Variables in Python

In Python, a **variable** is a symbolic name that refers to a value stored in memory. Variables are used to store and manipulate data in a program. Unlike some other programming languages, Python does not require explicit declaration of variables, as it dynamically assigns data types based on the value assigned.

Declaring Variables

A variable in Python is created when a value is assigned to it using the = operator:

```
x = 10       # Integer
y = 3.14     # Float
name = "John" # String
```

Python is dynamically typed, meaning that the type of a variable is determined at runtime and does not need to be explicitly stated.

Variable Naming Rules

While naming variables, the following rules must be followed:

- The name must start with a letter (A-Z or a-z) or an underscore _.
- It cannot start with a number.
- The name can contain letters, numbers, and underscores.
- Reserved keywords (like if, else, for) cannot be used as variable names.

Examples of valid and invalid variable names:

```
_valid_name = "Python"
name1 = "Alice"
1name = "Error"  # Invalid (Cannot start with a number)
if = "Error"     # Invalid (Cannot use a keyword)
```

Assigning Multiple Variables

Python allows multiple variable assignments in a single line:

```
a, b, c = 1, 2, 3
x = y = z = 100  # All variables hold the same value
```

Data Types in Variables

Python variables can store different data types:

```
num = 42        # Integer
pi = 3.1415     # Float
is_python = True # Boolean
```

```
text = "Hello"   # String
items = [1, 2, 3] # List
```

Type Conversion

Python provides built-in functions to convert variable types:

```
x = 5
y = str(x)  # Converts integer to string
z = float(x) # Converts integer to float
```

Global and Local Variables

A variable declared inside a function is **local**, while one declared outside is **global**:

```
global_var = "I am global"

def example():
    local_var = "I am local"
    print(local_var)

example()
print(global_var)
```