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?

Ans=>we can use several import statement for reinitializing the module

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

Ans=>we can create pacakage of it and we can call it anywhere and store.

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?

Ans=>import both modules separately don’t call them inside another.

Q4. Why is \_ \_all\_ \_ in Python?

Ans=>It declares the semantically "public" names from a module. If there is a name in \_\_all\_\_, users are expected to use it, and they can have the expectation that it will not change.

Q5. In what situation is it useful to refer to the \_ \_name\_ \_ attribute or the string '\_ \_main\_ \_'?

Ans=>\_\_name\_\_ It gets its value depending on how we execute the containing script

If you import this script as a module in another script, the \_\_name\_\_ is set to the name of the script/module.

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?

Ans=>

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?

Ans=>