Chapter 1: Python Fundamentals

Lesson 4: Input and Output

---

1. Introduction to Input and Output

Input and Output (I/O) are fundamental aspects of any programming language. In Python, I/O operations allow programs to interact with users or external systems.

- **Input**: Refers to data provided to the program, typically by the user.
- **Output**: Refers to data produced by the program, which is displayed to the user.

---

2. Taking Input in Python

Python provides a built-in function called `input()` to accept data from the user via the keyboard.

Syntax

```
variable_name = input("Prompt message: ")
```

- `Prompt message`: A string displayed to the user to guide them on what to enter.
- `variable_name`: Stores the user's input as a string.

Example

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

Key Points

1. The `input()` function always returns the user's input as a **string**.
2. If you need numeric input, you must explicitly convert it using type casting (`int()`, `float()`, etc.).

Example with Type Casting

```
age = int(input("Enter your age: "))
print(f"You will be {age + 10} years old in 10 years.")
```

---

3. Displaying Output in Python

The `print()` function is used to display output to the user.

Syntax

```
print(value1, value2, ..., sep=' ', end='\n')
```

- `value1, value2, ...`: Values to be printed.
- `sep`: Separator between values (default is a space `' '`).
- `end`: What to print at the end (default is a newline `'\n'`).

Examples

Basic Usage

```
print("Hello, World!")
```

Multiple Values

```
name = "Alice"
age = 25
print("Name:", name, "Age:", age)
```

Custom Separator

```
print("Python", "is", "fun", sep="-")
# Output: Python-is-fun
```

Custom End

```
print("Hello", end=" ")
print("World!")
# Output: Hello World!
```

---

4. Formatting Output

Python provides several ways to format output for better readability and presentation.

a. Using f-Strings (Recommended)

Introduced in Python 3.6, f-strings are concise and easy to use.

Syntax

```
f"String with {variable}"
```

Example

```
name = "Bob"
age = 30
print(f"My name is {name} and I am {age} years old.")
```

b. Using `.format()` Method

This method allows placeholders `{}` to be replaced with values.

Syntax

```
"String with {}".format(value)
```

Example

```
name = "Charlie"
age = 22
print("My name is {} and I am {} years old.".format(name, age))
```

c. Using `%` Operator (Old Style)

This is an older method but still supported.

Syntax

```
"String with %s" % value
```

Example

```
name = "Diana"
age = 28
print("My name is %s and I am %d years old." % (name, age))
```

---

5. Practical Examples

Example 1: User Profile

Create a program that asks for a user's details and displays them in a formatted way.

```
name = input("Enter your name: ")
age = input("Enter your age: ")
city = input("Enter your city: ")

print(f"User Profile:\nName: {name}\nAge: {age}\nCity: {city}")
```

---

6. Hands-On Exercises

1. Write a program that asks the user for their favorite color and prints a message like: `"Your favorite color is [color]. That's awesome!"`.
2. Write a program that asks the user for their full name and prints it.

---

7. Common Pitfalls

1. **Forgetting Type Conversion**: Always remember to convert input to the appropriate type when working with numbers.

```
# Incorrect
age = input("Enter your age: ")
print(age + 10)  # TypeError

# Correct
age = int(input("Enter your age: "))
print(age + 10)
```

---

8. Summary

- Use `input()` to take user input and `print()` to display output.
- Format output using f-strings, `.format()`, or `%` operators.
