

Agenda for Lab

1. Introduction to Python

Python overview: Why it's essential for data analytics and AI

2. Basic Syntax of Python:

- Variables (Used to store data for processing)
- Conditional Statements (Enable decision-making in the code, Syntax uses if, elif, and else)
- Loops (Two main types: for loop for iterating over sequences, and while loop for conditions)
- Operators (Arithmetic, Comparison, and Logical)
- Functions

3. Introduction to Data Analytics Libraries:

- NumPy: Numerical computations and array operations
- Pandas: Data manipulation and analysis
- Scikit-learn: Machine learning and predictive modeling
- Matplotlib: Data visualization and plotting



Agenda for Lab

4. Data Visualization Using Matplotlib

Visualizations:

- Line charts
- Bar graphs
- Scatter plots

5. Manipulation

Pandas:

- Data cleaning: Handling missing values
- Imputation techniques:
- Reshaping and merging datasets

Preparing the dataset for ML modelling

6. Feature Engineering

Importance of feature engineering in improving model accuracy

Techniques:

- Encoding categorical variables
- Normalization and scaling
- Polynomial features for non-linear relationships

Agenda for Lab

7. Dimensionality Reduction (PCA) and Clustering (K-means)

Principal Component Analysis (PCA):

- Reduces data dimensions while retaining variance
- Useful for visualization and improving model efficiency

K-means Algorithm:

- Groups data into clusters
- Use cases: Customer segmentation, anomaly detection

8. Supervised Modelling Techniques:

- Linear Regression: Predict numerical values based on input features
- Support Vector Machine: Effective for classification tasks

9. Neural Networks for Custom Dataset Modelling

Design and train neural networks using TensorFlow/Keras



Introduction to Python

Outlines

- ☐ Objectives
- ☐ What is Python?
- ☐ Why Learn Python?
- ☐ Features of Python
- ☐ Comparing Python to Other Languages
- ☐ Applications of Python






Introduction to Python

Python is **an interpreted, object-oriented, high-level programming language with dynamic semantics**. Its high-level built-in data structures, combined with dynamic typing and binding, make it attractive for rapid application development and scripting.

Python's simple syntax emphasizes readability, reducing program maintenance costs. It supports modularity and code reuse with modules and packages.








Introduction to Python

Objectives

- ❑  Understand what Python is and its key features.
- ❑  Explore Python's applications across various domains.
- ❑  Learn the step-by-step process for data analysis and present data in graphical format for the data trend understanding
- ❑  Run Python programs
- ❑  Conduct **Experiments** effectively

Introduction to Python

What is Python?

-  Python is a **high-level, interpreted programming language**.
-  Created by **Guido van Rossum** in 1991.
-  Known for its **simplicity, readability**, and versatility.
-  Widely used in areas like web development, data science, AI, and more.
- Key characteristics:
 -  Object-oriented programming support.
 -  Extensive libraries and frameworks.
 -  Active community for troubleshooting and resources.

Introduction to Python

Why Learn Python?

-  **Ease of Learning:**

Simple syntax that lowers the entry barrier for beginners.

-  **Cross-Platform Support:**

Python programs can run on various operating systems.

-  **Growing Demand:**

Python is among the top programming languages in demand.

-  **Versatility:**







Applicable to diverse domains like web development, automation, and scientific computing.

-  **Job Opportunities:**

Numerous career paths in Python development.

Introduction to Python

Features of Python

-  **Readable Syntax:** Designed for clarity and simplicity.
-  **Interpreted Language:** No need for separate compilation steps.
-  **Dynamic Typing:** Variables do not need explicit declaration.
-  **Open Source and Free:** Community-driven development with free access.
-  **Portable:** Write once, run anywhere.
-  **Extensive Standard Library:** Pre-built modules for common tasks.

Introduction to Python

Comparing Python to Other Languages

•Java:

- 🐍 Python programs run slower but are developed faster.
- 📄 Python code is 3-5 times shorter than Java.

•JavaScript:

- 📦 Python supports object-oriented programming and large-scale development.
- 📄 JavaScript focuses on functions and variables without class definitions.

•Perl:








- 🧰 Perl excels in Unix scripting; Python emphasizes readable and maintainable code.
- 🏗️ Python supports object-oriented and data-structure design methodologies.

•C++:

- ✂️ Python code is 5-10 times shorter than C++.
- 🔧 Serves as a glue language, combining C++ components efficiently

Introduction to Python

Applications of Python

-  **Data Analysis and Visualization:** Pandas, Matplotlib, Seaborn.
-  **Web Development:** Frameworks like Flask and Django.
-  **Machine Learning and AI:** TensorFlow, PyTorch, and Scikit-learn.
-  **Automation:** Task automation and scripting with libraries like Selenium.
-  **Game Development:** Libraries such as PyGame for 2D games.
-  **Internet of Things (IoT):** Used in embedded systems.
-  **Scientific Computing:** SciPy and NumPy for numerical computations.

Introduction to Python


Practical Session: Hands-on Learning

- Exercise 1:  Create a Python script to calculate the sum of two numbers.

```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
print("The sum is:", a + b)
```

Introduction to Python

Practical Session: Hands-on Learning

- Exercise 2:  Write a program to find the square of a numbers.

```
num = int(input("Enter a number: "))  
print("The square is:", num ** 2)
```

Introduction to Python

Practical Session: Hands-on Learning

- **Exercise 3:** Explore the basic usage of lists.

```
my_list = [1, 2, 3, 4]
print("Original List:", my_list)
my_list.append(5)
print("Updated List:", my_list)
```