Learn Python Programming from Scratch

Topic: Input and Output in Python

1. What is Input/Output?

Input and **Output** are fundamental concepts in programming that allow your program to interact with users and the external environment.

- **Input** is how you get data from the user or another source into your program
- **Output** is how you display data from your program to the user or another destination

In Python, input/output operations are straightforward and user-friendly:

- Use `print()` function to display output to the console
- Use `input()` function to get input from the user
- The `input()` function always returns a string, so type conversion may be needed

2. Why Input/Output is Important

Input/Output operations are essential because they:

- Enable user interaction with your programs
- Allow programs to receive data dynamically
- Make programs interactive and user-friendly
- Provide feedback and results to users
- Form the foundation for more complex applications

3. The print() Function

The `print()` function is Python's built-in function for displaying output. It can:

- Display text, numbers, variables, and expressions
- Handle multiple arguments separated by commas
- Format output in various ways
- Add separators and line endings

4. Basic Output with print()

The `print()` function is used to display information to the user. It's versatile and can handle various data types and formatting options.

In [1]:
```python
# Basic Output Examples with print()

# Simple text output
print("Hello, World!")
print("Welcome to Python programming!")

# Printing different data types
print(42)           # Integer
print(3.14)         # Float
print(True)         # Boolean

# Using variables in print()
name = "Alice"
age = 30
city = "New York"

print("Name:", name)
print("Age:", age)
print("City:", city)

# Multiple values in one print statement
print("Name:", name, "Age:", age, "City:", city)

# Using f-strings for formatted output (modern approach)
print(f"Welcome, {name}! You are {age} years old and live in {city}.")

# Using .format() method (alternative approach)
print("Hello, {}! You are {} years old.".format(name, age))

# Print with custom separators and endings
print("Python", "is", "awesome", sep="-")  # Custom separator
print("This is line 1", end=" | ")         # Custom ending
print("This continues on the same line")
```

```
Hello, World!
Welcome to Python programming!
42
3.14
True
Name: Alice
Age: 30
City: New York
Name: Alice Age: 30 City: New York
Welcome, Alice! You are 30 years old and live in New York.
Hello, Alice! You are 30 years old.
Python-is-awesome
This is line 1 | This continues on the same line
```

5. Getting User Input with input()

The `input()` function allows you to get input from the user. It displays a prompt message and waits for the user to enter data. The input is always returned as a string.

In [3]:
```python
# Basic Input Examples with input()

# Getting user input with prompts
name = input("Enter your name: ")
age = input("Enter your age: ")
city = input("Enter your city: ")

# Display the collected information
print("\n=== User Information ===")
print("Hello,", name)
print("You are", age, "years old.")
print("You live in", city)

# Using f-strings for better formatting
print(f"\nNice to meet you, {name}!")
print(f"It's great to know you're {age} years old and from {city}.")

# Note: input() always returns a string
print(f"\nData types:")
print(f"name is of type: {type(name)}")
print(f"age is of type: {type(age)}")  # This will be <class 'str'>
print(f"city is of type: {type(city)}")
```

```
=== User Information ===
Hello, John
You are 25 years old.
You live in Pune

Nice to meet you, John!
It's great to know you're 25 years old and from Pune.

Data types:
name is of type: <class 'str'>
age is of type: <class 'str'>
city is of type: <class 'str'>
```

6. Type Conversion with Input

Since `input()` always returns a string, you need to convert it to the appropriate data type for calculations or comparisons. Python provides built-in functions for type conversion.

In [1]:
```
# Type Conversion Examples

# Getting numeric input and converting to integers
print("=== Age Calculator ===")
current_age = input("Enter your current age: ")
print(f"Input received: '{current_age}' (type: {type(current_age)})")

# Convert string to integer for calculations
age_as_int = int(current_age)
next_year_age = age_as_int + 1
print(f"Next year, you will be {next_year_age} years old.")
```

```
=== Age Calculator ===
Input received: '23' (type: <class 'str'>)
Next year, you will be 24 years old.
```

In [2]:
```
# Getting floating-point input
print("\n=== Temperature Converter ===")
celsius = input("Enter temperature in Celsius: ")
celsius_float = float(celsius)
fahrenheit = (celsius_float * 9/5) + 32
print(f"{celsius_float}°C is equal to {fahrenheit}°F")
```

```
=== Temperature Converter ===
34.0°C is equal to 93.2°F
34.0°C is equal to 93.2°F
```

In [3]:
```python
# Multiple numeric inputs in one line
print("\n=== Simple Calculator ===")
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

print(f"Sum: {num1} + {num2} = {num1 + num2}")
print(f"Difference: {num1} - {num2} = {num1 - num2}")
print(f"Product: {num1} × {num2} = {num1 * num2}")
if num2 != 0:
    print(f"Division: {num1} ÷ {num2} = {num1 / num2}")
else:
    print("Cannot divide by zero!")
```

```
=== Simple Calculator ===
Sum: 29.0 + 13.0 = 42.0
Difference: 29.0 - 13.0 = 16.0
Product: 29.0 × 13.0 = 377.0
Division: 29.0 ÷ 13.0 = 2.230769230769231
```

7. Advanced print() Features

The `print()` function has several useful parameters that give you more control over the output format.

In [5]:
```python
# Advanced print() Features

print("=== Custom Separators ===")
# Default separator is a space
print("apple", "banana", "cherry")

# Custom separator
print("apple", "banana", "cherry", sep=", ")
print("2024", "12", "25", sep="-")
print("Python", "is", "awesome", sep=" *** ")

print("\n=== Custom Line Endings ===")
# Default end is newline (\n)
print("This is line 1")
print("This is line 2")

# Custom ending
print("Loading", end="")
print(".", end="")
print(".", end="")
print(".", end=" ")
print("Complete!")

print("\n=== Printing to Different Outputs ===")
import sys

# Print to standard error instead of standard output
print("This is an error message", file=sys.stderr)

# Multiple arguments with formatting
fruits = ["apple", "banana", "cherry"]
print("Available fruits:", *fruits, sep="\n- ")

print("\n=== Printing Special Characters ===")
print("Quotes: \"Hello, World!\"")
print("New line: First line\nSecond line")
print("Tab: Column1\tColumn2\tColumn3")
print("Backslash: C:\\Users\\Documents")
print("Unicode: Python 🐍 Programming")
```

```
=== Custom Separators ===
apple banana cherry
apple, banana, cherry
2024-12-25
Python *** is *** awesome

=== Custom Line Endings ===
This is line 1
This is line 2
Loading... Complete!

=== Printing to Different Outputs ===
Available fruits:
- apple
- banana
- cherry

=== Printing Special Characters ===
Quotes: "Hello, World!"
New line: First line
Second line
Tab: Column1	Column2 Column3
Backslash: C:\Users\Documents
Unicode: Python 🐍 Programming
```

```
This is an error message
```

8. Error Handling with Input

When working with user input, it's important to handle potential errors, especially when converting data types.

In [6]:
```
# Error Handling with Input

print("=== Safe Input Conversion ===")

# Basic error handling for integer input
try:
    user_age = input("Enter your age (must be a number): ")
    age = int(user_age)
    print(f"Your age is: {age}")
    print(f"In 10 years, you'll be {age + 10}")
except ValueError:
    print(f"Error: '{user_age}' is not a valid number!")

print("\n=== Robust Input Function ===")

def get_integer_input(prompt):
    """Get integer input with error handling"""
    while True:
        try:
            user_input = input(prompt)
            return int(user_input)
        except ValueError:
            print("Please enter a valid integer!")

def get_float_input(prompt):
    """Get float input with error handling"""
    while True:
        try:
            user_input = input(prompt)
            return float(user_input)
        except ValueError:
            print("Please enter a valid number!")

# Example usage (commented out to avoid infinite loops in notebook)
# age = get_integer_input("Enter your age: ")
# height = get_float_input("Enter your height in meters: ")
# print(f"Age: {age}, Height: {height}m")

print("Robust input functions defined successfully!")
```

```
=== Safe Input Conversion ===
Your age is: 35
In 10 years, you'll be 45

=== Robust Input Function ===
Robust input functions defined successfully!
Your age is: 35
In 10 years, you'll be 45

=== Robust Input Function ===
Robust input functions defined successfully!
```

9. Practical Examples and Exercises

Let's put together what we've learned with some practical examples and exercises you can try.

In [7]:
```
# Practical Examples - Personal Information Form
print("=== Example 1: Personal Information Form ===")
# Create a simple form to collect user information
print("Please fill out this form:")
print("-" * 30)
first_name = input("First Name: ")
last_name = input("Last Name: ")
age = int(input("Age: "))
email = input("Email: ")
phone = input("Phone: ")

print("\n" + "="*40)
print("           PROFILE SUMMARY")
print("="*40)
print(f"Name: {first_name} {last_name}")
print(f"Age: {age} years old")
print(f"Email: {email}")
print(f"Phone: {phone}")
print("="*40)
```

```
=== Example 1: Personal Information Form ===
Please fill out this form:
------------------------------

========================================
           PROFILE SUMMARY
========================================
Name: John Smith
Age: 43 years old
Email: john.edu@email.com
Phone: +1 234 56789
========================================
```

In [ ]:
```python
print("\n=== Example 2: Recipe Ingredient Calculator ===")
# Calculate ingredients based on servings
recipe_name = input("Enter recipe name: ") # Cake
original_servings = int(input("Original recipe serves how many people? ")) # 3
desired_servings = int(input("How many people do you want to serve? ")) # 5

multiplier = desired_servings / original_servings

print(f"\n{recipe_name} - Adjusted for {desired_servings} people")
print(f"Multiply all ingredients by {multiplier:.2f}")

# Example with specific ingredients
flour_cups = float(input("Original flour (cups): ")) # 4
sugar_cups = float(input("Original sugar (cups): ")) # 2

print(f"\nAdjusted ingredients:")
print(f"Flour: {flour_cups * multiplier:.2f} cups")
print(f"Sugar: {sugar_cups * multiplier:.2f} cups")
```

```
=== Example 2: Recipe Ingredient Calculator ===

Cake - Adjusted for 5 people
Multiply all ingredients by 1.67

Adjusted ingredients:
Flour: 6.67 cups
Sugar: 3.33 cups
```

In [9]:
```python
print("\n=== Example 3: Simple Mad Libs Game ===")
print("Let's create a funny story! Fill in the blanks:")
noun1 = input("Enter a noun: ")
adjective1 = input("Enter an adjective: ")
verb1 = input("Enter a verb: ")
noun2 = input("Enter another noun: ")
color = input("Enter a color: ")

story = f"""
Once upon a time, there was a {adjective1} {noun1} who loved to {verb1}.
Every day, the {noun1} would {verb1} around the {color} {noun2}.
It was the most {adjective1} sight anyone had ever seen!
"""

print("\n=== YOUR STORY ===")
print(story)
```

```
=== Example 3: Simple Mad Libs Game ===
Let's create a funny story! Fill in the blanks:

=== YOUR STORY ===

Once upon a time, there was a brave Prince who loved to battle.
Every day, the Prince would battle around the red panipat.
It was the most brave sight anyone had ever seen!
```

Key Takeaways

- `print()` displays output to the console and supports various formatting options
- `input()` gets user input as a string - always remember to convert types when needed
- **Type conversion** functions like `int()`, `float()`, `str()` help convert between data types
- **Error handling** with try-except blocks makes your programs more robust
- **f-strings** provide a modern, readable way to format output with variables
- Always validate user input to prevent crashes and improve user experience

Practice Exercises

Try these exercises to strengthen your understanding:

1. **Personal Calculator**: Create a program that asks for two numbers and performs all basic arithmetic operations
2. **Unit Converter**: Build a converter that transforms units (e.g., feet to meters, Celsius to Fahrenheit)
3. **Survey Form**: Design a comprehensive survey that collects and displays user information nicely
4. **Number Guessing Game**: Create a simple game where the user guesses a number (we'll learn conditionals later!)
5. **Receipt Generator**: Make a program that takes item names and prices, then generates a formatted receipt

---

Course Information

Learn Python Programming from Scratch
Author: Prakash Ukhalkar
Topic: Python Fundamentals - Input and Output Operations

Built with ❤️ for the Python learning community