PYTHON FOR AI STARTER TRACK – 2025 EDITION

" Your first step into the world of programming, data, and AI"

Market Demand Note

Python is the most popular programming language for data science, AI, and automation because of its simplicity, flexibility, and massive library support. From machine learning to APIs, Python is the backbone of modern data-driven solutions. Mastering Python basics and its data libraries opens doors to data analysis, AI model building, and automation.

Duration: 6 weeks | **Mode**: Online/Offline

📘 Python for Data & AI Track — 2025 Edition

Table of Contents

Python Starter Course – Table of Contents

- 4.5 Merging & Grouping Operations

Module 5: Data Visualization

- 5.1 Plotting with Matplotlib (Line, Bar, Histogram, Scatter)
- 5.2 Customizing Charts & Subplots
- 5.3 Introduction to Seaborn (Distplots, Boxplots, Heatmaps)
- 5.4 Inline Visualization in Jupyter

Module 6: Final Project

- 6.1 Project Selection & Dataset Familiarization
- 6.2 Data Cleaning & Exploration
- 6.3 Analysis & Statistical Summaries
- 6.4 Visualization & Storytelling
- 6.5 Final Notebook Submission & Presentation

Course Tools & Resources

- Platform: Jupyter Notebook (VS code recommended)
- Libraries: Python 3, Pandas, NumPy, Matplotlib, Seaborn
- Data Sources: Public datasets (Kaggle, Gov. of India portals, GitHub)
- Version Control: Basic introduction to GitHub for saving/sharing code
- Assessment: Weekly assignments, in-class coding labs, capstone project

Detailed Content

Week 1: Python Fundamentals (10 Hours)

Learning Objectives:
Set up Python, understand variables, data types, and basic operations, and start thinking like a programmer.

Topics Covered:

- Installing Python & Jupyter: Downloading Python 3.x via Anaconda/Miniconda; navigating Jupyter Notebook/Lab environments; running cells, markdown basics.
- Variables: Naming rules, assignment, dynamic typing.
- Data Types: Integers, floats, strings, booleans; type conversion (int(), str(), float()).
- Operators: Arithmetic (+, -, *, /, //, %, **), comparison (==, !=, >, <, etc.), logical (and, or, not), membership (in, not in).
- Print Statements: Basic printing, f-strings, formatting output.
- User Input: input() function, string manipulation, basic validation.

Practical Exercises:

- Exercise 1: Write code that asks the user for their name, age, and city, then prints a formatted greeting.
- Exercise 2: Calculate simple and compound interest, given user inputs.
- Exercise 3: Write a temperature converter (Celsius ↔ Fahrenheit).

Lab Work:

- Hands-on sessions: Working in Jupyter Notebook; executing code, saving, sharing notebooks.
- Debugging practice: Reading error messages, fixing syntax errors.
- Peer review: Pair up and review each other’s notebooks.

Week 2: Control Structures & Functions (10 Hours)

Learning Objectives:
Control program flow with conditions and loops, define reusable functions, and import Python libraries.

Topics Covered:

- Conditional Statements: if, elif, else blocks; nested conditions.
- Loops: for loops (range, iterating over sequences), while loops (break, continue).
- Functions: Defining (def), parameters, return values, scope (local vs. global), docstrings.
- Python Packages: Importing libraries (math, random), pip install basics.

- Basic Error Handling: try/except for simple user input validation.

Practical Exercises:

- Exercise 1: FizzBuzz problem (print numbers, replace multiples of 3/5 with “Fizz”/“Buzz”).
- Exercise 2: Number guessing game (random module).
- Exercise 3: Write a function to check if a number is prime.

Lab Work:

- Live coding demos: Instructor solves a problem step by step, students follow along.
- Code refactoring: Rewrite previous week’s code with functions.
- Homework: Build simple calculators (tip, BMI) using functions and conditionals.

Week 3: Data Structures & File Handling (10 Hours)

Learning Objectives:
Master Python’s core data structures and read/write data from files.

Topics Covered:

- Lists: Creation, indexing, slicing, append, insert, remove, list comprehension.
- Tuples: Immutability, tuple unpacking, use cases.
- Sets: Adding, removing, unions, intersections, differences.
- Dictionaries: Key-value pairs, CRUD operations, iterations, comprehensions.
- File I/O: open(), read(), write(), with (context managers), file modes.
- JSON & CSV: Reading/writing structured data, json and csv modules.

Practical Exercises:

- Exercise 1: Store user data (name, age, city) in a list of dictionaries and write to a JSON file.
- Exercise 2: Read a CSV of product prices, calculate total, average, and write results to a new file.
- Exercise 3: Remove duplicates from a list using a set.

Lab Work:

- Data manipulation: Clean and filter data from a CSV in Jupyter.
- Group project: Build a small “contact book” app that saves and loads entries from a file.

Week 4: Data Analysis with Pandas & NumPy (12 Hours)

Learning Objectives:
Import, clean, filter, and analyze tabular data with Pandas and NumPy.

Topics Covered:

- NumPy Basics: Arrays, array operations, broadcasting.
- Pandas Basics: Series, DataFrames, reading CSV/JSON/Excel, indexing, merging.
- Data Cleaning: Handling missing data, duplicate removal, data type conversion.
- Filtering & Sorting: Boolean indexing, query, sort_values.
- Basic Statistics: Mean, median, mode, std, describe(), groupby.

Practical Exercises:

- Exercise 1: Load a dataset (e.g., Bangalore weather, mobile phone prices), clean missing values, and filter by condition.
- Exercise 2: Calculate summary statistics for numeric columns, create groups by category.
- Exercise 3: Merge two datasets (e.g., products and sales), aggregate by group.

Lab Work:

- EDA (Exploratory Data Analysis): Students load, clean, and summarize a real dataset in Jupyter.
- Challenge: Find outliers or unusual patterns in the data.

Week 5: Data Visualization (10 Hours)

Learning Objectives:
Create and customize charts to communicate data insights.

Topics Covered:

- Matplotlib: Line plots, bar charts, histograms, scatter plots, titles, labels, legends.

- Customization: Colors, styles, figure size, subplots.
- Seaborn: Distplot, boxplot, pairplot, heatmaps.
- Inline Visualization: Show plots in Jupyter Notebook, export to file.

Practical Exercises:

- Exercise 1: Plot trends in Bangalore temperature data (time series line chart).
- Exercise 2: Compare product price distributions (histogram and boxplot).
- Exercise 3: Visualize correlation between variables (scatter matrix, heatmap).

Lab Work:

- Storytelling with Data: Write a Markdown summary of findings, supported by plots.
- Portfolio Piece: Students create a PDF/HTML report with code and visuals.

Week 6: Final Project (8 Hours)

Learning Objectives:
Combine all skills to analyze a real-world dataset and present insights.

Topics Covered:

- Project Scoping: Choose a dataset, define questions.
- Data Cleaning & Analysis: Apply Pandas, NumPy skills.
- Visualization: Generate compelling charts for key findings.
- Presentation: Notebook structure (Code + Markdown), clear explanations.

Project Ideas:

- Bangalore Air Quality: Trends, correlations with weather.
- E-Commerce Sales: Product categories, best/worst sellers.
- Student Performance: Pass/fail rates, subject comparisons.
- Restaurant Reviews: Sentiment trends, rating distribution.

Deliverables:

- Jupyter Notebook: Well-commented, with clear sections for code, output, and insights.
- 1-Page Summary: Key findings, supported by visuals.

- Short Presentation: 5-minute demo to class (what you did, what you found).

Assessment:

- Code Quality: Clean, efficient, commented.
- Analysis: Thoughtful, correct methods used.
- Visuals: Clear, informative, well-labeled.
- Communication: Explains findings in plain language.

Course Tools & Setup

- Platform: Jupyter Notebook/Lab on Ubuntu/Linux (or Windows/macOS).
- Data Sources: Public datasets (Kaggle, government portals, GitHub).
- Version Control: Encourage GitHub for saving and sharing projects.
- Open Source: All tools and libraries included are free (Python, Pandas, NumPy, Matplotlib, Seaborn).

Course Outcomes

- Solid Python foundation: Ready for more advanced topics (AI, data engineering, web dev).
- Portfolio projects: Real datasets, real analysis—great for resumes and interviews.
- Presentation skills: Learn to communicate technical work clearly.
- Confidence: Build, debug, and share working code.