Introduction To Python

Posted in **10 Computer Science**

These lecture notes are prepared by **Salman Ahmad**, who is both a **teacher** and a **web developer**. The purpose of this material is to make the topic clear and useful for students of Class 10 Computer Science.

Table of Contents

What is Python Programming?

Python is a high-level, easy-to-read programming language used to create software, websites, mobile applications, games, artificial intelligence (AI), data analysis applications, and many other types of programs.

Why is Python Called Versatile?

Python is called **versatile** because it can be used for many different types of work. Instead of learning a different programming language for every task, you can use Python for a wide variety of applications.

- Web Development
- Artificial Intelligence (AI)
- Machine Learning
- Data Science
- Data Analysis
- Automation
- Game Development
- Desktop Applications
- Mobile App Development
- Cybersecurity
- Web Scraping
- Cloud Computing
- Internet of Things (IoT)
- Robotics
- Scientific Computing
- Software Development
- Networking
- Blockchain

- Image Processing
- Computer Vision

What is Computer Programming?

Computer programming is the process of writing instructions (called **code**) that tell a computer what to do and how to do it.
A computer cannot think or make decisions on its own. It only follows the instructions given by a programmer.
It is also called **coding**, **programming**, or **software** in everyday language.

Setting Up Python Development Environment

Setting up a Python development environment means installing and configuring the tools needed to write and run Python programs.

Examples of Python Development Environments

These tools are commonly called Integrated Development Environments (IDEs).

- Visual Studio Code (VS Code)
- PyCharm
- Google Colab (Colab)
- Jupyter Notebook
- IDLE
- Spyder

What is Google Colab?

Google Colab is a cloud-based Python development environment. It lets you write and run Python code directly in your web browser, so you do not need to install Python on your computer.

Python Comments

What are Comments?

Comments are notes written inside a Python program to explain the code. They are ignored by the Python interpreter, so they are not executed.

Purpose of Comments

Comments are used to:

- Explain what the code does.

- Make the code easier to read and understand.
- Help other programmers understand the program.
- Remind yourself why you wrote a particular piece of code.

Types of Comments

Python mainly has two types of comments:

1. Single-Line Comment

A single-line comment starts with the `#` symbol.

Syntax

```
# This is a single-line comment
```

Example

```
# Print a welcome message
print("Welcome")
```

2. Multi-Line Comment

Python does not have a special multi-line comment symbol. A common way is to use triple quotes (`'''` or `"""`) to write comments that span multiple lines.

Example

```
"""
This program
prints a welcome
message.
"""

print("Welcome")
```

Variables in Python

What is a Variable?

A variable is a named location in memory that is used to store data. The value stored in a variable can be used or changed during the execution of a program.

Python Variables Example

```
x = 55
y = 66.6
name = "Salman"
age = 28
is_student = True

print(x)
print(y)
print(name)
print(age)
print(is_student)
```

Output:

```
55
66.6
Salman
28
True
```

Rules for Naming Variables

Python has some rules for naming variables.

1. Variable names must start with a letter or an underscore (_).

✅ **Correct**

```
name = "Ali"
_age = 20
```

❌ **Incorrect**

```
2name = "Ali"
```

2. Variable names can contain letters, numbers, and underscores.

✅ **Correct**

```
student1 = "Ahmed"
total_marks = 450
```

❌ **Incorrect**

```
student-name = "Ahmed"
```

(- is not allowed.)

3. Variable names cannot contain spaces.

✅ **Correct**

```
first_name = "Ali"
```

❌ **Incorrect**

```
first name = "Ali"
```

4. Variable names are case-sensitive.

This means uppercase and lowercase letters are treated as different names.

```
name = "Ali"
Name = "Ahmed"
```

```
print(name)
print(Name)
```

Output:

```
Ali
Ahmed
```

name and **Name** are two different variables.

5. Do not use Python keywords as variable names.

❌ **Incorrect**

```
if = 10
class = 5
```

These words are reserved by Python.

> **Note:** Reserved words (keywords) are predefined words in Python that have a special meaning. They cannot be used as variable names, function names, or class names.

Types of Variables in Python

In Python, variables are commonly classified based on the type of value they store.

1. Integer (int)

An integer stores whole numbers (without decimal points).

Example

```
age = 20
marks = 95
```

Explanation

20 and **95** are integers because they are whole numbers.

2. Float (float)

A float stores decimal numbers.

Example

```
height = 5.8
price = 99.99
```

Explanation

5.8 and **99.99** are floats because they contain decimal points.

3. String (str)

A string stores text. Text is written inside single (`' '`) or double (`" "`) quotes.

Example

```
name = "Ali"
city = 'Lahore'
```

Explanation

"Ali" and **'Lahore'** are strings because they contain text.

4. Boolean (bool)

A boolean stores only two values:

- **True**
- **False**

Example

```
is_student = True
is_logged_in = False
```

Explanation

- **True** means **Yes** or **Correct**.

- **False** means **No** or **Incorrect**.

Real-Life Example

Imagine a student registration form.

```
name = "Ali"
age = 20
height = 5.8
is_student = True
```

Here:

- **name** → String
- **age** → Integer
- **height** → Float
- **is_student** → Boolean

✅ **Correct**

```
marks = 90
student = "Ali"
```

> **Note:** You can skip the explanation. It is just added for better understanding.

1. Input Operation

An input operation allows the user to enter data into a program.

In Python, the `input()` function is used to take input from the user.

Input an Integer (int)

Use `int(input())` to take a whole number as input.

```
age = int(input("Enter your age: "))
```

```
print(age)
```

Sample Output:

```
Enter your age: 20
20
```

Input a Float (float)

Use `float(input())` to take a decimal number as input.

```
height = float(input("Enter your height: "))

print(height)
```

Sample Output:

```
Enter your height: 5.8
5.8
```

Input a String (str)

Use `str(input())` to take text as input.

```
name = str(input("Enter your name: "))

print(name)
```

Sample Output:

```
Enter your name: Ali
Ali
```

2. Output Operation

An output operation displays information to the user.

In Python, the `print()` function is used to display output.

Syntax

```
print(value)
```

Example

```
print("Hello, World!")
```

Output:

```
Hello, World!
```

Operators and Expressions in Python

What are Operators?

Operators are special symbols that perform operations on values or variables.

Example

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

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

Output:

```
15
```

Here:

- `+` is the operator.
- It adds the values of `a` and `b`.

Types of Operators

1. Arithmetic Operators

Arithmetic operators are used to perform mathematical calculations.

Operator	Meaning	Example
`+`	Addition	10 + 5 = 15
`-`	Subtraction	10 - 5 = 5
`*`	Multiplication	10 * 5 = 50
`/`	Division	10 / 5 = 2.0
`%`	Modulus (Remainder)	10 % 3 = 1
`**`	Exponent (Power)	2 ** 3 = 8
`//`	Floor Division	10 // 3 = 3

Python Arithmetic Operators Example

```
a = 10
b = 5

print("Addition:", a + b)
print("Subtraction:", a - b)
print("Multiplication:", a * b)
print("Division:", a / b)
print("Modulus:", a % 3)
```

```
print("Exponent:", 2 ** 3)
print("Floor Division:", 10 // 3)
```

Output:

```
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2.0
Modulus: 1
Exponent: 8
Floor Division: 3
```

2. Comparison (Relational) Operators

These operators compare two values and return **True** or **False**.

Operator	Meaning	Example
==	Equal to	5 == 5 → True
!=	Not equal to	5 != 3 → True
>	Greater than	10 > 5 → True
<	Less than	5 < 10 → True
>=	Greater than or equal to	10 >= 10 → True
<=	Less than or equal to	5 <= 10 → True

Python Comparison (Relational) Operators Example

```
a = 10
b = 5

print("Equal to:", a == b)
print("Not equal to:", a != b)
print("Greater than:", a > b)
print("Less than:", a < b)
print("Greater than or equal to:", a >= b)
print("Less than or equal to:", a <= b)
```

Output:

```
Equal to: False
Not equal to: True
Greater than: True
Less than: False
Greater than or equal to: True
Less than or equal to: False
```

3. Assignment Operators

These operators assign values to variables.

Operator	Meaning	Example
`=`	Assign	x = 10
`+=`	Add and assign	x += 5
`-=`	Subtract and assign	x -= 5
`*=`	Multiply and assign	x *= 2
`/=`	Divide and assign	x /= 2
`%=`	Modulus and assign	x %= 3
`//=`	Floor divide and assign	x //= 2
`**=`	Exponent and assign	x **= 2

Python Assignment Operators Example

```
x = 10
x += 5
print(x)

x = 10
x -= 5
print(x)

x = 10
x *= 2
print(x)
```

```
x = 10
x /= 2
print(x)

x = 10
x %= 3
print(x)

x = 10
x //= 3
print(x)

x = 5
x **= 2
print(x)
```

Output:

```
15
5
20
5.0
1
3
25
```

4. Logical Operators

Logical operators combine two or more conditions into a single expression and return **True** or **False**.

Operator	Meaning
`and`	True if both conditions are True.
`or`	True if at least one condition is True.
`not`	Reverses the result.

Python Logical Operators Example

```
x = 10
y = 5

print(x > 5 and y < 10)
print(x > 20 or y < 10)
print(not(x > 5))
```

Output:

```
True
True
False
```

Operator Precedence in Python

What is Operator Precedence?

Operator precedence is the order in which Python evaluates operators in an expression.

Simple Definition:

Operator precedence is the set of rules that determines which operation is performed first in an expression.

Operator Precedence Table

Precedence	Operator	Description
1 (Highest)	`()`	Parentheses
2	`**`	Exponent
3	`*`, `/`, `//`, `%`	Multiplication, Division, Floor Division, Modulus
4	`+`, `-`	Addition, Subtraction
5	`==`, `!=`, `>`, `<`, `>=`, `<=`	Comparison Operators
6	`not`	Logical NOT
7	`and`	Logical AND
8 (Lowest)	`or`	Logical OR