🐍 Python Beginner Guide (Complete Notes)

(Best for students, interviews, and ML foundation)

🔑 1. What is Python?

Python is a **high-level**, **interpreted**, and **object-oriented** programming language designed to be easy to read and write.

✓ Why Python?

- Simple and clean syntax
- Large community support
- Built-in libraries
- Used in AI, ML, Web Dev, Automation, Data Science, Scripting

🔑 2. Python Installation & IDE

Common IDEs:

IDE	Best For
IDLE	Beginners
VS Code	Projects
PyCharm	Professional development
Jupyter Notebooks	Data science / ML

Run code using:

```
python filename.py
```

🔑 3. Python Syntax

Indentation:

Python uses indentation instead of curly braces.

```
if 5 > 2:
    print("Yes")
```

📍 4. Comments

```
# Single line comment

'''
Multi-line
comment
'''
```

📍 5. Variables & Data Types

Python doesn't require type declaration.

```
x = 10
name = "Mohit"
pi = 3.14
```

Common Data Types:

Type	Example
int	10
float	10.5
str	"Hello"
bool	True, False
list	[1, 2, 3]
tuple	(1, 2, 3)
set	{1, 2, 3}
dict	{"name": "Mohit"}

📍 6. Input & Output

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

Formatted printing:

```
print(f"My name is {name}")
```

📍 7. Operators

Arithmetic:

```
a + b, a - b, a * b, a / b, a // b, a ** b, a % b
```

Comparison:

```
==, !=, >, <, >=, <=
```

Logical:

```
and, or, not
```

📍 8. Conditional Statements

```
age = 18

if age >= 18:
    print("Eligible")
elif age == 17:
    print("Almost there")
else:
    print("Not eligible")
```

📍 9. Loops

For Loop

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

While Loop

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

🔑 10. Functions

```
def greet(name):
    return f"Hello {name}"

print(greet("Mohit"))
```

🔑 11. Data Structures

List

```
fruits = ["Apple", "Mango"]
fruits.append("Banana")
print(fruits)
```

Tuple (Immutable)

```
t = (1, 2, 3)
```

Set (Unique Values)

```
s = {1, 2, 3}
s.add(4)
```

Dictionary

```
student = {"name": "Mohit", "age": 21}
print(student["name"])
```

🔑 12. String Methods

```
msg = "hello python"
print(msg.upper())
print(msg.capitalize())
print(msg.split())
```

📍 13. File Handling

```
# Write File
with open("data.txt", "w") as f:
    f.write("Hello World")

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

📍 14. Exception Handling

```
try:
    print(10 / 0)
except ZeroDivisionError:
    print("Error!")
finally:
    print("Done!")
```

📍 15. Object-Oriented Programming (OOP)

```
class Car:
    def __init__(self, brand):
        self.brand = brand

    def show(self):
        print("Brand:", self.brand)

c = Car("BMW")
```

```
c.show()
```

🔑 16. Modules & Packages

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

Custom module:

```
# file: mymodule.py
def hello():
    print("Hello from module")

# import
import mymodule
mymodule.hello()
```

🔑 17. Python in ML/NLP

Python is popular in **Natural Language Processing (NLP)** because:

✓ Syntax is close to natural English
✓ Libraries (NLTK, spaCy, transformers)
✓ Works great with AI frameworks like TensorFlow & PyTorch

Example:

```
import nltk
from nltk.tokenize import word_tokenize

text = "Python is amazing for NLP."
print(word_tokenize(text))
```

18. Mini Project Example

```
names = []

while True:
    n = input("Enter name (or 'stop'): ")
    if n == "stop":
        break
    names.append(n)

print("Names entered:", names)
```

Summary Checklist

Topic	Done
Variables & Data types	✓
Flow Control	✓
Loops	✓
Functions	✓
OOP	✓
Files	✓