Python

1. Python Basics

- Understand the fundamentals of Python programming
- Learn about Python data structures
- Learn about functions and modules

By completing these steps, you should get a good foundation in the basics of Python, allowing you to learn more advanced concepts later on and build useful applications.

Tasks

1. Understand the fundamentals of Python programming
    - ☐ Install Python on your computer
    - ☐ Learn about variables, data types, and basic syntax
    - ☐ Do a basic output exercise using print statements
2. Learn about Python data structures
    - ☐ Understand the basics of lists, tuples, and dictionaries
    - ☐ Work with conditional statements and loops
    - ☐ Do an exercise involving manipulating data structures
3. Learn about functions and modules
    - ☐ Understand function definition and calling
    - ☐ Learn about Python modules and libraries
    - ☐ Understand `__init__` and `__main__`
    - ☐ Do an exercise involving writing and using a custom function

2. Intermediate Python

- Learn about advanced data structures
- Learn about file handling and exceptions
- Learn about in Python

With the above steps completed, you'll be able to create more sophisticated programs and functionality for your users.

Tasks

1. Learn about advanced data structures
    - ☐ Learn about sets and frozensets
    - ☐ Work with nested data structures
    - ☐ Do an exercise involving advanced data structures
2. Learn about file handling and exceptions

- ☐ Learn how to read and write files
- ☐ Work with exception handling to catch and handle errors
- ☐ Do an exercise involving file handling and exception handling

3. Learn about testing in Python

   - ☐ Install the pytests unit testing library
   - ☐ Create some tests for functions you have written before
   - ☐ Try using a TDD approach to create some functions, e.g. a function to return the largest element of an array

3. Object oriented python

- Learn about classes and Objects
- Learn about inheritance
- Learn about polymorphism
- Learn about encapsulation

By completing the above steps, you will have a stronger understanding of object oriented programming, which will allow you to structure your projects and reuse code in different ways.

Tasks

1. Learn about classes and Objects

   - ☐ Create a simple class with attributes and methods.
   - ☐ Create an object of the class and access its attributes and methods.
   - ☐ Write a program that uses a class to model a real-world object.

2. Learn about inheritance

   - ☐ Create a subclass that inherits from a superclass.
   - ☐ Override a method in the subclass.
   - ☐ Use inheritance to create a hierarchy of classes that models a real-world scenario.

3. Learn about polymorphism

   - ☐ Create a program that uses polymorphism to process objects of different classes.
   - ☐ Use the isinstance() function to check the type of an object.
   - ☐ Write a program that demonstrates the concept of polymorphism in a real-world scenario.

4. Learn about encapsulation

   - ☐ Create a class with private attributes and methods.
   - ☐ Use getter and setter methods to access and modify private attributes.
   - ☐ Write a program that uses encapsulation to protect data in a real-world scenario.

4. Advanced Python

- Learn about regular expressions
- Learn about web scraping
- Learn about data visualization

These topics could easily take a day each - don't try to become an expert in all of it, just take a few hours to explore each one so that you know what is possible with Python and see a few interesting patterns.

Tasks

1. Learn about regular expressions
    - ☐ Understand the basics of regular expressions
    - ☐ Learn how to use the re module to search for and manipulate strings
    - ☐ Do an exercise involving regular expressions
2. Learn about web scraping
    - ☐ Understand the basics of web scraping
    - ☐ Learn how to use the requests and BeautifulSoup libraries to scrape websites
    - ☐ Do an exercise involving web scraping
3. Learn about data visualization
    - ☐ Learn about data visualization libraries like Matplotlib and Seaborn
    - ☐ Understand the basics of creating plots and charts
    - ☐ Do an exercise involving data visualization

5. Review

- Review and practice what you have learned
- Explore additional Python topics
- Work on a personal project

When you have time, it is important to keep practising: exercises, mini-projects, whatever keeps you motivated.

Tasks

1. Review and practice what you have learned
    - ☐ Go over the concepts covered in the previous days
    - ☐ Do some practice exercises to reinforce your understanding
2. Explore additional Python topics
    - ☐ Learn about advanced Python topics that interest you, such as multi-threading, network programming, or machine learning
    - ☐ Do some research and find resources to learn more about these topics

3. Work on a personal project
    - ☐ Choose a project that interests you and use Python to implement it
    - ☐ Practice your skills and explore new concepts while working on your project.

Resources

There are many free resources available online for learning Python programming. Here are some of the best ones:

Python.org is the official website for the Python programming language. It has a comprehensive tutorial for beginners, as well as documentation, downloads, and resources for Python users of all levels.

Codecademy is an online learning platform that offers interactive Python courses for beginners. The first lessons are free, and you can learn at your own pace.

Coursera offers a wide range of free online courses, including several on Python programming. You can learn from top instructors at leading universities and organizations around the world.

edX is another platform that offers free online courses, including several on Python programming. You can learn from top universities and institutions, and earn a certificate of completion for some courses.

YouTube has many excellent Python programming tutorials and courses available on YouTube, ranging from beginner to advanced levels. Some popular channels include Corey Schafer, sentdex, and Tech With Tim.

Python for Everybody is a free online course offered by the University of Michigan, and is designed for beginners. It covers the basics of programming using Python and is a great resource for those new to coding.

Automate the Boring Stuff with Python is a free online book by Al Sweigart teaches Python programming through practical examples that automate everyday tasks. It's a great resource for those who want to learn Python for practical purposes.

These are just a few of the many free resources available for learning Python programming. When choosing a resource, consider your learning style and goals, and look for reviews and recommendations from other learners. Good luck!

Project ideas

Here are some examples of Python projects that you can build to practice your skills:

Calculator Build a simple command-line calculator that can perform basic arithmetic operations like addition, subtraction, multiplication, and division.

File Organizer Build a script that can organize files in a directory based on their file type or other criteria. For example, you can move all images to an "Images" folder, all documents to a "Documents" folder, etc.

Password Manager Build a simple command-line password manager that can store and retrieve passwords for different accounts. You can add additional features like encryption and decryption of passwords.

Web Scraper Build a script that can scrape data from a website and store it in a database or a CSV file. You can use libraries like Beautiful Soup and Requests to make the task easier.

Word Count Tool Build a script that can count the number of words, lines, and characters in a text file. You can add additional features like sorting the results by frequency and generating a word cloud.

These projects will allow you to practice your Python skills without requiring a framework like Flask or Django. As you work on these projects, don't be afraid to experiment and add your own features and functionalities. Good luck!