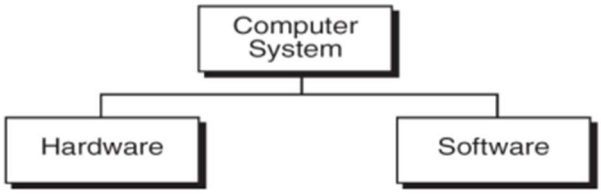
(Chapter- 01) Computer Systems

A computer system consists of hardware and software.



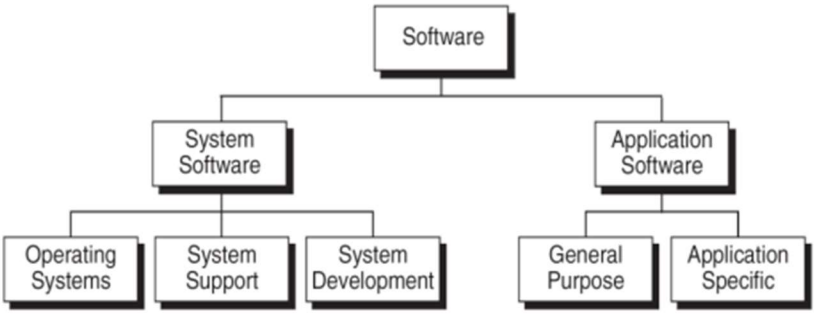
1. Hardware: Hardware refers to the physical components of a computer or any other electronic device that you can touch and see.

Examples:

- **Input Devices:** Keyboard, mouse, touchscreen, microphone (used to give commands to the computer).
- **Output Devices:** Monitor, printer, speakers (used to display or output the results of the computer's processing).
- **Processing Unit:** Central Processing Unit (CPU), often called the "brain" of the computer, processes data and instructions.
- **Storage Devices:** Hard drive, SSD (used to store data and programs permanently).
- **Memory:** RAM (Random Access Memory), temporarily stores data and instructions that the CPU needs while working.

2. Software: Software is a collection of instructions or programs that tell the hardware what to do. It is the non-physical part of a computer.

Or, It is consist of computer programs that instruct the execution of a computers



- **Types of Software:**
  - **System Software:** The most important software that manages hardware and software resources, examples include Windows, macOS, and Linux.
  - **Application Software:** Programs that help users do specific tasks like word processing (Microsoft Word), browsing the internet (Google Chrome), or playing games.

- **Examples of Software:**
    - **Mobile Applications:** WhatsApp, Instagram, Google Maps
    - **Web Applications:** Facebook (www.facebook.com), Amazon (www.amazon.com), YouTube (www.youtube.com)
    - **Gaming Software:** Free fire, GTA, Call of Duty
    - **Operating Systems:** Windows, macOS, Linux, Android, iOS

(Chapter- 02) Introduction of Programing

Computer Programs: A computer program is a set of instructions that execute the computers.

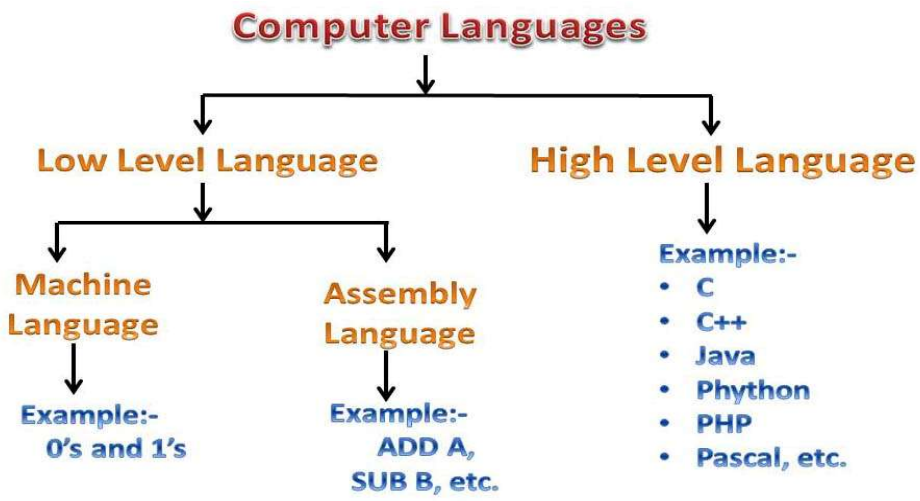
1. **Why is Programming Important?**

   Programming allows us to create software, apps, websites, and games. It's how we make computers do things, from simple tasks like adding numbers to complex ones like controlling a robot.

2. **How Does Programming Work?**
   1. **Writing Code:** Programmers write instructions, called code, in a programming language.
   2. **Executing Code:** The computer follows these instructions step by step to complete the task.
   3. **Debugging:** If something goes wrong, programmers find and fix the errors in the code, a process called debugging.

Computer Languages:

To communicate with a computer, users need a language that the computer can understand. Different languages have been developed to perform various tasks on a computer. These languages are mainly divided into two categories based on how they are interpreted.



Low-Level Languages

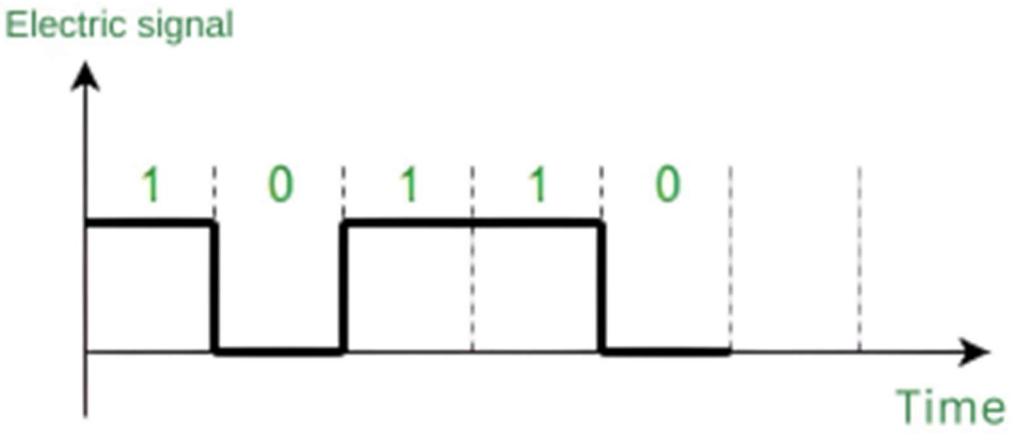
- Low-level languages are closer to the machine's hardware and offer less abstraction. They provide more control over hardware but are harder to read and write.

High-Level Languages

- High-level computer languages are designed to be easy to understand, using English-like words and symbols.
- Each instruction written in a high-level language is need to convert into machine language, which the computer can understand and execute.

Machine Language: Machine language is the simplest and most basic computer language. Computers can only understand this language, which uses 0s and 1s (binary code).

The number "0" means **no electric signal**, and "1" means there is **an electric signal**. Computers recognize these signals and understand machine language.



Computer Programming Language:

- A specific type of computer language designed to create software by providing instructions that a computer can execute.
- These languages are used to write algorithms and logic to perform specific tasks.

Side-by-side comparison between Machine Language and Computer Programing language

Aspect	Machine Language	Computer Programming Language
Definition	The lowest-level programming language in binary code	A higher-level language used to write software programs
Human Readability	Not human-readable (binary code)	Human-readable (using words and symbols)
Execution	Executed directly by the computer's hardware	Needs to be translated (compiled/interpreted) into machine language
Speed	Fastest execution, no translation needed	Slightly slower due to the translation process
Example	10110000 01100001	printf("Hello, World!"); in c- language
Portability	Not portable, specific to hardware/CPU	Portable, can run on different hardware with minimal changes

Compiler / Interpreter:

Both compilers and interpreters are tools used to convert programs written in high-level languages into machine code that a computer can understand and execute.

Aspect	Compiler	Interpreter
How It Works	Translates the entire program into machine code at once.	Translates and runs the program line by line.
Execution Time	Takes time to translate before running, but the program runs faster once translated.	Runs the program as it translates, which can be slower.
Error Handling	Shows all errors after translating the whole program.	Shows errors one at a time, as it runs through the program.
Example Languages	C, C++	Python, JavaScript
Use Case	Good for creating software that needs to run quickly.	Good for testing and debugging code easily.
Development Process	You need to recompile every time you make changes.	You can test parts of the code without recompiling.

File Extension

A file extension is a suffix at the end of a file name that indicates the type of files and folders.

It's usually is in small letters or numbers after a dot (.)

Examples:

- .txt – Text File (plain text)
- .docx – Microsoft Word Document
- .jpg or .png – Image Files
- .mp3 – Audio File
- .mp4 – Video File
- .pdf – Portable Document Format
- .html – HyperText Markup Language (web page)
- **.py – for Python Programing language**
- **.exe – for Software in windows**

(Chapter- 03) Introduction to Python

1. Python

Python is a **high-level**, **interpreted** programming language known for its simplicity and readability. Created by Guido van Rossum and first released in 1991, it emphasizes code readability using significant indentation. Python supports **multiple programming paradigms**, including procedural, object-oriented, and functional programming.

Key features of Python:

- **Easy Syntax:** Python's syntax is designed to be easy to read and write, making it accessible for beginners.
- **Interpreted Language:** Python code is executed line by line by the Python interpreter, without the need for compilation.
- **Extensive Libraries**: Python has a vast standard library and supports many third-party libraries for various fields, including data science, web development, automation, machine learning, and more.
- **Multi-Paradigm programming language:** Procedural, Object-Oriented, Functional and etc

2. Why we Learn Python Programing Language?

1. **Easy to Learn and Use**
2. **Versatility, Flexibility and Wide Range of Applications**
    - Web Development,
    - Data Science and Machine Learning,
    - Automation,
    - Game Development,
    - IoT and Robotics
3. **Huge Community Support & Open Source**
4. **High Demand in the Job Market (**Go to the Google Trends and compare different language for 5 or 10 years**)**
5. **Extensive Libraries and Frameworks**
6. **Strong Presence in AI, Machine Learning, and Data Science**

3. Installing Python and setting up the environment.

- **Download and Install Python:** https://www.python.org/downloads/
- **To check if Python was installed correctly, Type at CMD (Command Prompt) " python --version "**
- **Install any Python IDEs (Integrated Development Environment): IDLE, PyCharm, VSCode**

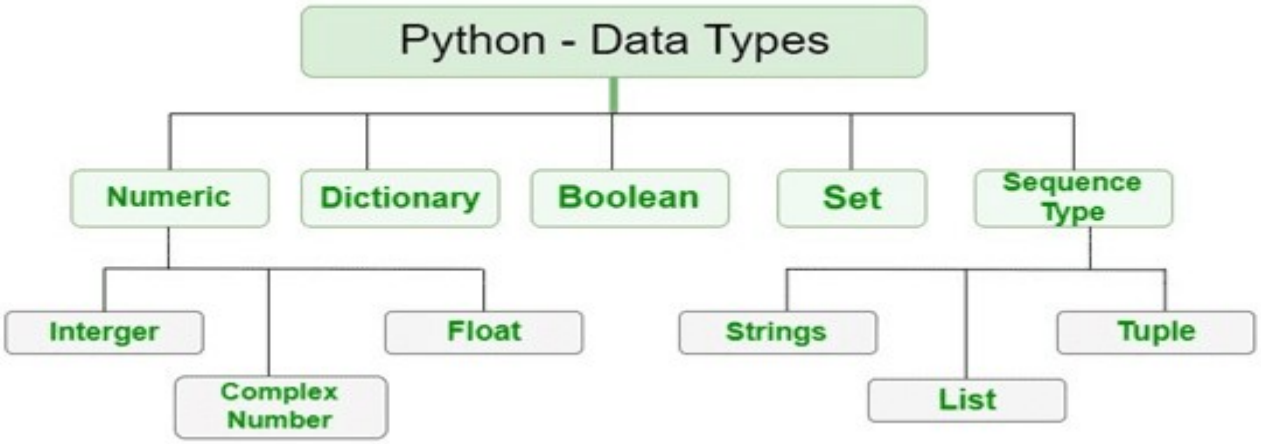
(Chapter- 04) Python Basics

Variable :- A variable is a name given to a memory location in Programs that holds a value. Variables allow you to store and manipulate data in a program.

age = 15
temperature = 36.5
grade = 'A'

What is Data types?

- Data types define the type of data that a variable can hold.
- They help the compiler to understand, how much memory to allocate for the variable and how to interpret the data.



Basic Data Types:

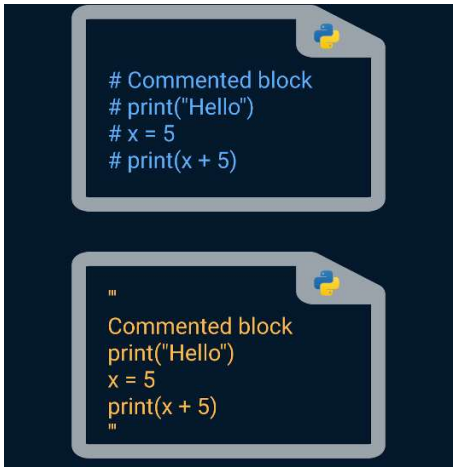
Name	Type	Description	Example
Integers	int	Whole numbers	3, 300, 200
Floating point	float	Numbers with a decimal point	2.3, 4.6, 100.0
Strings	str	Ordered sequence of characters	"hello", 'Sammy', "2000", "LL"
Lists	list	Ordered sequence of object	[10, "hello", 200.3]
Dictionaries	dict	Unordered Key: Value pairs	{"mykey": "value", "name" "Frankie"}
Tuples	tup	Ordered immutable sequence of objects	(10, "hello", 200.3)
Sets	set	Unordered collection of unique objects	{"a", "b"}

Comments:

In Python, a comment is a line of text that is ignored by the interpreter and is used to explain the code, make it more readable, or leave notes for yourself or other developers. Comments do not affect the execution of the program.

Single-line comment

''' Multi-line comment '''



Input and output functions:

- **Input Function: input():** It is used to take input from the user.

```python
# Taking input from the user
name = input("Enter your name: ")
age = input("Enter your age: ")

print("Hello, " + name + "! You are " + age + " years old.")
```

- **Print Function: print() :** It is use to display outputs data to the console.

```python
# Printing multiple values
print("Hello", "World!")  # Output: Hello World!

# Custom separator and ending
print("Hello", "World", sep=", ", end="!\n")  # Output: Hello, World!
```

Operators: It is special symbols that perform operations on variables and values.

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

1. Arithmetic Operators: These operators are used to perform basic mathematical operations.

Operator	Description	Example
+	Addition	`5 + 3` results in `8`
-	Subtraction	`5 - 3` results in `2`
*	Multiplication	`5 * 3` results in `15`
/	Division	`5 / 2` results in `2.5`
//	Floor Division	`5 // 2` results in `2`
%	Modulus (Remainder)	`5 % 2` results in `1`
**	Exponentiation	`5 ** 2` results in `25`

2. Comparison Operators: These operators compare two values and return a Boolean result (True or False).

Operator	Description	Example
==	Equal to	5 == 3 results in False
!=	Not equal to	5 != 3 results in True
>	Greater than	5 > 3 results in True
<	Less than	5 < 3 results in False
>=	Greater than or equal to	5 >= 5 results in True
<=	Less than or equal to	5 <= 5 results in True

3. Logical Operators : These operators are used to combine conditional statements.

Operator	Description	Example
and	Returns True if both statements are true	True and False results in False
or	Returns True if at least one statement is true	True or False results in True
not	Returns True if the statement is false	not True results in False

4. Assignment Operators: These operators are used to assign values to variables.

Operator	Description	Example
=	Assigns value	x = 5
+=	Adds and assigns	x += 2 (equivalent to x = x + 2)
-=	Subtracts and assigns	x -= 2 (equivalent to x = x - 2)
***=**	Multiplies and assigns	x *= 2 (equivalent to x = x * 2)
/=	Divides and assigns	x /= 2 (equivalent to x = x / 2)
//=	Floor divides and assigns	x //= 2 (equivalent to x = x // 2)
%=	Modulus and assigns	x %= 2 (equivalent to x = x % 2)
****=**	Exponentiates and assigns	x **= 2 (equivalent to x = x ** 2)

5. Bitwise Operators: These operators are used to perform operations on binary numbers.

Operator	Description	Example
&	Bitwise AND	5 & 3 results in 1
`	`	Bitwise OR
^	Bitwise XOR	5 ^ 3 results in 6
~	Bitwise NOT	~5 results in -6
<<	Left Shift	5 << 1 results in 10
>>	Right Shift	5 >> 1 results in 2

6. Membership Operators: These operators test for membership in a sequence (like lists, strings, or tuples).

Operator	Description	Example
in	Returns True if the value is found in the sequence	3 in [1, 2, 3] results in True
not in	Returns True if the value is not found in the sequence	4 not in [1, 2, 3] results in True

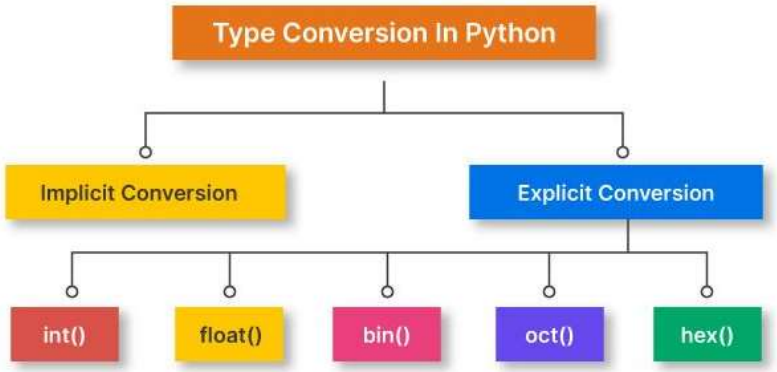
7. Identity Operators: These operators check if two variables point to the same object in memory.

Operator	Description	Example
is	Returns True if both variables point to the same object	x is y
is not	Returns True if both variables do not point to the same object	x is not y

Type conversion or Type Casting :

The conversion of one data type into the other data type is known as type casting in python or type conversion in python.

Python supports a wide variety of functions or methods like: int(), float(), str(), ord(), hex(), oct(), tuple(), set(), list(), dict(), etc. for the type casting in python.



Explicit typecasting:

The conversion of one data type into another data type, done via developer or programmer's intervention or manually as per the requirement, is known as explicit type conversion.

It can be achieved with the help of Python's built-in type conversion functions such as int(), float(), hex(), oct(), str(), etc .

Example of explicit typecasting:

```
string = "15"
string_number = int(string)          #throws an error if the string is not a valid
sum= 7 + string_number
print("The Sum of both the numbers is: ", sum)
```

Output:

The Sum of both the numbers is 22

Implicit type casting: According to the level, one data type is converted into other by the Python interpreter itself (automatically). This is called, implicit typecasting in python.

Python converts a smaller data type to a higher data type to prevent data loss.

Example of implicit type casting:

```
# Python automatically converts
# a to int
a = 7
print(type(a))

# Python automatically converts b to float
b = 3.0
print(type(b))

# Python automatically converts c to float as it is a float addition
c = a + b
print(c)
print(type(c))
```

Output:

```
<class 'int'>
<class 'float'>
10.0
<class 'float'>
```