# **Module 3.1 Python Data Types & Variables**

## **Lesson Introduction**

Today’s Topics:

1. Primitive & Non-Primitive Data Types
2. Variables & Identifiers
3. Literals & Constants
4. Type Casting & Conversion
5. Comments & Documentation
6. Command-line Arguments (sys.argv)

## **Real-World Hook**

Have you ever filled out an online form where you enter your name, age, and email?

* Your name is **text** → stored as string.
* Your age is **a number** → stored as integer.
* The “subscribe to newsletter” checkbox → stored as boolean (True/False).

Computers need to store different types of information, and **data types & variables** are the tools that make it possible.

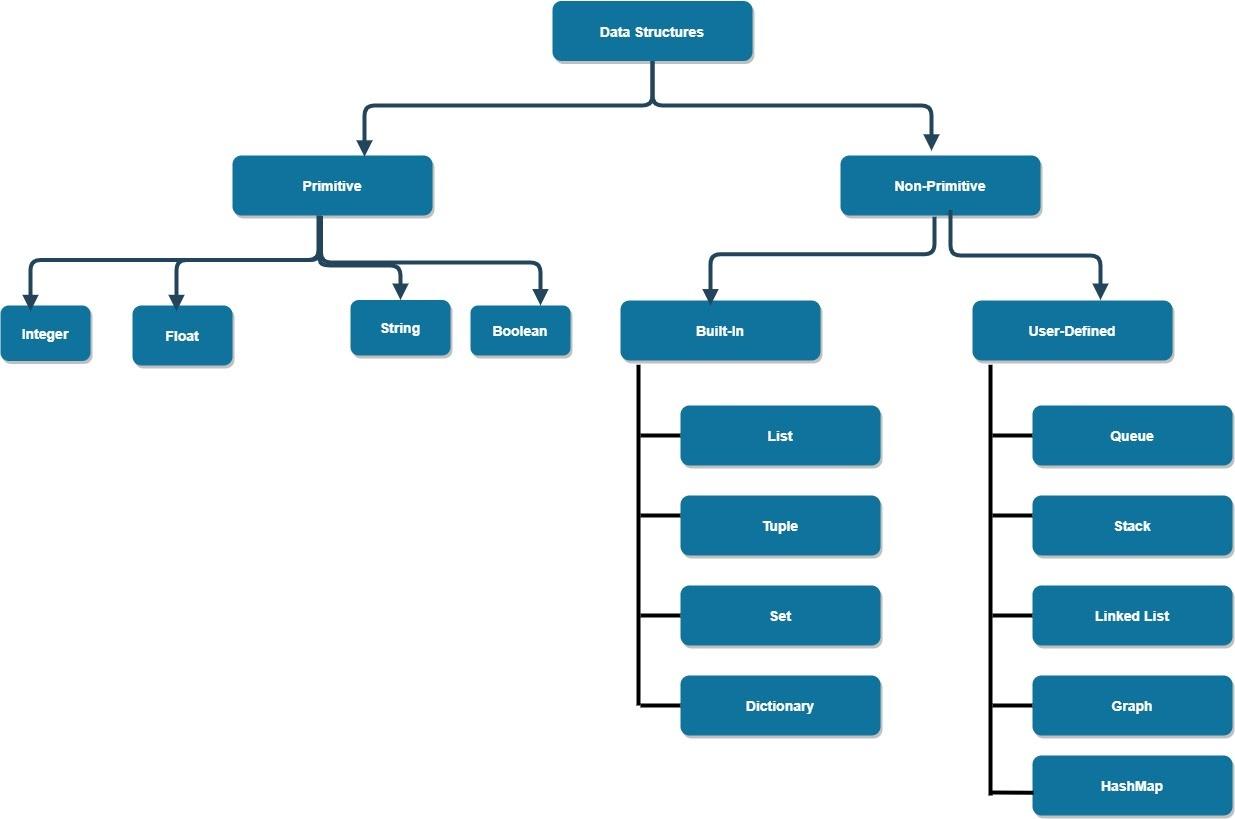
## **Introduce Topic as a Solution**

Without clear data types, computers wouldn’t know how to handle numbers vs text vs yes/no answers.  
 Variables act like **labeled containers** where we store these values for processing.

## **Theory**

### **1. What are Data Types?**

* **Definition**: Classification that tells Python what kind of value a variable holds.
* **Why**: Ensures correct storage, memory allocation, and operations.
* **When**: Every time we store a value.
* **How**: Python assigns data types automatically (dynamic typing).



### **2. Types of Data Types**

#### **Primitive Data Types**

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Example** |
| int | Whole numbers | age = 25 |
| float | Decimal numbers | pi = 3.14 |
| bool | True or False | is\_valid = True |
| str | Sequence of characters | name = "Alice" |

#### **Non-Primitive Data Types**

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Example** |
| list | Ordered, mutable | [1, 2, 3] |
| tuple | Ordered, immutable | (4, 5, 6) |
| set | Unordered, unique elements | {1, 2, 3} |
| dict | Key-value pairs | {"name": "Bob"} |

### **3. Variables & Identifiers**

* **Variable**: A named reference to stored data.

Name =”raj” ,student = ‘abhisek”

* **Rules for Identifiers**:  
  1. Start with letter/underscore, not a digit. 1name =”raj” wrong
  2. Can contain letters, numbers, \_.
  3. No spaces, no reserved keywords.

**Syntax:**

name = "Alice" # variable assignment

### **4. Literals & Constants**

* **Literal**: Fixed value like 100, "Hello".
* **Constant**: Variable whose value should not change (written in CAPS).

### **5. Type Casting & Conversion**

**Explicit Casting**:

a = int("10") # string → int

**Implicit Conversion**:

a = 5 + 2.5 # int converted to float

### **6. Comments & Documentation**

* # for single-line
* """ """ for multi-line/docstrings

### **7. Command-line Arguments**

Use sys.argv to pass external input when running scripts.

### **Advantages**

✅ Flexible data handling  
 ✅ Easy to read and write  
 ✅ Reduces manual memory management

### **Disadvantages**

❌ Can lead to unexpected type changes  
 ❌ Type errors if used incorrectly

### **Limitations**

* Python is slower for type-heavy tasks than C/C++.

**Overcoming Limitations**:

* Use **NumPy** for numeric arrays.
* Use **type hints** for better control.

### **Alternative**

Other languages like **Java** and **C++** use static typing.

## **Practical Example (with Dataset)**

**Dataset: Student Info (10 rows sample, 50+ rows possible)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Age** | **Grade** | **Enrolled** |
| 1 | Alice | 20 | 85.5 | True |
| 2 | Bob | 22 | 90.0 | False |
| 3 | Charlie | 21 | 78.0 | True |
| … | … | … | … | … |

**Example Code:**

students = [

{"ID": 1, "Name": "Alice", "Age": 20, "Grade": 85.5, "Enrolled": True},

{"ID": 2, "Name": "Bob", "Age": 22, "Grade": 90.0, "Enrolled": False}

]

for student in students:

print(f"{student['Name']} - Grade: {student['Grade']}")

## **Business Scenario**

A university student management system needs to store student details (name, age, grade, enrollment status).  
 Using proper data types ensures grades can be calculated correctly, names displayed properly, and enrollment status checked quickly.

## **Practice Session**

**Dataset**: Same 50-row student dataset.

1. Print names of all enrolled students.
2. Convert all grades to integers and display.
3. Calculate average age.
4. Display names in uppercase.
5. Count number of students enrolled (True).
6. Get highest grade.
7. Find students younger than 21.
8. Create a tuple of all student names.
9. Convert dataset into dictionary with ID as key.
10. Accept sys.argv to filter students by grade threshold.

## **Case Study**

**Project**: Student Data Analyzer

* Load student dataset
* Categorize into "Pass" or "Fail" based on grade ≥ 50
* Generate summary statistics (avg grade, total enrolled)
* Output results in a new list/dictionary

## **Assignments**

1. Create your own dataset of products (name, price, stock, availability).
2. Write code to:  
   * Filter available products
   * Calculate total stock value
   * Convert product names to uppercase

## **Quiz**

1. What is the difference between int and float?
2. Give an example of implicit type conversion in Python.
3. How do you declare a constant in Python?
4. What does sys.argv[0] contain?
5. Name two non-primitive Python data types.

## **Common Interview Questions (Theoretical)**

1. Difference between primitive and non-primitive data types.
2. How does Python handle dynamic typing?
3. What are the rules for naming identifiers?
4. Explain mutable vs immutable with examples.
5. Difference between list and tuple.

## **Common Interview Questions (Practical)**

1. Write a program to swap two variables without using a third variable.
2. Given a list of numbers, convert them all to strings.
3. Parse command-line arguments to sum two numbers.
4. Count how many students have grades above 80 from dataset.
5. Convert a dictionary into a list of tuples.