Week 1: Introduction to Python Programming

1 Introduction to Python

Python is a widely used, high-level programming language, celebrated for its simplicity and ease of learning. It's highly versatile and can be used for various purposes including web development, data analysis, artificial intelligence, and automation.

1.1 Why Learn Python?

- **Readable**: Python code is clean and easy to understand.
- **Versatile**: Supports multiple programming paradigms such as object-oriented, functional, and procedural programming.
- **Large Community**: Extensive documentation and a large user base make it easy to find help and resources.
- **Library Support**: Python has libraries for virtually any task you can imagine (e.g., web frameworks, data manipulation, machine learning).

1.2 Basic Syntax

Python programs are written in plain text and executed line by line. Python files have the extension `.py`. Semicolons are not required, and indentation defines code blocks.

2 Hello World Program

The simplest Python program outputs the message ”Hello, World!” to the screen.

2.1 Objective

Understand basic Python syntax and output.

2.2 Code Example

```python
# Hello World in Python
print("Hello, World!")
```

The `print()` function is used to display text or data on the console.

3 Basic Arithmetic Operations

Python supports all basic arithmetic operations such as addition, subtraction, multiplication, and division.

3.1 Objective

Learn how to use arithmetic operators in Python.

3.2 Code Example

```python
# Basic arithmetic operations
a = 11
b = 5

print("Addition:", a + b)         # Output: 16
print("Subtraction:", a - b)      # Output: 6
print("Multiplication:", a * b)   # Output: 55
print("Division:", a / b)         # Output: 2.2
print("Integer Division:", a // b)         # Output: 2
print("Modulas:", a % b)         # Output: 1
```

Key Points:

- Python supports operators such as +, -, *, and /.
- Division always returns a float, even if both operands are integers.

4 Variables and Data Types

Variables store data in Python, and Python automatically infers their type based on the assigned value.

4.1 Objective

Understand how to define variables and work with different data types.

4.2 Code Example

```python
name = "Alice"   # String
age = 25         # Integer
height = 5.6     # Float

print(name, age, height)
```

Key Points:

- Python is dynamically typed, meaning you don't need to declare a variable's type.
- Common data types include `int`, `float`, and `str`.

5 User Input

Python provides the `input()` function to take input from the user. The input is always stored as a string.

5.1 Objective

Learn how to take user input and process it in Python.

5.2 Code Example

```python
name = input("Enter your name: ")
color = input("Enter your favorite color: ")

print(f"Hello {name}, your favorite color is {color}.")
```

Key Points:

- `input()` takes a prompt as an argument and always returns a string.
- Use `f-strings` to embed variables into strings.

Note: An f-string (formatted string literal) is a concise way to embed expressions inside string literals in Python. Introduced in Python 3.6, f-strings allow you to directly insert variables and expressions inside curly braces within a string, making string formatting simpler and more readable compared to older methods such as str.format() or string concatenation.

The syntax for f-strings is as follows:

```
f"Your string with {expression} or {variable}"
```

6 Simple Calculations with User Input

Convert user input to integers using `int()` to perform arithmetic operations.

6.1 Objective

Perform arithmetic operations based on user input.

6.2 Code Example

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

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

7 Temperature Converter

Convert temperatures from Celsius to Fahrenheit using the formula:

$$\text{Fahrenheit} = \left(\text{Celsius} \times \frac{9}{5}\right) + 32$$

7.1 Objective

Work with variables and apply formulas.

7.2 Code Example

```
celsius = 25
fahrenheit = (celsius * 9/5) + 32
print("Fahrenheit:", fahrenheit)
```

8 Distance Converter

Convert kilometers to miles using the conversion factor:

$$\text{Miles} = \text{Kilometers} \times 0.621371$$

8.1 Objective

Use a formula to convert units.

8.2 Code Example

```
kilometers = 5
miles = kilometers * 0.621371
print("Miles:", miles)
```

9 Working with Strings

Python provides many built-in functions for manipulating strings.

9.1 Objective

Perform string manipulations such as case conversions.

9.2 Code Example

```python
name = input("Enter your name: ")
print("Uppercase:", name.upper())
print("Lowercase:", name.lower())
```

Key Points:

- `upper()` converts a string to uppercase.
- `lower()` converts a string to lowercase.

10 Type Conversion

Convert between types using Python's built-in functions.

10.1 Objective

Learn how to convert data types explicitly in Python.

10.2 Code Example

```python
num_str = "123"
num_int = int(num_str)  # Converts string to integer
print(num_int)
```

11 Simple String Formatting

Use `f-strings` to embed variables into strings.

11.1 Objective

Format output using Python's f-string syntax.

11.2 Code Example

```python
name = "Alex"
age = 22
print(f"{name} is {age} years old.")
```

12 Summary and Recap

- **Variables and Data Types**: Python automatically infers variable types.
- **Arithmetic and Input**: Python supports all basic arithmetic operations.
- **String Manipulation**: Python offers several built-in string methods.
- **Type Conversion**: Use functions like `int()` and `float()` to convert between types.

13 Problems To Be Solved:

1. String Concatenation: • Write a Python program that takes two strings as input from the user and concatenates them. Print the resulting string.
2. Basic Arithmetic: • Write a program that takes two numbers as input and calculates the sum, difference, product, and quotient. Print each result.
3. Area of a Rectangle: • Create a program that takes the length and width of a rectangle as input from the user and calculates the area. Print the area.
4. String Length: • Write a Python program that takes a string as input from the user and prints the length of the string. [hint: use `len` function.]
5. List Indexing: • Write a program that creates a list of five numbers and prints the second and fourth numbers in the list using indexing. [Hint: Google it and learn how to do get value from a list.]
6. Basic Data Type Conversion: • Write a program that takes a string representation of a number (e.g., "5") from the user, converts it to an integer, and prints the square of that number. [Hint: use ** operator for calculating power of a value. For example 5 ** 2 = 25]
7. Formatting Output: • Create a program that takes a user's name and age as input and prints a formatted message, such as: "Hello, [name]! You are [age] years old."
8. using Built-in Functions: • Write a program that takes a list of five numbers and prints the minimum, maximum, and sum of the numbers using built-in functions.