Lesson Plan:

Predefined Objects in Python

PW SKILLS

There are several predefined objects and data types in Python. These objects are readily available for use without the need for any additional imports. Here are some of the most common predefined objects in Python

1. Numeric Types:

- int: Integer numbers **(e.g., 42, -10).**
- float: Floating-point numbers **(e.g., 3.14, -0.5).**
- complex: Complex numbers with real and imaginary parts **(e.g., 2 + 3j).**

2. Sequence Types:

- str: Strings of characters **(e.g., "hello", 'world').**
- list: Mutable sequences of elements **(e.g., [1, 2, 3])**.
- tuple: Immutable sequences of elements **(e.g., (1, 2, 3))**.

3. Mapping Types:

- dict: Mutable collections of key-value pairs **(e.g., {'a': 1, 'b': 2}).**

4. Set Types:

- set: Mutable collections of unique elements **(e.g., {1, 2, 3}).**
- frozenset: Immutable collections of unique elements **(e.g., frozenset({1, 2, 3})).**

5. Boolean Type:

- bool: Boolean values representing True or False.

6. NoneType:

- None: A special object representing the absence of a value or a null value.

In the above you came across these two terminologies: **Mutable** and **Immutable**. Let's understand what they really mean:

Mutable	Immutable
Objects whose state can be modified after creation are called mutable objects.	Objects whose state cannot be modified after creation are called immutable objects.
Lists, dictionaries, sets, and user-defined classes (unless explicitly designed to be immutable) are examples of mutable objects.	Examples include integers, floating-point numbers, strings, tuples, and frozensets.
Mutable objects can be altered in place; that is, you can change their contents without creating a new object.	Immutable objects cannot be changed once they are created. Any operation that appears to modify an immutable object actually creates a new object.