

**Unit 1 : Introduction to Python :-** Installing Python, basic syntax, interactive shell, editing saving and running a script; The concept of data types, variables, assignments; immutable variables; numerical types, operators(Arithmetic Operator, Relational Operator, Logical or Boolean Operator, Assignment Operator, Ternary Operator, Bitwise Operator, Increment or Decrement Operator) and expressions; comments in the program, understanding error messages.

## Installing Python

### Step 1: Download the Python Installer

- सबसे पहले किसी browser को open करो और official website पर जाओ [python.org/downloads](https://python.org/downloads).
- वहाँ पर latest **Python 3 release** (जैसे Python 3.13.0) ढूँढो.
- अब **Windows Installer (64-bit)** पर click करो (ज़्यादातर PC 64-bit होते हैं).
- इससे एक .exe file download हो जाएगी.

### Step 2: Run the Installer

- Download पूरा होने के बाद उस .exe file पर **double-click** करो.
- ✓ Important: एक option होगा "**Add Python 3.x to PATH**", उसे जरूर check करो (ताकि Python command prompt में आसानी से run हो सके).
- अब दो option मिलेंगे:
  - **Install Now** → Normal users के लिए best है.
  - **Customize Installation** → अगर तुम्हें manually settings change करनी हैं (जैसे folder location).

### Step 3: Complete the Installation

- अब screen पर आने वाले instructions follow करो → License accept करो → Installation process start हो जाएगा.
- Installation में कुछ minutes लग सकते हैं.
- अंत में एक option दिखेगा "**Disable path length limit**" → इसे enable करना better है (long folder names की problem avoid करने के लिए).

### Step 4: Verify Installation

- अब **Command Prompt (cmd)** या **PowerShell** open करो.
- Type करो:
- `python --version`
- अगर installation सही हुआ है तो output मिलेगा:
- `Python 3.13.0`

इसका मतलब Python successfully install हो गया है और use करने के लिए ready है.

## Basic Python Syntax:

Learn all the basic Python syntaxes you need to start coding. This guide covers comments, variables, functions, loops, and more — explained simply for beginners.



हर **programming language** का अपना **syntax** होता है — यानी वो नियम (rules) और structure जिनका पालन करके हम code लिखते हैं।

- कुछ भाषाएँ (जैसे **C**, **C++**, **Java**, **JavaScript**) का syntax आपस में मिलता-जुलता है।
- जबकि Python जैसी भाषाएँ readability और simplicity पर ज़्यादा ध्यान देती हैं, इसलिए इनका syntax काफी अलग और user-friendly है।

**Syntax** किसी भी programming language के rules का set होता है, जो बताता है कि उस language में code कैसे लिखा जाएगा। अगर हम इन rules को follow नहीं करेंगे तो code error देगा।

Programming language के syntax के कुछ main elements ये हैं:

- **Keywords** → ये reserved words होते हैं जिनका खास meaning होता है। Example: `if`, `else`, `for` आदि। इन्हें variable नाम की तरह use नहीं कर सकते।
- **Variables** → Data values को store करने के लिए variables use होते हैं। इनके नाम, type और scope होते हैं।
- **Control Flow Statements** → ये program के execution का sequence तय करते हैं। जैसे: `if-else`, `for loop`, `while loop`।
- **Indentation** → Python में indentation बहुत important है। ये बताता है कि कौन सा code किस block से belong करता है। (जैसे loop या function के अंदर का code)।
- **Comments** → Code को explain करने के लिए use होते हैं ताकि पढ़ने वाले को आसानी से समझ आए। Python में `#` symbol से single-line comment लिखा जाता है।

Simple words में, **syntax** वो **grammar** है जो **programming language** को **readable** और **executable** बनाती है।

## Basic Python Syntax

आइए **basic Python syntax** को देखें, जो ऊपर बताए गए items से मिलकर बना है और इसमें ये चीजें भी शामिल हैं:

- **Functions**
- **Data types**
- **Classes और Objects**
- **Lists, Tuples, और Dictionaries**

नीचे एक simple Python script दिया गया है, जो basic syntax के ज्यादातर elements को show करता है

```
# Variable Declaration and Assignment
```

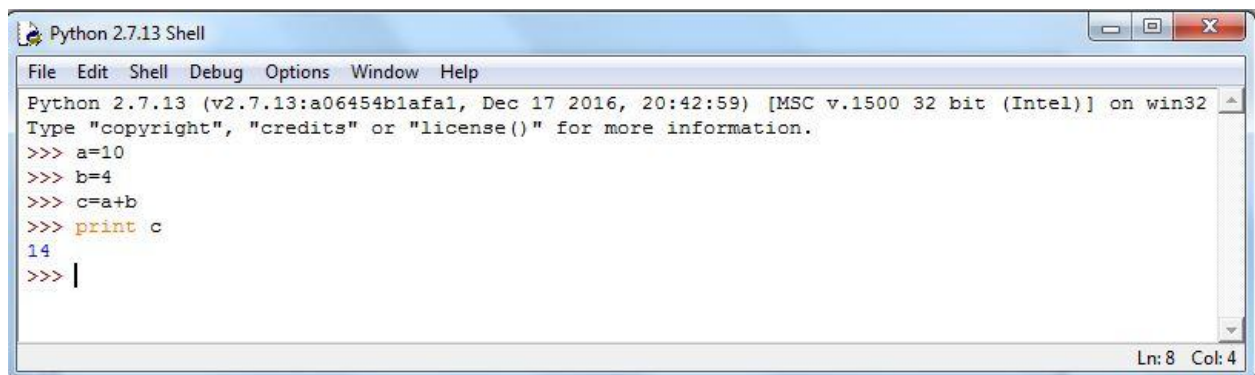
```
x = 5          # Integer variable assignment ('=' symbol से value assign की गई,  
यहाँ x = 5 है)  
y = "Hello!"   # String variable assignment (Strings हमेशा quotes "" में लिखी जाती हैं)
```

यहाँ पर:

- x एक **integer type variable** है जिसकी value 5 है।
- y एक **string type variable** है जिसकी value "Hello!" है।

## Interactive shell

**Python Interactive Shell** (जिसे Python Interpreter या **IDLE** भी कहते हैं) एक **command-line interface** है जहाँ आप **Python code line by line execute** कर सकते हो और तुरंत उसका output देख सकते हो।



## Features (फीचर्स)

### 1. Immediate Feedback (तुरंत Result)

- आप कोई भी Python statement लिखते हो → Enter दबाते हो → तुरंत उसका result दिखता है।
- Example:
  - >>> 2 + 3
  - 5

### 2. Line by Line Execution (Step by Step Run)

- Shell में आप code को एक-एक line लिखकर test कर सकते हो।
- इससे beginners के लिए Python सीखना easy हो जाता है।

### 3. Testing and Learning (Practice के लिए Best)

- Beginners छोटे-छोटे programs test कर सकते हैं।
- कोई error हो तो तुरंत पता चल जाता है।

### Example (Python Shell में)

```
>>> print("Hello Python")
Hello Python
```

```
>>> x = 10
>>> x * 2
20
```

✓ मतलब यह कि **Python Interactive Shell** coding सीखने और छोटे-छोटे experiments करने के लिए बहुत useful है।

### Editing, Saving, and Running a Script

- A script is a Python program saved with a .py extension.
- Steps:
  1. Open any editor (Notepad, VS Code, PyCharm).
  2. Write Python code.
  3. Save as filename.py.
  4. Run with `python filename.py`

## Python Data Types

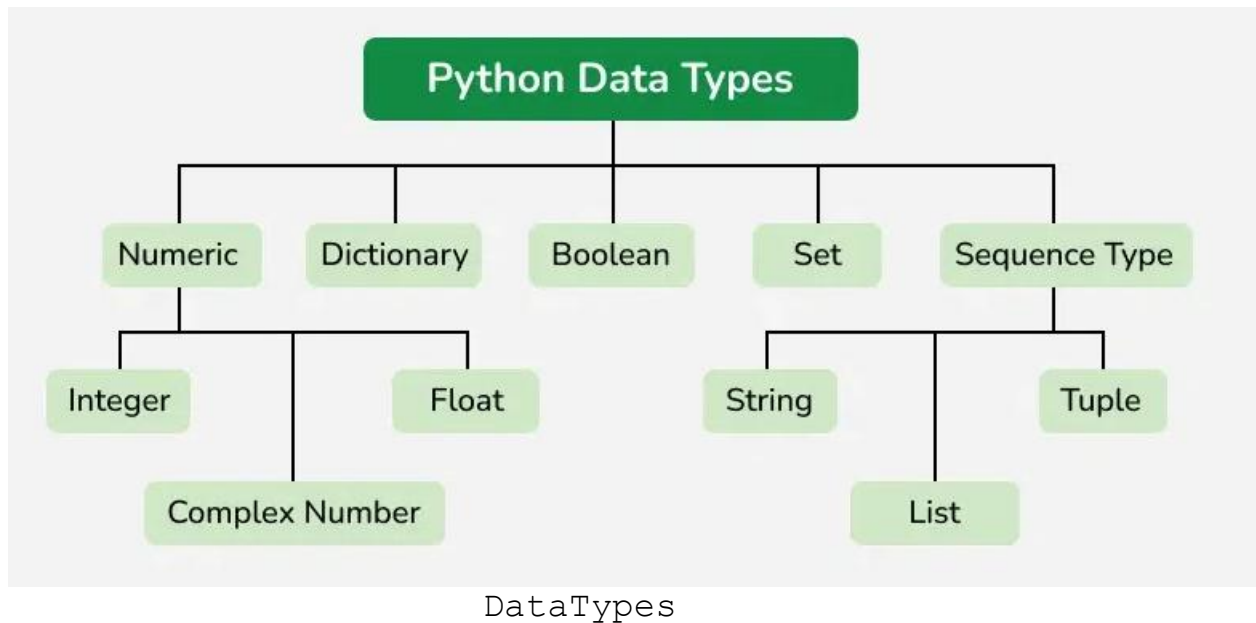
**Python Data types** का मतलब है data items का **classification** (वर्गीकरण)।

ये बताता है कि किसी value का प्रकार (type) क्या है और उस पर कौन-से operations perform किए जा सकते हैं।

क्योंकि Python programming में सब कुछ **object** होता है, इसलिए Python data types वास्तव में **classes** होते हैं और variables उन classes के **instances (objects)** होते हैं।

नीचे दिए गए standard या built-in data types Python में available हैं:

- **Numeric** → int, float, complex
- **Sequence Type** → string, list, tuple
- **Mapping Type** → dict
- **Boolean** → bool
- **Set Type** → set, frozenset
- **Binary Types** → bytes, bytearray, memoryview



यह code variable 'x' को अलग-अलग Python **data types** की values assign करता है – जैसे **int, float, list, tuple और string**।

हर बार जब नई value assign होती है, तो पुरानी value replace हो जाती है।

इसका मतलब यह है कि variable 'x' हर बार नया data type और value ले लेता है, जो सबसे **recent assignment** होता है।

```
# int, float, string, list and set
x = 50
x = 60.5
x = "Hello World"
x = ["geeks", "for", "geeks"]
x = ("geeks", "for", "geeks")
```

## 1. Numeric Data Types in Python

Numeric data type ऐसे data को represent करता है जिसमें numeric value होती है। Numeric value **integer**, **floating number**, या **complex number** हो सकता है। Python में ये values अलग-अलग classes के रूप में defined हैं: **int**, **float** और **complex class**।

- **Integers (int class)** → इसमें positive या negative whole numbers (बिना fraction या decimal के) होते हैं। Python में integer की length पर कोई limit नहीं है।
- **Float (float class)** → इसमें real numbers होते हैं जो decimal point के साथ represent किए जाते हैं। Float values में scientific notation (जैसे  $2.5e3$ ) का भी use किया जा सकता है।
- **Complex Numbers (complex class)** → Complex number को (real part) + (imaginary part) $j$  की form में लिखा जाता है।

Example:

- `a = 5`
- `print(type(a))`     # int
- 
- `b = 5.0`
- `print(type(b))`     # float
- 
- `c = 2 + 4j`
- `print(type(c))`     # complex

**Output:**

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

## 2. Sequence Data Types in Python

Sequence data type ordered collection होता है जिसमें multiple values को efficiently store किया जा सकता है। Python में sequence types हैं: **String**, **List**, **Tuple**

### (a) String Data Type

- Python Strings, Unicode characters का array होती है।
- Python में कोई separate "character" data type नहीं है, एक character को भी length-1 की string माना जाता है।

- String को 'single quotes', "double quotes", या '''triple quotes''' से बनाया जा सकता है।
- Indexing से हम string के characters access कर सकते हैं।

```
s = 'Welcome to the Geeks World'
print(s)
print(type(s))    # str

print(s[1])        # e
print(s[2])        # l
print(s[-1])       # d
```

### Output:

```
Welcome to the Geeks World
<class 'str'>
e
l
d
```

## (b) List Data Type

- List array की तरह होती है, लेकिन इसमें mixed data types store कर सकते हैं।
- Square brackets [] का use करके list create होती है।

```
a = [1, 2, 3]
print(a)

b = ["Geeks", "For", "Geeks", 4, 5]
print(b)
```

### Output:

```
[1, 2, 3]
['Geeks', 'For', 'Geeks', 4, 5]
```

### Access Items (Indexing + Negative Indexing)

```
a = ["Geeks", "For", "Geeks"]
print(a[0])    # Geeks
print(a[2])    # Geeks
print(a[-1])   # Geeks
print(a[-3])   # Geeks
```

## (c) Tuple Data Type



- Tuple भी ordered collection है, लेकिन **immutable** है (यानि बदल नहीं सकते)।
- Parentheses () या सिर्फ comma , से tuple बनाया जा सकता है।

```
tup1 = ()
tup2 = ('Geeks', 'For')
print(tup2)    # ('Geeks', 'For')

tup1 = tuple([1, 2, 3, 4, 5])
print(tup1[0])    # 1
print(tup1[-1])   # 5
print(tup1[-3])   # 3
```

### 3. Boolean Data Type in Python

- Boolean data type में केवल दो values होती हैं: **True** और **False**
- ये `bool` class से represent होती हैं।
- Python case-sensitive है, इसलिए `true` (छोटे अक्षरों में) keyword नहीं है।

```
print(type(True))    # bool
print(type(False))   # bool
print(type(true))     # error
```

### 4. Set Data Type in Python

- Set unordered collection है जिसमें duplicate elements नहीं होते।
- Curly braces {} या `set()` function से create होता है।

```
s1 = set("GeeksForGeeks")
print(s1)

s2 = set(["Geeks", "For", "Geeks"])
print(s2)

set1 = set(["Geeks", "For"])
for i in set1:
    print(i, end=" ")

print("Geeks" in set1)    # True
```

## 5. Dictionary Data Type in Python

- Dictionary data को **key-value pair** में store करता है।
- Keys unique और immutable होने चाहिए।
- Dictionary को {} या dict() से create किया जाता है।

```
d = {1: 'Geeks', 2: 'For', 3: 'Geeks'}
print(d)

d1 = dict({1: 'Geeks', 2: 'For', 3: 'Geeks'})
print(d1)

print(d['name'])    # For
print(d.get(3))     # Geeks
```

## Exercise Questions

### Q1. Basic List Operations

```
fruits = ["apple", "banana", "orange"]
print(fruits)

fruits.append("grape")
print(fruits)

fruits.remove("orange")
print(fruits)
```

#### Output:

```
['apple', 'banana', 'orange']
['apple', 'banana', 'orange', 'grape']
['apple', 'banana', 'grape']
```

### Q2. Basic Tuple Operations

```
coordinates = (3, 5)
print(coordinates)
print("X-coordinate:", coordinates[0])
print("Y-coordinate:", coordinates[1])
```

#### Output:

```
(3, 5)
X-coordinate: 3
Y-coordinate: 5
```

## Introduction Variables

Programming में **Variables** का use data को **store** और **manipulate** करने के लिए किया जाता है। Python में variables बहुत ज़रूरी हैं क्योंकि ये code की structure बनाने में help करते हैं। Developers variables का use करके data के साथ interact करते हैं और programs को robust और adaptable बनाते हैं।

यह tutorial Python variables, उनके rules, types और examples को explain करेगा।

## What Are Variables in Python?

Python में variables **containers** की तरह काम करते हैं, जो data values (जैसे numbers, text या complex structures) को store करते हैं।

Python में आपको variable का type declare करने की ज़रूरत नहीं होती। बस value assign कर दीजिए और Python अपने आप type handle कर लेता है।

Variable Python में memory के किसी location को point करता है जहाँ data store होता है। हम program के दौरान उसका value कभी भी change कर सकते हैं।

Variables से code readable और efficient बनता है, और ये dynamic applications बनाने में बहुत काम आते हैं।

## Rules for Python Variables

Python variables के नाम रखने के लिए कुछ rules होते हैं ताकि code readable और error-free रहे:

1. Variable का नाम letter या underscore ( `_` ) से शुरू होना चाहिए। (e.g., `_name`, `age`)
2. Variable का नाम number से start नहीं हो सकता।
3. सिर्फ letters, numbers और underscore का use करें (कोई space या special character नहीं)।
4. Python keywords (जैसे `if`, `class`, `while`) variable name नहीं हो सकते।
5. Variable names case-sensitive होते हैं (`Age` और `age` अलग माने जाते हैं)।
6. हमेशा meaningful names रखें (जैसे `total_price` सही है, `tp` avoid करें)।

अगर आप ये simple rules follow करेंगे तो code clean और bug-free रहेगा।

## Examples Showing Python Variable Rules

### Invalid Example ✖

```
2name = "Varun"  
student age = 14  
if = True
```

#### ✖ Error क्यों?

- 2name number से start हो रहा है।
- student age में space है।
- if एक Python keyword है।

### Valid Example ✔

```
name2 = "Varun"  
student_age = 14  
is_enrolled = True  
  
print(name2, student_age, is_enrolled)
```

✔ यह बिना error के run करेगा।

## Variable Assignment

Variable assignment का मतलब है variable को किसी value से store करना, इसके लिए = operator का use किया जाता है।

## Declaration and Initialization of Variables

Python dynamically typed language है, इसलिए आपको type declare करने की ज़रूरत नहीं है। Variable उस समय create होता है जब उसे कोई value assign की जाती है।

```
name = "Varun"  
age = 25  
print(name)  
print(age)
```

### Output:

```
Varun  
25
```

## Redeclaring Variables in Python

Python में आप variables को दुबारा declare कर सकते हैं, मतलब आप उनकी value या type change कर सकते हैं।

```
item_count = 10  
print(item_count)  
  
item_count = 5  
print(item_count)
```

### Output:

```
10  
5
```

## Assigning Values to Multiple Variables

आप एक ही value multiple variables में assign कर सकते हैं।

```
a = b = c = 100  
print(a)  
print(b)  
print(c)
```

### Output:

```
100  
100  
100
```

## Assigning Different Values to Multiple Variables

आप एक ही line में multiple variables को different values भी assign कर सकते हैं।

```
name, age, city = "Amit", 28, "Delhi"

print(name)
print(age)
print(city)
```

### Output:

```
Amit
28
Delhi
```

### Immutable Variables

- Variables that **cannot be changed** once created.
- Example: int, float, str, tuple are immutable.

```
x = "hello"
```

```
x[0] = "H" # Error (string is immutable)
```

## Python Operators (ऑपरेटर्स in Python)

Python में **Operators** special symbols या keywords होते हैं जो variables और values पर operations perform करते हैं।

ये expressions की base होते हैं, जिनकी मदद से हम data को modify कर सकते हैं और computations (गणनाएँ) कर सकते हैं।

Python में कई तरह के Operators होते हैं, हर एक का अलग purpose है।

### Types of Python Operators

Python language में ये operators होते हैं:

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Assignment Operators
4. Logical Operators
5. Bitwise Operators
6. Membership Operators
7. Identity Operators

## 1. Python Arithmetic Operators

- Arithmetic operators का use mathematical operations जैसे **addition, subtraction, multiplication, division** करने में होता है।
- ये integers, variables और expressions पर काम करते हैं।
- Normal operators के अलावा modulus (%), exponent (\*\*), और floor division (//) भी होते हैं।

### Operators Table

Operator	Name	Example
+	Addition	10 + 20 = 30
-	Subtraction	20 - 10 = 10
*	Multiplication	10 * 20 = 200
/	Division	20 / 10 = 2
%	Modulus	22 % 10 = 2
**	Exponent	4**2 = 16
//	Floor Division	9//2 = 4

### Example Code

```
a = 21
b = 10
print("a + b :", a + b)
print("a - b :", a - b)
print("a * b :", a * b)
print("a / b :", a / b)
print("a % b :", a % b)
print("a ** b :", a ** b)
print("a // b :", a // b)
```

### Output

```
a + b : 31
a - b : 11
a * b : 210
a / b : 2.1
a % b : 1
a ** b : 16679880978201
```

a // b : 2

---

## 2. Python Comparison (Relational) Operators

- ये operators दो values को compare करने के लिए use होते हैं।
- Result हमेशा **Boolean (True/False)** होता है।

Operator	Name	Example
==	Equal	4 == 5 (False)
!=	Not Equal	4 != 5 (True)
>	Greater Than	4 > 5 (False)
<	Less Than	4 < 5 (True)
>=	Greater Than or Equal to	4 >= 5 (False)
<=	Less Than or Equal to	4 <= 5 (True)

---

## 3. Python Assignment Operators

- Assignment operators का use **variables** को **value assign** करने के लिए होता है।
- सबसे basic assignment operator है =।

Operator	Name	Example
=	Assignment	a = 10
+=	Addition Assignment	a += 5 → a = a + 5
-=	Subtraction Assignment	a -= 5 → a = a - 5
*=	Multiplication Assignment	a *= 5 → a = a * 5
/=	Division Assignment	a /= 5 → a = a / 5
%=	Modulus Assignment	a %= 5 → a = a % 5



Operator	Name	Example
<b>**=</b>	Exponent Assignment	$a ** = 2 \rightarrow a = a ** 2$
<b>//=</b>	Floor Division Assignment	$a //= 3 \rightarrow a = a // 3$

---

## 4. Python Bitwise Operators

- Bitwise operators numbers के **binary representation** पर काम करते हैं।

Operator	Name	Work
<b>&amp;</b>	Binary AND	दोनों bits 1 हों तो result 1 होगा
<b> </b>	Binary OR	किसी एक bit 1 हो तो result 1 होगा
<b>^</b>	Binary XOR	सिर्फ एक bit 1 हो तो result 1 होगा
<b>~</b>	Ones Complement	सभी bits को invert करता है
<b>&lt;&lt;</b>	Left Shift	Bits को left shift करता है
<b>&gt;&gt;</b>	Right Shift	Bits को right shift करता है

---

## 5. Python Logical Operators

- Logical operators Boolean expressions को evaluate करने के लिए use होते हैं।

Operator	Description	Example
<b>and</b>	दोनों operands True हों तो True	(a and b)
<b>or</b>	कोई भी operand True हो तो True	(a or b)

Operator	Description	Example
not	Operand का उल्टा result देता है	not(a and b)

## 6. Python Membership Operators

- Membership operators check करते हैं कि कोई value sequence (list, tuple, set, string) में present है या नहीं।

Operator	Description	Example
in	Value present हो तो True	x in y
not in	Value present ना हो तो True	x not in y

## 7. Python Identity Operators

- Identity operators check करते हैं कि दो **objects same memory address** पर point कर रहे हैं या नहीं।

Operator	Description	Example
is	दोनों objects same हों तो True	x is y
is not	Objects different हों तो True	x is not y

## Increment Operator (+=)

Python में `i++` जैसा operator नहीं होता।

Increment करने के लिए हम use करते हैं:

```
x = 5
x += 1    # same as x = x + 1
print("Incremented value:", x)
```

**Output → 6**

---

## **Decrement Operator (--)**

Python में `i--` operator भी नहीं होता।

Decrement करने के लिए:

```
x = 10
x -= 1    # same as x = x - 1
print("Decrement value:", x)
```

**Output → 9**

## **Comments in the Program**

- Comments are notes written for humans, not executed by Python.
- Single-line comment → `#`
- Multi-line comment → `''' ... '''`
- `# This is a single-line comment`
- `'''`
- `This is`
- `a multi-line comment`
- `'''`

## **Understanding Error Messages**

- Errors tell us what went wrong in the code.
- Example:
- `print(Hello)`

✗ Error: `NameError: name 'Hello' is not defined`

✓ Fix: `print("Hello")`