**Data Types in Python**

A **data type** defines the kind of value a variable can hold and the operations that can be performed on it. Data types are essential because they determine **how data is stored in memory** and **how it can be processed**.

Python provides several **built-in data types**:

**🔹 1. Numeric Data Types**

* **int**: Whole numbers. Example: x = 5
* **float**: Numbers with decimals. Example: y = 3.14
* **complex**: Numbers with real and imaginary parts. Example: z = 2 + 3j

**🔹 2. Sequence Types**

* **str**: Strings (sequence of characters). Example: text = "Hello"
* **list**: Ordered, mutable collections. Example: my\_list = [1, 2, 3]
* **tuple**: Ordered, immutable collections. Example: my\_tuple = (1, 2, 3)

**🔹 3. Mapping Type**

* **dict**: Key–value pairs. Example: student = {"name": "John", "age": 30}

**🔹 4. Set Types**

* **set**: Unordered collection of unique elements. Example: my\_set = {1, 2, 3}
* **frozenset**: Immutable version of a set. Example: f\_set = frozenset([1, 2, 3])

**🔹 5. Boolean Type**

* **bool**: Represents True or False. Example: is\_valid = True

**🔹 6. Binary Types**

* **bytes**: Immutable byte sequences. Example: data = b"Hello"
* **bytearray**: Mutable byte sequences. Example: arr = bytearray(b"Hello")

**🔹 7. None Type**

* **NoneType**: Represents the absence of a value. Example: result = None

**🔹 8. Custom Data Types**

* Defined using **classes and objects**. Example:
* class Car:
* def \_\_init\_\_(self, brand, year):
* self.brand = brand
* self.year = year

**📌 Strings in Python**

**1. String Basics**

* Strings are enclosed in 'single', "double", or '''triple''' quotes.
* Strings are **immutable** (cannot be modified directly).
* Access characters using indexing:
* text = "Python"
* print(text[0]) # P

**2. String Operations**

* **Concatenation**: "Hello" + " World" → "Hello World"
* **Slicing**: text[2:5] extracts characters from index 2 to 4.
* **Formatting**:
* name = "Alice"
* print(f"Hello, {name}!") # f-string

**3. Useful Methods**

* len(text), upper(), lower(), strip(), replace(), split(), join(), startswith(), etc.

**4. Escape Sequences**

* \n (newline), \t (tab), \\ (backslash).

**📌 Numeric Data Types**

* **Integers (int)**: Whole numbers, positive or negative.
* **Floats (float)**: Numbers with decimals.
* Both support arithmetic (+ - \* / % \*\* //).

**Example:**

a = 10

b = 3.5

print(a + b) # 13.5

⚠️ Floating-point precision issues may occur:

print(0.1 + 0.2) # 0.30000000000000004

👉 Use the **math** or **decimal** module for precise calculations.

**📌 Regular Expressions (Regex)**

Regular expressions are used for **pattern matching and text processing**.

* Python’s re module provides regex functionality.

**Common Metacharacters:**

* . → Any character
* \* → Zero or more repetitions
* + → One or more repetitions
* ? → Zero or one occurrence
* [] → Character set
* | → OR
* ^ → Start of string
* $ → End of string

Example:

import re

text = "My email is test@example.com"

match = re.search(r"\b[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b", text)

print(match.group()) # test@example.com