Reading Data from Keyboard

Theory:

In Python, the **input() function** is used to **take data (input)** from the user through the **keyboard** during program execution.
When the program reaches an input() statement, it **pauses** and waits for the user to type something and press **Enter**.

🔹 1. Syntax of input()

variable_name = input("Message to the user: ")

- **variable_name** → The name of the variable that stores the user's input.
- **Text inside parentheses and quotes** → The message or prompt displayed to the user.

Example:

name = input("Enter your name: ")

If the user types Christian, then the variable name will store the string "Christian".

🔹 2. Default Data Type

By default, the data entered through input() is stored as a **string**, even if the user types numbers.

Example:

age = input("Enter your age: ")

print(type(age))

If the user enters 22, the output will be:

<class 'str'>

That means the input is stored as a **string**, not an integer.

🔹 3. Type Conversion

If you need to use the input as a **number** for calculations, you must **convert** it to another data type such as:

- int() → for integers (whole numbers)

- float() → for decimal numbers

Example:

```
num1 = int(input("Enter an integer: "))
num2 = float(input("Enter a float number: "))

print("Sum =", num1 + num2)
```

Output:

```
Enter an integer: 10
Enter a float number: 5.5
Sum = 15.5
```

🔹 4. Combining Text and Variables

You can combine input values with other text using:

- **+ operator** (string concatenation)
- **.format() method**
- **f-strings** (formatted string literals)

Example using f-string:

```
name = input("Enter your name: ")
age = int(input("Enter your age: "))
print(f" Hello, {name}! You are {age} years old.")
```

Output:

```
Enter your name: Christian
Enter your age: 22
Hello, Christian! You are 22 years old.
```

🔹 5. Important Notes

- Always use the correct data type conversion before performing arithmetic operations.

- If the user enters an invalid input (like typing letters when int() is expected), Python will show a **ValueError**.
- You can handle such errors later using **try-except** statements.
