🧑‍🏫 Lecture Topic: Python Data Types

🧠 What are Data Types in Python?

In Python, **data types** specify the type of value a variable holds. Since Python is **dynamically typed**, you don't need to declare a variable's type explicitly — Python infers it based on the value assigned.

Example:

```
x = 10     # Python knows this is an integer
y = "Hi"   # Python knows this is a string
```

🧩 Categories of Python Data Types

Python's built-in data types are mainly categorized into:

1. **Numeric Types**
2. **Text Type**
3. **Boolean Type**
4. **Sequence Types**
5. **Set Types**
6. **Mapping Type**
7. **Binary Types** (Advanced – optional for beginners)

1️⃣ Numeric Data Types

➢ `int`: Integer numbers

Used for whole numbers (positive/negative)

```
x = 100
print(type(x))   # Output: <class 'int'>
```

➢ `float`: Decimal numbers

Used for real numbers with decimals.

```
price = 99.99
print(type(price))  # Output: <class 'float'>
```

➢ `complex`: Complex numbers

Used for mathematical operations with imaginary parts.

```
z = 2 + 3j
print(type(z))  # Output: <class 'complex'>
```

2️⃣ Text Type

➢ `str`: String

Used to store text (characters, sentences, etc.)

```
name = "Azhar"
message = 'Welcome to Python!'
print(type(name))  # Output: <class 'str'>
```

💡 **Strings are immutable (cannot be changed after creation).**

3️⃣ Boolean Type

➢ `bool`: True or False

Used for decision-making in conditional logic.

```
is_active = True
is_admin = False
print(type(is_active))  # Output: <class 'bool'>
```

4️⃣ Sequence Types

➢ `list`: Ordered, changeable collection

Can hold mixed data types.

```
fruits = ["apple", "banana", "cherry"]
fruits.append("mango")
print(fruits)  # ['apple', 'banana', 'cherry', 'mango']
```

➢ `tuple`: Ordered, unchangeable collection

Faster and used for fixed data.

```
colors = ("red", "green", "blue")
print(colors[0])  # Output: red
```

➢ `range`: Immutable sequence of numbers

Mostly used in loops.

```
r = range(5)
print(list(r))  # Output: [0, 1, 2, 3, 4]
```

5️⃣ Set Type

➢ `set`: Unordered, unique items

Used to remove duplicates.

```
nums = {1, 2, 3, 4, 4, 2}
print(nums)  # Output: {1, 2, 3, 4}
```

✅ Useful in checking membership:

```
print(3 in nums)  # True
```

6️⃣ Mapping Type

➤ `dict`: Key-value pairs

Like a real-world dictionary with lookup capability.

```
person = {
    "name": "Azhar",
    "age": 24,
    "is_student": True
}
print(person["name"])  # Output: Azhar
```

7️⃣ Type Casting (Type Conversion)

You can convert between types manually:

```
a = 5
b = float(a)   # 5.0
c = str(a)     # "5"
```

🔍 Summary Table

Type	Example	Mutable	Ordered	Use Case
`int`	`x = 10`	❌	❌	Counting, indexing
`float`	`price = 10.5`	❌	❌	Financial, scientific apps

`str`	`name = "Ali"`	❌	✅	Text messages, input
`bool`	`is_true = False`	❌	❌	Logic, conditions
`list`	`nums = [1, 2, 3]`	✅	✅	Collections, iteration
`tuple`	`t = (1, 2)`	❌	✅	Read-only data
`set`	`{1, 2, 3}`	✅	❌	Unique values
`dict`	`{"key": "value"}`	✅	✅*	Key-based lookup

🧪 Practice Exercise

1. Create a list of 5 of your favorite foods.
2. Convert the list to a set.
3. Add one new food using `.add()`.
4. Create a dictionary with keys: "name", "age", "city".
