What is Python?

Python is a high-level, general-purpose programming language known for its readability and versatility. It's widely used in various fields, including web development, data science, machine learning, and software development. Python is an interpreted language, meaning code is executed line by line. Its simple syntax, similar to English, makes it beginner-friendly and popular among experienced programmers.

Here's a more detailed breakdown:

Versatile and General-Purpose: Python isn't specialized for a particular task but can be used in many applications.

Readability and Beginner-Friendly: Its syntax is designed to be easy to understand, even for those new to programming.

Interpreted Language: Python code is executed directly, line by line, without needing to be compiled first.

Object-Oriented: Python supports object-oriented programming principles, allowing for modular and reusable code.

Large Standard Library: Python comes with a vast collection of pre-written code modules for various tasks.

Widely Used: It's a popular choice for web development, data analysis, machine learning, and software development.

Cross-Platform: Python runs on various operating systems like Windows, macOS, and Linux

What is Modules and Pip in Python?

In Python, modules are reusable blocks of code that extend the language's functionality. pip is the package manager for Python, used to install and manage these external modules.

Reusability: Modules are designed to be reused in different parts of a program or even in different programs.

Organization: They help organize code, making it easier to maintain and understand.

Extensibility: Modules provide access to a wide range of functionalities without requiring you to write the code yourself.

Examples: Standard Python modules like math, random, or datetime are pre-installed with the language. External modules can be installed using pip, such as numpy, pandas, or requests.

pip (Package Installer for Python):

- **Installation:** pip is the primary tool for installing Python packages, including external modules.
- **Dependency Management:** It helps manage dependencies between different packages.
- **Virtual Environments:** pip can be used with virtual environments to isolate project dependencies.
- **Command-line usage:** The basic command is pip install <module_name>

Comments in python

A **comment** is a line in your code that is **ignored by Python when your program runs**. It's used to **explain code** or leave **notes for yourself or others**.

1. Single-line Comment

Use # at the beginning of the line.

```
# This is a single-line comment

print("Hello, World!")  # This prints a message
```

2. Multi-line Comment

Python doesn't have a true multi-line comment, but you can use triple quotes (''' or """) to simulate it.

```
'''
This is a
multi-line comment
'''
print("Welcome to Python")
```

What is an Escape Sequence?

Escape sequences start with a **backslash** \, and are used to **represent special characters** in a string that can't be typed directly.

Escape Sequence	**Meaning**	**Example**
\n	New Line	"Hello\nWorld"
\t	Tab Space	"Name:\tAlice"
\\	Backslash	"This is a backslash: \\"
\'	Single Quote	'It\'s a sunny day'
\"	Double Quote	"He said, \"Hi!\""

Variables and Data types in python

What is a Variable?

A **variable** is a **name** that stores a value in memory.
It acts like a **container** for data.

Rules for Naming Variables:

- Must start with a letter or underscore (_)
- Cannot start with a number
- Can only contain letters, numbers, and underscores
- **Case-sensitive** (name ≠ Name)

Python Data Types

Python has built-in data types to classify different kinds of data.

1. Basic Data Types in Python:

Data Type	Example	Description
int	10, -5, 1000	Whole numbers
float	3.14, -2.0	Decimal numbers
str	"Hello"	Text - (string of characters)
bool	True, False	Boolean values

2.Collection Data Types:

Data Type	Example	Description
list	[1, 2, 3]	Ordered, changeable, allows duplicates
tuple	(1, 2, 3)	Ordered, unchangeable, allows duplicates
set	{1, 2, 3}	Unordered, no duplicates
dict	{"name": "Alice", "age": 25}	Key-value pairs

Exercise:1

1. Add two numbers
2. Find the square and cube of a number
3. Calculate area of a rectangle
4. Convert temperature (Celsius ↔ Fahrenheit)
5. Simple interest calculator

Typecasting in python

Typecasting in Python means converting one data type into another. It's useful when you want to perform operations between different data types or format data in a specific way.

There are two types of typecasting in Python:

1. Implicit Typecasting

Python automatically converts one data type to another without your involvement.

```
a = 5        # int
b = 2.0      # float

result = a + b
print(result)          # Output: 7.0
print(type(result))    # Output: <class 'float'>
```

2. Explicit Typecasting

You manually convert the data type using type functions like:

- `int()` – converts to integer
- `float()` – converts to float
- `str()` – converts to string
- `bool()` – converts to boolean

Examples:

```
# String to int
num_str = "10" #print("Hello" 6,9,0,sep="$",end="JJA")
num_int = int(num_str)
print(num_int)          # Output: 10
# Float to int
f = 7.9
print(int(f))           # Output: 7 (decimal part is removed)

# Int to string
```

```
x = 100
print(str(x))           # Output: '100'

# Int to float
y = 5
print(float(y))         # Output: 5.0
```

taking input from user

Basic Example:

```
name = input("Enter your name: ")
print("Hello,", name)
```

using typecasting:

```
age = int(input("Enter your age: "))    # Converts input string
to integer
print("You are", age, "years old.")
```

Exercise 2:

1. Take your full name as input and print: "Nice to meet you, <full name>!"
2. Input three numbers and print their total sum.
3. Take two floating-point numbers and print their multiplication result.
4. Take your birth year and current year as input, then print your age.
5. Input the length of one side of a square and print its area and perimeter.
   (Area = side × side, Perimeter = 4 × side)

operators in python

1. Arithmetic Operators (Used for mathematical operations)

Operator	**Description**	**Example**
+	Addition	`5 + 3 = 8`
-	Subtraction	`5 - 2 = 3`
*	Multiplication	`4 * 2 = 8`
/	Division	`8 / 2 = 4.0`

Operator	Description	Example
`//`	Floor Division	`8 // 3 = 2`
`%`	Modulus (remainder)	`7 % 3 = 1`
`**`	Exponent (power)	`2 ** 3 = 8`

2. Comparison Operators (Returns True or False)

Operator	Description	Example
`==`	Equal to	`5 == 5 → True`
`!=`	Not equal to	`5 != 3 → True`
`>`	Greater than	`6 > 2 → True`
`<`	Less than	`2 < 5 → True`
`>=`	Greater than or equal	`5 >= 5 → True`
`<=`	Less than or equal	`4 <= 6 → True`

3. Assignment Operators (Used to assign values to variables)

Operator	Description	Example
`=`	Assign	`x = 5`
`+=`	Add and assign	`x += 3 → x = x + 3`
`-=`	Subtract and assign	`x -= 2 → x = x - 2`
`*=`	Multiply and assign	`x *= 4`
`/=`	Divide and assign	`x /= 2`
`//=`	Floor divide and assign	`x //= 3`

Operator	Description	Example
`%=`	Modulus and assign	`x %= 2`
`**=`	Exponent and assign	`x **= 2`

4. Logical Operators (Used to combine conditional statements)

Operator	Description	Example
`and`	Returns `True` if both are true	`x > 5 and x < 10`
`or`	Returns `True` if at least one is true	`x < 3 or x > 7`
`not`	Reverses the result	`not(x > 5)`