Welcome to

Python for Programming and Data Science

Agenda

- Introduction to Python
- Setup environment
- Printing Output
- Arithmetic Operations
- Variable and Data Types
- Conditional Statements
- Practice problems

Why learn Python and Data Science?

- Most versatile programming language
- Easy to use syntax
- Used for Web Development, Software Engineering, Data Science, Machine Learning etc
- Competitive Edge in the Job Market
- Driving Business Growth through Data
- Growing Demand for Data Scientists and Python Developers
- AI revolution

Setup Python Environment

Cloud Options:

- Repl.it
- programiz
- Google Colab

Local setup

- PyCharm
- Visual Basic
- Jupyter Notebook

Print Function

- The print() function in Python is used to display information or output to the console. It allows you to present text, variables, or expressions, making it a fundamental tool for debugging, logging, and providing information about the program's execution.

Examples:
print(“Hello World”)
print(“Hello” + “World”)
print(“Hello\nWelcome”)

Input Function

- This function is commonly used to create interactive programs where the user can provide input or make choices, such as entering their name, selecting options, or providing numerical values. The input() function helps make Python programs more dynamic and user-friendly by enabling interaction between the program and the user during runtime.

Examples:
input(“What is your name?”)
input(“How old are you?”)
input(“Enter a number”)

Variables

- Variables in programming are used to store and manipulate data. They serve as named containers that hold values, such as numbers, strings, lists, or more complex objects. Variables provide a way to reference and work with data throughout a program without having to repeatedly write the actual values.

  Examples:

  name = “Wasi”

  age = 25

- Naming Convention:
  - Can’t start with symbols or numbers
  - Variables must be meaning
  - Camel Casing. (eg. firstName).
  - Dash naming. (eg. first_Name)

Variables

- Variables in programming are used to store and manipulate data. They serve as named containers that hold values, such as numbers, strings, lists, or more complex objects. Variables provide a way to reference and work with data throughout a program without having to repeatedly write the actual values.

  Examples:

  name = “Wasi”

  age = 25

- Naming Convention:
  - Can’t start with symbols or numbers
  - Variables must be meaning
  - Camel Casing. (eg. firstName).
  - Dash naming. (eg. first_Name)

Data Types

In Python, data types define the kind of data that a variable can hold. They specify what operations you can perform on the data and how the data is stored in memory. Python has several built-in data types, including:

- **Integers (int)**: Used to represent whole numbers, positive or negative, without a decimal point. For example: 5, -10, 1000.
- **Floating-Point Numbers (float)**: Used to represent numbers with decimal points or in scientific notation. For example: 3.14, 0.001, 2.5e-3.
- **Strings (str)**: Used to represent sequences of characters, like text. Strings are enclosed in either single (') or double (") quotes. For example: "Hello, Python", '12345'.
- **Booleans (bool)**: Used to represent binary values - either True or False. Booleans are often used in conditional statements and logic operations.

Understanding and correctly using data types is essential in Python because it determines how you can manipulate and work with your data in your code. Python is dynamically typed, which means you don't need to explicitly declare the data type of a variable; it's determined automatically based on the assigned value

Arithmetic Operations

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)
- Exponential (**)
- Assignment (=)

Conditional Statements

The most commonly used conditional statement in Python is the "if" statement, which checks if a given condition is true and executes a block of code if it is. Python also provides the "else" statement to provide an alternative block of code to execute when the condition is false, and the "elif" (short for "else if") statement to handle multiple conditions sequentially. These statements enable programmers to create dynamic and responsive code, making decisions within their programs based on various inputs or circumstances. Conditional statements are a fundamental building block of Python programming, enabling developers to write flexible and powerful applications.

Code block example:

```
if marks >= 50:
        print(“Pass”)
else:
        print(“Fail”)
```

Practice Problems

- Bill Split
- BMI calc
- Calculate Area of circle
- Grade Sheet