Day 1: Introduction to Python

- Install Python and an integrated development environment (IDE) of your choice
- Learn about Python syntax and basic data types (strings, integers, floats, and booleans)
- Complete 10 simple programming exercises using basic data types

Difficult Tasks:

1. Write a Python program that takes in a sentence and returns the number of vowels in the sentence.
2. Write a Python program that prints out the first 50 prime numbers.
3. Write a Python program that takes in a list of numbers and returns the sum of all even numbers in the list.
4. Write a Python program that takes in a list of words and returns a new list with only the words that contain the letter "a".
5. Write a Python program that takes in a list of numbers and returns a new list with only the prime numbers.

Day 2: Conditionals and Loops

- Learn about conditional statements (if/else) and loops (for/while)
- Complete 10 programming exercises using conditionals and loops

Difficult Tasks:

1. Write a Python program that takes in a list of numbers and returns the largest number in the list.
2. Write a Python program that takes in a sentence and returns the number of words in the sentence.
3. Write a Python program that takes in a list of numbers and returns the product of all the numbers in the list.
4. Write a Python program that takes in a list of words and returns a new list with only the words that start with a capital letter.
5. Write a Python program that takes in a list of numbers and returns a new list with only the odd numbers.

Day 3: Functions

- Learn about functions and how to define and call them in Python
- Complete 10 programming exercises using functions

Difficult Tasks:

1. Write a Python program that takes in a list of numbers and returns the median of the list.
2. Write a Python program that takes in a sentence and returns the sentence with the words in reverse order.
3. Write a Python program that takes in a list of numbers and returns the difference between the largest and smallest number in the list.
4. Write a Python program that takes in a list of words and returns a new list with only the words that end with the letter "s".

5. Write a Python program that takes in a list of numbers and returns a new list with the numbers sorted in descending order.

Day 4: Lists and Tuples

- Learn about lists and tuples in Python
- Complete 10 programming exercises using lists and tuples

Difficult Tasks:

1. Write a Python program that takes in a list of numbers and returns the second largest number in the list.
2. Write a Python program that takes in a list of words and returns the longest word in the list.
3. Write a Python program that takes in two lists of numbers and returns a new list with only the common numbers in both lists.
4. Write a Python program that takes in a list of words and returns a new list with the words sorted in alphabetical order.
5. Write a Python program that takes in a list of numbers and returns a new list with each number squared.

Day 5: Dictionaries

- Learn about dictionaries in Python
- Complete 10 programming exercises using dictionaries

Difficult Tasks:

1. Write a Python program that takes in a list of words and returns a dictionary with the number of times each word appears in the list.
2. Write a Python program that takes ina dictionary and returns a new dictionary with the keys and values swapped.
3. Write a Python program that takes in two dictionaries and returns a new dictionary with the keys and values combined.
4. Write a Python program that takes in a dictionary and returns a new dictionary with only the keys that start with the letter "a".
5. Write a Python program that takes in a list of dictionaries and returns a new list with the dictionaries sorted by the value of a specified key.

Day 6: File Input and Output

- Learn about reading and writing files in Python
- Complete 10 programming exercises using file input and output

Difficult Tasks:

1. Write a Python program that reads in a file and returns the number of words in the file.
2. Write a Python program that reads in a file and returns the longest word in the file.
3. Write a Python program that reads in two files and writes a new file with the contents of both files combined.

4. Write a Python program that reads in a file and returns a dictionary with the number of times each word appears in the file.
5. Write a Python program that reads in a file and prints out the words in the file in alphabetical order.

Day 7: Sets

- Learn about sets in Python
- Complete 10 programming exercises using sets

Difficult Tasks:

1. Write a Python program that takes in two sets of numbers and returns a new set with only the common numbers in both sets.
2. Write a Python program that takes in a list of numbers and returns a set with the unique numbers in the list.
3. Write a Python program that takes in two sets of words and returns a new set with only the common words in both sets.
4. Write a Python program that takes in a set of numbers and returns the sum of all the numbers in the set.
5. Write a Python program that takes in a list of words and returns a new set with the unique letters in all the words.

Day 8: Object-Oriented Programming Basics

- Learn about classes, objects, and methods in Python
- Complete 10 programming exercises using OOP basics

Difficult Tasks:

1. Write a Python program that defines a class for a car and includes methods for accelerating and braking the car.
2. Write a Python program that defines a class for a bank account and includes methods for depositing and withdrawing money.
3. Write a Python program that defines a class for a book and includes methods for getting the title, author, and number of pages.
4. Write a Python program that defines a class for a person and includes methods for getting the name, age, and gender of the person.
5. Write a Python program that defines a class for a rectangle and includes methods for calculating the area and perimeter of the rectangle.

Day 9: Inheritance and Polymorphism

- Learn about inheritance and polymorphism in Python
- Complete 10 programming exercises using inheritance and polymorphism

Difficult Tasks:

1. Write a Python program that defines a class for a vehicle and includes methods for accelerating and braking. Define a subclass for a car that includes a method for opening the doors.

2. Write a Python program that defines a class for a shape and includes methods for calculating the area and perimeter. Define subclasses for a rectangle and a circle that override the methods to calculate their respective areas and perimeters.
3. Write a Python program that defines a class for a pet and includes methods for getting the name and species of the pet. Define subclasses for a dog and a cat that override the method for getting the species to return "dog" or "cat", respectively.
4. Write a Python program that defines a class for a person and includes methods for getting the name, age, and gender. Definea subclass for a student that includes a method for getting the student's grade point average.
5. Write a Python program that defines a class for a shape and includes methods for calculating the area and perimeter. Define a subclasses for a square and a rectangle that inherit from the shape class and include methods for calculating the area and perimeter specific to their shapes.

Day 10: Advanced OOP Concepts

- Learn about advanced OOP concepts such as abstract classes, interfaces, and decorators in Python
- Complete 10 programming exercises using advanced OOP concepts

Difficult Tasks:

1. Write a Python program that defines an abstract class for a shape and includes abstract methods for calculating the area and perimeter. Define subclasses for a rectangle and a circle that implement the abstract methods.
2. Write a Python program that defines an interface for a logger and includes methods for logging information. Define a class that implements the interface and includes a decorator that logs information before and after each method call.
3. Write a Python program that defines a class for a bank account and includes methods for depositing and withdrawing money. Define a subclass for a savings account that includes a method for calculating interest and a decorator that logs information before and after each method call.
4. Write a Python program that defines a class for a vehicle and includes methods for accelerating and braking. Define a subclass for a bike that includes a method for pedaling and a decorator that logs information before and after each method call.
5. Write a Python program that defines a class for a shape and includes methods for calculating the area and perimeter. Define a subclass for a triangle that includes a method for calculating the area using Heron's formula and a decorator that logs information before and after each method call.

Day 11: File Handling

- Learn how to read and write files in Python using built-in functions and libraries
- Practice reading and writing to files with 10 programming exercises

Difficult Tasks:

1. Write a Python program that reads a CSV file containing customer data and outputs a report showing the total revenue, number of orders, and average order value.
2. Write a Python program that reads a text file containing a list of numbers and calculates the sum, mean, and standard deviation.

3. Write a Python program that reads a JSON file containing product data and outputs a report showing the top-selling products, the average rating, and the total revenue generated.
4. Write a Python program that reads a binary file containing image data and outputs a thumbnail image.
5. Write a Python program that reads a log file containing server data and outputs a report showing the number of requests, the average response time, and the percentage of successful requests.

Day 12: Regular Expressions

- Learn about regular expressions and their use cases in Python
- Practice working with regular expressions in 10 programming exercises

Difficult Tasks:

1. Write a Python program that uses regular expressions to extract email addresses from a text file.
2. Write a Python program that uses regular expressions to validate whether a given string is a valid URL.
3. Write a Python program that uses regular expressions to extract phone numbers from a CSV file.
4. Write a Python program that uses regular expressions to remove HTML tags from a web page.
5. Write a Python program that uses regular expressions to extract dates from a text file and convert them to a different format.

Day 13: Error Handling

- Learn about error handling in Python and how to handle different types of errors
- Practice error handling in 10 programming exercises

Difficult Tasks:

1. Write a Python program that handles exceptions raised by a user-defined function and logs the errors to a file.
2. Write a Python program that handles exceptions raised by a third-party library and displays a custom error message.
3. Write a Python program that handles exceptions raised by file I/O operations and prompts the user to enter a valid file path.
4. Write a Python program that handles exceptions raised by network operations and retries the operation after a specified delay.
5. Write a Python program that handles exceptions raised by database operations and rolls back the transaction if an error occurs.

Day 14: Multithreading

- Learn about multithreading in Python and how to use the threading module
- Practice multithreading with 10 programming exercises

Difficult Tasks:

1. Write a Python program that uses multithreading to download multiple files simultaneously and displays the progress for each download.

2. Write a Python program that uses multithreading to process a large CSV file and outputs the results to a text file.
3. Write a Python program that uses multithreading to calculate the prime numbers within a given range and displays the results in real-time.
4. Write a Python program that uses multithreading to encrypt and decrypt a file and measures the performance improvement over single-threaded operations.
5. Write a Python program that uses multithreading to download multiple web pages and extract information from them concurrently.

Day 15: Multithreading and Multiprocessing

- Learn about multithreading and multiprocessing in Python
- Practice using threads and processes in 10 programming exercises

Difficult Tasks:

1. Write a Python program that uses multithreading to perform multiple tasks simultaneously, such as downloading files and processing data.
2. Write a Python program that uses multiprocessing to perform computationally intensive tasks, such as image processing or machine learning.
3. Write a Python program that creates a multithreaded server that handles incoming client connections and processes requests in parallel.
4. Write a Python program that creates a multiprocessing pool that distributes tasks across multiple processes, optimizing performance for CPU-bound tasks.
5. Write a Python program that combines multithreading and multiprocessing to create a high-performance application that can handle multiple concurrent requests and process them efficiently.

Day 16: Networking

- Learn about networking in Python and how to use the socket module
- Practice networking with 10 programming exercises

Difficult Tasks:

1. Write a Python program that creates a TCP server and handles incoming client connections, sending and receiving messages.
2. Write a Python program that creates a UDP server and handles incoming client connections, sending and receiving messages.
3. Write a Python program that uses the requests library to send HTTP requests and receive responses from a web server.
4. Write a Python program that uses the ftplib library to upload and download files to an FTP server.
5. Write a Python program that uses the smtplib library to send emails through a SMTP server.

Day 17: Web Development with Flask

- Learn about web development in Python using the Flask framework
- Practice creating web applications with Flask in 10 programming exercises

Difficult Tasks:

1. Write a Python program that creates a simple Flask application that displays a list of products with prices and ratings.
2. Write a Python program that creates a Flask application that allows users to register and login, and displays a dashboard with user-specific information.
3. Write a Python program that creates a Flask application that integrates with a third-party API and displays relevant data to the user.
4. Write a Python program that creates a Flask application that allows users to upload and download files securely.
5. Write a Python program that creates a Flask application that integrates with a database and allows users to perform CRUD operations on data.

Day 18: Web Scraping with BeautifulSoup

- Learn about web scraping in Python using the BeautifulSoup library
- Practice web scraping with 10 programming exercises

Difficult Tasks:

1. Write a Python program that uses BeautifulSoup to scrape data from a website and outputs the results to a CSV file.
2. Write a Python program that uses BeautifulSoup to scrape data from a website and inserts the results into a database.
3. Write a Python program that uses BeautifulSoup to scrape data from a website and generates a report showing the top results.
4. Write a Python program that uses BeautifulSoup to scrape data from a website and performs sentiment analysis on the results.
5. Write a Python program that uses BeautifulSoup to scrape data from a website and generates a word cloud from the results.

Day 19: Data Visualization with Matplotlib

- Learn about data visualization in Python using the Matplotlib library
- Practice data visualization with 10 programming exercises

Difficult Tasks:

1. Write a Python program that uses Matplotlib to visualize data from a CSV file, displaying it as a bar chart or line chart.
2. Write a Python program that uses Matplotlib to visualize data from a database, displaying it as a scatter plot or bubble chart.
3. Write a Python program that uses Matplotlib to visualize data from a web API, displaying it as a heatmap or choropleth map.
4. Write a Python program that uses Matplotlib to create interactive visualizations, allowing users to explore and interact with data.
5. Write a Python program that uses Matplotlib to create animated visualizations, showing changes over time or in response to user input.

Day 20: Final Project

- Create a final project that incorporates all the topics covered in the previous 19 days, including OOP, file handling, regular expressions, error handling, multithreading, multiprocessing, networking, web development, web scraping, and data visualization.
- This project should be a fully functional application that demonstrates your skills and understanding of Python.

Difficult Tasks:

1. Write a Python program that creates a web application that allows users to upload and analyze data files using regular expressions and data visualization.
2. Write a Python program that uses multithreading and multiprocessing to process and analyze large datasets in real-time, displaying results using data visualization.
3. Write a Python program that creates a networked application that allows users to collaborate on a real-time data analysis project using web scraping and data visualization.
4. Write a Python program that uses OOP to create a machine learning model that can analyze and classify data, and then displays the results using data visualization.
5. Write a Python program that uses all the concepts covered in the previous 19 days to create a complex and robust application that solves a real-world problem, such as fraud detection or predictive maintenance.