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?

Yes we can use several import statements in the same program for a same module. But we need to importing the different functions from the same module. If we do importing same module several time without specifying any particular functions to be imported, then, we will only be importing the same function.

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

Python Modules are simply stored with ‘.py’ extension. It has list of functions performing many different operations that can be used by just calling their names and passing the necessary parameters to them.

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?

To avoid circular importing error we need to first instantiate first module object and then use it by calling second module.

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

\_\_all\_\_ is not used for importing. Instead, we use **from <module\_name> import \*** to import all the functions from the python module.

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

When we need to run top level code in python we use \_\_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?

The benefit of including PC in RPN interpreter is to remember the calculation of the RPN interpreter as RPN interpreter does not stores any address to perform its operations.

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?