



PYTHON TOPICS

Introduction to Python

- What is Python?
- Features of Python
- Installing Python
- Python IDLE / VS Code

Python Basics

- Syntax
- Comments
- Variables
- Data Types
- Input and Output
- Type Conversion

Operators

- Arithmetic Operators
- Comparison Operators
- Logical Operators
- Assignment Operators
- Membership Operators

Conditional Statements

- if
- if else
- if elif else
- Nested if

Loops

- for loop
- while loop
- break
- continue
- pass

Strings

- String creation
- Indexing
- Slicing
- String methods
- Formatting

Lists, Tuples & Sets

- List
- Tuple
- Set
- Basic methods

Difference between them

Dictionary

- Creating dictionaries
- Keys and values
- Adding / deleting data
- Dictionary methods

Functions

- Creating functions
- Parameters
- Arguments
- Return value
- Lambda functions

Modules & Packages

- Import
- Built-in modules
- Creating your own module

File Handling

- Open file
- Read file
- Write file
- Append file
- Close file

Exception Handling

- try
- except
- else
- finally

Object-Oriented Programming

- Class
- Object
- Constructor
- Inheritance
- Encapsulation
- Polymorphism

Database Connectivity

- SQL basics
- SQLite / MySQL
- Insert
- Update
- Delete



Select	To-Do List
Python Projects	Quiz Application
Calculator	Billing System
Student Management System	Simple Login System

If you want, I can also give you **Python full syllabus with simple English + Tamil meaning + code + output**, chapter by chapter.

Introduction to Python o What is Python? o Features of Python o Installing Python o Python IDLE / VS Code

INTRODUCTION TO PYTHON

WHAT IS PYTHON?

Python is a **simple and easy-to-learn programming language**. It is used to create software, websites, applications, automation, and more.

FEATURES OF PYTHON

- Easy to learn and use
- Simple syntax
- Free and open source
- Works on different operating systems
- Supports object-oriented programming
- Large number of libraries
- Used in AI, Data Science, Web Development, etc.

INSTALLING PYTHON

- Download Python from the official Python website.
- Run the installation file.
- Select **Add Python to PATH**.
- Click **Install Now**.
- Check the installation using Python IDLE or Command Prompt.

PYTHON IDLE / VS CODE

PYTHON IDLE:

A simple software that comes with Python. It is used to write and run Python programs.

VS CODE:

Visual Studio Code is a popular code editor. It provides many useful tools and extensions for writing Python programs.

more details

Sure 👍 Here are **more detailed and simple notes** for **Introduction to Python**:

INTRODUCTION TO PYTHON

WHAT IS PYTHON?

Python is a **high-level, easy-to-learn programming language**. It was created by **Guido van Rossum** and first released in **1991**.

Python is popular because its syntax is simple and easy to understand. We can use Python to give instructions to a computer and create different types of programs.

SIMPLE EXAMPLE

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

Output:

Hello, World!



The print() function is used to display information on the screen.

USES OF PYTHON

PYTHON IS USED IN MANY AREAS:

-  **Web Development** – Creating websites and web applications
 -  **Software Development** – Creating computer applications
 -  **Artificial Intelligence (AI)** – Creating intelligent systems
 -  **Data Science** – Analyzing and processing data
 -  **Machine Learning** – Teaching computers to learn from data
 -  **Cybersecurity** – Security testing and automation
 -  **Automation** – Automating repetitive tasks
 -  **Game Development** – Creating simple games
-

FEATURES OF PYTHON

EASY TO LEARN

Python has a simple and readable syntax. Beginners can understand Python programs easily.

Example:

```
name = "Priya"
print(name)
```

Easy to Read

Python code looks similar to simple English. This makes programs easier to read and understand.

```
age = 20
```

```
if age >= 18:
```

```
    print("Eligible")
```

Free and Open Source

Python is **free to download and use**.

Open source means that its source code is publicly available and can be improved by developers.

Interpreted Language

Python executes code **line by line** using an interpreter.

This helps programmers find and correct errors easily.

Cross-Platform

Python programs can run on different operating systems, such as:

Windows

Linux

macOS

The same Python program can usually be used on different computers with little or no change.



Large Library Support

Python has many built-in and external libraries.

Libraries provide ready-made functions and tools that help programmers perform tasks easily.

Examples:

math – Mathematical operations

random – Random numbers

datetime – Date and time operations

Object-Oriented Programming Support

Python supports **Object-Oriented Programming (OOP)**.

OOP helps programmers organize programs using:

Classes

Objects

Widely Used

Python is used by students, beginners, developers, companies, and researchers around the world.

Installing Python

Python must be installed on your computer before you can write and run Python programs.

Steps to Install Python

Step 1: Download the Python installation file from the official Python website.

Step 2: Open the downloaded installation file.

Step 3: Before installation, select:

Add Python to PATH

This allows Python to be used from the Command Prompt or Terminal.

Step 4: Click:

Install Now

Step 5: Wait for the installation to complete.

Step 6: Open Python IDLE or Command Prompt to check whether Python is installed correctly.

Testing Python Installation

Open Command Prompt and type:

python --version

The installed Python version will be displayed.

Example:

Python 3.x.x

Python IDLE

What is Python IDLE?

IDLE stands for **Integrated Development and Learning Environment**.

It is a simple program that is usually installed along with Python.

Python IDLE is useful for beginners because it allows us to:

Write Python programs



Run Python programs

Test commands

Learn Python easily

Using Python IDLE

When IDLE is opened, we can see the Python Shell.

Example:

```
>>> print("Welcome to Python")
```

Output:

Welcome to Python

The >>> symbol is called the **Python prompt**.

Python Script in IDLE

We can also create and save Python programs.

Steps:

Open **Python IDLE**

Click **File**

Select **New File**

Write the Python program

Save the file with .py extension

Click **Run**

Select **Run Module**

Example Program

```
name = "Student"  
print("Welcome", name)
```

Output:

Welcome Student



VS Code

What is VS Code?

VS Code stands for **Visual Studio Code**.

It is a popular and powerful **code editor** used for writing programs in many programming languages, including Python.

For Python programming, we usually install the **Python extension** in VS Code.

Features of VS Code

Easy code editing

Syntax highlighting

Error checking

Auto-completion

Debugging tools

Extensions support

Integrated Terminal

Steps to Use Python in VS Code

**Step 1**

Install **VS Code**.

Step 2

Open VS Code.

Step 3

Install the **Python Extension**.

Step 4

Create a new file.

Step 5

Save the file with the .py extension.

Example:

hello.py

Step 6

Write a Python program:

```
print("Hello Python")
```

Step 7

Run the program.

Output:

Hello Python

Python IDLE vs VS Code**Python IDLE**

Simple and beginner-friendly

Comes with Python installation

Basic features

Good for learning

Easy to use

VS Code

Advanced and professional

Installed separately

Many advanced features

Good for projects and development

Supports extensions

Simple Conclusion

Python is one of the easiest programming languages for beginners. It has simple syntax, many useful features, and is used in modern technologies such as **AI, Data Science, Web Development, and Automation**. Beginners can start learning Python using **Python IDLE**, while **VS Code** provides more advanced tools for larger projects

Python Basics**Syntax**

Python syntax means the rules used to write Python programs.

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

Comments

Comments are notes in a program. Python ignores comments.

```
# This is a comment
```

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

Variables

Variables are used to store data.



```
name = "Priya"
age = 20
```

Data Types

Common Python data types:

- str – Text → "Hello"
- int – Whole number → 10
- float – Decimal number → 10.5
- bool – True or False → True

Input and Output

Input: Get data from the user.

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

Output: Display data on the screen.

```
print(name)
```

Type Conversion

Type conversion means changing one data type into another.

```
age = "20"
new_age = int(age)
```

```
print(new_age)
```

Example:

- str() – Convert to text
- int() – Convert to integer
- float() – Convert to decimal

Operators in Python – Details with Coding

Operators are special symbols used to perform operations on values and variables.

Arithmetic Operators

Used for mathematical calculations.

Operator	Meaning	Example
+	Addition	a + b
-	Subtraction	a - b
*	Multiplication	a * b
/	Division	a / b
%	Modulus (remainder)	a % b
**	Power	a ** b
//	Floor Division	a // b

```
a = 10
b = 3
```

```
print("Addition:", a + b)
print("Subtraction:", a - b)
```



```
print("Multiplication:", a * b)
print("Division:", a / b)
print("Remainder:", a % b)
print("Power:", a ** b)
print("Floor Division:", a // b)
```

Output:

Addition: 13
Subtraction: 7
Multiplication: 30
Division: 3.3333333333333335
Remainder: 1
Power: 1000
Floor Division: 3

Comparison Operators

Used to compare two values. The result will be **True** or **False**.

Operator	Meaning	Example
==	Equal to	a == b
!=	Not equal to	a != b
>	Greater than	a > b
<	Less than	a < b
>=	Greater than or equal	a >= b
<=	Less than or equal	a <= b

```
a = 10
b = 20
```

```
print(a == b)
print(a != b)
print(a > b)
print(a < b)
print(a >= b)
print(a <= b)
```

Output:

False
True
False
True
False
True

Logical Operators



Used to combine two or more conditions.

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

a = 10

b = 20

print(a < 15 and b > 15)

print(a > 15 or b > 15)

print(not(a > b))

Output:

True

True

True

Simple Example:

age = 20

if age >= 18 and age <= 60:

print("Eligible")

Output:

Eligible

Assignment Operators

Used to assign or update values in variables.

Operator	Example	Meaning
=	a = 10	Assign
+=	a += 5	Add and assign
-=	a -= 5	Subtract and assign
*=	a *= 5	Multiply and assign
/=	a /= 5	Divide and assign
%=	a %= 5	Modulus and assign
**=	a **= 2	Power and assign
//=	a //= 2	Floor divide and assign

a = 10

a += 5

print(a)

a -= 3



```
print(a)
```

```
a *= 2
```

```
print(a)
```

Output:

```
15
```

```
12
```

```
24
```

Simple Tamil Meaning

Arithmetic Operators – கணித கணக்குகள் செய்ய பயன்படும்.

Comparison Operators – இரண்டு மதிப்புகளை ஒப்பிட பயன்படும்.

Logical Operators – இரண்டு அல்லது அதற்கு மேற்பட்ட conditions-ஐ இணைக்க பயன்படும்.

Assignment Operators – Variable-க்கு value assign அல்லது update செய்ய பயன்படும்

Conditional Statements

Conditional statements are used to **make decisions** in Python.

if Statement

Used when a condition is **True**.

```
age = 18
```

```
if age >= 18:  
print("You can vote")
```

Output:

```
You can vote
```

if else Statement

Used when there are **two choices**.

```
age = 16
```

```
if age >= 18:  
print("You can vote")  
else:  
print("You cannot vote")
```

Output:

```
You cannot vote
```

if elif else Statement

Used when there are **multiple conditions**.

```
mark = 75
```

```
if mark >= 90:  
print("Grade A")  
elif mark >= 60:
```



```
print("Grade B")
else:
print("Grade C")
```

Output:

Grade B

Nested if

An if statement **inside another if statement** is called Nested if.

```
age = 20
```

```
if age >= 18:
if age >= 21:
    print("Eligible for all activities")
else:
    print("Adult")
```

Output:

Adult

Simple Tamil Meaning:

Conditional Statements = நிபந்தனையின் அடிப்படையில் முடிவு எடுக்கும் Statements.

if → ஒரு நிபந்தனை

if else → இரண்டு முடிவுகள்

if elif else → பல நிபந்தனைகள்

Nested if → ஒரு if-க்குள் மற்றொரு if

5. Loops

Loops are used to **repeat a block of code.**

1. for loop

Used to repeat through a sequence.

```
for i in range(1, 6):
```

```
    print(i)
```

Output:

1

2

3

4

5

Example: Print student names

```
names = ["Priya", "Ravi", "Anu"]
```



for name in names:

```
    print(name)
```

2. while loop

Repeats while a condition is True.

```
i = 1
```

```
while i <= 5:
```

```
    print(i)
```

```
    i = i + 1
```

3. break

break stops the loop immediately.

```
for i in range(1, 10):
```

```
    if i == 5:
```

```
        break
```

```
    print(i)
```

Output:

1

2

3

4



4. continue

continue skips the current iteration.

```
for i in range(1, 6):
```

```
    if i == 3:
```

```
        continue
```

```
    print(i)
```

Output:

1

2

4

5



5. pass

pass is a placeholder. It does nothing.

```
for i in range(5):
```

```
    pass
```

6. Strings

A **String** is text written inside quotes.

String Creation

```
name = "Priya"
```

```
city = 'Chennai'
```

```
print(name)
```

```
print(city)
```

Indexing

Each character has a position starting from 0.

```
text = "PYTHON"
```

```
print(text[0])
```

```
print(text[2])
```

Output:

```
P
```

```
T
```

Slicing

Used to get part of a string.

```
text = "PYTHON"
```

```
print(text[0:3])
```

```
print(text[2:6])
```

Output:

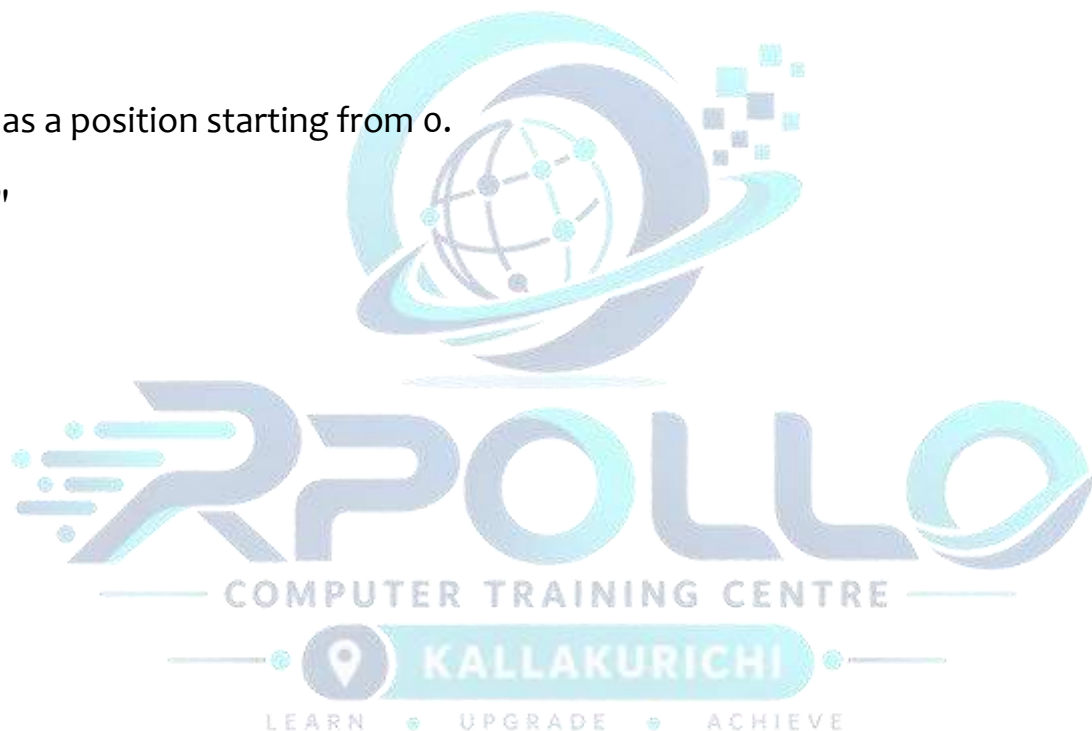
```
PYT
```

```
THON
```

String Methods

```
name = "python programming"
```

```
print(name.upper())
```





```
print(name.lower())
```

```
print(name.title())
```

```
print(name.replace("python", "Java"))
```

String Formatting

```
name = "Priya"
```

```
age = 20
```

```
print(f"My name is {name} and I am {age} years old")
```

7. Lists, Tuples & Sets

List

A list is **ordered and changeable**.

```
courses = ["Python", "Java", "C++"]
```

```
print(courses)
```

```
courses.append("HTML")
```

```
print(courses)
```

Basic List Methods

```
numbers = [10, 20, 30]
```

```
numbers.append(40)
```

```
numbers.remove(20)
```

```
numbers.pop()
```

```
print(numbers)
```

Tuple

A tuple is **ordered but cannot be changed**.

```
colors = ("Red", "Green", "Blue")
```

```
print(colors[0])
```

Set



A set stores **unique values**.

```
numbers = {1, 2, 3, 3, 4, 4}
```

```
print(numbers)
```

Duplicates are automatically removed.

Basic Set Methods

```
numbers = {10, 20, 30}
```

```
numbers.add(40)
```

```
numbers.remove(20)
```

```
print(numbers)
```

Difference

Feature	List	Tuple	Set
Ordered	Yes	Yes	No
Changeable	Yes	No	Yes
Duplicate values	Yes	Yes	No
Brackets	[]	()	{}

8. Dictionary

A dictionary stores data as **Key : Value** pairs.

Creating Dictionary

```
student = {  
    "name": "Priya",  
    "age": 20,  
    "course": "Python"  
}
```

```
print(student)
```

Keys and Values

```
print(student["name"])
```

```
print(student["course"])
```

Adding Data

```
student["city"] = "Chennai"
```



```
print(student)
```

Deleting Data

```
del student["age"]
```

```
print(student)
```

Dictionary Methods

```
student = {  
    "name": "Priya",  
    "age": 20  
}
```

```
print(student.keys())
```

```
print(student.values())
```

```
print(student.items())
```

9. Functions

A function is a **reusable block of code**.

Creating Functions

```
def welcome():  
    print("Welcome to Python")
```

```
welcome()
```

Parameters

Parameters receive data.

```
def greet(name):  
    print("Hello", name)
```

```
greet("Priya")
```

Here name is the **parameter**.

Arguments

The value sent to a function is called an **argument**.

```
def add(a, b):  
    print(a + b)
```

```
add(10, 20)
```



Here 10 and 20 are arguments.

Return Value

```
def add(a, b):
```

```
    return a + b
```

```
result = add(10, 20)
```

```
print(result)
```

Output:

30

Lambda Function

A lambda is a small one-line function.

```
add = lambda a, b: a + b
```

```
print(add(10, 20))
```

10. Modules & Packages

A **module** is a Python file containing reusable code.

Import

```
import math
```

```
print(math.sqrt(25))
```

Built-in Modules

Random

```
import random
```

```
print(random.randint(1, 10))
```

Date and Time

```
import datetime
```

```
print(datetime.datetime.now())
```

Creating Your Own Module

Create a file called:

```
my_module.py
```

Write:



```
def welcome():
```

```
    print("Welcome to Python")
```

Then create another Python file:

```
import my_module
```

```
my_module.welcome()
```

11. File Handling

File handling is used to **store and manage data in files**.

Open File

```
file = open("data.txt", "r")
```

Read File

```
file = open("data.txt", "r")
```

```
print(file.read())
```

```
file.close()
```

Write File

```
file = open("data.txt", "w")
```

```
file.write("Welcome to Python")
```

```
file.close()
```

Append File

Append adds new content without deleting old content.

```
file = open("data.txt", "a")
```

```
file.write("\nPython is easy")
```

```
file.close()
```

Better Method: with open()

```
with open("data.txt", "r") as file:
```

```
    print(file.read())
```

The file closes automatically.



12. Exception Handling

Exception handling is used to handle **errors without stopping the program**.

try and except

try:

```
number = int(input("Enter a number: "))  
print(number)
```

except:

```
print("Invalid input")
```

else

Runs when there is no error.

try:

```
number = int(input("Enter number: "))
```

except:

```
print("Error")
```

else:

```
print("You entered:", number)
```

finally

Runs whether an error occurs or not.

try:

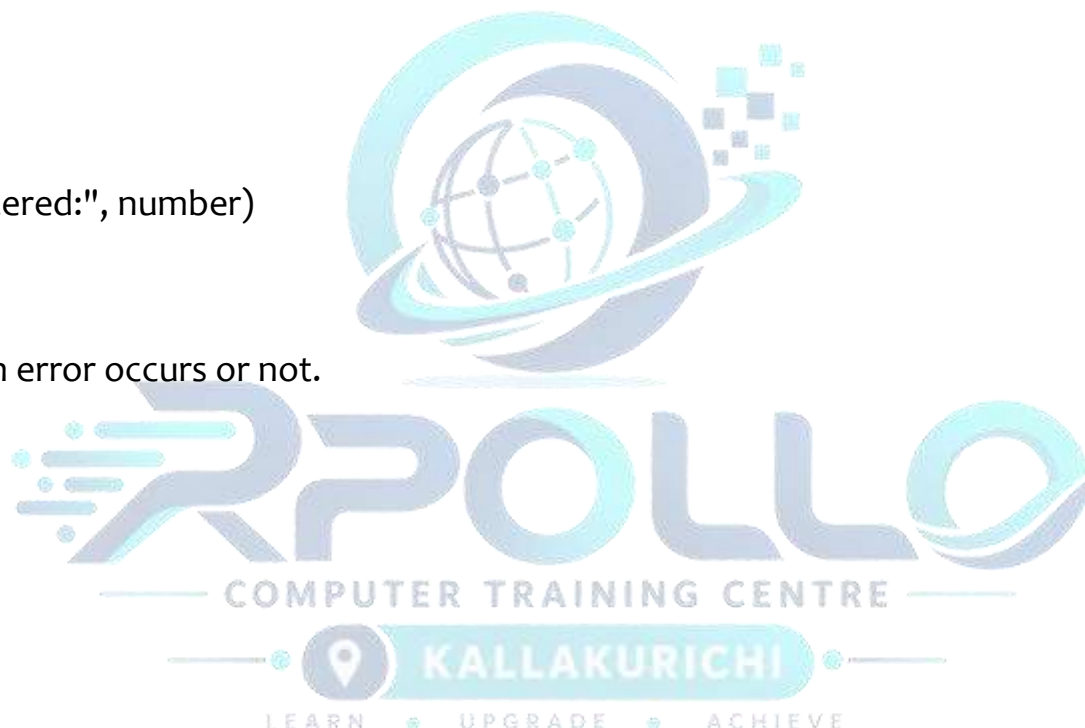
```
print(10 / 2)
```

except:

```
print("Error")
```

finally:

```
print("Program finished")
```



13. Object-Oriented Programming (OOP)

Class and Object

A **Class** is a blueprint.

An **Object** is created from a class.

class Student:

```
def welcome(self):  
    print("Welcome Student")
```

```
s1 = Student()
```



```
s1.welcome()
```

Constructor

A constructor automatically runs when an object is created.

```
class Student:
```

```
    def __init__(self, name):
```

```
        self.name = name
```

```
    def display(self):
```

```
        print(self.name)
```

```
s1 = Student("Priya")
```

```
s1.display()
```

Inheritance

One class can use another class's properties.

```
class Parent:
```

```
    def show(self):
```

```
        print("This is Parent")
```

```
class Child(Parent):
```

```
    def display(self):
```

```
        print("This is Child")
```

```
c = Child()
```

```
c.show()
```

```
c.display()
```

Encapsulation

Used to protect data.

```
class Student:
```

```
    def __init__(self):
```

```
        self.__marks = 90
```

```
    def show_marks(self):
```

```
        print(self.__marks)
```





```
s = Student()
```

```
s.show_marks()
```

Polymorphism

The same method name can work differently.

```
class Dog:
```

```
    def sound(self):
```

```
        print("Bark")
```

```
class Cat:
```

```
    def sound(self):
```

```
        print("Meow")
```

```
animals = [Dog(), Cat()]
```

```
for animal in animals:
```

```
    animal.sound()
```

14. Database Connectivity

Python can connect with databases such as **SQLite** and **MySQL**.

SQL Basics

Create Table

```
CREATE TABLE students (
```

```
    id INTEGER,
```

```
    name TEXT,
```

```
    age INTEGER
```

```
);
```

SQLite Connection

```
import sqlite3
```

```
connection = sqlite3.connect("college.db")
```

```
print("Database Connected")
```



```
connection.close()
```

Create Table Using Python

```
import sqlite3
```

```
connection = sqlite3.connect("college.db")
```

```
cursor = connection.cursor()
```

```
cursor.execute("""
```

```
CREATE TABLE IF NOT EXISTS students (
```

```
    id INTEGER PRIMARY KEY,
```

```
    name TEXT,
```

```
    age INTEGER
```

```
)
```

```
""")
```

```
connection.commit()
```

```
connection.close()
```

Insert Data

```
cursor.execute(
```

```
    "INSERT INTO students (name, age) VALUES (?, ?)",
```

```
    ("Priya", 20)
```

```
)
```

```
connection.commit()
```

Select Data

```
cursor.execute("SELECT * FROM students")
```

```
data = cursor.fetchall()
```

```
for row in data:
```

```
    print(row)
```

Update Data

```
cursor.execute(
```

```
    "UPDATE students SET age = ? WHERE name = ?",
```





```
(21, "Priya")
)
```

```
connection.commit()
```

Delete Data

```
cursor.execute(
    "DELETE FROM students WHERE name = ?",
    ("Priya",)
)
```

```
connection.commit()
```

MySQL

For MySQL, Python commonly uses a MySQL connector.

```
import mysql.connector
```

```
connection = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="college"
)
```

```
print("MySQL Connected")
```



15. Python Projects

1. Calculator

```
num1 = float(input("Enter first number: "))
```

```
num2 = float(input("Enter second number: "))
```

```
choice = input("Enter +, -, *, /: ")
```

```
if choice == "+":
```

```
    print("Answer:", num1 + num2)
```

```
elif choice == "-":
```



```
print("Answer:", num1 - num2)
```

```
elif choice == "*":
```

```
print("Answer:", num1 * num2)
```

```
elif choice == "/":
```

```
print("Answer:", num1 / num2)
```

```
else:
```

```
print("Invalid choice")
```

2. Student Management System

```
students = {}
```

```
while True:
```

```
print("\n1. Add Student")
```

```
print("2. View Students")
```

```
print("3. Exit")
```

```
choice = input("Enter choice: ")
```

```
if choice == "1":
```

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

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

```
    students[name] = age
```

```
    print("Student Added Successfully")
```

```
elif choice == "2":
```

```
    for name, age in students.items():
```

```
        print("Name:", name, "Age:", age)
```

```
elif choice == "3":
```

```
    break
```





else:

print("Invalid Choice")

3. To-Do List

tasks = []

while True:

print("\n1. Add Task")

print("2. View Task")

print("3. Delete Task")

print("4. Exit")

choice = input("Enter choice: ")

if choice == "1":

task = input("Enter task: ")

tasks.append(task)

elif choice == "2":

for task in tasks:

print(task)

elif choice == "3":

task = input("Enter task to delete: ")

tasks.remove(task)

elif choice == "4":

break

4. Quiz Application

score = 0

answer = input("What is the capital of India? ")

if answer.lower() == "delhi":





```
print("Correct!")

score += 1

else:

    print("Wrong!")

print("Your Score:", score)
```

5. Billing System

```
product = input("Enter product name: ")
price = float(input("Enter price: "))
quantity = int(input("Enter quantity: "))
```

```
total = price * quantity
```

```
print("\n---- BILL ----")
print("Product:", product)
print("Price:", price)
print("Quantity:", quantity)
print("Total:", total)
```

6. Simple Login System

```
username = "admin"
password = "1234"
```

```
user = input("Enter username: ")
pwd = input("Enter password: ")

if user == username and pwd == password:

    print("Login Successful")

else:

    print("Invalid Username or Password")
```

Python Topics 5–15 Summary

Loops → Strings → Collections → Dictionary → Functions → Modules → File Handling → Exception Handling → OOP → Database → Projects

This covers the main **intermediate to advanced Python topics** with simple examples and practical code.

