Unit: 1

Python Introduction

The Python Language:

Python ek high-level, easy to learn programming language hai jo 1991 me **Guido van Rossem** ne devlop ki thi. Iska main focus simplicity aur readability par hota hai, isliye beginners ke liye python ek best choice mana jata hai.

Python ek interpreted aur object oriented programming language hai, jisme code likhna almost English jaisa lagta hai isme curly braces {} ki jagah indentation ka use hota hai, jo code ko clean aur understandable banate hain.

Main features of Python:

- Easy to learn & simple syntax, kam code me kaam ho jata hai.
- **Open source**- free hai, koi license fees nhi.
- **Platform independent**- windows, linux, macOS sub par chalti hai.
- **Large library support**- ready made modules (NumPy, Pandas, Django, etc).
- **Interpreted language**- line-by-line execute hoti hai.
- **Object Oriented & functional**- dono programming styles support karti hain.

Python ka Use:

1. Web development- Django, Flask
2. Artificial Intelligence & Machine Learning
3. Data Science & Data Analysis
4. Desktop Applications
5. Game Development

The Standard library and Extension Modules:

1. Python Standard Library kya hoti hai?

Definition:
Python ke saath jo built-in modules aate hain, unhe **Standard Library** kehte hain.

Matlab:

- Aapko alag se install nahi karna padta
- Direct use kar sakte hain using import

Example:

```
import math
print(math.sqrt(16))
```

Standard Library ke Features

✓ Pre-written code (time save karta hai)
✓ Cross-platform (Windows, Linux, Mac sab par chalega)
✓ Large collection of modules
✓ Efficient & optimized

Important Standard Library Modules

(A) Math Module: Mathematical operations ke liye

```
import math
print(math.pi)      # 3.14
print(math.sqrt(25)) # 5
print(math.factorial(5)) # 120
```

(B) Random Module: Random values generate karne ke liye

```
import random
print(random.randint(1, 10))
print(random.choice([1,2,3,4]))
```

(C) Datetime Module: Date aur time handle karne ke liye

```
import datetime
print(datetime.datetime.now())
```

(D) OS Module: Operating system se interact karne ke liye

```
import os
print(os.getcwd())  # current directory
os.mkdir("test")    # folder create
```

(E) Sys Module: Python interpreter ke control ke liye

```
import sys
print(sys.version)
print(sys.path)
```

(F) File Handling (Built-in)

```
file = open("test.txt", "w")
file.write("Hello World")
file.close()
```

(G) JSON Module: Data ko JSON format me convert karne ke liye

```
import json
data = {"name": "Deepak"}
json_data = json.dumps(data)
print(json_data)
```

(H) Collections Module: Advanced data structures provide karta hai

```
from collections import Counter
print(Counter([1,2,2,3]))
```

(I) Re Module (Regular Expressions)

```
import re
pattern = r"\d+"
text = "abc123"
print(re.findall(pattern, text))
```

---

2. Python Extension Modules kya hote hain?

Definition:
Jo modules Python ke standard library me nahi hote aur external sources se install karne padte hain, unhe **Extension Modules** kehte hain.

Install using:

```
pip install module_name
```

Popular Extension Modules

(A) NumPy: Scientific computing ke liye

```
import numpy as np
arr = np.array([1,2,3])
print(arr)
```

(B) Pandas: Data analysis ke liye

```
import pandas as pd
data = pd.DataFrame({"A":[1,2]})
print(data)
```

(C) Matplotlib: Graph aur charts banane ke liye

```
import matplotlib.pyplot as plt
plt.plot([1,2,3],[4,5,6])
plt.show()
```

(D) Requests: HTTP requests bhejne ke liye

```
import requests
response = requests.get("https://api.github.com")
print(response.status_code)
```

(E) Flask / Django: Web development frameworks

- Flask → lightweight
- Django → full framework

Advantages

✅ **Standard Library**

- Ready to use
- Reliable
- No installation needed

✅ **Extension Modules**

- Powerful features
- Advanced functionality
- Real-world applications (AI, ML, Web)

Summary

Standard Library = Python ke built-in tools
Extension Modules = Extra powerful tools (install karne padte hain)

Python Implementation:

1. Python Implementation kya hota hai?

Definition:
Python implementation ka matlab hai **Python language ko actual me run karne ka system** — yani kaise Python code execute hota hai internally.

Python code kaise machine tak pahuchta hai aur run hota hai.

Python Execution Process (Step-by-Step)

Python directly machine code me convert nahi hota, balki intermediate step hota hai

Step 1: Source Code (.py file)

Aap Python code likhte hain

print("Hello World")

Step 2: Compilation → Bytecode

Python code convert hota hai **Bytecode** me (.pyc file)

Ye machine independent hota hai

Step 3: Python Virtual Machine (PVM)

Bytecode ko execute karta hai

Ye hi actual execution karta hai

Flow:

Source Code (.py)
↓
Bytecode (.pyc)
↓
Python Virtual Machine (PVM)
↓
Output

Bytecode kya hota hai?

- Intermediate code hota hai
- Machine dependent nahi hota
- Faster execution ke liye use hota hai

Stored in:

__pycache__ folder

Python Virtual Machine (PVM)

Definition:
Ek virtual engine jo bytecode ko execute karta hai

✓ Runtime environment provide karta hai
✓ Memory manage karta hai
✓ Error handling karta hai

Python Interpreted Language kaise hai?

Python **compiled + interpreted dono** hai:

- Pehle compile hota hai (bytecode me)
- Phir interpret hota hai (PVM ke through)

Isliye Python ko **interpreted language** kaha jata hai

Different Python Implementations

Python sirf ek nahi hai — multiple implementations hain

(A) CPython (Most Important)

✓ Official implementation
✓ C language me likha gaya hai
✓ Default Python interpreter

Sabse zyada use hota hai

(B) Jython

✓ Java me likha gaya
✓ Java Virtual Machine (JVM) par run hota hai

Java libraries use kar sakte hain

(C) IronPython

✓ .NET framework ke liye
✓ C# ecosystem me use hota hai

(D) PyPy

✓ Fast implementation
✓ Just-In-Time (JIT) compiler use karta hai

CPython se faster ho sakta hai

(E) MicroPython

✓ Embedded systems ke liye
✓ IoT devices me use hota hai

Memory Management in Python

Python automatic memory management use karta hai

✓ Heap Memory

- Objects yahan store hote hain

✓ Stack Memory

- Function calls store hote hain

Garbage Collection

Python unused memory ko automatically free karta hai

Reference counting + Garbage collector

Dynamic Typing

Python me variable ka type define nahi karna padta

```
x = 10
x = "Hello"
```

Type automatically change ho jata hai

Python me Portability

✓ Same code different OS par run ho sakta hai
✓ Reason → Bytecode + PVM

Advantages of Python Implementation

✓ Easy execution
✓ Platform independent
✓ Automatic memory management
✓ Fast development

Disadvantages

❌ Slow compared to C/C++
❌ High memory usage
❌ Mobile development me limited use

Python Development and versions:

1. Python Development kya hai?

Definition:
Python development ka matlab hai Python language ka use karke **software, websites, tools, automation scripts, AI models, etc. banana**.

Simple words me:

Python ka use karke real-world applications banana hi Python development hai

---

Python ka Development History

Python ko develop kiya tha:

Guido van Rossum

🗓️ Year: 1989 (development start)
🗓️ First release: 1991

Goal:

- Simple language banana
- Readable code
- Easy learning

Python Development ke Major Goals

✓ Code readability (easy to understand)
✓ Simple syntax
✓ Cross-platform support

✓ Rapid development (fast coding)
✓ Large community support

Python Development ke Major Fields

(A) Web Development

- Frameworks: Django, Flask
- Websites & web apps

(B) Data Science & AI

- Libraries: NumPy, Pandas, TensorFlow
- Machine learning, AI models

(C) Automation / Scripting

- Tasks automate karna (file handling, scraping)

(D) Software Development

- Desktop apps
- Tools development

(E) Cybersecurity

- Penetration testing tools
- Ethical hacking scripts

(F) Cloud & DevOps

- Automation scripts
- Deployment tools

Python Versions Overview

Python ke development me do major versions aaye hain:

(A) Python 2.x

🗓️ Released: 2000

Features:

✓ Easy syntax
✓ Fast performance

Problems:

❌ Unicode support weak
❌ Future development stop ho gaya

🗓️ End of Life: 1 January 2020

Ab use nahi hota (deprecated)

(B) Python 3.x (Current)

🗓️ Released: 2008

Yeh modern Python hai (recommended)

✓ Key Improvements:

1. Better Unicode Support

```
print("नमस्ते")
```

---

2. Print Function

```
print("Hello")   # Python 3
```

Python 2 me:

```
print "Hello"
```

---

3. Integer Division

```
5 / 2 = 2.5   # Python 3
```

---

4. Improved Libraries

- Modern modules
- Better performance

5. Cleaner Syntax

- More readable code

Python Version Evolution (Important)

Version	Year	Key Features
Python 1.0	1994	Basic features
Python 2.0	2000	Garbage collection
Python 2.7	2010	Last version of Python 2
Python 3.0	2008	Major upgrade
Python 3.5+	2015	Async programming
Python 3.8+	2019	Walrus operator
Python 3.10+	2021	Pattern matching
Python 3.11+	2022	Faster execution

Python Version Features (Advanced)

(A) Async Programming

```
async def hello():
    print("Hello")
```

(B) Walrus Operator (:=)

```
if (n := 5) > 3:
    print(n)
```

(C) Pattern Matching

```
match x:
   case 1:
      print("One")
```

Python Development Tools

✓ IDEs

- VS Code
- PyCharm

✓ Package Manager

- pip

✓ Version Control

- Git

Python Development Process

1. Code likhna
2. Run karna
3. Debugging
4. Testing
5. Deployment

Why Python Popular hai?

✓ Easy to learn
✓ Huge community
✓ Versatile (har field me use)
✓ Fast development

Installation from source code and Binaries:

1. Installation kya hota hai?

Definition:
Kisi software ko system me setup karna taaki wo properly run kare, use **installation** kehte hain.

Python installation ke 2 main methods hote hain:

1. **Binary Installation**

2. **Source Code Installation**

---

1. Binary Installation (Easy Method)

Binary kya hota hai?

Binary file matlab **pre-compiled software**
Already machine-readable format me hota hai

Example:

- .exe (Windows)
- .msi
- .deb, .rpm (Linux)

Python Binary kaha milta hai?

Official website: Python

Steps (Windows):

1. Python website se installer download karein
2. .exe file run karein
3. "Add Python to PATH" select karein
4. Install Now par click karein
5. Installation complete

Verify Installation:

python --version

Advantages of Binary Installation

✓ Easy & fast
✓ No technical knowledge needed
✓ Ready-to-use
✓ Beginner friendly

Disadvantages

❌ Customization kam hota hai
❌ Optimization control limited

---

2. Source Code Installation (Advanced Method)

Source Code kya hota hai?

Human-readable code jise compile karna padta hai

Python Source kaha milta hai?

Same official site se .tar.gz ya .zip format me

Steps (Linux / macOS):

Step 1: Download source code

https://www.python.org/ftp/python/3.x.x/Python-3.x.x.tar.xz

Step 2: Extract file

tar -xf Python-3.x.x.tar.xz

Step 3: Configure

./configure

Step 4: Build (Compile)

make

Step 5: Install

sudo make install

Verify:

python3 --version

Advantages of Source Installation

✓ Full control milta hai
✓ Custom features enable kar sakte hain
✓ Optimized build bana sakte hain

Disadvantages

❌ Complex process
❌ Time-consuming
❌ Errors aa sakte hain compilation me

🔹 **Kab kaunsa use karein?**

Binary Installation use karein:

- Beginners hain
- Quick setup chahiye
- Windows use kar rahe hain

Source Installation use karein:

- Linux developer hain
- Custom Python build chahiye
- Performance optimize karna hai

The Python Interpreter:

Python Interpreter kya hota hai?

Definition:
Python Interpreter ek program hota hai jo Python code ko **read, translate aur execute** karta hai.

Interpreter ka kaam kya hota hai?

✓ Code ko read karta hai
✓ Bytecode me convert karta hai
✓ Execute karta hai
✓ Errors detect karta hai

Python Interpreter ka Working Process

Python interpreter internally 2 steps me kaam karta hai:

Step 1: Compilation to Bytecode

Python code (.py) → convert hota hai bytecode (.pyc)

Step 2: Execution via PVM

Bytecode → execute hota hai **Python Virtual Machine (PVM)** ke through

Flow Diagram:

Python Code (.py)
↓
Bytecode (.pyc)
↓
Python Virtual Machine
↓
Output

Interpreter vs Compiler

Feature	Interpreter	Compiler
Execution	Line by line	Whole program
Speed	Slow	Fast
Error detection	Line-wise	After compilation
Example	Python	C, C++

Python interpreter use karta hai, isliye debugging easy hoti hai

---

Python Interactive Mode (REPL)

REPL = Read, Evaluate, Print, Loop

Direct terminal me code run kar sakte hain

```
>>> 5 + 5
10
```

✓ Testing ke liye best
✓ Quick calculation

Script Mode

Code file me likh kar run karte hain

```
print("Hello World")
```

Run:

```
python file.py
```

Types of Python Interpreters

(A) CPython (Most Important)

✓ Default interpreter
✓ C language me likha gaya
✓ Most widely used

(B) PyPy

✓ Fast interpreter
✓ JIT (Just-In-Time) compiler use karta hai

(C) Jython

✓ Java-based interpreter
✓ JVM par run hota hai

(D) IronPython

✓ .NET framework ke liye
✓ C# ecosystem me use hota hai

(E) MicroPython

✓ Embedded systems ke liye
✓ IoT devices me use hota hai

Python Interpreter ke Features

✓ Platform independent
✓ Easy debugging
✓ Dynamic typing support
✓ Automatic memory management

Advantages

✓ Easy to use
✓ Fast development
✓ Error detection easy
✓ No separate compilation needed

Disadvantages

❌ Execution slow hota hai
❌ Memory usage zyada hota hai

Core Python language and Built – ins:

1. Built-ins ka matlab kya hota hai?

Definition:
Python me jo **functions, objects, aur features pehle se hi available hote hain (without import)** unhe **Built-ins** kehte hain.

Lexical Structure:

Lexical Structure kya hota hai?

Definition:
Programming language me jo **basic building blocks (tokens)** hote hain, unki structure ko **Lexical Structure** kehte hain.

Tokens kya hote hain?

Token = smallest unit of program

Python me tokens ke types:

1. Keywords
2. Identifiers
3. Literals
4. Operators
5. Delimiters

---

3. Types of Tokens

(A) Keywords

Reserved words hote hain (special meaning ke saath)

✓ Inhe variable name ke roop me use nahi kar sakte

Examples:

if, else, while, for, True, False, None

(B) Identifiers

Variables, functions, classes ke naam

Rules:

✓ Letters, digits, underscore allowed
✓ Number se start nahi hona chahiye
✓ Keywords use nahi kar sakte

Example:

```
name = "Rahul"
_age = 20
```

(C) Literals

Fixed values jo program me directly use hoti hain

Types:

✓ Numeric → 10, 3.14
✓ String → "Hello"
✓ Boolean → True, False
✓ None → None

(D) Operators

Operations perform karne ke liye

Types:

✓ Arithmetic → + - * /
✓ Relational → == != > <
✓ Logical → and or not
✓ Assignment → =

(E) Delimiters (Separators)

Symbols jo structure define karte hain

Examples:

```
( ) { } [ ] , : . ;
```

Indentation (Very Important)

Python me indentation syntax ka part hai

```
if True:
    print("Hello")
```

✓ Space/tab use hota hai
✓ Block define karta hai

Galat indentation → error

Comments

Code explain karne ke liye

Single-line:

This is comment

Multi-line:

"""
This is
multi-line comment
"""

Whitespace

Space, tab, newline

✓ Python me meaningful hota hai (especially indentation)

Case Sensitivity

Python **case-sensitive** language hai

name ≠ Name

Line Structure

✓ **Single line:**

a = 10

✓ **Multi-line:**

```
total = (1 + 2 +
        3 + 4)
```

Escape Characters

Special characters in strings

Escape	Meaning
\n	New line
\t	Tab
\\	Backslash

String Quotes

✓ Single ' '
✓ Double " "
✓ Triple ''' ''' or """ """

Data Types:

Data Type kya hota hai?

Definition:
Data type batata hai ki variable me kaunsa type ka data store ho raha hai aur us par kaunse operations perform ho sakte hain.

“Data ka nature + us par allowed operations” = Data Type

Python me Data Types ki khas baat

✓ Dynamically typed language
✓ Variable ka type automatically decide hota hai
✓ Same variable me different type store ho sakta hai

```
x = 10
x = "Hello"
```

Python ke Built-in Data Types (Main Categories)

Python me data types ko major categories me divide kiya jata hai

(A) Numeric Data Types

1. int (Integer)

Whole numbers

```
a = 10
b = -5
```

---

2. float (Floating Point)

Decimal numbers

x = 3.14

---

3. complex

Real + imaginary part

c = 2 + 3j

Important:

- Python me integer size unlimited hota hai
- Float approximate values store karta hai

(B) Sequence Data Types

1. String (str)

Characters ka sequence

name = "Deepak"

Features:

Immutable
Indexing possible
Slicing possible

name[0] # D
name[0:3] # Dee

---

2. List

Ordered collection

list = [1, 2, 3, "Hello"]

Features:

Mutable
Different data types store kar sakta hai

list[0] = 10

---

3. Tuple

List jaisa but immutable

t = (1, 2, 3)

Features:

Faster than list
Data change nahi kar sakte

(C) Set Data Types

1. set

Unique elements ka collection

s = {1, 2, 3, 3}
# Output: {1,2,3}

Features:

Duplicate allowed nahi
Unordered

2. frozenset

Immutable set

fs = frozenset([1, 2, 3])

(D) Mapping Data Type

dict (Dictionary)

Key-value pairs

d = {"name": "Deepak", "age": 20}

Features:

Keys unique hote hain
Mutable
Fast lookup

(E) Boolean Data Type

x = True
y = False

Conditions me use hota hai

(F) None Type

x = None

No value represent karta hai

Mutable vs Immutable

✓ Mutable (change ho sakta hai)

- list
- dict
- set

✓ Immutable (change nahi hota)

- int
- float
- str
- tuple

Type Checking

```
x = 10
print(type(x))
```

Type Conversion

✓ Implicit (automatic)

```
x = 5 + 2.5  # result float
```

✓ Explicit (manual)

```
int("10")
float(5)
str(100)
```

Important Concepts

(A) Indexing

```
name = "Python"
name[0]  # P
```

(B) Slicing

```
name[0:3]  # Pyt
```

(C) Length

len("Python") # 6

Example Program

```
a = 10
b = 3.14
name = "Deepak"
lst = [1, 2, 3]

print(type(a))
print(type(name))
```

Advantages of Data Types

✓ Memory efficient
✓ Correct operations ensure karta hai
✓ Error reduce karta hai

Variables:

Variable kya hota hai?

Definition:
Variable ek **naam (name)** hota hai jisme data store kiya jata hai.

Variable = data store karne ka container (box)

Example

```
name = "Rahul"
age = 20
```

Yahan:

- name → variable
- "Deepak" → value
- = → assignment operator

Variable kaise kaam karta hai?

Python me variable ek **reference** hota hai jo memory location ko point karta hai

```
x = 10
```

x memory me stored value 10 ko refer karta hai

Rules for Naming Variables

✓ Letter (a-z, A-Z) ya _ se start hona chahiye
✓ Number se start nahi kar sakte
✓ Special characters allowed nahi (@ # $ ❌)
✓ Keywords use nahi kar sakte (if, for ❌)
✓ Case-sensitive hota hai

Valid & Invalid Examples

✓ Valid:

```
name = "Ram"
_age = 20
user1 = "abc"
```

❌ Invalid:

```
1name = "Ram"   # number se start
my-name = "abc" # hyphen allowed nahi
```

Dynamic Typing

Python me variable ka type declare nahi karna padta

```
x = 10      # int
x = "Hello"  # string
```

Same variable ka type change ho sakta hai

Multiple Assignment

✓ Ek hi line me multiple variables:

```
a, b, c = 1, 2, 3
```

✓ Same value assign:

```
x = y = z = 10
```

Variable Types (Scope ke basis par)

(A) Local Variable

Function ke andar define hota hai

```
def func():
    x = 10  # local variable
```

(B) Global Variable

Function ke bahar define hota hai

```
x = 20  # global variable
```

Global keyword:

```
x = 10

def func():
    global x
    x = 20
```

Variable Memory Behavior

```
a = 10
b = a
```

Dono same value ko refer karte hain

Deleting Variables

```
x = 10
del x
```

Variable delete ho jata hai

Type Checking

```
x = 10
print(type(x))
```

Naming Conventions (Best Practice)

✓ Use meaningful names
✓ Snake_case use karein

```
user_name = "Deepak"
total_marks = 90
```

Constants (Convention)

Python me constant officially nahi hota, but uppercase use karte hain

```
PI = 3.14
```

Example Program

```
name = "Deepak"
age = 20

print(name)
print(age)
```

Expression and Operators:

1. Expression kya hota hai?

Definition:
Expression ek combination hota hai:
values + variables + operators
jo evaluate hokar ek result deta hai

Example:

```
x = 5 + 3
```

5 + 3 = expression
result = 8

Types of Expressions

(A) Arithmetic Expression

```
5 + 3
10 * 2
```

(B) Relational Expression

```
5 > 3   # True
```

(C) Logical Expression

```
(5 > 3) and (2 < 4)
```

(D) Assignment Expression

x = 10

(E) Bitwise Expression

5 & 3

---

2. Operator kya hota hai?

Definition:
Operator ek symbol hota hai jo operation perform karta hai.

Example:
+, -, *, /

Types of Operators in Python

(A) Arithmetic Operators

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
//	Floor Division
%	Modulus
**	Power

Example:

```
a = 10
b = 3

print(a + b)   # 13
print(a // b)  # 3
print(a % b)   # 1
```

(B) Relational (Comparison) Operators

Operator	Meaning
==	Equal
!=	Not equal
>	Greater
<	Less
>=	Greater or equal
<=	Less or equal

```
print(10 > 5)   # True
```

(C) Logical Operators

Operator	Meaning
and	AND
or	OR
not	NOT

```
True and False  # False
```

(D) Assignment Operators

Operator	Example
=	x = 5
+=	x += 2
-=	x -= 2
*=	x *= 2
/=	x /= 2

(E) Bitwise Operators

Operator	Meaning
&	AND
`	`
^	XOR
~	NOT
<<	Left shift
>>	Right shift

(F) Membership Operators

Operator	Meaning
in	present
not in	not present

```
"a" in "apple"   # True
```

(G) Identity Operators

Operator	Meaning
is	same object
is not	not same

```
a = [1,2]
b = a

print(a is b)  # True
```

Operator Precedence (Priority)

Kaunsa operator pehle chalega

Priority	Operators
High	**
	* / // %

Priority Operators

	+ -
	== != > <
	and
Low	or

Example:

```
5 + 3 * 2   # 11 (not 16)
```

Associativity

Same priority me kaun pehle chalega

- Left to Right
- Exception: ** (Right to Left)

```
2 ** 3 ** 2  # 512
```

Example Program

```
a = 10
b = 5

result = (a + b) * 2
check = a > b and b < 10

print(result)
print(check)
```

Numeric Operations:

Numeric Operations kya hote hain?

Definition:
Numbers (int, float, complex) par jo mathematical operations perform kiye jate hain, unhe **Numeric Operations** kehte hain.

Numeric Data Types (Recap)

✓ int → Integer (10, -5)
✓ float → Decimal (3.14)
✓ complex → (2 + 3j)

Basic Arithmetic Operations

(A) Addition (+)

a = 10 + 5 # 15

(B) Subtraction (-)

a = 10 - 3 # 7

(C) Multiplication (*)

a = 5 * 2 # 10

(D) Division (/)

a = 10 / 2 # 5.0

Always float return karta hai

(E) Floor Division (//)

a = 10 // 3 # 3

Decimal part hata deta hai

(F) Modulus (%)

a = 10 % 3 # 1

Remainder deta hai

(G) Exponentiation (**)

a = 2 ** 3 # 8

Power calculate karta hai

Mixed Type Operations

a = 5 + 2.5 # 7.5

Result automatically higher type me convert hota hai (float)

Type Conversion

✓ Explicit Conversion

int(3.9) # 3
float(5) # 5.0

✓ Implicit Conversion

```
5 + 2.0   # 7.0
```

Numeric Functions (Built-in)

abs() → absolute value

```
abs(-5)   # 5
```

round() → rounding

```
round(3.6)   # 4
```

pow() → power

```
pow(2, 3)   # 8
```

min() / max()

```
min(1, 2, 3)   # 1
max(1, 2, 3)   # 3
```

Math Module (Advanced Operations)

```
import math
math.sqrt(16)   # 4
math.factorial(5)  # 120
math.pi
```

Operator Precedence (Important)

Priority	Operators
High	**
	* / // %
Low	+ -

Example:

```
5 + 3 * 2   # 11
```

Special Numeric Values

Infinity

```
float("inf")
```

NaN (Not a Number)

float("nan")

Complex Number Operations

```
a = 2 + 3j
b = 1 + 2j

print(a + b)
print(a * b)
```

Example Program

```
a = 10
b = 3

print(a + b)
print(a / b)
print(a // b)
print(a % b)
print(a ** b)
```

Advantages

✓ Easy calculations
✓ Automatic type conversion
✓ Powerful built-in functions

Sequence Operations:

1. Sequence kya hota hai?

Definition:
Sequence ek ordered collection hota hai jisme elements ek fixed order me store hote hain.

Python ke Sequence Types

str (String)
list
tuple

Sequence Operations kya hote hain?

Sequence par perform hone wale common operations:

✓ Indexing
✓ Slicing
✓ Concatenation
✓ Repetition
✓ Membership
✓ Length

Important Sequence Operations

(A) Indexing

Element ko position ke basis par access karna

```
name = "Python"
print(name[0])   # P
print(name[-1])  # n
```

Index start hota hai 0 se
Negative index last se start hota hai

(B) Slicing

Sequence ka part nikalna

```
name = "Python"
print(name[0:3])   # Pyt
print(name[2:])    # thon
print(name[:4])    # Pyth
```

(C) Concatenation (+)

Do sequences ko jodna

```
a = "Hello"
b = "World"

print(a + b)   # HelloWorld
```

(D) Repetition (*)

Sequence ko repeat karna

```
print("Hi" * 3)   # HiHiHi
```

(E) Membership (in / not in)

Check karna element present hai ya nahi

```
print("a" in "apple")   # True
```

(F) Length (len())

Total elements count

```
print(len("Python"))   # 6
```

---

5. List Specific Operations

Add elements

```
lst = [1, 2]
lst.append(3)
```

Remove elements

```
lst.remove(2)
```

Sort

```
lst = [3, 1, 2]
lst.sort()
```

Tuple Operations

Tuple immutable hota hai

```
t = (1, 2, 3)
```

✓ Only access allowed
❌ Modification allowed nahi

String Operations

Upper / Lower

```
name = "python"

print(name.upper())  # PYTHON
```

Split

```
"hello world".split()
```

Sequence Functions (Common)

✓ len() → length
✓ max() → largest
✓ min() → smallest

```
lst = [1, 2, 3]
print(max(lst))
print(min(lst))
```

Example Program

```
lst = [1, 2, 3]

print(lst[0])       # indexing
print(lst[1:3])     # slicing
print(lst + [4,5])  # concatenation
print(lst * 2)      # repetition
print(2 in lst)     # membership
```

Important Points

✓ Sequence ordered hota hai
✓ Indexing aur slicing important hai
✓ Strings & tuples immutable hote hain
✓ Lists mutable hoti hain

Dictionary Operations

1. Dictionary kya hota hai?

Definition:
Dictionary ek **key-value pair** based data structure hota hai.

Simple:

Data ko key : value ke form me store karta hai

Example:

```
d = {"name": "Deepak", "age": 20}
```

name → key
"Rahul" → value

Dictionary ke Features

✓ Unordered (Python 3.7+ me insertion order maintain hota hai)
✓ Mutable (change ho sakta hai)
✓ Keys unique hoti hain
✓ Values duplicate ho sakti hain

Dictionary Operations

(A) Accessing Values

```
d = {"name": "Deepak", "age": 20}
print(d["name"])      # Deepak
print(d.get("age"))   # 20
```

get() safe hota hai (error nahi deta)

(B) Adding / Updating Elements

```
d["city"] = "Delhi"   # add
d["age"] = 25         # update
```

(C) Removing Elements

1. pop()

```
d.pop("age")
```

2. del

```
del d["name"]
```

3. clear()

```
d.clear()
```

(D) Checking Key

```
"name" in d   # True
```

(E) Length

```
len(d)
```

Important Dictionary Methods

keys()

```
d.keys()
```

Sab keys return karta hai

values()

```
d.values()
```

Sab values return karta hai

items()

d.items()

Key-value pairs return karta hai

update()

d.update({"age": 30})

copy()

new_d = d.copy()

Looping in Dictionary

```
for key in d:
   print(key, d[key])
```

Using items():

```
for k, v in d.items():
   print(k, v)
```

Nested Dictionary

```
d = {
   "student": {
      "name": "Deepak",
      "age": 20
   }
}
```

Dictionary Comprehension

```
square = {x: x*x for x in range(5)}
```

Example Program

```
d = {"a": 1, "b": 2}

# add
d["c"] = 3

# update
d["a"] = 10
```

```
# access
print(d["a"])

# delete
d.pop("b")

print(d)
```

The Print Statement:

1. Print Statement kya hota hai?

Definition:

print() ek **built-in function** hai jo output screen par display karta hai.

Basic Syntax

```
print("Hello World")
```

Print ke Examples

Multiple values print karna

```
name = "Deepak"
age = 20

print(name, age)
```

Separator use karna

```
print("A", "B", "C", sep="-")
```

Output: A-B-C

End parameter

```
print("Hello", end=" ")
print("World")
```

Output: Hello World

Formatted Output (f-string)

```
name = "Rahul"
print(f"My name is {name}")
```

The Conditional Statements:

Conditional Statements kya hote hain?

Definition:

Conditional statements ka use **decision lene ke liye** hota hai.

Types of Conditional Statements

(A) if Statement

```
age = 18

if age >= 18:
   print("Adult")
```

(B) if-else Statement

```
age = 16

if age >= 18:
   print("Adult")
else:
   print("Minor")
```

(C) if-elif-else Statement

```
marks = 75

if marks >= 90:
   print("A Grade")
elif marks >= 60:
   print("B Grade")
else:
   print("C Grade")
```

(D) Nested if

```
age = 20

if age >= 18:
   if age >= 21:
      print("Eligible for everything")
```

Important Concepts:

Indentation (Very Important ⚠️)

Python me indentation zaroori hota hai

```
if True:
    print("Correct")
```

Galat indentation → error

Conditions me Operators

```
if a > b:
if a == b:
if a != b:
```

Logical Operators

```
if a > 5 and b < 10:
```

Short-hand if (One-line)

```
if a > b: print("A is greater")
```

✓ Ternary Operator

```
result = "Adult" if age >= 18 else "Minor"
```

Example Program

```
name = "Rahul"
age = 20

print(f"Name: {name}")

if age >= 18:
    print("You can vote")
else:
    print("You cannot vote")
```

Looping Control Flow Statement:

Looping kya hota hai?

Definition:
Looping ka use ek hi block of code ko **baar-baar (repeat)** karne ke liye hota hai.

Python me Loop ke Types

Python me mainly 2 loops hote hain:

✓ for loop
✓ while loop

---

1. for Loop

Definition:

for loop ka use tab hota hai jab hume pata ho kitni baar loop chalana hai.

Syntax:

```
for variable in sequence:
   # code block
```

Example:

```
for i in range(5):
   print(i)
```

Output: 0 1 2 3 4

---

range() function

range(start, stop, step)

Examples:

```
range(5)       # 0 to 4
range(1, 5)    # 1 to 4
range(1, 10, 2) # odd numbers
```

Loop through list

```
lst = [1, 2, 3]
```

```
for i in lst:
    print(i)
```

---

2. while Loop

Definition:

while loop tab tak chalta hai jab tak condition true ho

Syntax:

```
while condition:
    # code
```

Example:

```
i = 1

while i <= 5:
    print(i)
    i += 1
```

Loop Control Statements

(A) break

Loop ko turant stop kar deta hai

```
for i in range(5):
    if i == 3:
        break
    print(i)
```

Output: 0 1 2

(B) continue

Current iteration skip karta hai

```
for i in range(5):
    if i == 2:
        continue
    print(i)
```

Output: 0 1 3 4

(C) pass

Kuch nahi karta (placeholder)

```
for i in range(5):
    pass
```

Nested Loops

Loop ke andar loop

```
for i in range(3):
    for j in range(2):
        print(i, j)
```

---

else with Loop

Python me loop ke sath else bhi use hota hai

```
for i in range(3):
    print(i)
else:
    print("Loop finished")
```

Infinite Loop

```
while True:
    print("Hello")
```

Ye kabhi stop nahi hoga (dangerous 😄)

Example Program

```
for i in range(1, 6):
    if i == 3:
        continue
    print(i)
```
