part-3

Complex Numbers in Python

Introduction to Data Types

- we can work with different data types, such as integers, floats, and complex numbers.
- We can assign values to variables and check their data types using the `type()`function.

Integer Data Type

- We can assign an integer value to a variable, such as `num = 7`
- The `type(num)`function will return `int` as the data type

Float Data Type

- We can assign a decimal number to the `num` variable, such as `num = 7.`.
- The `type(num)` function will return `float` as the data type.

Complex Data Type

- We can assign a complex number to the `num` variable.
- The `type(num)`function will return `complex` as the data type.

Accessing Real and Imaginary Parts of Complex Numbers

- To get the real part of a complex number, we can use the `num.real` attribute.
- To get the imaginary part of a complex number, we can use the `num.imag` attribute.
- Both the real and imaginary parts are stored as `float` data types.

Arithmetic Operations with Data Types

- We can perform various arithmetic operations, such as addition, subtraction, multiplication, and division, with differe nt data types.

- The output data type may change depending on the operation performed.

Addition

- We can add two integers or floats using the `+` operator.
- The output data type will be the same as the input data types.

Subtraction

- We can subtract two integers or floats using the `-`operator.
- The output data type will be the same as the input data types.

Multiplication

- We can multiply two integers or floats using the `*`operator.
- The output data type will be the same as the input data types.

Division

- We can divide two integers or floats using the `/`operator.
- The output data type will be a `float`, even if the input data types are integers.

Integer Division

- To get an integer output for division, we can use the `//`operator.
- This will perform floor division, rounding down the result to the nearest integer.

Exponentiation

- We can raise a number to a power using the `**`operator.
- This will perform the exponentiation operation.

Modulus

- We can find the remainder of a division operation using the `%`operator.
- This will return the remainder of the division

Arithmetic Operations with Complex Numbers

- We can perform arithmetic operations, such as division, with complex numbers.

- The output will be a complex number.

Type Conversions in Python

- Python supports both implicit and explicit type conversions.
- Implicit conversions happen through arithmetic operations, while explicit conversions use built-in functions.

Explicit Type Conversions

- We can convert a string representation of a number to an integer using the `int()`function.
- We can convert a number to a float using the `float()`function.
- We can convert a number to a complex number using the `complex()`function.

Implicit Type Conversions

- When performing arithmetic operations, Python will automatically convert the data types to ensure the operation ca n be performed.
- For example, adding an integer and a float will result in a float output.

Representing Binary, Hexadecimal, and Octal Numbers

- In Python, we can represent integer numbers in binary, hexadecimal, and octal formats.
- We can use the prefixes `0b`, `0x`, and `0o` to represent binary, hexadecimal, and octal numbers, respectively.

Working with Fractions

- Python has a `fractions` module that allows us to work with fractions.
- We can perform operations like addition, subtraction, and finding the reciprocal of fractions.

Number Functions in Python

- Python provides various built-in functions for working with numbers, such as `abs()`, `math.pi`, `math.e`, `math.exp()`, `math.log()`, `math.factorial()`, and `math.sqrt()`
- These functions allow us to perform common mathematical operations on numbers.

Trigonometric Functions

- Python's `math` module provides trigonometric functions like `math.cos()`and `math.tan()`

- These functions take angles in radians as input and return the corresponding trigonometric values.

Random Number Functions

- Python's `random` module provides functions like `random.randrange()` and `random.choice()`
- to generate random numbers and select random elements from sequences.

Function	Description
`random.randrange(start, stop, step)`	Generates a random integer within the specified range
`random.choice(sequence)`	Selects a random element from the given sequence