Python Day 1

Python Day 1:

Objective:

By the end of Day 1, students will have a thorough understanding of Python's basic syntax, including variables, data types, operators, and input/output. They will complete multiple mini-projects and exercises to reinforce these concepts.

1.1 What is Python?

Overview: - **Python** is a general-purpose, high-level language, known for its readability and ease of use. - **Interpreted**: Python code is executed line by line, making it beginner-friendly for learning. - **Dynamically Typed**: You don't need to specify the data type when declaring variables; Python automatically handles it.

Key Features of Python: - **Ease of learning**: Python syntax is designed to be simple and readable, making it an ideal language for beginners. - **Extensive standard libraries**: Python includes many libraries (modules) for handling common tasks such as file I/O, web development, data analysis, etc. - **Cross-platform**: Python code can run on different operating systems (Windows, Mac, Linux) without modification.

Discussion: - Real-world uses of Python (Google, Netflix, NASA use Python). - Why Python is a good starting language for beginners (simple syntax, readable code, fast to learn).

1.2 Setting Up the Python Environment

Steps for Installation:

1. **Download Python**:
   - Guide students through installing Python from python.org. Make sure to check "Add Python to PATH" during installation.
2. **Setting up IDE**:
   - Install **VSCode** or **PyCharm** and set them up for Python development. Show how to create new Python files and run code in the IDE.
3. **Creating Virtual Environments**:
   - Explain virtual environments and why they're useful for isolating project dependencies. Guide students to create a virtual environment for their project using:

```
python -m venv myenv
source myenv/bin/activate  # Mac/Linux
myenv\Scripts\activate  # Windows
```

Hands-On Exercise 1: - **Create a virtual environment** for your first Python project. - Verify your installation by running the command `python --version`.

1.3 Basic Syntax and Structure

Variables:

- Variables store data that you can reference and manipulate later. Python does not require explicit variable type declarations, making it more flexible but also requiring care to avoid type errors.
- **How to Declare a Variable**:

```
name = "Alice"
age = 30
height = 5.6
is_student = True
```

Key Points to Explain: - Variables are case-sensitive (`Name` is different from `name`). - You can assign different types of values (integer, float, string, boolean) to variables.

Data Types:

- **String**: Used for text data.

```
name = "John"
```

- **Integer**: Whole numbers.

```
age = 25
```

- **Float**: Numbers with decimals.

```
height = 5.9
```

- **Boolean**: Logical values (`True` or `False`).

```
is_student = True
```

Operators:

- **Arithmetic Operators**: Perform basic mathematical operations.
  - + (addition), - (subtraction), * (multiplication), / (division), % (modulus), ** (exponentiation), // (floor division).

  Example:

```
x = 10
y = 2
print(x + y)  # Outputs: 12
print(x / y)  # Outputs: 5.0
print(x % y)  # Outputs: 0 (remainder)
```

Comparison Operators:

- Used to compare two values and return `True` or `False`.
  - == (equal), != (not equal), >, <, >=, <=.

Mini-Project 1: Basic Operations Program (15 minutes): - **Goal**: Create a simple program that accepts two numbers from the user and performs basic operations (addition, subtraction, multiplication, division). - **Steps**: 1. Ask the user for two numbers. 2. Perform addition, subtraction, multiplication, and division. 3. Print the results.

Example Code:

```
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))

print(f"Addition: {num1 + num2}")
print(f"Subtraction: {num1 - num2}")
print(f"Multiplication: {num1 * num2}")
print(f"Division: {num1 / num2}")
```

Discussion: - Explain how Python handles different types of arithmetic operations, and introduce potential errors (like division by zero).

1.4 Input and Output

Why This Is Important:

Understanding how to take input from users and provide meaningful output is the first step to creating interactive programs.

Input from User:

- Use the `input()` function to take input from the user. The input is always returned as a string, so it must be converted to the appropriate data type if needed.

```
name = input("What is your name? ")
age = int(input("What is your age? "))  # Converting to integer
```

Output to User:

- Use the `print()` function to display data.

```
print("Hello", name)
print(f"You are {age} years old.")  # Using formatted strings
```

Practice Exercise 2: - Create a program that takes the user's name, age, and city, and then prints a welcome message.

Example Code:

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

print(f"Hello {name}, you are {age} years old and you live in {city}.")
```

Discussion: - Explain how to use type conversion (e.g., converting a string to an integer using `int()`).

1.5 Project 2: Simple Calculator

Goal:

Write a Python program that behaves like a basic calculator. The user should input two numbers, and the program should perform addition, subtraction, multiplication, and division.

Steps:

1. Ask the user to input two numbers.
2. Perform the four basic operations.
3. Display the result of each operation.

Why This Is Important: This project combines user input, variables, data types, and operators into a cohesive, interactive program.

Example Code:

```
# Simple Calculator Program

# Input
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

# Perform calculations
add_result = num1 + num2
sub_result = num1 - num2
mul_result = num1 * num2
div_result = num1 / num2

# Output results
print(f"Addition: {add_result}")
print(f"Subtraction: {sub_result}")
```

```
print(f"Multiplication: {mul_result}")
print(f"Division: {div_result}")
```

Practice Challenge: - Modify the program to handle edge cases (e.g., division by zero). Implement a check before performing division.

1.6 Logical and Comparison Operators

Logical Operators:

- **and**: Returns `True` if both conditions are true.
- **or**: Returns `True` if at least one condition is true.
- **not**: Reverses the result, returning `False` if the condition is true.

Example:

```
is_raining = True
is_warm = False

if is_raining and not is_warm:
    print("It's cold and raining.")
```

Comparison Operators:

- Comparison operators return `True` or `False` after comparing two values:

```
    a = 5
    b = 10
    print(a < b)  # Outputs: True
    print(a == b)  # Outputs: False
```

Practice Exercise 3: - Write a program that compares two numbers entered by the user and prints which one is larger, or if they are equal.

Example Code:

```
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))

if num1 > num2:
    print(f"{num1} is greater than {num2}")
elif num1 < num2:
    print(f"{num2} is greater than {num1}")
else:
    print("Both numbers are equal.")
```

1.7 Project 3: Temperature Converter

Goal:

Create a Python program that converts temperatures between Celsius and Fahrenheit.

Steps:

1. Ask the user if they want to convert from Celsius to Fahrenheit or Fahrenheit to Celsius.
2. Take the temperature input from the user.
3. Perform the conversion using the appropriate formula:
   - Celsius to Fahrenheit: `F = (C × 9/5) + 32`
   - Fahrenheit to Celsius: `C = (F - 32) × 5/9`
4. Display the converted temperature.

Why This Is Important: This project reinforces the concept of conditional statements (`if-else`), user input, and arithmetic operations. It also introduces basic logic that allows the program to "choose" the correct conversion formula based on user input.

Example Code:

```
# Temperature Converter

# Ask the user for conversion type
conversion_type = input("Convert (1) Celsius to Fahrenheit or (2) Fahrenheit
to Celsius: ")

# Perform the conversion
if conversion_type == "1":
    celsius = float(input("Enter temperature in Celsius: "))
    fahrenheit = (celsius * 9/5) + 32
    print(f"{celsius}°C is {fahrenheit}°F")
elif conversion_type == "2":
    fahrenheit = float(input("Enter temperature in Fahrenheit: "))
    celsius = (fahrenheit - 32) * 5/9
    print(f"{fahrenheit}°F is {celsius}°C")
else:
    print("Invalid input. Please enter 1 or 2.")
```

Practice Challenge: - Allow the user to perform multiple conversions in one run of the program by using a loop.

1.8 Additional Exercises and Recap (15 minutes)

Recap:

- **Key Concepts** covered:
  - Variables and data types (string, integer, float, boolean).

- Arithmetic, comparison, and logical operators.
- Input and output functions.
- Conditional logic (`if-else` statements).

Additional Mini-Projects for Practice:

1. **Odd/Even Number Checker**:
   - Write a program that takes a number from the user and prints whether it's odd or even.

   Example:

```python
number = int(input("Enter a number: "))
if number % 2 == 0:
    print(f"{number} is even.")
else:
    print(f"{number} is odd.")
```

2. **Simple Interest Calculator**:
   - Write a program that calculates the simple interest for a principal amount, rate of interest, and time.

   Example:

```python
principal = float(input("Enter the principal amount: "))
rate = float(input("Enter the rate of interest: "))
time = float(input("Enter the time (years): "))

interest = (principal * rate * time) / 100
print(f"Simple Interest: {interest}")
```

Homework:

- Modify the **temperature converter** to use a loop, allowing multiple conversions in a single run.
- Write a program that calculates the Body Mass Index (BMI) based on user input for height and weight.

Day 1 Summary:

By the end of **Day 1**, students will have: - **Set up their Python environment** and installed an IDE. - **Understood the basics of Python syntax** (variables, operators, data types). - Completed several **mini-projects**, including: - A **basic calculator**. - A **temperature converter**. - Other small, focused exercises.