 **Simple Topics** (Days 1–4, 7, 9): **2–3 hours per day**  
These cover foundational concepts and lighter topics like syntax, functions, and modules.

 **Moderate Topics** (Days 5–6, 8, 10–14, 16–19): **3–4 hours per day**  
These include data structures, OOP, file handling, automation, APIs, and testing.

 **Advanced and Project Days** (Days 11–12, 15, 20–25): **4–6 hours per day**  
Advanced topics and capstone projects require more time for understanding and practice.

**Python Full Course: Day-Wise Learning Plan**

**Day 1: Introduction to Python**

* What is Python?
* Installing Python and Setting Up the Environment
* Writing Your First Python Script
* Understanding Python's Syntax and Code Structure

**Day 2: Python Basics**

* Variables and Data Types
* Input and Output
* Type Conversion
* Comments and Documentation

**Day 3: Control Flow**

* Conditional Statements (if, elif, else)
* Loops (for, while)
* Break and Continue
* Nested Loops

**Day 4: Functions**

* Defining Functions
* Function Arguments and Return Values
* Default and Keyword Arguments
* Lambda Functions
* Scope (Global vs Local Variables)

**Day 5: Data Structures (Part 1)**

* Lists: Creating, Accessing, and Modifying
* Tuples: Immutable Collections
* Sets: Unique Elements Only
* Basic Operations on Lists, Tuples, and Sets

**Day 6: Data Structures (Part 2)**

* Dictionaries: Key-Value Pairs
* Dictionary Methods: get(), update(), etc.
* List Comprehensions
* Working with Strings and String Methods

**Day 7: Object-Oriented Programming (OOP) (Part 1)**

* Classes and Objects
* Attributes and Methods
* Instance and Class-Level Attributes

**Day 8: Object-Oriented Programming (OOP) (Part 2)**

* Inheritance
* Polymorphism
* Encapsulation (Getters and Setters)

**Day 9: Modules and Packages**

* Importing Built-in Modules (math, os, etc.)
* Creating Custom Modules
* Using pip to Install Packages
* Setting Up Virtual Environments

**Day 10: File Handling**

* Reading and Writing Files
* Working with CSV and JSON Files
* Error Handling (try, except, finally)

**Day 11: Advanced Topics (Part 1)**

* Decorators
* Generators
* Context Managers (with Statement)

**Day 12: Advanced Topics (Part 2)**

* Regular Expressions with re
* Example: Email Validation

**Day 13: Python for Data Handling (Part 1)**

* Introduction to numpy and pandas
* Manipulating Arrays with numpy
* Creating and Using DataFrames in pandas

**Day 14: Python for Data Handling (Part 2)**

* Filtering and Sorting Data with pandas
* Grouping and Aggregation
* Plotting with matplotlib and seaborn

**Day 15: Web Development with Python**

* Introduction to Flask and Django
* Building a Simple Web App with Flask
* Handling Requests and Responses

**Day 16: Python for Automation (Part 1)**

* Automating Tasks with os and shutil
* Automating Email with smtplib

**Day 17: Python for Automation (Part 2)**

* Using selenium for Web Automation
* Example: Log into a Website and Scrape Data

**Day 18: Python and APIs**

* Consuming APIs with requests
* Parsing JSON Responses
* Building Your Own API with Flask

**Day 19: Testing in Python**

* Writing Unit Tests with unittest and pytest
* Debugging Tips
* Logging with the logging Module

**Day 20: Advanced Python Concepts**

* Working with Threads and Multiprocessing
* Asynchronous Programming with asyncio
* Performance Optimization Tools (cProfile, timeit)

**Day 21-25: Capstone Projects**

* Build a **To-Do App** using Flask and a Database
* Create a **Web Scraper** to Extract Data from Websites
* Develop a **REST API** for a Blog or E-commerce Application
* Analyze a **Dataset** with pandas and Visualize Insights
* Automate **Daily Tasks** such as File Organization or Email Notifications

**Python Full Course: Comprehensive Guide**

**1. Introduction to Python**

* **What is Python?**  
  Python is a high-level, interpreted programming language known for its simplicity and readability. It is widely used for web development, data analysis, artificial intelligence, scientific computing, and more.
* **Installing Python and Setting Up the Environment**  
  Install Python from the official [Python website](https://www.python.org/downloads/). Learn how to set up environments using:
  + IDLE (Python’s Integrated Development and Learning Environment)
  + VS Code with Python extension
  + Jupyter Notebook for interactive computing
* **Writing Your First Python Script**  
  Create and run a simple hello\_world.py script to print "Hello, World!".
* **Understanding Python's Syntax and Code Structure**  
  Learn about indentation, statements, and basic program flow.

**2. Python Basics**

* **Variables and Data Types**
  + Integers, Floats, Strings, Booleans
  + Mutable (lists, dictionaries) vs Immutable (tuples, strings) types
* **Input and Output**
  + Using input() for user input
  + Using print() for output
  + String formatting methods: f-strings, format(), %
* **Type Conversion**
  + Implicit and explicit type conversion (e.g., int(), str(), float())
* **Comments and Documentation**
  + Single-line comments (#)
  + Multi-line comments (''' or """)

**3. Control Flow**

* **Conditional Statements**
  + Syntax of if, elif, else
  + Nested conditions
  + Example: Determine if a number is positive, negative, or zero.
* **Loops**
  + for loop: Iterating over ranges, lists, dictionaries
  + while loop: Executing as long as a condition is true
* **Break and Continue**
  + Interrupt loop execution with break
  + Skip iterations using continue
* **Nested Loops**
  + Using loops inside other loops for complex iteration tasks

**4. Functions**

* **Defining Functions**
  + Syntax: def function\_name(parameters):
  + Example: Create a function to calculate factorials.
* **Function Arguments and Return Values**
  + Positional and keyword arguments
  + Returning multiple values
* **Default and Keyword Arguments**
  + Setting default values for parameters
  + Using keyword arguments for clarity
* **Lambda Functions**
  + Anonymous functions with lambda keyword
  + Example: Sorting a list of tuples by the second element
* **Scope**
  + Local vs global variables
  + Using global and nonlocal keywords

**5. Data Structures**

* **Lists**
  + Creating, accessing, modifying, and slicing lists
  + List methods: append(), remove(), sort(), etc.
* **Tuples**
  + Immutable collections
  + Unpacking tuples
* **Sets**
  + Unique elements only
  + Set operations: union, intersection, difference
* **Dictionaries**
  + Key-value pairs
  + Iterating over keys and values
  + Dictionary methods: get(), update(), pop(), etc.
* **List Comprehensions**
  + Compact syntax for generating lists
  + Example: [x\*\*2 for x in range(10) if x % 2 == 0]
* **Working with Strings**
  + String methods: split(), join(), replace()
  + Formatting and f-strings

**6. Object-Oriented Programming (OOP)**

* **Classes and Objects**
  + Defining classes and creating objects
  + Example: Create a Car class with attributes like brand and speed.
* **Attributes and Methods**
  + Instance and class-level attributes
  + Instance and static methods
* **Inheritance**
  + Deriving classes from a parent class
  + Using super() to access parent class methods
* **Polymorphism**
  + Overloading methods
  + Using common methods across different classes
* **Encapsulation**
  + Private and protected members
  + Using getter and setter methods

**7. Modules and Packages**

* **Importing Modules**
  + Built-in modules like math, os, sys
  + Importing specific functions or using aliases
* **Creating Custom Modules**
  + Writing and importing your own Python files
* **Using pip to Install Packages**
  + Installing and managing external libraries
  + Example: Install requests for HTTP operations
* **Virtual Environments**
  + Creating isolated Python environments using venv

**8. File Handling**

* **Reading and Writing Files**
  + Open files in different modes (r, w, a)
  + Reading file content using read(), readline(), and readlines()
  + Writing to files using write()
* **Working with CSV and JSON Files**
  + Use csv module for CSV files
  + Parse and write JSON using json module
* **Error Handling**
  + Using try, except, and finally blocks
  + Catching specific exceptions

**9. Advanced Topics**

* **Decorators**
  + Creating and applying decorators
  + Example: Logging function calls
* **Generators**
  + Using yield to create generator functions
  + Example: Generate an infinite sequence of numbers
* **Context Managers**
  + Using with for resource management
  + Example: File handling with with open()
* **Regular Expressions**
  + Pattern matching with re module
  + Example: Validate email addresses

**10. Python for Data Handling**

* **Introduction to numpy and pandas**
  + Array manipulation with numpy
  + Dataframes and Series in pandas
* **Data Manipulation with pandas**
  + Filtering and sorting data
  + Grouping and aggregation
* **Plotting with matplotlib and seaborn**
  + Create line, bar, and scatter plots
  + Visualize data distributions with seaborn

**11. Web Development with Python**

* **Basics of Web Frameworks**
  + Introduction to Flask and Django
* **Building a Simple Web App**
  + Using Flask to create routes and templates
  + Example: Build a "Hello, World!" web application
* **Handling Requests and Responses**
  + Processing form data
  + Returning HTML or JSON responses

**12. Python for Automation**

* **Automating Tasks with os and shutil**
  + Perform file operations programmatically
  + Example: Organize files into folders
* **Using selenium for Web Automation**
  + Automate browser tasks
  + Example: Log into a website and scrape data
* **Automating Email with smtplib**
  + Sending emails programmatically
  + Example: Send automated reports

**13. Python and APIs**

* **Consuming APIs with requests**
  + Send HTTP GET and POST requests
  + Example: Fetch weather data from an API
* **Parsing JSON Responses**
  + Extract data from API responses
* **Building Your Own API with Flask**
  + Create RESTful APIs
  + Example: A simple to-do API

**14. Testing in Python**

* **Writing Unit Tests**
  + Using unittest and pytest frameworks
  + Example: Test a simple calculator function
* **Debugging Tips**
  + Using pdb for debugging
  + Debugging in IDEs like VS Code
* **Logging**
  + Implementing logging with the logging module
  + Setting logging levels (DEBUG, INFO, WARNING, ERROR)

**15. Advanced Python Concepts**

* **Working with Threads and Multiprocessing**
  + Creating and managing threads
  + Parallel processing with multiprocessing
* **Asynchronous Programming**
  + Using asyncio for asynchronous tasks
  + Example: Fetch multiple URLs concurrently
* **Performance Optimization**
  + Profiling and optimizing Python code
  + Tools like cProfile and timeit

**16. Capstone Projects**

* Build a **To-Do App** using Flask and a database
* Create a **Web Scraper** to extract data from websites
* Develop a **REST API** for a blog or e-commerce application
* Analyze a **Dataset** with pandas and visualize insights
* Automate **Daily Tasks** such as file organization or email notifications

**Basic Syntax and Variables**

1. Write a program to print "Hello, World!".
2. Create a variable to store your name and print it.
3. Write a program to swap two numbers using a temporary variable.
4. Write a program to calculate the area of a rectangle (length × breadth).

**Input and Output**

1. Write a program to take input from the user and print it.
2. Ask the user for their age and print whether they are a minor, an adult, or a senior citizen.

**Conditional Statements**

1. Write a program to check if a number is even or odd.
2. Write a program to find the largest of three numbers.
3. Write a program to check if a number is positive, negative, or zero.

**Loops**

1. Write a program to print numbers from 1 to 10 using a for loop.
2. Write a program to print the multiplication table of a given number.
3. Write a program to calculate the sum of all numbers in a list using a while loop.

**Functions**

1. Write a function to find the square of a number.
2. Write a function to return the maximum of two numbers.
3. Write a program with a function that takes a string as input and returns it in reverse.

**Lists**

1. Write a program to create a list of 5 numbers and print the sum of the elements.
2. Write a program to find the largest number in a list.
3. Write a program to count the number of vowels in a given string.

**Strings**

1. Write a program to check if a string is a palindrome.
2. Write a program to replace all spaces in a string with underscores.
3. Write a program to count the frequency of each character in a string.

**Dictionaries**

1. Write a program to create a dictionary with keys as numbers (1 to 5) and their squares as values.
2. Write a program to retrieve a value from a dictionary using a key.

**Miscellaneous**

1. Write a program to generate a random number between 1 and 100.
2. Write a program to calculate the factorial of a number using recursion.

**File Handling**

1. Write a program to read a text file and count the number of lines, words, and characters in it.
2. Write a program to write a list of numbers to a file and then read and print them.

**Object-Oriented Programming**

1. Create a Person class with attributes name and age. Add a method to display this information.
2. Write a class Rectangle with methods to calculate area and perimeter. Create objects and demonstrate its usage.
3. Implement a class hierarchy for vehicles, where Car and Bike inherit from a base class Vehicle.

**Data Structures**

1. Write a program to merge two dictionaries.
2. Implement a stack using a list and add push, pop, and peek operations.
3. Implement a queue using a list with enqueue and dequeue operations.

**Advanced Loops and Logic**

1. Write a program to generate the Fibonacci sequence up to a given number.
2. Write a program to check if a number is a prime number.
3. Write a program to find all the prime numbers between two given numbers.

**Error Handling**

1. Write a program to divide two numbers and handle division by zero using exception handling.
2. Write a program to handle a file not found exception.

**List and String Manipulations**

1. Write a program to find the second largest number in a list.
2. Write a program to remove duplicate elements from a list.
3. Write a program to check if two strings are anagrams of each other.

**Algorithms**

1. Write a program to implement bubble sort.
2. Write a program to implement binary search.
3. Write a program to find the greatest common divisor (GCD) of two numbers using the Euclidean algorithm.

**Libraries and Modules**

1. Write a program to fetch the current date and time using the datetime module.
2. Write a program to create a bar chart using the matplotlib library.

**Recursion and Functional Programming**

1. Write a program to find the factorial of a number using recursion.
2. Write a program to calculate the sum of digits of a number using recursion.
3. Write a program using map, filter, and reduce to process a list of numbers.

**Miscellaneous**

1. Write a program to implement a basic calculator using functions for add, subtract, multiply, and divide.
2. Write a program to shuffle a deck of cards and draw 5 random cards.
3. Write a program to count the frequency of words in a text file.