in other Python code.

A3. One way to avoid circular importing in Python is to refactor the code so that the circular dependency is eliminated. This can involve moving some code to a new module or redesigning the dependencies between modules. Another way is to use the lazy import technique, where one of the modules is imported inside a function or a method rather than at the top-level of the importing module.

A4. The `__all__` variable in Python is used to define a list of names that should be considered public API of a module. When a module is imported using the `from module import *` syntax, only the names listed in `__all__` are imported. This helps to prevent namespace pollution and makes it clear which names are intended to be used externally.

A5. The `__name__` attribute in Python is a special variable that is automatically set to the name of the current module. When a module is run as the main program, the `__name__` attribute is set to the string `"__main__"`. This allows the module to check if it is being imported or run directly as the main program and take appropriate actions.

A6. Attaching a program counter to an RPN interpreter application can provide benefits such as:

- Easier debugging: With a program counter, it is easier to identify which line of code caused an error.
- Improved error reporting: A program counter can be used to generate more informative error messages.
- Better performance: By keeping track of the current line of code, an RPN interpreter can avoid re-evaluating lines of code that have already been executed.

A7. To create a basic programming language like RPN, the minimum expressions or statements needed would include:

- Arithmetic operations (addition, subtraction, multiplication, division)
- Conditional statements (if-else)
- Looping constructs (for and while loops)
- Input/output statements (read and write)
- Variable assignment and retrieval.