UNIT 1: INTRODUCTION TO PYTHON

What is Python?

Python is a high-level, interpreted, general-purpose programming language.

- Created by Guido van Rossum and released in 1991.
- Designed to be easy to read and write.
- Supports multiple programming paradigms. (Procedural, Object-Oriented, Functional)
- Has a large standard library and community support.
- Used in Web Development, Data Science, AI, Automation, Scripting and more.

Did You Know?

Python was named after the TV show "Monty Python's Flying Circus"!

Key Features of Python

- Easy to Learn – Simple syntax
- Readable – Code looks like English
- Interpreted – Executes code line by line
- Portable – Run on Windows, Mac, Linux
- Open Source – Free and community driven
- Versatile – Many domains and applications

Why Python?

- Beginner friendly
- Large community and support
- Huge collection of libraries
- High demand in industry
- Great for rapid development

Applications of Python

- Web Development (Django, Flask)
- Data Science & Analytics
- Machine Learning & AI
- Automation & Scripting
- Game Development
- IoT, Networking, Cyber Security and more

Hello, Python!

```
# This is a comment
print("Hello, Python!")
```

Use print() to display output

Python Workflow

Write Code (.py file) → Save File → Run / Execute (Python Interpreter) → Output

Remember

- ✓ Indentation is important in Python.
- ✓ Python is case-sensitive.
- ✓ Practice programs regularly to improve skills.

Keep Coding
Keep Growing! ☆

Features & Applications of Python

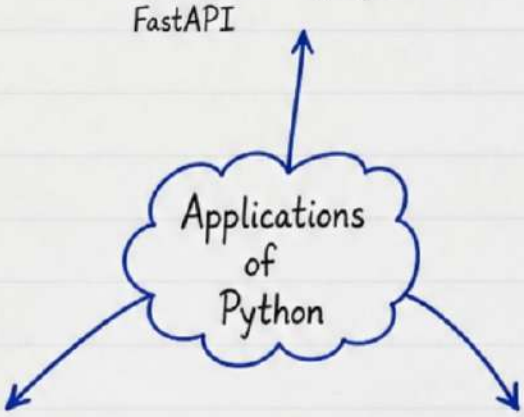
Features of Python

- ☆ **Easy to Learn & Use**
  Simple syntax similar to English.
  Beginner friendly.
- ☆ **High-Level Language**
  Abstracts complex details,
  Easy to write & maintain.
- ☆ **Interpreted Language**
  Runs line by line, no compilation needed.
- ☆ **Platform Independent**
  Runs on Windows, Mac, Linux, and more.
- ☆ **Large Standard Library**
  Built-in libraries for various tasks.
- ☆ **Open Source**
  Free to use and supported by a huge community.
- ☆ **Extensible**
  Can be integrated with C, C++, Java etc.
- ☆ **Object-Oriented**
  Supports classes, objects, inheritance and more.

Applications of Python

Web Development

- Build websites and web apps using frameworks.
- Frameworks: Django, Flask, FastAPI



Artificial Intelligence

- Used in Machine Learning and Deep Learning.
- Libraries: TensorFlow, PyTorch, Keras

Data Science

- Used for data analysis, visualization and insights.
- Libraries: NumPy, Pandas, Matplotlib, Seaborn

Memory Trick

Remember "WAD" for Applications of Python

WAD → Web, AI, Data

W → Web Development
A → Artificial Intelligence
D → Data Science

Think:
"Websites need Brains to handle Data!"

Installation & Running First Program

1) Installation Steps

Download IDE
(VS Code / Arduino IDE / Any C/C++ IDE)

↓

Install IDE
Follow the setup wizard and complete installation.

↓

Install Compiler / Tools
For C/C++ → MinGW / GCC
For Arduino → Built-in

↓

Verify Installation
Open terminal / command prompt and check version.

↓

Create New File
Save the file with .c extension.

↓

Build and Run
Compile the program and run the output.

2) First Program – "Hello, World!"

```c
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

3) Output (Terminal)

```
Command Prompt                                  — □ ×

C:\Programs\hello> hello.exe
Hello, World!

C:\Programs\hello> _
```

4) Common Mistakes

- Forgetting header file → #include <stdio.h>
- Missing semicolon (;)
- Case sensitive errors → main() is not Main()
- Saving file with wrong extension → .txt instead of .c
- Not compiling the file before running
- Ignoring error messages in terminal

TIP: Always read error messages carefully. They help you fix the problem faster.

Syntax, Comments & Keywords

→ SYNTAX:

Syntax refers to the set of rules that define the structure of a program. Following proper syntax is essential for the program to run without errors.

INDENTATION MATTERS

CORRECT INDENTATION	INCORRECT INDENTATION

`if (x > 0):`
`  print ("Positive")`
`else:`
`  print ("Non-positive")`
✓
	`if (x > 0):`
	`  print ("Positive")`
	`else:`
	`  print ("Non-positive")`
	✗

→ COMMENTS:

Comments are used to explain code. They are ignored by the interpreter.

1) SINGLE-LINE COMMENT

Use # to write a comment

Example:

```
x = 10      # This is a comment
y = x + 5   # another comment
```

2) MULTI-LINE COMMENT

Use triple quotes (`'''...'''`) or triple double quotes (`"""..."""`)

Example:

```
'''
This is a
multi-line comment
that spans multiple lines.
'''
```

→ KEYWORDS:

Keywords are reserved words in Python that have special meanings and cannot be used as identifiers.

List of Python Keywords (as of Python 3.x):

- False
- None
- True
- and
- as
- assert
- async
- await
- break
- class
- continue
- def
- del
- elif
- else
- except
- finally
- for
- from
- global
- if
- import
- in
- is
- lambda
- nonlocal
- not
- or
- pass
- raise
- return
- try
- while
- with
- yield

> 💡 Remember:
> Keywords cannot be used as variable names, function names, or any other identifiers.

GOOD PRACTICES:

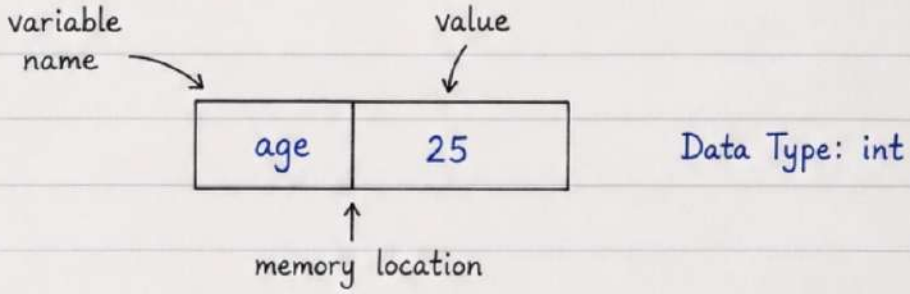
- Follow proper indentation (use 4 spaces per level).
- Use comments to explain complex logic.
- Use meaningful variable and function names.

Variables & Data Types

What is a Variable?

A variable is a named location in memory used to store data.
Think of it as a box (memory) with a label (variable name) and a value inside.

Variable Representation:



Common Data Types in Python

Data Type	Description	Example	Example Value
int	Integer (whole number)	age	25
float	Floating point number	price	19.99
str	String (text)	name	"Alice"
bool	Boolean (True or False)	is_active	True
list	Ordered collection	marks	[85, 90, 78]
tuple	Ordered, immutable collection	point	(10, 20)
set	Unordered collection (unique)	unique_ids	{1, 2, 3}
dict	Key-value pairs	student	{'name': 'Avi', 'age': 20}

Key Points

- Variables must be declared before use.
- Python is dynamically typed (type is determined at runtime).
- Variable names are case-sensitive.
- Use meaningful names for better readability.

Interview Question

Q. What is the difference between list and tuple?

Ans: List is mutable (can be changed) and uses [ ]. Tuple is immutable (cannot be changed) and uses ( ).

Example

```
age = 25
price = 19.99
name = "Alice"
marks = [85, 90, 78]
student = {'name': 'Avi', 'age': 20}
```

Type Casting & User Input

① Type Casting (Type Conversion)

★ Converting one data type into another.
Helps when we need to perform operations between incompatible types.

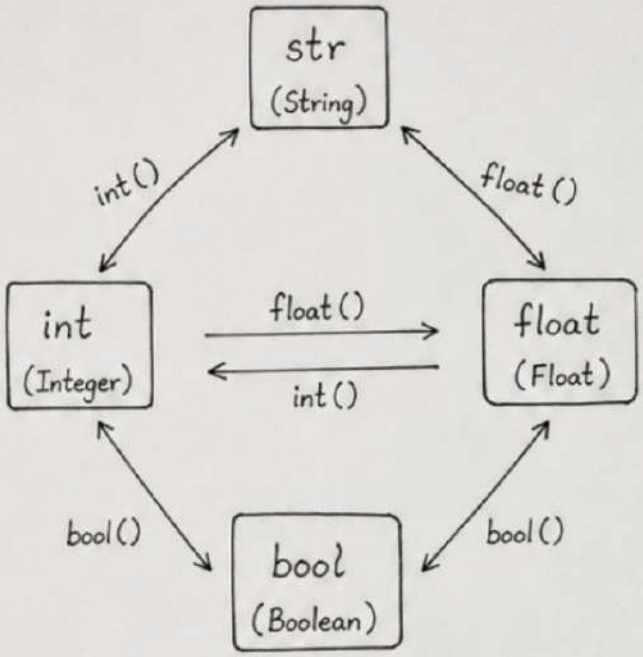
★ Implicit Conversion (Automatic)

Python automatically converts smaller type to larger type.
Example : int → float

★ Explicit Conversion (Manual)

We convert the type using built-in functions.
Example : str → int , float → int, etc.

TYPE CONVERSION DIAGRAM



Examples :

- int ("25") → 25
- float ("3.14") → 3.14
- str (100) → "100"
- bool(0) → False , bool(1) → True

---

② User Input in Python

★ We use the input() function to take input from the user.

Syntax : `input (prompt)`

★ The input() function always returns data as a string.
So, we may need to type cast it to the required type.

Code Example :

```
name   = input ("Enter your name : ")
age    = int (input ("Enter your age : "))
height = float (input ("Enter your height (in m) : "))

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

Memory Trick

"SIFB"

S → String (text)
I → Integer (whole number)
F → Float (decimal number)
B → Boolean (True / False)

Input is always a String first !

Note : Always check the data type you are working with.
Use type casting when needed to avoid errors.

Date :
Page No

Output Formatting

→ Output formatting in Python helps us display values in a clean, readable and structured way.

1. f-strings (Python 3.6+)

- Easy and readable
- Prefix string with 'f' or 'F'
- Variables are placed inside { }

2. .format() method

- Works in all Python versions
- Uses curly braces { } as placeholders
- More flexible for complex formatting

	f-strings	.format() method
Syntax	f"text {variable}"	"text { }".format(variable)
Example	name = "Alice" age = 20 msg = f"My name is {name} and I am {age} years old."	name = "Alice" age = 20 msg = "My name is { } and I am { } years old.".format(name, age)
Output	My name is Alice and I am 20 years old.	My name is Alice and I am 20 years old.

Real-world Example

→ Suppose we are building a billing system for a store. We want to generate a bill for the customer.

Using f-strings:

```
item = "Wireless Mouse"
price = 750
qty = 2
total = price * qty
bill = (f"Item: {item}\n"
        f"Quantity: {qty}\n"
        f"Total Amount: ₹{total}")
print(bill)
```

⇒

Using .format() method:

```
item = "Wireless Mouse"
price = 750
qty = 2
total = price * qty
bill = ("Item: {}\n"
        "Quantity: {}\n"
        "Total Amount: ₹{}"
        .format(item, qty, total))
print(bill)
```

Output:

```
Item: Wireless Mouse
Quantity: 2
Total Amount: ₹1500
```

Key Takeaway

- Use f-strings for modern, clean and readable code.
- Use .format() when working with dynamic or older Python versions.