Python Lesson: User Input and Data Handling

Lesson Introduction

So far, Python programs have used values written directly in the code. This lesson introduces user input, which allows a program to pause, receive information from the user, and change its behavior based on that input. You will learn how Python handles user input, why input values must be converted, and different ways input can be provided to a program.

1. Purpose of User Input

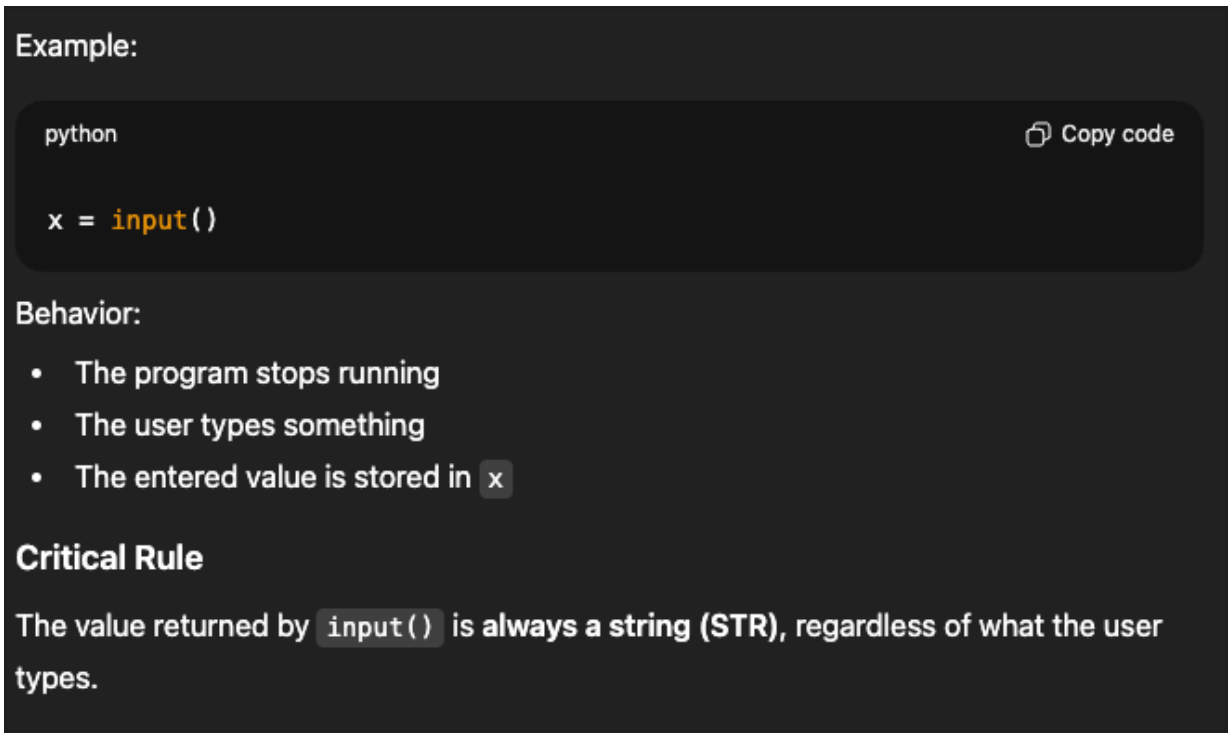
Programs that use fixed values always behave the same way. User input allows programs to:

- Wait for user interaction
- Accept data from the keyboard or command line
- Respond differently based on the input provided

This is a key step toward building interactive programs.

2. The input() Function

Python receives input using the built-in input() function.



Behavior:

- The program stops running
- The user types something
- The entered value is stored in x

Critical Rule

The value returned by input() is **always a string (STR)**, regardless of what the user types.

3. Prompting the User

When input() is used without instructions, the user may not know what to enter.
A prompt message explains what input is expected:

4. Type Conversion

Because input() returns a string, Python cannot perform arithmetic unless the value is converted.

Incorrect usage:

```python
x = input("Enter number: ")
print(x + 5)  # Error
```

Converting to Integer

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

Converting to Float

```python
x = float(input("Enter number: "))
```

After conversion, Python can perform mathematical operations.

5. Inline Type Conversion

Instead of converting in two steps:

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

Python allows conversion in one step:

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

This shortens the code without changing how it works.

6. Accepting a Single Character

Using `input()` allows the user to type multiple characters.

To store only the **first character**, use indexing:

python Copy code

```python
ch = input()[0]
```

Explanation:

- User enters text
- Python keeps only the first character
- Strings allow indexing starting from position 0

7. Evaluating Expressions with eval()

The `eval()` function allows Python to evaluate expressions typed by the user.

Example:

```python
result = eval(input("Enter expression: "))
```

If the user types:

```
2 + 6 - 1
```

Python evaluates the expression and returns:

```
7
```

Important note:

- `eval()` executes the input as Python code
- It should be used cautiously

8. Summary Table of Key Concepts

Concept	Description	Example
input()	Receives user input as a string	x = input()
Prompt message	Explains what the user should enter	input("Enter number: ")
String result	Input is always stored as a string	"5"
int()	Converts input to integer	int(input())
float()	Converts input to decimal number	float(input())
Inline conversion	Converts input in one line	x = int(input())

String indexing	Extracts first character	input()[0]
eval()	Evaluates expressions from input	eval(input())
sys.argv	Command-line input storage	sys.argv[1]
Importing sys	Required to use sys.argv	import sys