🐍COMPLETE PYTHON NOTES – DETAILED (PLACEMENT + REAL-TIME)

Bro ye notes **basic se advanced**, **real-time examples**, **interview + project mindset** ke saath likhe gaye hain. Isko follow kiya to Python solid ho jayega.

🐓WHAT IS PYTHON? (REAL-TIME VIEW)

Definition

Python is a **high-level, interpreted, object-oriented programming language** used for backend development, automation, data analysis, ML, and web apps.

Real-time use

- 🏦Bank software → account handling
- Websites → backend logic (Django / Flask)
- ⚓ML → data processing
- Automation → file handling, scripts

🐕VARIABLES & DATA TYPES (WITH EXAMPLES)

Variable

Variable is a **container to store data**.

```
name = "Karan"
age = 22
salary = 25000.50
```

Data Types

Type	Example	Real-time use
int	10	age, quantity
float	99.5	price, marks
str	"India"	name, city
bool	True	login status

```
is_logged_in = True
```

🐗TYPE CASTING (VERY IMPORTANT)

```
x = "10"
y = int(x)   # converting string to int
```

Real-time

Input hamesha **string** hota hai → calculation ke liye convert karna padta hai.

🐚INPUT & OUTPUT

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

Real-time

- Login form
- ATM input
- Registration form

🐜OPERATORS (WITH REAL USE)

Arithmetic

```
bill = 500
tax = 50
print(bill + tax)
```

Comparison

```
age = 18
print(age >= 18)
```

Logical

```
username = True
password = True
print(username and password)
```

🐝CONDITIONAL STATEMENTS (DECISION MAKING)

```
age = 17
if age >= 18:
    print("Eligible for voting")
else:
    print("Not eligible")
```

Real-time

- Login validation
- Eligibility check
- Result system

🐟LOOPS (AUTOMATION CONCEPT)

for loop

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

while loop

```
i = 1
while i <= 5:
    print(i)
    i += 1
```

Real-time

- Print bills
- Process records
- Repeat tasks

🐠FUNCTIONS (REUSABILITY)

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

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

Real-time

- Login function
- Payment function
- Validation function

🐡COLLECTIONS (DATA MANAGEMENT)

LIST (Mutable)

```
students = ["A", "B", "C"]
students.append("D")
```

Real-time: Student list, product list

TUPLE (Immutable)

```
days = ("Mon", "Tue")
```

Real-time: Fixed data (days, months)

SET (Unique values)

```
ids = {1,2,3,3}
```

Real-time: Unique user IDs

DICTIONARY (MOST IMPORTANT)

```
student = {
    "name": "Karan",
    "age": 22,
    "course": "CSE"
}
```

Real-time: JSON, API data, database rows

🌈STRING METHODS (REAL USE)

```
msg = " hello world "
print(msg.strip())
print(msg.upper())
```

Used in form validation, text processing

🐓🐓FILE HANDLING (VERY DETAILED + REAL-TIME)

File handling is used to **store data permanently** in files. Jab program close hota hai tab bhi data safe rehta hai.

🌧️Why File Handling?

- Data save karna (logs, reports)
- User records store karna
- Project data maintain karna

Real-time examples: - ATM transaction history - Login logs - Student records

---

🌧️File Modes (IMPORTANT)

Mode	Meaning	Use
r	Read	File read karne ke liye
w	Write	New file / overwrite
a	Append	Data add karne ke liye
r+	Read + Write	Both

🌧️Writing into a File

```
f = open("data.txt", "w")
f.write("Hello Python
")
f.write("File Handling")
f.close()
```

Agar file nahi hogi → create ho jayegi   Agar file hogi → overwrite ho jayegi

---

🌧️Reading from a File

```
f = open("data.txt", "r")
content = f.read()
print(content)
f.close()
```

🌧️Read Line by Line

```
f = open("data.txt", "r")
for line in f:
    print(line)
f.close()
```

Used when file is very large

🌧️Append Mode (Most Used)

```
f = open("data.txt", "a")
f.write("
New record added")
f.close()
```

Real-time: Daily logs add karna

🌧️with Statement (BEST PRACTICE)

```
with open("data.txt", "r") as f:
    print(f.read())
```

Automatically file close ho jati hai

🌧️Real-Time Mini Example (Student Record)

```
with open("students.txt", "a") as f:
    name = input("Enter name: ")
    f.write(name + "
")
```

🌧️File Handling with Exception

```python
try:
    f = open("data.txt", "r")
    print(f.read())
except FileNotFoundError:
    print("File not found")
finally:
    f.close()
```

🌧️Interview One-Liners

- File handling is used for permanent data storage
- `with` statement is safer than open()
- Append mode does not overwrite data

🐓🐕 EXCEPTION HANDLING (ERROR CONTROL)

```python
try:
    print(10/0)
except ZeroDivisionError:
    print("Cannot divide by zero")
```

Real-time: Prevent app crash

🐓🐗OOPS (MOST IMPORTANT FOR PLACEMENT)

Class & Object

```python
class Student:
    def __init__(self, name):
        self.name = name

s1 = Student("Karan")
```

🦙FOUR PILLARS OF OOPS

🐓Encapsulation

Data ko protect karna

```python
class Bank:
    def __init__(self):
        self.__balance = 5000
```

🐕Inheritance

Parent → Child

```python
class Person:
    pass
class Student(Person):
    pass
```

🐗Polymorphism

Same method, different output

```python
class Dog:
    def speak(self):
        print("Dog")
class Cat:
    def speak(self):
        print("Cat")
```

🐚Abstraction

Hide implementation (basic idea)

🐓🐚MODULES

```python
import math
print(math.sqrt(16))
```

🐓🐜PYTHON INTERVIEW ONE-LINERS

- Python is interpreted
- Everything is object
- List is mutable

- Tuple is immutable
- Dictionary stores key-value

🌀FINAL ADVICE (BRO LEVEL)

Code daily GitHub push daily Logic > theory Revise interview questions

---

If you master this file → Python backend ready