PYTHON PROGRAMMING AND DATA STRUCTURES

Unit I

Basics of Python Programming: Introduction to Python, Features of Python, Programming Modes - Interactive Mode & Script Mode, Identifiers, Naming Conventions, Keywords (Reserved Words), **Built-in Data Types**, Literals - Integer, Float, Complex, Boolean, String, Variables, **Operators**, Expressions, Assignment Statements, Input/Output Statements, Python Syntax (Lines, Comments, Indentation) Operators & Operands, **Classification of Operators -** Arithmetic Operators, Relational Operators, Logical Operators, Bitwise Operators, Assignment, Augmented Assignment, Identity Operators, Expressions & Precedence Rules.

---

computer languages:-

a computer language is a way of communicating with a computer.

→a language acts as a bridge between humans and computers. helps us create software, websites, mobile apps ,database, AI systems.

-→ programming languages are used to create programs that control the behaviour of a system, to express algorithms or as a mode of human communication.

Programming Languages :-

Programming = Programming is the process of giving instructions to a computer to perform a task or solve a problem.

Languages = Language is a system of communication used to share ideas, thoughts, instructions, or information.

->Language = a way to communicate.
Every language has words, grammar, and rules.
Without language, we cannot share information.

Programming languages are special languages for communicating with computers

What is a Programming Language?:-

a programming language is a special language used to write instructions that a computer can understand and execute.

(or)

it is a way for humans to communicate with computers and tell them what to do.
*It is used to write programs.
*It follows specific rules and syntax.

*Computers cannot understand human languages, so programming languages act as a bridge.
*Examples: Python, C, Java, JavaScript, C++.

Types of Programming Languages :-

1).Low-Level Languages :-

A Low-Level Language is a type of programming language that is very close to the computer's hardware.
It is difficult for humans to read, write, and understand, but very easy for the computer (CPU) to understand.

Why is it called "Low-Level"?

Because:

✓ It is very close to the computer hardware
✓ It is not similar to human language
✓ It is hard for humans to read and write

Types of low level languages:-

Machine Level Language?

Machine Level Language is the lowest level programming language and the only language that the computer's CPU can understand directly.

It is written compl etely in binary digits (0s and 1s).

Ex:- 10100011 11001010
00001111 00110011
→machine level language is used in microcontrollers,
embedded systems(washing machines ,Smart tv, remote controls, AC Controller),
firm ware (permanent software- BIOS used to start the computer and run the program),
device drivers, robotics, CPU Testing, and computer boot process(booting is the process of starting the computer.1.power on, 2. POST test(Power on self test-post checks CPU, RAM, Keyboard, Hard disk), 3.BIOS runs(Basic input output system,),
4.Bootloader loads(once BIOS finds a bootable device, it loads the bootloader),
5. Operating system loads into RAM(system files load), 6. computer is ready).

2.assembly language:-(second Generation language)

-assembly language developed in the mid-1950s was a great leap forward.it used symbolic codes, also known as mnemonic codes, which are easy to remember abbreviations, rather than numbers.
-uses mnemonics (symbols and short codes) instead of binary.
-easier than machine language but still hardware-dependent.
-needs an assembler to convert into machine code.

ex:-
ADD for adding, CMP for compare and MUL for multiply.
MOV AX,4 (stores the value 4 in the AX register of the CPU) MOV means Move
MOV BX,6 (stores the value 6 in the Bx register of the CPU
ADD AX,BX (adds the content of the AX and BX registers and stores the result in the AX register)

high-level languages:-(Third Generation Language)

-close to human language, easy to read/write.
-requires compiler/interpreter to convert into machine language.
Examples of High -Level Languages :-
Python, java, C++, C, JavaScript, PHP, Ruby, Swift, Kotlin, GO

Where High-Level Languages Are Used?
-Websites (JavaScript, PHP)
-Mobile Apps (Java, Kotlin, Swift)
-AI & Machine Learning (Python)
-Games (C++, C#)
-Banking Systems (Java)
-Desktop Apps
These languages are translated into machine code using compilers or interpreters.

What is a Compiler?

A Compiler is a software program that
--> converts the entire source code into machine code at once
and then executes it.

Example languages that use a compiler:

C, C++, Go, Swift
Java (partly compiler)

How it works:
-You write the program

-The compiler checks the whole code
-Converts everything into machine code
Then the program runs

Advantages:
-Very fast execution
-Errors are shown together after compiling
-Better for large and complex programs

What is an Interpreter?
An Interpreter is a software program that converts and executes the code line by line.

Example languages that use an interpreter:
Python
JavaScript
PHP
Ruby

How it works:
-Reads the first line → executes
-Reads the second line → executes
-Reads the third line → executes
(Continues like this)

Advantages:
-Easy to find errors (it stops exactly at the error line)
-Good for beginners
-Faster development

Feature	Compiler	Interpreter
Execution	Whole program at once	Line by line execution
Speed	Faster execution	Slower execution
Error Display	Shows all errors after compiling	Shows error immediately at that line
Output	Creates an executable file	Does not create an executable file
Examples	C, C++	Python, JavaScript

Features of Python

Python is a popular high-level programming language known for its simplicity and power. Below are the important features explained clearly:

1.Simple and Easy to Learn

Python has a vary simple syntax similar to English. It is easy for beginners to understand and write programs.

Ex:-

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

2. Interpreted Language

Python is an interpreted language, which means code is executed line by line. It does not need compilation like C or C++.

3. High-Level Language

Python hides complex details like memory management. Programmers can focus on logic instead of hardware-level operatons.

4. Object-Oriented

Python supports OOP concepts like:

- Class
- Object
- Inheritance
- Polymorphism
- Encapsulation

5. Platform Independent (Portable)

Python programs can run on different operating systems like:

- Windows
- Linux
- Mac

No need to change the code.

6. Dynamically Typed

In Python, you don’t need to declare the data type of a variable.

Example:

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

7. Large Standard Library

Python provides many built-in modules and libraries like:

- math
- random
- datetime

- os

This reduces the need to write extra code.

8. Free and Open Source

Python is free to use and open-source. Anyone can download and modify it.

9. Supports Multiple Programming Paradigms

Python supports:

- Procedural Programming
- Object-Oriented Programming
- Functional Programming

Applications of Python

1. Web Applications

You can build complete websites using Python.
-Python Web Frameworks:
-Django
-Flask
Examples of websites built using Python: Instagram, YouTube, Spotify
You can create:
College websites
E-commerce websites
Student result portals
News websites

2. Mobile Applications

Python can be used to build simple mobile apps using:
-Kivy (python library that helps you create mobile apps)
-BeeWare

You can create:
-Calculator apps
-To-do list apps (add, edit tasks , delete tasks, view pending work)CRUD Operation.
-Simple learning apps

3. Desktop Applications

Python can create computer applications using:
-Tkinter
-PyQt
-wxPython

You can create:

-Attendance software
-Billing systems
-Student management systems

4. Machine Learning & Artificial Intelligence Applications

Python is No.1 in AI/ML.
Using libraries like:
-NumPy
-Pandas
-TensorFlow
-Scikit-Learn

You can create:
-Face detection systems
-Voice recognition
-Chatbots
Recommendation systems (like YouTube, Amazon)

5. Data Science & Data Analytics Applications

Using Python you can:
-Analyze data
-Visualize charts
-Predict future trends

Libraries:
-Matplotlib
-Seaborn
-Pandas
Used in banking, hospitals, marketing, and business analytics.

6. Game Development

Python can create mini-games using Pygame.
Examples:
-Car racing game
-Snake game
-Shooting game
-Puzzle game

Programming Modes:-

Programming mode is the method of writing and executing programs in a programming language. In Python, it includes Interactive Mode and Script Mode

1).Interactive Mode:-

Interactive Mode is a programming mode in Python where the user can **enter commands one by one and get the output immediately**. It works at the Python prompt (>>>) and does not require saving the program in a file.

In this mode, Python follows the **Read–Evaluate–Print Loop (REPL)** process, where it reads the input, executes it, and displays the result instantly.

Ex:-

```
>>> a = 10
>>> b = 5
>>> a + b
15
```

2).Script Mode

Script Mode is a programming mode in Python where we **write the program in a file and then execute it**. The file is saved with a **.py extension**.

In this mode, the entire program is written at once and executed together, instead of line by line.

Ex:-

```
a = 10
b = 5
print(a + b)
```

Python Introduction:-

Python is a high-level, interpreted, general-purpose, object oriented programminglanguage used to develop different types of applications easily and quickly, which was introduced by Guido-van Rossum in 1991. python is used for web development from server side, software development etc.

1). why python is a high-level language ?

a language that is:

-> close to human language

-> easy to read & write (we write simple English like words)

2). why python is an interpreted language ?

a language where :

-> code is executed line by line

-> errors are show immediately.

-> python uses an interpreter because reads one line execute it then goes to next line.

3). why python is a General-Purpose Language ?

General Purpose means a language that can be used for many types of applications.

→ python is a general purpose programming language because python can be used to develop :

* web applications
* mobile applications
* desktop software
* AI & ML applications
* Games etc.

Python Syntax Compared to Other Programming Languages:-

What is Syntax?

Syntax is the set of rules that defines how a program must be written so the computer can understand it.

Syntax = grammar of a programming language

C Programming:-

```
#include <stdio.h>
int main() {
   printf("Hello");
   return 0;
}
```

Java Programming:-

```
class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello, World");
   }
}
```

Python programming :-

```
print("Hello, world")
```

Python syntax is simple and readable compared to other programming languages like C and Java. Python uses indentation instead of braces, does not require semicolons, and does not need data type declarations. Due to its English-like syntax and less code, Python is easy to learn and widely used, especially by beginners.

Python installation and setup :-

Python installation means installing python software on a computer so that we can write and run python programs.

steps to install python (windows):-

1). open browser
2). go to official website (www.python.org)
3). click download python (latest version)
4). run installer
5). installation complete

how to check python installation :-

-> using command prompt
1. press window + R
2. type cmd --> Enter
3. Type (python --version)

Python IDLE (Integrated Development and Learning Environment):-

it is default editor that comes with python to write and execute programs.

optional Editors/ IDEs.

-> VS Code
-> PyCharm
-> Jupyter Notebook etc.

How to create a python file :-

We create a Python file by writing code and saving it as a .py file or extension.
It can be run using Python in the command prompt or an IDE.

Valid Python File Names

✓ Must end with .py
✓ Can use letters, numbers, and underscore _
✓ Should not start with a number

Ex:-

hello.py
test.py
my_program.py
python1.py
student_data.py
calculator_app.py

invalid Python File Names:-

- missing .py extension (Ex:- hello, program.txt , etc..)
- spaces in file name (Ex:- my program.py, hello world.py)
- starting with a number (Ex:- 12test.py, 278.py)
- special characters (my@code.py, test#.py, etc.)
- using python keywords (print.py, if.py, for.py, class.py)

Basics Functions in python :-

print()

it is a predefine function in python and it is used to display output on the screen.

Ex:-

```
print('computer science')
print("sri venkateswara university")
print('''sgdc''')
ab=1202
print(ab)
```

id():- (Shows memory address)

it is a predefine function in python. the main objective of id() function is to know the address of variable or identifier where our value is stored at memory allocation.

Ex:-1

```
eid=1003
print(eid)
print(type(eid))
print(id(eid))
```

Ex:-2

```
a = 10
b = 10
print(id(a))
print(id(b))
```

Ex:-3

```
a = 10
b = 20
print(id(a), id(b))
```

type():-

it is a predefine function in python. it tells the data type of a value or variable.

Ex:1

```
a = 10
print(type(a))
```

Ex:-2
```
b = 10.5
print(type(b))
```

Ex:-3
```
c = "Python"
print(type(c))
```

Ex:-4
```
print(type(100))
print(type("Hello"))
print(type(3.14))
```

input():-

The input() function in Python is used to take input from the user during program execution.
Ex:-
Syntax of input()
```
variable_name = input("message")
```

Example:
```
name = input("Enter your name: ")
print(name)
```

what is predefine function ?

Predefined functions are built-in functions provided by Python. These functions are already defined during the development of Python and are available to the programmer for direct use without writing their definitions.

(or)

During the development of the Python programming language, some useful functions were already created and provided by the Python developers. These functions are called predefined functions or built-in functions.

Ex:-

print() -- display output
input() -- take input from user
type() -- find data type
id() -- memory address
len() -- length of data

sum() -- total of values
max() -- biggest value
min() -- smallest value

Comments in python :-

comments are non-executable lines in a python program. they are written to explain the code and help humans understand what the program is doing.
→ python ignores comments while running the program.

Types of comments in python :-

1). single-line comment

written using the # (hash) symbol.

Ex:-

```
# This is a single-line comment
print("Hello World")  # This prints output
```

2). multi-line comment:-

written using triple quotes
Ex:-

```
"""
This is a multi-line comment
Used to explain big programs
"""
```

Data Types in python:-

Data types specify the type of data stored in a variable. python supports numeric, string, Boolean, list, tuple, set, dictionary and none data types.
Data type= type of data stored in a variable.

A. Numeric Data Types :-

used to store numbers.

1). int(integer):-

int data type is used to store whole numbers in python. it can be represent as positive number or negative number.

Ex:-

```
x1=1600
print(x1, type(x1))
x2=-455
print(x2,type(x2))
```

2). float:-(decimal numbers)

it can be represent as decimal point number or floating point number. it may be either positive float decimal or negative floating decimal number.

Ex:-

```
x1=123.36
print(x1,type(x1),id(x1))
x2=-561.54
print(x2,type(x2),id(x2))
```

Ex:-

```
x1=1.3*10*10
print(x1,type(x1), id(x1))
```

Ex:-

```
x1=1.3e2
print(x1, type(x1), id(x1))
```

3). complex:-

complex data type in python is used to represent numbers having real part and an imaginary part.

it is written in the form a +bj, where j represents the imaginary unit. python allows only j or J, not i.

Ex:- x1=100+500j

100 ----> real part

500j ---> imaginary part

Ex:-

```
x1=200+5j
x2=-250-5j
print(x1, type(x1))
print(x2, type(x2))
```

Ex:- (finding real and imaginary part)

```
x1=500+20.21j
print("Real Part:",x1.real)
print("imaginary part:",x1.imag)
```

Ex:-
x1 = 2j + 5J
print(x1, "real part", x1.real, "imaginary part", x1.imag)

output:-
7j real part 0.0 imaginary part 7.0

4). string data type (str):-

String is a data type in Python used to store text such as characters, words, or sentences.
Strings are written inside single quotes, double quotes, or triple quotes.

- while working with string data type space also consider as one character.
- python supports positive index which starts from 0 to end -1. it is also known as forward direction.
- python also supports negative direction which starts from -1 to end +1. it is also known as backword direction.
- string is immutable data type

Ex:-
```
str1='core python'
str2="Advance Python"
str3='''Django'''
str4="""javascript"""
str5='120'
str6="123.67"
print(str1,str2,str3,str4,str5,str6)
```

Accessing Characters in String (Indexing) :-

Indexing means accessing individual characters from a string using their position number.

In Python, every character in a string has a position (index).

- Indexing starts from 0, not from 1
- Spaces are also counted as characters

Positive Indexing :-

Positive indexing starts from 0 (left to right).
Ex:-
name = "PYTHON"

P	Y	T	H	O	N
0	1	2	3	4	5

How to Access a Character

Syntax:

string_name[index]

Examples:

```
name = "PYTHON"
print(name[0])
print(name[3])
print(name[5])
```

Example with Sentence (Including Space)

```
text = "Hello World"
print(text[0])
print(text[5])
print(text[6])
print(text[10])
```

H	e	l	l	o		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

Negative Indexing:-

Negative indexing means counting characters from the end of the string (right to left).
Python gives minus (-) numbers for characters from the end.

Ex:-

P	Y	T	H	O	N
-6	-5	-4	-3	-2	-1

Ex:-

```
name = "PYTHON"
print(name[-1])
print(name[-2])
print(name[-3])
print(name[-6])
```

slice operator in string :-

the main objective of slice operator is to make the pieces of string object. slice operator is applicable for positive direction and negative direction as well.

Slice Operator Syntax

string[start : end : step]

start → where to begin (included)

end → where to stop (excluded)

step → jump count (optional)

Ex:-
```
s = "PYTHON"
print(s[0:4])
```

Ex:-
```
s="PYTHON"
print(s[2:5])
```

Ex:- (Starts from beginning automatically.)
```
s="PYTHON"
print(s[:3])
```

Ex:-
```
s="PYTHON"
print(s[3:])
```

Ex:- Reverse String (Negative Step)
```
str1="Core Python"
print(str1[::-1])
```

Ex:-
```
str1 = "Core Python"
print(str1[-6:-1])
```

5). Boolean data type (bool):-

Boolean data type in python is used to represent logical values. It has only two possible values: True and False. The Boolean data type is written as bool in python. When we compare two values using operators such as ==, >, <, != the result will always be either True or False.

Ex:-
```
a = True
b = False
print(a)
print(b)
```

Ex:-
```
x = 10
y = 5
print(x > y)
print(x < y)
print(x == y)
```

Ex:-

```
x1=15
x2=170
print(x1)
print(x2)
x3=x1==x2
print("the result is:",x3)
x4=x1!=x2
print("the result is:",x4)
```

B). sequence data types:-

A sequence is an ordered collection of items, which can be of similar or different data types. In sequence data types, the elements are arranged in a specific order and can be accessed using indexing and slicing. The main sequence data types in python are list, tuple, string and range.

6). List

List is a collection of different or same type values stored in a single variable, written using square brackets []. It is ordered, indexed, mutable(make the changes), and allows duplicate values.

mutable object:-(stateful object)

Mutable data types can be changed after creation.

Ex:- list, set,dict,bytearray

immutable (stateless object):-

once we create an object we cannot perform operations or we cannot modify that object then it is said to immutable object.

Ex:- int, float, str, complex, tuple, bytes.

how to create a python list?

the list can be created using either the list constructor or using square brackets [].

Syntax of List

```
list_name = [value1, value2, value3]
```

1. using list() constructor :-

Ex:-

```
chars = list("Python")
print(chars)
```

2. using square brackets[]. the items inside the square brackets.

Ex:-
numbers = [10, 20, 30, 40]
print(numbers)

Ex:-
a1=[10,"Sgdc",9.24,98,"mpl"]
print(a1)

7). Tuple

tuple is a sequence data type in python used to store multiple values in a single variable. A tuple is written using round brackets( ), and the elements are separated by commas. Tuple are ordered and allow duplicate values, but they are immutable, which means we cannot change add, or remove elements after creation.

Elements in a tuple can be accessed using indexing, and slicing, starting from index 0.

how to create a tuple:-

1.using parenthesis():-
2.using a tuple() constructor
Ex:-
t = (10, 20, 30)
print(t)
Ex:
t = (1, "Python", 3.5, True)
print(t)

Ex:-
n1=(10,20,"python",12.34)
print(n1,type(n1))

8). Range

The range() function in Python is used to generate a sequence of numbers within a specified range.
It is most commonly used in loops, especially for loops, to repeat an action a specific number of times.
range() returns a sequence of numbers starting from a value and ending before a value.

syntax:-

for x1 in range(begin,end(end-1),step):

Ex:-

```
for x1 in range(10):
   print(x1)
```

Ex:-

```
for x1 in range(10):
   print(x1,end=" ")
```

Ex:-

```
for a1 in range(3,16):
   print(a1,end=" ")
```

Ex:-

```
for i in range(5):
   print(i, end=" ")
```

9). set data type:-

Set is a built-in data type in python used to store multiple values in single variable. A set is written using curly brackets { } and the elements are separated by commas. Set are unordered and unindexed , which means the items do not have a fixed position and cannot be accessed using indexing.

One important feature of a set is that it does not allow duplicate values, if duplicate elements are given, it automatically removes them. Sets are also mutable, so we can add or remove elements using methods like add() and remove().

Creating a Set

1). Using curly braces { }

Ex:-

```
s={10, 20, 30, 40}
print(s,type(s))
```

2).Using set() constructor

Ex:-

```
s = set([10, 20, 30, 40])
print(s)
```

1. Unordered

-->Elements have no fixed position

-->Indexing and slicing are not allowed

Ex:-

```
s = {10, 20, 30}
print(s[0])
```

output:-
TypeError: 'set' object is not subscriptable

2.No Duplicate Values

```
s = {1, 2, 2, 3, 3}
print(s)
```

3. Mutable:-

Adding elements

```
s = {10, 20, 30}
s.add(40)
print(s)
```

Removing elements

```
s = {10, 20, 30}
s.remove(20)
print(s)
```

10). Dictionary data type :-

Dictionary is a built-in data type in python used to store data in key-value pairs. It is written using curly brackets { }, where each key is separated from its value by a colon : and pairs are separated by commas.

Dictionaries are unordered, immutable and do not allow duplicate keys (but values can be duplicated). Each value in a dictionary is accessed using its unique key instead of an index number.

Creating a Dictionary

Using curly braces { }

```
d1 = {1: 'Shri', 2: 'Gnanambica', 3: 'Degree'}
print(d1)
```

Using dict() constructor

```
d2 = dict(a="Shri", b="Gnanambica", c="Degree")
print(d2)
```

Ex:-

```
Product_info={"pid":1001,"pname":"laptop", "price":45000}
print(Product_info, type(Product_info))
```

Ex:-

```
n1={0:100,1:200,2:300,3:400}
print(n1, type(n1))
```

11). None data type :-

None is a special data type in python that represents the absence of a value. It is written as None and belongs to the data type called NoneType. None means “no value” or “nothing”. It is commonly used to initialize variables when no value is assigned yet, or as the default return value of a function.

Ex:-

```
x = None
print(x)
print(type(x))
```

Ex:-

```
e1=35000
print("emp salary:",e1)
e2=None
print(e2)
```

12). Bytes data type:-

Bytes data type in Python is used to store binary data in the form of values from 0 to 255. It is immutable and mainly used in file handling, networking, and hardware communication. Bytes can be created using the bytes() constructor or by using 'b' prefix.

ASCII values

Capital letters: A–Z → 65–90
Small letters: a–z → 97–122
Digits: 0–9 → 48–57

How to Create Bytes Data Type?

There are two common ways.

1. Using bytes() constructor:-

Important Rules of bytes
Allowed:-

✓ Integers from 0 to 255
✓ Byte literals using b" "
✓ Used for binary data

--> The bytes() constructor accepts only integer values between 0 and 255. Float values and strings are not allowed

Ex:-

```
b = bytes([65, 66, 67, 68])
```

```
print(b)
```

Output:
b'ABCD'

Explanation:
65 → A
66 → B
67 → C
68 → D
These numbers are converted using ASCII values.

2. Using b prefix (byte literal)

In Python, the b prefix is used to represent byte literals. Inside b"", only characters are allowed. To create bytes using numeric values, the bytes() constructor must be used.

Ex:-

```
x = b"Gnanambica"
print(x)
print(list(x))
```

In the given code, the string "Gnanambica" is converted into bytes using the b prefix.
The list() function converts the bytes object into a list of ASCII values of each character.

Bytes is IMMUTABLE (cannot be changed)

Example:

```
b = bytes([65, 66, 67])
b[0] = 90
print(b)
```

output:-
TypeError: 'bytes' object does not support item assignment

Accessing Bytes Elements:-

You can access bytes using indexing.
Ex:-

```
b = bytes([65, 66, 67])
print(b[0])
print(b[1])
```

Ex:-

```
a1=b"Python"
print(a1[0])
print(a1[1])
```

(it prints numbers because bytes store numeric values).

Python Code to Find ASCII Value of a Character:

ord()

ord = ordinal value

ord() is a built-in Python function that is used to find the ASCII / Unicode value of a character. The ord() function accepts only one character at a time. If multiple characters are given, python raises an error. To get Unicode values of multiple characters, we must apply ord() to each character separately.

Ex:-

```
ch='A'
value=ord(ch)
print(value)
```

Ex:-

```
a1='?'
print(ord(a1))
```

Ex:-

```
ch=input('Enter a character:')
print('ASCII value:', ord(ch))
```

chr()

chr() is a built-in Python function that is used to convert an ASCII / Unicode value into a character.

chr = character

-->It accepts an integer

-->Returns a character

converting ASCII Value → Character:-

Ex:-

```
num=67
print(chr(num))
```

Ex:-

```
num=128512
print(chr(num))
```

Ex:-

```
num = int(input("Enter ASCII value: "))
print("Character is:", chr(num))
```

13). Bytearray data type:-

Bytearray is a mutable sequence of bytes in python. It stores binary data as integers between 0 and 255. Unlike bytes, bytearray allows modification of elements. It is used in file handling, network programming, and situations where binary data needs to be changed.

Ex:-

```
b1=bytearray([65,66,67,68])
print(b1)
```

Ex:- (Accessing Elements)

```
b1=bytearray([10,20,30])
print(b1[0])
print(b1[1])
```

Ex:- (Modifying Element)

```
b1=bytearray([65,66,67])
b1[0]=90
print(b1)
```

14) Frozenset data type:-

Frozenset is a built-in data type in python that is similar to a set but immutable. This means once a frozenset is created, its elements cannot be changed, added or removed. Frozenset is unordered, does not allow duplicate values, and supports mathematical operations.

How to Create a Frozenset

Using frozenset() constructor:-

```
fs = frozenset([10, 20, 30, 40])
print(fs, type(fs))
```

Ex:-

```
s = {1, 2, 3}
fs = frozenset(s)
print(fs)
```

Ex:-

```
fs = frozenset("python")
print(fs)
```

Ex:-

```
d = {frozenset([1,2,3]): "Python"}
print(d)
```

When to Use Frozenset?

✓ You want data that must not change

✓ You need a set as a dictionary key

✓ You want data safety

Operators in Python:-

Operators in python are special symbols used to perform operations on variables and operands (values). They are used to perform mathematical calculations, compare values, assign values, and apply logical conditions in a program.

A=10
B=5
C=a+b
In this example + is a operator and a, b are operands.

Python has seven types of operators that we can use to perform different operations and produce a result.

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

1). Arithmetic Operator:-

Arithmetic operators in Python are used to perform basic mathematical operations such as addition, subtraction, multiplication, division, modulus, floor division, and exponentiation on numeric values.

there are seven arithmetic operators we can use to perform different mathematical operations, such as:

1. + (Addition)
2. - (subtraction)
3. * (multiplication)
4. / (division)
5. // (floor division)
6. % (modulus)
7. ** (Exponentiation)

Ex:- **Addition Operator (+)** --> Used to add two or more values.

```
a = 10
b = 20
print(a + b)
```

Ex:- **Subtraction Operator (-)** -->Used to subtract one value from another.

```
a = 20
b = 10
print(a - b)
```

Ex:- **Multiplication Operator (*)** --> Used to multiply values.

```
a = 5
b = 4
print(a * b)
```

Ex:- **Division Operator (/)** --> Used to divide two numbers.

```
a = 10
b = 2
print(a / b)
```

Ex**:- the floor division // r**ounds the result down to the nearest whole number

```
x = 15
y = 2
print(x // y)
```

output: 7

Ex:- **% (modulus)**--> returns the remainder after division of two numbers.

```
a=10
b=3
print(a%b)
```

Ex**:-** ** **(Exponentiation)**--> used to calculate the power of a number.

Syntax

```
result = base ** exponent
```

base → number to be multiplied

exponent → how many times it is multiplied

```
a=2
b=3
print(a**b)
```

Explanation:-

2 ** 3 = 2 × 2 × 2 = 8

Ex:-

```
a = 15
b = 4
print("Addition:", a + b)
print("Subtraction:", a - b)
print("Multiplication:", a * b)
print("Division:", a / b)
print("Floor Division:", a // b)
print("Modulus:", a % b)
print("Exponentiation:", a ** b)
```

Output:-

Addition: 19
Subtraction: 11
Multiplication: 60
Division: 3.75
Floor Division: 3
Modulus: 3
Exponentiation: 50625

2. Assignment Operator:-

in python, assignment operator are used to assigning value to the variable. Assign operator is denoted by = symbol. there are shorthand assignment operators in python, for example a+=2 which is equivalent to a=a+2.

Operator	Example	Same As
=	a=10	a=10
+=	a+=30	a=a+30
-=	a-=15	a=a-15
*=	a*=10	a=a*10
/=	a/=5	a=a/5
%=	a%=5	a=a%5
**=	a**=4	a=a**4
//=	a//=5	a=a//5
>>=	a>>=5	a=a>>5
<<=	a<<=5	a=a<<5

Ex:-

Simple Assignment (=)

```
marks = 85
print(marks)
```

Ex:- **Add and Assign (+=)**

```
score = 50
score += 20
print(score)
```

Ex:- **Subtract and Assign (-=)**

```
balance = 1000
balance -= 250
print(balance)
```

Ex:- **Multiply and Assign (*=)**

```
salary = 20000
salary *= 2
print(salary)
```

Ex:- **Divide and Assign (/=)**

```
distance = 100
distance /= 4
print(distance)
```

Ex:- **Floor Divide and Assign (//=)**

```
students = 50
groups = 6
students //= groups
print(students)
```

Ex:-

```
a=10
print('initial value:',a)
a+=5
print('addition:',a)
a-=3
print('subtraction:',a)
a*=2
print('multiplication:',a)
a/=4
print('Division:',a)
```

```
a//=2
print('floor_division:',a)
a%=3
print('modulus:',a)
a**=2
print('Exponentiation:',a)
```

3). Relational Operator(Comparison Operator):-

Relational Operator are also called comparison operators. It is used to compare two values or variables. These operators check the relationship between the values and return a Boolean result, which is either True or False.

The main relational operators in python are:

==(equal to), != (not equal to), > (greater than), < (less than), >= (greater than or equal to), <= (less than or equal to).

Ex:- (**Equal to (==)** --> Checks whether both values are equal.)

```
a=10
b=10
print(a==b)
```

Ex:- (**Not Equal to (!=)** --> Checks whether values are different.)

```
a = 10
b = 5
print(a != b)
```

Ex:- (**Greater Than (>)**--> Checks if the left value is greater than the right value.)

```
a = 20
b = 10
print(a > b)
```

Ex:- (**Less Than (<)** --> Checks if the left value is smaller than the right value.)

```
a = 5
b = 10
print(a < b)
```

Ex:- (**Greater Than or Equal to (>=)** --> Checks if the value is greater or equal)

```
marks = 35
print(marks >= 35)
```

Ex:- (**Less Than or Equal to (<=)** --> Checks if the value is less or equal.)

```
age = 18
print(age <= 18)
```

4). Logical operator:-

Logical operator in python are used to combine two or more conditions and return a Boolean result, which is either True or False. They are mainly used in decision-making statements such as if, while and other conditional expressions.

Python has three logical operators: and, or and not.

Types of Logical Operators in Python:-

and:-

the and operator returns True only when both conditions are True. If any one condition is False, the result will be False.

Condition 1	**Condition 2**	**Result**
True (1)	True(1)	True(1)
True(1)	False(0)	False(0)
False(0)	True(1)	False(0)
False(0)	False(0)	False(0)

Ex:-

```
a = 10
b = 20
print(a > 5 and b > 15)
```

ex:-

```
username=input("Enter the username:")
password=input("Enter the password:")
if(username=="sgdc" and password=="sgdc@123"):
   print("Welcome to Shri Gnanambica Degree College")
else:
   print("Provide Valid Details")
```

or:-

the or operator returns True if at least one condition is True. If both conditions are False, the result will be False.

Condition 1	Condition 2	Result
True (1)	True(1)	True(1)
True(1)	False(0)	True(1)
False(0)	True(1)	True(1)
False(0)	False(0)	False(0)

Ex:-

```
a = 10
b = 20
print(a > 50 or b > 15)
```

Ex:-

```
first_name=input("Enter the First Name:")
Lat_name=input("Enter the last name:")
username=input("Enter the username:")
password=input("Enter the password:")
if(first_name== "shri" or Lat_name=="gnanambica") and (username=="sgdc" or password=="sgdc123"):
   print("welcome to Shri Gnanambica Degree College")
else:
   print("please provide valid details")
```

not:-

the not operator is used to reverse the result. If the condition is True, it becomes False. If the condition is False, it becomes True.

True False
False True

Ex:-

```
s='shri'
print(not h)
```

5). Membership operator:-

Membership operator are used to check whether a value is present in a sequence or collection such as list, tuple, string, set or dictionary.

Python has two membership operators:

1). in
2). not in

1). in

The in operator is used to check whether a value exists in a sequence. If the value is present, it returns True, otherwise False.

Ex:-
a = [10,20,30,40]
print(20 in a)

Ex:-
name = "python"
print('p' in name)

2). not in

The not in operator is used to check whether a value is not present in a sequence.
If the value is not present → True
If the value is present → False
Ex:-
a = [10,20,30,40]
print(50 not in a)

Ex:-
text = "python"
print('z' not in text)
Ex:-
fruits = ("apple", "banana", "mango")
print("apple" in fruits)

Ex:-
student = {"name": "Ravi", "age": 20}
print("name" in student)

(Note:- In dictionaries, membership checks only keys, not values.)

6). Identity operator:-

Identity operator are used to check whether two variables refer to the same object in memory. These operators compare the memory location of objects, not just their values.
Python has two identity operators:
1). is
2). is not

1). is

The is operator returns True if both variables refer to the same object in memory.
If they refer to different objects, it returns False. (same memory location)

Ex:-
a = 10
b = 10
print(a is b)

Ex:-
Ex:-
s1=(12, "python")
print(id(s1))
s2=(12, "python")
print(id(s2))
print(s1 is s2)

2). is not

The is not operator returns True if the variables refer to different objects in memory.
Ex:-
c1="python"
d1="python"
print(c1 is d1)

Ex:-
a=10
print(id(a))
b=10
print(id(b))
print(a is not b)

7). Bitwise operator:-

Bitwise operator used to performing bitwise operations in integers. To perform bitwise, we first need to convert integer value to binary(0 and 1) value.
The bitwise operator operates on values bit by bit, so it is called bitwise. It always returns the result in decimal format.
in python has 6 bitwise operators listed below.
1. & bitwise and
2. | bitwise or
3. ^ bitwise xor
4. ~ bitwise 1's complement.or bitwise not
5. << bitwise left-sift
6. >> bitwise right shift.

1. & bitwise and :-

The **Bitwise AND operator (&)** is used to perform an **AND operation on the binary representation of two numbers**. It compares each bit of the numbers and returns **1 only when both bits are 1**, otherwise it returns **0**.

0 & 0 is 0
1 & 0 is 0
0 & 1 is 0
1 & 1 is 1

Ex:-

```
a = 5
b = 3
result = a & b
print(result)
```

Ex:-

```
a = 6
b = 4
print(a & b)
```

Ex:-

```
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
print("Bitwise AND:", num1 & num2)
```

2. | bitwise or:-

The **Bitwise OR operator (|) pipe symbol** is used to perform an **OR operation on the binary bits of two numbers**. It compares each bit of the numbers and returns **1 if at least one of the bits is 1**, otherwise it returns **0**.

A	B	A | B
0	0	0
0	1	1
1	0	1
1	1	1

Ex:-
a = 5
b = 3
print(a | b)

Ex:-
a = 6
b = 4
print(a | b)

3. Bitwise XOR operator (^ caret symbol):-

The **Bitwise XOR (Exclusive OR) operator ^** is used to compare the **binary bits of two numbers**.

It returns **1 when the bits are different** and **0 when the bits are the same**.

A	**B**	**A ^ B**
0	0	0
0	1	1
1	0	1
1	1	0

Ex:-

a = 5
b = 3
print(a ^ b)

Ex:-
a = 10
b = 7
print(a ^ b)

4.Bitwise NOT Operator (~) :-

The **Bitwise NOT operator (~)** is used to **invert (reverse) all the bits** of a number. It changes **1 to 0** and **0 to 1** in the binary representation.

~n = -(n + 1)

Ex:-
a = 5
print(~a)

Ex:-
a = 10
print(~a)

5. << bitwise left-sift

The left shift operator (<<) is a bitwise operator used to shift the bits of a number to the left side by a specified number of positions. It is written as a << b, where a is the value to be shifted and b is the number of positions.

When the bits are shifted to the left, the empty positions on the right side are filled with zeros (0). The bits that move out from the left side are discarded.

Ex:-
a = 5
print(a << 1)

Explanation:

Binary of 5: 00000101
After left shift by 1 position: 00001010
Decimal value of 00001010 is 10

Output:

10

6). Right-Shift operator (>>)

The right shift operator (>>) is a bitwise operator used to shift the bits of a number to the right side by a specified number of positions. It is written as a >> b, where a is the value to be shifted and b is the number of positions.

When the bits are shifted to the right, the empty positions on the left side are filled with zeros for positive numbers. The bits that move out from the right side are discarded.

Ex:-

```
a = 20
print(a >> 1)
```

Explanation:

Binary of 20: 00010100
After right shift by 1 position: 00001010

Decimal value of 00001010 is 10

Output:
10

Type Conversion:-

Type conversion in Python is the process of changing one data type into another, which is essential for data manipulation and compatibility in operations. There are two primary types of conversion: implicit (automatic) and explicit (manual, also known as type casting).

Implicit Type Conversion:-

In implicit conversion, Python automatically converts one data type into another during expression evaluation.
In implicit type conversion of data types in Python, the Python interpreter automatically converts one data type to another without any user involvement.
Ex:-

```
x = 10
y = 10.6
z = x + y
print("x:", type(x))
print("y:", type(y))
print("z =", z)
print("z :", type(z))
```

Explicit Type Conversion:-

Explicit type conversion requires the programmer to manually convert data types using built-in functions. This is necessary when Python cannot perform an automatic conversion or when you need control over the resulting type, though it may result in data loss.

common types casting functions:-

function	converts to
int()	integer
float()	float
str()	string
complex()	complex
list()	list
tuple()	tuple()
set()	set()

Ex:- (String to Integer)

```
x = "25"
y = int(x)
print(y, type(y))
```

Ex:-

```
a = "12.5"
b = float(a)
print(b)
```

Ex:-

```
num = 100
print(str(num))
```

Ex:- (complex conversion)

```
x = complex(5)
y = complex(2, 3)
print(x)
print(y)
```

Ex:-
s = "python"
print(list(s))

5 Marks

1. Explain Identifiers in Python?
2. What are keywords in Python?
3. Define Variables in Python. How are Variables created and assigned values?
4. Explain Arithmetic and Relational Operators in Python with examples.
5. List any five features of Python

10 Marks

1. Explain Built-in Data Types in Python with suitable examples and programs.
2. What are Operators in Python? Explain the Classification of Operators with examples
3. Explain in detail the Features of Python. Why is Python considered a high-level language?
4. Explain Expressions and Operator Precedence Rules in Python with examples.
5. What are Identity Operators and Bitwise Operators? Explain with examples and truth tables.

B. Devendra Msc-CS
Dept of Computers
Shri Gnanambica Degree College(A)