**Python Learning Plan: 30 Days to Mastery**

Welcome to your 30-day journey to mastering Python! Below is a structured plan to help you learn Python comprehensively. Each day, you will focus on specific topics, practice with problems, and gradually build your skills. Let's get started!

**Day 1: Basics**

**Problem:** Write a program that prints "Hello, World!" to the console.

**What to Learn:** You need to understand basic Python syntax and how to print to the console. **Resources:**

* [Python.org Official Python Tutorial](https://docs.python.org/3/tutorial/index.html)
* [Codecademy's Python Course](https://www.codecademy.com/learn/learn-python-3)
* [W3Schools Python Tutorial](https://www.w3schools.com/python/)

**Problem:** Create a program that calculates the sum of two numbers entered by the user.

**What to Learn:** Practice basic input and output in Python.

**Resources:** Refer to the resources mentioned earlier.

**Problem:** Write a Python script that converts a temperature in Celsius to Fahrenheit using the formula **F = (C \* 9/5) + 32**.

**What to Learn:** Basic mathematical operations and conversions in Python.

**Resources:** Refer to the resources mentioned earlier.

**Day 2: Control Flow**

**Problem:** Implement a guessing game. **What to Learn:** You need to know how to use conditionals (if statements) and implement loops (while or for). **Resources:**

* [Control Flow in Python](https://docs.python.org/3/tutorial/controlflow.html)
* [Python Conditional Statements](https://www.programiz.com/python-programming/if-elif-else)
* [Python Loops](https://www.learnpython.org/en/Loops)

**Problem:** Create a program that calculates the factorial of a number entered by the user. **What to Learn:** Learn about loops and mathematical operations in Python. **Resources:** Refer to the resources mentioned earlier.

**Problem:** Write a script that determines whether a year is a leap year or not. **What to Learn:** Work with conditionals and boolean expressions in Python. **Resources:** Refer to the resources mentioned earlier.

**Day 3: Functions and Modules**

**Problem:** Define a function to calculate the factorial of a number. **What to Learn:** Learn how to define and use functions in Python. **Resources:**

* [Python Functions](https://docs.python.org/3/tutorial/controlflow.html#defining-functions)
* [Python Function Arguments](https://www.programiz.com/python-programming/function-argument)

**Problem:** Define functions to find the maximum and minimum of three numbers. **What to Learn:** Practice defining and calling functions with multiple arguments. **Resources:** Refer to the resources mentioned earlier.

**Problem:** Import a Python module and use functions from it. **What to Learn:** Understand modules and how to import and use them. **Resources:**

* [Python Modules](https://docs.python.org/3/tutorial/modules.html)
* [Python Built-in Modules](https://docs.python.org/3/library/index.html)

**Day 4: Lists and Loops**

**Problem:** Write a program to find all prime numbers in a given range. **What to Learn:** Learn about lists, loops (for and while), and basic mathematical operations. **Resources:**

* [Python Lists](https://docs.python.org/3/tutorial/introduction.html#lists)
* [Python For Loops](https://www.learnpython.org/en/Loops)
* [Prime Number Generation in Python](https://www.geeksforgeeks.org/python-program-to-print-all-prime-numbers-in-an-interval/)

**Problem:** Create a program that finds the factorial of all numbers from 1 to 10 and stores the results in a list. **What to Learn:** Practice using loops and lists together. **Resources:** Refer to the resources mentioned earlier.

**Problem:** Write a script that calculates the sum of all multiples of 3 and 5 below 100. **What to Learn:** Learn to use loops, conditionals, and basic arithmetic operations. **Resources:** Refer to the resources mentioned earlier.

**Day 5: String Manipulation**

**Problem:** Count the number of vowels in a string. **What to Learn:** Understand string manipulation and basic string methods in Python. **Resources:**

* [Python Strings](https://docs.python.org/3/tutorial/introduction.html#strings)
* [Python String Methods](https://docs.python.org/3/library/stdtypes.html#string-methods)
* [Counting Vowels in Python](https://www.geeksforgeeks.org/python-program-to-count-number-of-vowels-using-sets-in-given-string/)

**Problem:** Write a program to reverse a string. **What to Learn:** Practice string manipulation and slicing in Python. **Resources:** Refer to the resources mentioned earlier.

**Problem:** Create a script that checks if a given string is a palindrome. **What to Learn:** Work with string indexing and comparisons. **Resources:** Refer to the resources mentioned earlier.

**Day 6: Dictionaries and Sets**

**Problem:** Write a program to count the frequency of each word in a given text. **What to Learn:** Learn about dictionaries, sets, and basic text parsing. **Resources:**

* [Python Dictionaries](https://docs.python.org/3/tutorial/datastructures.html#dictionaries)
* [Python Sets](https://docs.python.org/3/tutorial/datastructures.html#sets)
* [Text Parsing in Python](https://docs.python.org/3/library/re.html)

**Problem:** Create a script that finds and lists the unique words in a text file. **What to Learn:** Explore dictionary and set operations. **Resources:** Refer to the resources mentioned earlier.

**Problem:** Write a program to find common elements between two lists using sets. **What to Learn:** Understand set operations and list manipulation. **Resources:** Refer to the resources mentioned earlier.

**Day 7: Project Day**

**Problems:**

* Build a to-do list application.
* Create a basic calculator program.
* Generate a multiplication table. **What to Learn:** Apply what you've learned so far to create practical programs. **Resources:** Refer to the resources mentioned in the previous days for guidance and inspiration. You can also search for specific tutorials on building these projects.

**Day 8: Classes and Objects**

**Problem:** Create a Python class representing a bank account with methods for deposit, withdrawal, and balance inquiry. **What to Learn:** Learn how to define and use classes and objects in Python. **Resources:**

* [Python Classes and Objects](https://docs.python.org/3/tutorial/classes.html)
* [Python Object-Oriented Programming (OOP)](https://realpython.com/python3-object-oriented-programming/)

**Problem:** Implement a class that models a basic geometric shape (e.g., Circle) with methods to calculate its area and perimeter. **What to Learn:** Practice creating and working with classes, methods, and properties. **Resources:**

* [Object-Oriented Programming (OOP) in Python](https://www.datacamp.com/community/tutorials/python-object-oriented-programming/)
* [Python Classes and Objects - GeeksforGeeks](https://www.geeksforgeeks.org/python-classes-and-objects/)

**Problem:** Write a program that uses a class to represent a student with attributes like name, age, and grade. **What to Learn:** Further practice creating and using classes, constructors, and attributes. **Resources:**

* [Classes and Objects in Python](https://www.learnpython.org/en/Classes_and_Objects)
* [Python Classes and Objects Tutorial - Programiz](https://www.programiz.com/python-programming/class)

**Day 9: Inheritance and Polymorphism**

**Problem:** Build a class hierarchy for different types of vehicles (e.g., Car, Bicycle) with common and specialized methods. **What to Learn:** Explore inheritance and how to create class hierarchies in Python. **Resources:**

* [Python Inheritance](https://docs.python.org/3/tutorial/classes.html#inheritance)
* [Inheritance and Polymorphism in Python](https://realpython.com/inheritance-composition-python/)

**Problem:** Demonstrate method overriding by creating subclasses of a base class. **What to Learn:** Understand how to override methods in derived classes. **Resources:**

* [Python Method Overriding](https://www.geeksforgeeks.org/method-overriding-in-python/)
* [Python Inheritance and Method Overriding - Programiz](https://www.programiz.com/python-programming/inheritance)

**Problem:** Build a program that demonstrates polymorphism with different shapes (e.g., Circle, Rectangle, Triangle). **What to Learn:** Practice creating classes, inheritance, and polymorphism with geometric shapes. **Resources:**

* [Python Polymorphism Example - Tutorialspoint](https://www.tutorialspoint.com/python3/python_polymorphism.htm)
* [Python Polymorphism - JournalDev](https://www.journaldev.com/21541/python-polymorphism)

**Day 10: Advanced OOP**

**Problem:** Create a class-based representation of a library catalog with methods for checking out and returning books. **What to Learn:** Explore more advanced class design and interaction. **Resources:**

* [Python Class Design](https://realpython.com/python3-object-oriented-programming/#class-design)
* [Python Classes and Objects - TutorialsPoint](https://www.tutorialspoint.com/python/python_classes_objects.htm)

**Problem:** Implement a class-based quiz game with questions and answers, incorporating inheritance and polymorphism. **What to Learn:** Combine multiple OOP concepts in a single project. **Resources:**

* [Python Quiz Game Project - GeeksforGeeks](https://www.geeksforgeeks.org/python-quiz-game-project/)
* [Building a Simple Quiz - Programiz](https://www.programiz.com/python-programming/examples/quiz)

**Problem:** Build a program that demonstrates polymorphism with different animals (e.g., Dog, Cat, Bird). **What to Learn:** Practice creating classes, inheritance, and polymorphism with animal types. **Resources:**

* [Python Polymorphism Example - Tutorialspoint](https://www.tutorialspoint.com/python3/python_polymorphism.htm)
* [Python Polymorphism - JournalDev](https://www.journaldev.com/21541/python-polymorphism)

**Day 11: File Handling and Exception Handling**

**Problem:** Search for files with a specific extension in a directory and its subdirectories. **What to Learn:** Learn how to navigate and search for files in directories using Python. **Resources:**

* [Python File Handling](https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files)
* [Python File Handling - W3Schools](https://www.w3schools.com/python/python_file_handling.asp)

**Problem:** Read data from a JSON file and perform data manipulation. **What to Learn:** Practice reading and manipulating data from JSON files. **Resources:**

* [Python JSON](https://docs.python.org/3/library/json.html)
* [Working with JSON Data in Python](https://realpython.com/python-json/)

**Problem:** Implement a program that handles user-defined exceptions in a custom Python module. **What to Learn:** Understand how to handle exceptions effectively and create custom exception classes.

**Resources:**

* [Python Exception Handling](https://docs.python.org/3/tutorial/errors.html)
* [Python Custom Exceptions - GeeksforGeeks](https://www.geeksforgeeks.org/python-exception-handling-using-custom-exceptions/)

**Day 12: Data Analysis and Visualization**

**Problem:** Read data from a CSV file and perform data analysis. **What to Learn:** Learn how to read and analyze data from CSV files using libraries like pandas. **Resources:**

* [Working with CSV Files in Python](https://realpython.com/python-csv/)
* [Python CSV Documentation](https://docs.python.org/3/library/csv.html)

**Problem:** Create visualizations from data using Matplotlib or Seaborn. **What to Learn:** Explore data visualization techniques in Python. **Resources:**

* [Matplotlib Documentation](https://matplotlib.org/stable/contents.html)
* [Seaborn Documentation](https://seaborn.pydata.org/introduction.html)

**Problem:** Perform exploratory data analysis (EDA) on a dataset, including summary statistics and data visualization. **What to Learn:** Practice EDA techniques using libraries like pandas and Matplotlib/Seaborn. **Resources:**

* [Data Analysis with pandas - DataCamp](https://www.datacamp.com/community/tutorials/pandas-tutorial-dataframe-python)
* [Python Data Visualization - Real Python](https://realpython.com/tutorials/data-viz/)

**Day 13: Project Day**

**Problems:**

* Build a personal finance manager application.
* Perform sentiment analysis on social media data.
* Create a basic content management system (CMS) for a website. **What to Learn:** Apply what you've learned to create practical projects. **Resources:** Refer to the resources mentioned in the previous days for guidance and inspiration. You can also search for specific tutorials on building these projects.

**Day 14: Project Day**

**Problems:**

* Continue working on your larger projects from the previous days. **What to Learn:** Dedicate this day to finalizing your larger projects, adding finishing touches, and ensuring they are ready for presentation or deployment. **Resources:** Refer to the resources mentioned in the previous days for guidance and inspiration.

**Day 15: Web Scraping and APIs**

**Problem:** Scrape data from a website using BeautifulSoup. **What to Learn:** Learn web scraping basics and use BeautifulSoup for parsing HTML. **Resources:**

* [Web Scraping using Python and BeautifulSoup](https://realpython.com/beautiful-soup-web-scraper-python/)
* [Beautiful Soup Documentation](https://www.crummy.com/software/BeautifulSoup/bs4/doc/)

**Problem:** Analyze a large dataset and generate summary statistics. **What to Learn:** Explore data analysis techniques using libraries like pandas. **Resources:**

* [pandas Documentation](https://pandas.pydata.org/docs/)
* [Data Analysis with pandas - DataCamp](https://www.datacamp.com/community/tutorials/pandas-tutorial-dataframe-python)

**Problem:** Read data from an Excel file and perform data visualization. **What to Learn:** Learn how to read Excel files and visualize data with libraries like pandas and Matplotlib. **Resources:**

* [Working with Excel Files in Python](https://realpython.com/openpyxl-excel-spreadsheets-python/)
* [Data Visualization in Python](https://realpython.com/tutorials/data-viz/)

**Day 16: Advanced Functions and Modules**

**Problem:** Sort a list of dictionaries using lambda functions. **What to Learn:** Explore advanced sorting techniques using lambda functions. **Resources:**

* [Sorting Lists of Dictionaries](https://docs.python.org/3/howto/sorting.html#sortinghowto)
* [Python Lambda Functions](https://realpython.com/python-lambda/)

**Problem:** Create a Python module for statistical calculations. **What to Learn:** Practice module creation and importing custom modules. **Resources:**

* [Creating Modules in Python](https://docs.python.org/3/tutorial/modules.html)
* [Creating and Using Modules in Python](https://realpython.com/python-modules-packages/)

**Problem:** Generate random data following a specified distribution. **What to Learn:** Learn how to generate random data with different distributions. **Resources:**

* [Random Module in Python](https://docs.python.org/3/library/random.html)
* [Generating Random Data in Python - Real Python](https://realpython.com/tutorials/random/)

**Day 17: Database Interaction**

**Problem:** Connect to a SQLite database and perform CRUD operations. **What to Learn:** Understand database interaction using SQLite and SQL queries. **Resources:**

* [Python SQLite Documentation](https://docs.python.org/3/library/sqlite3.html)
* [SQLite Tutorial](https://www.sqlitetutorial.net/)

**Problem:** Fetch data from an external API and store it in a local database. **What to Learn:** Learn how to make API requests and store data in a database. **Resources:**

* [Python Requests Library](https://docs.python-requests.org/en/master/)
* [APIs and Databases in Python - Real Python](https://realpython.com/api-integration-in-python/)

**Problem:** Query a database to retrieve and display specific information. **What to Learn:** Practice SQL queries to retrieve data from a database. **Resources:**

* [SQL Basics](https://learn.microsoft.com/en-us/sql/sql-basics?view=sql-server-ver15)
* [SQL Queries Tutorial - SQLZoo](https://sqlzoo.net/wiki/SQL_Tutorial)

**Day 18: Data Visualization**

**Problem:** Use Matplotlib to plot data from a CSV file. **What to Learn:** Learn the basics of data visualization with Matplotlib. **Resources:**

* [Matplotlib Documentation](https://matplotlib.org/stable/contents.html)
* [Python Data Visualization - Real Python](https://realpython.com/tutorials/data-viz/)

**Problem:** Create interactive plots using Plotly. **What to Learn:** Explore interactive data visualization with Plotly. **Resources:**

* [Plotly Python Documentation](https://plotly.com/python/)
* [Plotly Express Tutorial - Plotly](https://plotly.com/python/plotly-express/)

**Problem:** Generate word clouds from text data. **What to Learn:** Learn how to create word clouds for text data visualization. **Resources:**

* [Word Cloud Python Library](https://github.com/amueller/word_cloud)
* [Generating Word Clouds in Python - Real Python](https://realpython.com/generating-word-clouds-python/)

**Day 19: Web Development**

**Problem:** Build a simple web application using Flask or Django. **What to Learn:** Learn web development basics with Python frameworks. **Resources:**

* [Flask Documentation](https://flask.palletsprojects.com/en/2.1.x)
* [Django Documentation](https://docs.djangoproject.com/en/4.1/)

**Problem:** Create a RESTful API with Flask or Django. **What to Learn:** Understand how to build RESTful APIs and serve data. **Resources:**

* [Flask RESTful](https://flask-restful.readthedocs.io/en/latest/)
* [Django REST framework](https://www.django-rest-framework.org/)

**Problem:** Develop a basic e-commerce website with user authentication. **What to Learn:** Implement user authentication and build a dynamic web application. **Resources:**

* [Django Authentication](https://docs.djangoproject.com/en/4.1/topics/auth/)
* [Flask-Login (for Flask)](https://flask-login.readthedocs.io/en/latest/)

**Day 20: Project Day**

**Problems:** Continue working on your larger projects from the previous days.

**What to Learn:** Dedicate this day to finalizing your larger projects, adding finishing touches, and ensuring they are ready for presentation or deployment.

**Resources:** Refer to the resources mentioned in the previous days for guidance and inspiration.

**Day 21: Advanced Web Development**

**Problem:** Implement user authentication with role-based access control in a web application. **What to Learn:** Learn how to handle user roles and permissions in a web app. **Resources:**

* [Role-Based Access Control (RBAC) - Wikipedia](https://en.wikipedia.org/wiki/Role-based_access_control)
* [Django Role-Based Access Control - Django](https://django-role-based-access-control.readthedocs.io/en/latest/)

**Problem:** Create a blog or forum application with user-generated content. **What to Learn:** Build a dynamic web application with user-generated content and comments. **Resources:**

* [Building a Blog Application with Django - DjangoGirls](https://tutorial.djangogirls.org/)
* [Flask Mega-Tutorial Part XV: Facelift - Miguel Grinberg](https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-xv-a-better-ui)

**Problem:** Implement real-time features using WebSocket technology. **What to Learn:** Explore WebSocket communication and real-time updates in web apps. **Resources:**

* [WebSocket in Python - WebSockets for Fun and Profit](https://realpython.com/websockets/)
* [Django Channels Documentation](https://channels.readthedocs.io/en/latest/)

**Day 22-24: Continue working on larger projects**

**Problems:** Dedicate these days to refining and improving your major projects. **What to Learn:** Focus on improving the functionality, user interface, and documentation of your projects. **Resources:** Use your existing project resources and seek help from online forums and communities if needed.

**Day 25: Machine Learning**

**Problem:** Build a machine learning model using scikit-learn. **What to Learn:** Apply machine learning techniques to solve classification or regression problems. **Resources:**

* [scikit-learn Documentation](https://scikit-learn.org/stable/documentation.html)
* [Machine Learning with scikit-learn - DataCamp](https://www.datacamp.com/community/tutorials/machine-learning-python)

**Problem:** Create a recommendation system. **What to Learn:** Learn how recommendation systems work and implement one using Python. **Resources:**

* [Building a Movie Recommendation System in Python - Real Python](https://realpython.com/build-recommendation-engine-collaborative-filtering/)
* [Introduction to Recommendation Systems in Python - DataCamp](https://www.datacamp.com/community/tutorials/recommendation-engine-python)

**Problem:** Implement a natural language processing (NLP) project. **What to Learn:** Explore NLP techniques like sentiment analysis or text classification. **Resources:**

* [Natural Language Processing in Python - DataCamp](https://www.datacamp.com/community/tutorials/natural-language-processing-tutorial)
* [Text Classification with Python - Real Python](https://realpython.com/nltk-nlp-python/)

**Day 26: Networking and APIs**

**Problem:** Develop a multi-client chat application. **What to Learn:** Learn about network programming and socket communication in Python. **Resources:**

* [Socket Programming in Python - GeeksforGeeks](https://www.geeksforgeeks.org/socket-programming-python/)
* [Building a Python Chat Application - Real Python](https://realpython.com/python-sockets/#building-a-python-chat-application)

**Problem:** Create a web crawler. **What to Learn:** Understand web scraping and crawling techniques using libraries like BeautifulSoup and requests. **Resources:**

* [Web Scraping and Crawling with Python - Real Python](https://realpython.com/tutorials/web-scraping/)
* [Web Scraping using Python and BeautifulSoup - DataCamp](https://www.datacamp.com/community/tutorials/tutorial-python-beautifulsoup-datacamp-tutorials)

**Problem:** Build a RESTful API with documentation. **What to Learn:** Learn how to create a RESTful API using frameworks like Flask or Django and document its endpoints. **Resources:**

* [API Documentation with Swagger and ReDoc - Flask-RESTPlus](https://flask-restplus.readthedocs.io/en/stable/swaggerui.html)
* [Django Rest Framework Documentation - API Guide](https://www.django-rest-framework.org/topics/documenting-your-api/)

**Day 27: Advanced Topics**

**Problem:** Implement multi-threading or multi-processing in Python. **What to Learn:** Explore concurrency and parallelism in Python for better performance. **Resources:**

* [Threading in Python](https://realpython.com/intro-to-python-threading/)
* [Multiprocessing in Python - GeeksforGeeks](https://www.geeksforgeeks.org/multiprocessing-python-set-2/)

**Problem:** Explore decorators for enhancing functions and methods. **What to Learn:** Understand how decorators can modify the behavior of functions and methods. **Resources:**

* [Python Decorators - Real Python](https://realpython.com/primer-on-python-decorators/)
* [Understanding Python Decorators in 12 Easy Steps - DataCamp](https://www.datacamp.com/community/tutorials/decorators-python)

**Problem:** Learn asynchronous programming using the **asyncio** library. **What to Learn:** Understand asynchronous programming concepts and how to use **async** and **await**. **Resources:**

* [Asynchronous Programming in Python - GeeksforGeeks](https://www.geeksforgeeks.org/asynchronous-programming-in-python-an-introduction/)
* [Asyncio - The Python Standard Library](https://docs.python.org/3/library/asyncio.html)

**Day 28: Finalize Projects**

**Problem:** Focus on finalizing your major projects. **What to Learn:** Ensure that your projects are well-documented, organized, and fully functional. **Resources:** Use your existing project resources and seek feedback from peers if possible.

**Day 29-30: Review and Showcase**

**Problems:**

* Review and refactor your code from the past month, focusing on code quality and optimization.
* Create a portfolio to showcase your projects and skills to potential employers or collaborators.
* Prepare for technical interviews or further learning based on your interests and career goals.

**Resources:** Use this time to thoroughly review your code, seek feedback, and prepare for showcasing your work. Consider creating a personal website or GitHub repository to showcase your projects.

Congratulations on completing your one-month Python learning journey! By following this plan and continuously practicing and building projects, you'll be well on your way to mastering Python. Remember that learning programming is an ongoing process, so keep exploring new topics and projects as you continue your programming journey. Good luck!