Python Programming Notes

**1. What is Programming?**

Programming is the process of designing and building an executable computer program to accomplish a specific task or solve a particular problem.

It involves writing instructions (code) in a programming language that a computer can understand and execute.

**Key Points:**

* It is a way to communicate with computers.
* Involves problem-solving, logical thinking, and creativity.
* Used to automate tasks, analyze data, develop software, and more.
* Follows a structured approach (algorithms) to achieve desired results.

**2. What is a Program?**

A **program** is a set of instructions written in a programming language that tells a computer what operations to perform. Programs can range from simple (e.g., printing "Hello, World!") to complex (e.g., operating systems, games, AI models).

**Key Points:**

* A program is written in a programming language (Python, Java, C++, etc.).
* It follows a sequence of steps to perform a task.
* Programs are executed by a computer’s CPU (Central Processing Unit).
* Examples: Web browsers, video games, mobile apps, calculators.

**3. Input / Processing / Output (IPO Model)**

The **IPO model** describes the basic structure of a computer program:

1. **Input** – Data provided to the program (e.g., user input, files, sensors).
2. **Processing** – The program manipulates the input (calculations, logic, algorithms).
3. **Output** – The result produced after processing (e.g., display, file, sound).

**Example (Simple Calculator Program):**

* **Input:** Two numbers (5 and 3) and an operation (add).
* **Processing:** Adds the numbers (5 + 3 = 8).
* **Output:** Displays the result (8).

**Key Points:**

* All programs follow this fundamental structure.
* Input can come from users, files, databases, or APIs.
* Processing involves computations, decision-making, and loops.
* Output can be displayed, stored, or sent to another system.

**4. What is a Programming Language?**

A **programming language** is a formal language used to write instructions that a computer can execute. It provides syntax (rules) and semantics (meaning) to write programs.

**Types of Programming Languages:**

1. **Low-Level Languages** (Close to machine code):

These languages are **very close to the hardware**, meaning they directly communicate with the computer's CPU. They're hard for humans to read and write but **very efficient for the computer**.

* + **Machine Language** (Binary: 01010101)
  + **Assembly Language** (Uses mnemonics: MOV, ADD)

1. **High-Level Languages** (Easier for humans):

These are programming languages that are **closer to human language** and **easier to read, write, and understand** than low-level languages. They are designed to be **portable**, meaning they can run on different types of hardware with little or no modification.

* + **Python, Java, C++, JavaScript, Ruby**
  + More readable, requires a compiler/interpreter.

**Key Points:**

* Programming languages act as a bridge between humans and computers.
* High-level languages are easier to write and maintain.
* Each language has strengths (Python for AI, JavaScript for web).

**5. Difference Between Interpreter and Compiler**

Both **interpreters** and **compilers** convert high-level code into machine code, but they work differently:

| **Feature** | **Interpreter** | **Compiler** |
| --- | --- | --- |
| **Execution** | Line-by-line execution. | Entire program is compiled at once. |
| **Speed** | Slower (translates each time). | Faster (executes compiled code directly). |
| **Error Handling** | Stops at first error. | Shows all errors after compilation. |
| **Portability** | Needs interpreter installed. | Compiled code runs independently. |
| **Examples** | Python, JavaScript, Ruby. | C, C++, Java (partially compiled). |

**Key Points:**

* **Interpreted languages** are flexible (good for scripting).
* **Compiled languages** are efficient (good for performance-critical apps).
* Some languages (Java) use both (compiled to bytecode, then interpreted by JVM).

The **Python** programming language was named **after the British comedy group "Monty Python"**, not the snake.

When **Guido van Rossum**, the creator of Python, started developing the language in the late 1980s, he wanted a name that was **short, unique, and a bit mysterious**. He was a fan of the BBC comedy series *"Monty Python’s Flying Circus"*, so he decided to call the language **Python** as a tribute.

So, even though the logo sometimes includes snakes, the name itself comes from **comedy, not reptiles**.

# What is Python

* Python is a dynamically typed, general-purpose programming language that supports an object-oriented programming approach as well as a functional programming approach.
* Python is also an interpreted and high-level programming language.
* It was created by Guido Van Rossum in 1989.

Statically Typed

Define variable and then store info

Var number int;

Number = 1

C, C++

Dynamically Typed:

Variables are determined in runtime

Name = “Kabir”

Number = 1

Python

**Note:** Python is a case sensitive language.

# Comments in Python:

**Comments** are used to explain code and make it more readable. Python ignores comments during execution.

Example:

# This is a single-line comment

x = 5 # This is an inline comment

'''

This is a multi-line comment

using triple single quotes

'''

print("Hello")

# Variables:

Variable is like a container that holds data. Very similar to how our containers in kitchen holds sugar, salt etc Creating a variable is like creating a placeholder in memory and assigning it some value.

In Python its as easy as writing:

a = 1

b = True

c = "Harry"

d = None

## Variable Naming Conventions in Python

* 1. **Use lowercase letters** with words separated by underscores (snake\_case).
  2. **Must start with a letter** or underscore (\_), and **not a number**.
  3. **Avoid using Python keywords** (like if, for, class, def).
  4. **Be descriptive** — use meaningful names that describe the value or purpose.
  5. **Case-sensitive** — Name, name, and NAME are all different variables.

## Print Statement With Variable and Input Function:

In Python, the input() function is used to take input from the user, and it always returns the input as a string. The print() function is used to display text, messages, or variable values on the screen. By combining these two functions, you can ask the user for information and then display a customized message using their input.

# Python Libraries

A library is a collection of modules that provide related functionality and tools.

math is a standard library containing math-related modules and functions. We should initially install and library and then import before we can use it. Some libraries are built-in in python and other are external libraries that needs to be installed.

Example

import math

print(math.sqrt(25))

# What is a Data Type?

Data type specifies the type of value a variable hold. This is required in programming to do various operations without causing an error. In python, we can print the type of any operator using type function:

a = 1

print(type(a))

b = "1"

print(type(b))

By default, python provides the following built-in data types:

## 1. Numeric data: int, float

* int: 3, -8, 0
* float: 7.349, -9.0, 0.0000001

## 2. Text data: str

str: "Hello World!!!", "Python Programming"

## 3. Boolean data:

Boolean data consists of values True or False.

## 4. Sequenced data: list, tuple

### list:

A list is an ordered collection of data with elements separated by a comma and enclosed within square brackets. Lists are mutable and can be modified after creation.

**Example:**

list1 = [8, 2.3, [-4, 5], ["apple", "banana"]]

print(list1)

Output:

[8, 2.3, [-4, 5], ['apple', 'banana']]

### Tuple:

A tuple is an ordered collection of data with elements separated by a comma and enclosed within parentheses. Tuples are immutable and can not be modified after creation.

**Example:**

tuple1 = (("parrot", "sparrow"), ("Lion", "Tiger"))

print(tuple1)

Output:

(('parrot', 'sparrow'), ('Lion', 'Tiger'))

## 5. Mapped data: dict

**dict:** A dictionary is an unordered collection of data containing a key:value pair. The key:value pairs are enclosed within curly brackets.

**Example:**

dict1 = {"name":"Sakshi", "age":20, "canVote":True}

print(dict1)

Output:

{'name': 'Sakshi', 'age': 20, 'canVote': True}

**Operators in Python – Class Notes**

**📘 Definition:**

Operators are special symbols or keywords in Python that perform operations on variables and values. Python provides several types of operators to perform different kinds of tasks, such as arithmetic, comparison, logical, assignment, etc.

To build something like a calculator, we primarily use **arithmetic operators**.

**🔢 Arithmetic Operators in Python**

These operators are used to perform basic mathematical operations.

| **Operator** | **Name** | **Example** | **Explanation** |
| --- | --- | --- | --- |
| + | Addition | 15 + 7 | Adds two numbers → 22 |
| - | Subtraction | 15 - 7 | Subtracts right from left → 8 |
| \* | Multiplication | 5 \* 7 | Multiplies two numbers → 35 |
| \*\* | Exponentiation | 5 \*\* 3 | 5 raised to power 3 → 125 |
| / | Division | 5 / 3 | Divides and gives float → 1.666… |
| % | Modulus | 15 % 7 | Remainder of division → 1 |
| // | Floor Division | 15 // 7 | Division without remainder → 2 |

**💡 Extra Tips:**

* Use \*\* for powers like squares, cubes, etc.
* / always returns a **float**, even if the result is whole.
* // gives you the **whole number** part only.
* % is useful to check if a number is **even or odd**:  
  Example: 5 % 2 → 1 (odd), 4 % 2 → 0 (even)

# Typecasting in python

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(), tuple(), list(), dict(), etc. for the type casting in python.

Two Types of Typecasting:

1. Explicit Conversion (Explicit type casting in python)
2. Implicit Conversion (Implicit 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"

number = 7

string\_number = int(string) #throws an error if the string is not a valid integer

sum= number + string\_number

print("The Sum of both the numbers is: ", sum)

Output:

The Sum of both the numbers is 22

## Implicit type casting:

Python **automatically** converts one data type to another **during an operation**, if needed. Data types in Python do not have the same level i.e. ordering of data types is not the same in Python. Some of the data types have higher-order, and some have lower order. While performing any operations on variables with different data types in Python, one of the variable's data types will be changed to the higher data type. 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))

Ouput:

<class 'int'>

<class 'float'>

10.0

<class 'float'>