Numeric Data Types

In Python, numeric data types represent numerical values and are fundamental for performing mathematical operations. The primary numeric data types in Python include integers, floating-point numbers, and complex numbers. Here are details about each of these numeric data types:

1. **Integers (`int`):**
   - Integers are whole numbers without a decimal point.
   - Example:

     ```python
     x = 10
     y = -5
     ```

   - Python supports arbitrarily large integers, allowing you to work with very large or very small numbers.

2. **Floating-Point Numbers (`float`):**
   - Floating-point numbers, or floats, represent real numbers and can have decimal points or be written in scientific notation.
   - Example:

     ```python
     a = 3.14
     b = -2.5e2  # Scientific notation: -250.0
     ```

   - Floating-point arithmetic may involve some rounding errors due to the representation limitations in the computer.

3. **Complex Numbers (`complex`):**
   - Complex numbers are written in the form `a + bj`, where `a` and `b` are real numbers, and `j` represents the imaginary unit.
   - Example:

     ```python
     z = 2 + 3j
     ```

   - Complex numbers support arithmetic operations like addition, subtraction, multiplication, and division.

4. **Numeric Operations:**
   - Python supports a wide range of numeric operations for integers and floating-point numbers, including addition (`+`), subtraction (`-`), multiplication (`*`), division (`/`), modulus (`%`), exponentiation (`**`), and floor division (`//`).

```python
    a = 5
    b = 2

    addition = a + b       # Result: 7
    subtraction = a - b    # Result: 3
    multiplication = a * b # Result: 10
    division = a / b       # Result: 2.5
    modulus = a % b        # Result: 1
    exponentiation = a ** b # Result: 25
    floor_division = a // b # Result: 2
    ```

5. **Conversion Between Numeric Types:**
   - You can convert between numeric types using built-in functions like `int()`, `float()`, and `complex()`.

    ```python
    x = 10
    y = float(x)      # Convert to float: 10.0
    z = complex(x)    # Convert to complex: (10+0j)
    ```

6. **Type Checking:**
   - The `type()` function can be used to check the type of a variable.

    ```python
    num = 3.14
    print(type(num))  # Result: <class 'float'>
    ```

7. **Math Module:**
   - Python's `math` module provides additional mathematical functions for more complex operations.

    ```python
    import math

    square_root = math.sqrt(25)    # Result: 5.0
    logarithm = math.log10(100)    # Result: 2.0
    trig_function = math.sin(0)    # Result: 0.0
    ```

8. **Random Module:**
   - The `random` module provides functions for generating random numbers.

    ```python
    import random

    random_number = random.randint(1, 10)  # Generates a random integer between 1 and 10
    ```

9. **Decimal Module:**
   - The `decimal` module is useful for precise decimal arithmetic.

   ```python
   from decimal import Decimal

   a = Decimal('0.1')
   b = Decimal('0.2')
   result = a + b  # Result: Decimal('0.3')
   ```

Understanding and working with Python's numeric data types is essential for performing calculations, writing mathematical algorithms, and handling various types of numerical data in your programs.