**Variables in Python**

**1. Introduction**

In Python, **variables** are used to store data that can be referenced and manipulated later in the program. They are fundamental building blocks in any programming language and allow developers to create programs that work with dynamic data.

**2. What is a Variable?**

A **variable** is a named location in memory used to store data. Once a variable is created, you can use it to perform operations, store results, and pass it to functions or expressions.

**Example:**

x = 10

name = "Mounika"

In the above example:

* x is a variable storing the integer 10
* name is a variable storing the string "Mounika"

**3. Rules for Naming Variables**

* Must start with a **letter (A-Z, a-z)** or an **underscore (\_)**.
* Cannot start with a **digit**.
* Can contain letters, digits, and underscores.
* **Case-sensitive** (Name and name are different).
* Cannot use **reserved keywords** (e.g., class, if, for, etc.).

**Valid Names:**

age, \_name, total\_marks, student1

**Invalid Names:**

1age, total-marks, class (reserved keyword)

**4. Variable Declaration and Assignment**

Python is a **dynamically-typed** language. This means you do not need to declare the data type of a variable explicitly.

**Syntax:**

variable\_name = value

**Examples:**

a = 5 # integer

b = 3.14 # float

name = "John" # string

is\_valid = True # boolean

**5. Multiple Assignments**

Python allows assigning values to multiple variables in a single line.

**Examples:**

x, y, z = 1, 2, 3

name = age = city = "Unknown"

**6. Data Types in Variables**

| **Type** | **Example** |  |
| --- | --- | --- |
| int | x = 10 -- | Integer value |
| float | x = 3.14 -- | Decimal (floating-point) |
| str | x = "Hello" -- | String (text) |
| bool | x = True -- | Boolean value (True/False) |
| list | x = [1,2,3] -- | Ordered, mutable sequence |
| tuple | x = (1,2) -- | Ordered, immutable sequence |
| dict | x = {"a": 1} | Key-value pairs |
| set | x = {1,2,3} | Unordered collection of unique items |

**7. Type Checking and Type Casting**

**Checking type:**

x = 5

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

**Type casting:**

x = "123"

y = int(x) # converts string to integer

**8. Constants in Python**

Python does not have built-in constant types, but by convention, variables written in **all uppercase** are treated as constants.

PI = 3.14159

MAX\_USERS = 100

**9. Global and Local Variables**

* **Global Variable**: Defined outside a function and accessible throughout the program.
* **Local Variable**: Defined inside a function and accessible only within that function.

**Example:**

x = 100 # global

def my\_function():

y = 50 # local

print(y)

my\_function()

print(x)

**Global keyword:**

If you want to modify a global variable inside a function:

count = 0

def increment():

global count

count += 1

**10. Deleting Variables**

You can delete a variable using the del keyword.

x = 10

del x

Trying to use x after deletion will raise a NameError.

**11. Dynamic Typing in Python**

Python allows variables to change their type during program execution.

x = 10

print(type(x)) # int

**12. Memory Management**

Python uses **reference counting** and a **garbage collector** to manage memory. When a variable is no longer in use, its memory is automatically freed.

**13. Best Practices**

* Use **descriptive variable names** (e.g., student\_name instead of s).
* Follow **snake\_case** convention for variable names.
* Avoid using **single-character variables** unless in loops (i, j, k).
* Use **constants** for fixed values.

**15. Example Program**

name = "Alice"

age = 25

height = 5.6

is\_student = True

print("Name:", name)

print("Age:", age)

print("Height:", height)

print("Student:", is\_student)

**Output:**Name: Alice

Age: 25

Height: 5.6

Student: True