# Input from User with input() in Python

Premanand S

Assistant Professor
School of Electronics Engineering
Vellore Institute of Technology
Chennai Campus

*premanand.s@vit.ac.in*

November 20, 2024

# Introduction to `input()`

- `input()` is a built-in Python function used to prompt the user for input.
- It always returns input as a **string**.

## Example (Python Code)

```
# Basic Syntax
variable = input("Enter a value: ")
```

# Capturing Basic User Input

- Use `input()` to prompt the user and store the result in a variable.
- Provide a message in the prompt to guide the user.

## Example (Python Code)

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

# Converting Input to Other Data Types

- `input()` returns a string, so conversion may be needed.
- Common conversions: `int()`, `float()`, `bool()`.

## Example (Python Code)

```
age = int(input("Enter your age: "))
height = float(input("Enter your height: "))
is_student = input("Are you a student? (yes/no): ").
                                        lower() == 'yes'
```

# Handling Invalid Input with `try-except`

- **ValueError** occurs if input cannot be converted.
- Use `try-except` to manage invalid inputs.

## Example (Python Code)

```
try:
    age = int(input("Enter your age: "))
except ValueError:
    print("Invalid input! Please enter a number.")
```

# Advanced Prompting Techniques

- Use multi-line prompts for complex instructions.
- Input validation loop: Repeatedly prompt until valid input is received.

## Example (Python Code)

```
while True:
    try:
        age = int(input("Enter your age: "))
        break
    except ValueError:
        print("Please enter a valid integer.")
```

# Using `input()` with Lists and Multiple Values

- Use `split()` to capture multiple values separated by spaces.
- Convert list elements if needed.

## Example (Python Code)

```
numbers = input("Enter numbers separated by space: ").split()
numbers = [int(num) for num in numbers]
```

# Inputting Data for Complex Structures

- Prompt for key-value pairs to build dictionaries.
- Example: Creating a dictionary from user input.

## Example (Python Code)

```
user_data = {}
user_data['name'] = input("Enter your name: ")
user_data['age'] = int(input("Enter your age: "))
user_data['scores'] = [int(x) for x in input
                       ("Enter scores: ").split()]
```

# Input Validation and Custom Error Messages

- Validate inputs to ensure they meet required conditions.
- Use custom error messages to guide users.

## Example (Python Code)

```
age = int(input("Enter age (must be positive): "))
if age <= 0:
    print("Age must be positive.")
```

# Parsing CSV-style Input Using `split()`

- For comma-separated input, use `split(',')` to create a list.

## Example (Python Code)

```
data = input("Enter names separated by commas: ")
names = data.split(',')
```

# Secure Input with getpass

- Use `getpass` to securely gather passwords without displaying them.

**Example (Python Code)**

```
from getpass import getpass
password = getpass("Enter your password: ")
```

# Validating Complex Data Formats with Regular Expressions

- Use `re.fullmatch()` for format validation like email or phone number.

## Example (Python Code)

```
import re
email = input("Enter your email: ")
if re.fullmatch(r"[^@]+@[^@]+\.[^@]+", email):
    print("Valid email")
else:
    print("Invalid email")
```

# Simulating Default Values in `input()` Prompt

- Use a conditional to handle empty input as a default.

## Example (Python Code)

```
name = input("Enter your name (default: anand): ") or "prem"
print("Hello,", name)
```

# `input()` understanding

# 1. What is the `input()` function in Python?

- The `input()` function in Python is used to take input from the user.
- It returns the input as a string.

## Example (Python Code)

```
name = input("Enter your name: ")
print(f"Hello, {name}!")
```

# 2. How does `input()` work in Python?

- The `input()` function displays a prompt message (optional) to the user and waits for input.
- After the user presses Enter, the function returns the input as a string.

## Example (Python Code)

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

# 3. Can you give an example of using `input()` to get user input?

- Here is an example of using `input()` to get user input and print a message.

**Example (Python Code)**

```
name = input("Enter your name: ")
print(f"Hello, {name}!")
```

# 4. What is the default return type of `input()`?

- The `input()` function always returns a string.
- Even if a number is entered, it will be returned as a string.

## Example (Python Code)

```
age = input("Enter your age: ")
print(type(age))
# Output: <class 'str'>
```

# 5. How do you convert the input from `input()` into an integer or a float?

- Since `input()` returns a string, you can use `int()` or `float()` to convert the input to the respective type.

**Example (Python Code)**

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

# 6. What happens if the user presses Enter without typing anything?

- If the user presses Enter without typing anything, `input()` returns an empty string `""`.

## Example (Python Code)

```
name = input("Enter your name: ")
# If Enter is pressed without input,
# it prints "Your name is: "
print("Your name is:", name)
```

# 7. How would you implement input validation for an integer input using `input()`?

- Use a loop and `try-except` to validate that the input is an integer.

## Example (Python Code)

```
while True:
    try:
        age = int(input("Enter your age: "))
        break
    except ValueError:
        print("Please enter a valid integer.")
```

# 8. Can you take multiple inputs in one line using `input()`

- Yes, you can use `split()` to capture multiple inputs separated by spaces.

**Example (Python Code)**

```
numbers = input("Enter numbers separated by space: ").split()
numbers = [int(num) for num in numbers]
print("Sum of numbers:", sum(numbers))
```

# 9. What is the use of the `input()` function's prompt argument?

- The prompt argument is an optional message that is displayed to the user to inform them of what input is expected.

**Example (Python Code)**

```
name = input("Please enter your name: ")
```

# 10. How can you handle multiple inputs of different types from the user in a single line?

- Use `split()` to separate the inputs and convert each input to the appropriate type.

**Example (Python Code)**

```
name, age = input("Enter your name and age: ").split()
age = int(age)
print(f"Name: {name}, Age: {age}")
```

# 11. How do you prevent users from entering empty input?

- Check if the input string is empty and prompt the user again if necessary.

## Example (Python Code)

```
while True:
    name = input("Enter your name: ").strip()
    if name:
        break
    print("Name cannot be empty.")
```

# 12.Can `input()` be used for reading multi-line input?

- By default, `input()` reads a single line. However, you can implement multi-line input using a loop.

## Example (Python Code)

```
print("Enter your multi-line input (press Enter on an empty
                                                line to finish):")
lines = []
while True:
    line = input()
    if line == "":
        break
    lines.append(line)
print("You entered:", "\n".join(lines))
```

# 13.How would you handle user input for a login system, ensuring secure password input using `input()`

- Use the `getpass` module to securely handle password input without displaying it.

**Example (Python Code)**

```
from getpass import getpass
username = input("Enter your username: ")
password = getpass("Enter your password: ")
```

# 14. Can you implement a user input prompt with a default value if the user does not provide any input?

- Use a conditional expression to provide a default value when input is empty.

**Example (Python Code)**

```
name = input("Enter your name (default: anand): ") or "Prem"
print(f"Hello, {name}!")
```