Python Data Types Explained

Why Do We Need Data Types in Python?

1. Efficient Memory Management:

   Python assigns memory based on the type and size of the value.

2. Type-Specific Operations:

   Data types define what operations are valid (e.g., you can add integers but not strings with numbers).

3. Clarity and Debugging:

   Helps in understanding and debugging code more easily.

4. Optimization:

   Python internally optimizes operations based on data types for performance.

Built-in Data Types in Python

1. Numeric Types

int: Integer values (e.g., 10)

float: Decimal values (e.g., 3.14)

complex: Complex numbers (e.g., 2+3j)

2. Text Type

str: A sequence of Unicode characters (e.g., 'Hello')

3. Sequence Types

list: Mutable sequence (e.g., [1, 2, 3])

tuple: Immutable sequence (e.g., (4, 5, 6))

range: Sequence of numbers (e.g., range(5))

4. Set Types

Python Data Types Explained

set: Unordered collection of unique elements (e.g., {1, 2, 3})

frozenset: Immutable set

5. Mapping Type

dict: Collection of key-value pairs (e.g., {'name': 'Alice'})

6. Boolean Type

bool: Represents True or False

7. Binary Types

bytes: Immutable byte data (e.g., b'hello')

bytearray: Mutable byte data

memoryview: View of byte data

8. None Type

NoneType: Represents the absence of value (e.g., a = None)

Example Code

```
a = 10          # int
b = 3.14        # float
c = 'Hello'     # str
d = [1, 2, 3]   # list
e = {'a': 1}    # dict
f = True        # bool
g = None        # NoneType

print(type(d))   # Output: <class 'list'>
```