7-Day Python Learning Plan (Basic to Advanced)

This structured plan covers Python essentials to advanced concepts in 1 hour daily, focusing on practical exercises.

Day 1: Python Basics

Topics to Cover:

- Python Syntax:
  - Variables, data types (`int` , `float` , `str` , `bool`).
- Input and Output:
  - `print()` and `input()` .
- Basic Operators:
  - Arithmetic (`+` , `-` , `*` , `/` , `//` , `%`), Comparison (`==` , `!=` , `<` , `>`), Logical (`and` , `or` , `not`).

Practical Task:

1. Write a script to calculate the area of a circle given its radius.
2. Create a simple program that asks for a user's name and age, then prints a message.

Resources:

- Python Basics by W3Schools

Day 2: Control Flow

Topics to Cover:

- Conditionals:
  - `if` , `elif` , `else` .
- Loops:
  - `for` , `while` .
  - `break` , `continue` .
- Range Function:
  - Generate sequences with `range()` .

Practical Task:

1. Write a program to print all even numbers between 1 and 50.
2. Create a number guessing game where the user has to guess a random number.

Resources:

- MDN: Python Control Flow

Day 3: Functions and Modules

Topics to Cover:

- Functions:
  - Defining and calling functions.
  - Parameters and return values.
  - Default and keyword arguments.
- Importing Modules:
  - Built-in modules (`math` , `random`).
  - Writing and importing your own module.

Practical Task:

1. Write a function to check if a number is prime.
2. Create a module with helper functions (e.g., factorial, sum of a list) and use it in a program.

Resources:

- Python Functions

Day 4: Data Structures

Topics to Cover:

- Lists:
  - Operations (`append` , `pop` , `slice`).
- Tuples:
  - Immutable sequences.

- Dictionaries:
  - Key-value pairs, methods (`keys()` , `values()` , `items()`).
- Sets:
  - Unique elements, set operations (`union` , `intersection`).

Practical Task:

1. Create a program to manage a to-do list using a list.
2. Write a script to count the frequency of each word in a sentence using a dictionary.

Resources:

- Python Data Structures

Day 5: File Handling and Exception Management

Topics to Cover:

- File Handling:
  - Open, read, write, and close files (`open()` , `read()` , `write()`).
- Exception Handling:
  - `try` , `except` , `finally` .
  - Raising exceptions with `raise` .

Practical Task:

1. Write a program to read a text file and count the number of lines.
2. Implement a calculator that handles invalid input using exception handling.

Resources:

- File Handling in Python

Day 6: Object-Oriented Programming

Topics to Cover:

- Classes and Objects:
  - Defining classes, creating objects.

- Attributes and Methods:
  - Instance variables, class methods, static methods.
- Inheritance:
  - Creating subclasses, overriding methods.
- Encapsulation and Polymorphism.

Practical Task:

1. Create a class `Car` with attributes like `brand`, `model`, and `year`, and methods to start and stop the car.
2. Create a subclass `ElectricCar` that extends the functionality of `Car`.

Resources:

- Python OOP

Day 7: Advanced Concepts and Mini Project

Topics to Cover:

- Generators and Iterators:
  - Creating and using generators.
- Decorators:
  - Writing and applying decorators.
- Working with APIs:
  - Using `requests` to fetch data.
- Regular Expressions:
  - Pattern matching with `re`.

Practical Project:

1. **Weather App**:
   - Use the `requests` library to fetch weather data from an API.
   - Parse the data and display it in a user-friendly format.
2. **Text File Analyzer**:
   - Analyze a text file for word count, unique words, and frequency.

Resources:

- Python Generators

- Requests Library

Bonus Advanced Topics

- **Data Science**: Learn `pandas` and `numpy`.
- **Web Development**: Basics of Flask or Django.
- **Testing**: Use `unittest` for testing Python programs.
- **Concurrency**: Explore `asyncio` for asynchronous programming.

This plan ensures a practical learning experience while covering essential Python concepts and advanced topics. Let me know if you'd like resources for any specific day!