**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&#39;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&#39;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 &#39;\_ \_main\_ \_&#39;?**

**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&#39;d need to render a basic**

**programming language like RPN primitive but complete— that is, capable of carrying out any**

**computerised task theoretically possible?**

**SOLUTIONS**

*1. Yes, it is permissible to use several import statements to import the same module. The goal would be to access different attributes or functions of the module using different names. This is beneficial when you want to avoid naming conflicts or when you want to use shorter names to refer to attributes or functions of the module.*

*2. Some characteristics of a module in Python include:*

* *A module is a file containing Python definitions and statements.*
* *A module can define functions, classes and variables.*
* *A module can be imported into other Python scripts.*

*3. To avoid circular importing, you can refactor the modules so that the code that causes the circular importing is moved to a new module, which can then be imported by the modules that need it. Another approach is to use lazy importing, which means that the module is imported only when it is needed, instead of importing it at the beginning of the script.*

*4. The \_ all \_ variable in Python is used to specify which names should be imported when a user does "from module import \*". It helps to prevent importing unwanted or unnecessary names and can improve code readability.*

*5. The \_ name \_ attribute or the string '\_ main \_' is useful when you want to check if a Python script is being run as the main program or if it is being imported as a module into another script. This is often used to create modules that can be run as a script, but also imported into other scripts as a library.*

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

* *Ability to track the progress of the interpreter through the RPN script, making it easier to debug errors.*
* *Ability to implement advanced features like breakpoints and step-by-step execution, allowing for more fine-grained control over the execution of the script.*
* *Ability to optimize the execution of the script by identifying frequently executed code and optimizing it.*

*7. The minimum expressions or statements needed to render a basic programming language like RPN primitive but complete would include:*

* *Expressions for arithmetic operations (addition, subtraction, multiplication, division, etc.)*
* *Expressions for conditional statements (if/else)*
* *Expressions for loops (while, for)*
* *Variables and assignments*
* *Input/output functions*
* *Functions and procedures*