**Exercise 1.1:** explain the differences between interactive mode and script mode in Python.

In Python, there are two primary ways to interact with the language and execute code: interactive mode and script mode.

These modes serve different purposes and have distinct characteristics.

Interactive Mode:

Interactive mode is also known as the Python REPL (Read-Eval-Print Loop).

In this mode, you can enter Python commands and statements one at a time, and the interpreter immediately executes them and displays the results.

It is an excellent environment for quick experimentation, testing code snippets, and learning Python, as you can see the output immediately after entering each line.

Interactive mode is often used for debugging and exploring Python modules and libraries.

Script Mode:

Script mode is used for creating Python programs or scripts.

In this mode, you write a sequence of Python statements in a text file (usually with a '.py' extension) and then execute the entire script as a single unit.

You typically use a text editor or integrated development environment (IDE) to write and edit your Python scripts.

To run a Python script, you need to execute it from the command line or an integrated development environment.

Differences:

Interactive mode allows you to enter and execute Python code line by line, providing immediate feedback, while script mode involves writing a program in a file and executing it as a whole.

Interactive mode is great for quick testing and exploration, while script mode is used for creating larger, more structured programs.

Script mode is typically used for building standalone Python applications, running batch processes, and creating reusable code modules.

**Exercise 2.1:** explain the concept of data types in Python.

In Python, data types are classifications that specify which type of value a particular variable can hold.

Data types define the characteristics of data and determine how the data can be used and manipulated.

Python is dynamically typed, which means you don't need to explicitly declare the data type of a variable; it is determined automatically based on the value assigned to it.

Python supports several built-in data types, which can be categorized into the following main groups:

1.Numeric Types:

* int: Represents integers, such as 3, -42, or 0.
* float: Represents floating-point numbers, including decimal numbers, like 3.14 or -0.5.
* complex: Represents complex numbers in the form a + bj, where a and b are real numbers, and j is the imaginary unit.

2.Text Type:

* str: Represents strings of characters, like "Hello, World!" or 'Python'.

3.Boolean Type:

* bool: Represents Boolean values, either True or False. Booleans are used for logical operations and control flow.

4.Sequence Types:

* list: Represents ordered, mutable sequences. You can create lists containing various data types.
* tuple: Represents ordered, immutable sequences. Once you define a tuple, you cannot change its elements.
* range: Represents an immutable sequence of numbers, often used in loops.

5.Mapping Type:

* dict: Represents a dictionary, which is an unordered collection of key-value pairs. Each key is unique, and you can use it to access the corresponding value.

6.Set Types:

* set: Represents an unordered collection of unique elements. Sets are used for tasks like removing duplicates or testing membership.
* frozenset: Similar to sets but immutable.

7.Binary Types:

* bytes: Represents a sequence of bytes (immutable).
* bytearray: Represents a mutable sequence of bytes.
* memoryview: Represents a memory view of a sequence of bytes.

8.None Type:

* NoneType: Represents a special value None, used to indicate the absence of a value or a null value.

**Exercise 4.1:** explain arithmetic operators.

* Addition (+):

The addition operator is used to add two or more numbers.

* Subtraction (-):

The subtraction operator is used to subtract one number from another.

* Multiplication (\*):

The multiplication operator is used to multiply two or more numbers.

* Division (/):

The division operator is used to divide one number by another. In Python 3, this returns a float.

* Modulus (%):

The modulus operator returns the remainder of the division of one number by another.

* Exponentiation (\*\*):

The exponentiation operator raises a number to a specified power.

* Floor Division (//):

The floor division operator returns the largest integer less than or equal to the result of dividing one number by another.

**Exercise 4.2:**

Order of Operations in Python, operations follow the standard order of operations (PEMDAS):

1. Parentheses
2. Exponents
3. Multiplication and Division
4. Addition and Subtraction