Chapter 3: Input and Output

Letting the user talk to Python—and Python reply back.

Introduction

In the previous chapters, you learned how to use `print()` to make Python speak and how to store data in variables.

Now, we flip it.
We make **the user** speak—and **Python** listen.

That's where `input()` comes in.

Together, `input()` and `print()` let you build interactive, fun programs.

The `input()` Function

```python
name = input("What is your name? ")
print("Hello", name)
```

Example Interaction:

```
What is your name? Ravi
Hello Ravi
```

→ The text inside `input()` is the **prompt**.
→ What the user types is stored in a variable.
→ The result is **always a string**.

Doing Math with Input

Remember: `input()` gives you a **string**, not a number.

If you want to do math, you must **convert** it.

```python
num = input("Enter a number: ")
num = int(num)
print("Double of your number is", num * 2)
```

Without conversion, this will fail:

```
age = input("Age: ")
print(age + 1)  # ❌ TypeError: can’t add str + int
```

→ Fix it with:

```
age = int(age)
print(age + 1)
```

Formatting Strings (Making Output Look Good)

There are **4 ways** to combine variables with text:

1. Commas in `print()`

```
name = "Ravi"
print("Hello", name)
```

✓ Simple
✓ Adds spaces automatically

---

2. Plus (`+`) Operator

```
print("Hello " + name)
```

✓ Good when combining **strings only**
✗ Will error if you forget to convert a number:

```
print("Age: " + 12)  # ❌ TypeError
```

---

3. f-Strings (Modern Style)

```
name = "Ravi"
age = 14
print(f"Hello {name}, you are {age} years old.")
```

✓ Cleanest and most recommended
✓ Fast and readable
✗ Don't forget the `f` before the string

---

4. `.format()` Method (Older Style)

```python
print("Hello {}, you are {} years old.".format(name, age))
```

✓ Works fine
✗ Slightly older syntax than f-strings

Special Characters in Strings

```python
print("Line 1\nLine 2")  # New line
print("Name:\tJohn")     # Tab space
```

Output:

```
Line 1
Line 2
Name:	John
```

→ You can format text layout using `\n` and `\t`.

Mini Quiz or Challenge

1. What will this print?

```python
name = input("Name: ")
print("Hello " + name)
```

2. How can you combine a number and a string in one print line?

3. What will happen with this?

```python
print("Age: " + 12)
```

Tips, Mistakes, and Mini Rules

✓ Always **convert input** before doing math
✓ Use `f"{}` for clean output
✓ Use commas to mix types easily
✗ Don't mix strings and numbers with `+` unless types match
✗ Don't forget the `f` when using f-strings

Summary Recap

- `input()` lets users type into your program

- It always returns a **string**
- Use `int()` or `float()` to convert input when doing math
- Combine with `print()` to make conversational programs
- Format output using:
  - Commas
  - Plus `+`
  - f-strings `f"{}"`
  - `.format()`

Mini-Project Exercise

🎯 *Create a mini greeting app.*

Ask for name, age, and favorite color.
Use an `f-string` to print a friendly message.

```python
name = input("Name: ")
age = input("Age: ")
color = input("Favorite color: ")

print(f"Hi {name}! You are {age} years old and you like {color}.")
```

Practice Exercises

✅ Basic Problems

1. Ask the user’s name and greet them
2. Ask for two numbers and print their sum
3. Ask for name and age, then use `.format()` to print
4. Ask for a city and print "Welcome to "
5. Ask for favorite number and print double of it
6. Ask for something funny and print it back with a smiley
7. Print name and school on two lines using `\n`

🚀 Intermediate Problems

A1. Ask the user their birth year and calculate their age
A2. Ask for 3 hobbies and print them in a sentence
A3. Make a short bio-generator:
Ask name, age, city, and interest → print a paragraph using an f-string

---