multiverse

Contents

Name:
Date:

Python

Python is a popular programming language that is widely used for developing web applications, data analysis, artificial intelligence, and much more. It's known for its simplicity and ease of use, making it an excellent language for programmers of all ability levels.

In this journey, you'll learn how to write basic Python programs, work with variables, control flow statements, and functions. You'll also learn about Python's built-in data structures, such as lists, tuples, and dictionaries, and how to manipulate them.

As you progress, you'll delve into more advanced topics, such as object-oriented programming, modules, and libraries. You'll also learn how to work with external data sources, such as files and databases, and how to develop web applications using Python.

Programming in Python can be a challenging but incredibly rewarding experience. With determination and practice, you'll be able to create powerful programs that can solve complex problems and automate tasks.

Learning Plan Tasks

1. Python Basics
2. Intermediate Python
3. Object Oriented Python
4. Advanced Python
5. Review
6. Build Something Using Python!

#checkoutTheDocs

- **Python**: Official Documentation
- **Python**: Beginner Guides and Documentation
- **Coursera**: Python Crash Course
- **Coursera**: IT Certificate - Automation with Python
- **Codecademy**: Python for Programmers
- **Python for Everybody**
- **YouTube**: Some popular channels include Corey Schafer, sentdex, and Tech With Tim.

1. Python Basics

This section covers the following topics:

- 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.

Name:
Date:

Tasks

1. Understand the Fundamentals of Python Programming
   - ☐ Install Python on your computer.
   - ☐ Create your first Python file in VSCode (or your preferred coding editor).
   - ☐ Use a `print` statement to output your name and `"Hello World"` to the console. Run your file using the command `python3 FILE_NAME`.
   - ☐ Learn about variables, data types, and basic Python syntax.
2. Learn About Python Data Structures
   - ☐ Understand the basics of Python's 3 of the the built-in data types: lists, tuples, and, dictionaries.
   - ☐ Work with conditional statements and loops
   - ☐ Do an exercise involving manipulating data structures. For example, given a list of numbers, double all these numbers using loops and
     - Can't find a dataset? Ask ChatGPT to create a dataset for you!
3. Learn About Functions and Modules
   - ☐ Understand function definition and calling
   - ☐ Do an exercise involving writing and using a custom function
   - ☐ Learn About Python modules and libraries
   - ☐ Find at least one Python module and/or library and write a function that utilizes it
   - ☐ Understand `__init__.py` and `__main__`
4. Build Something
   - **Option #1 - Rock, Paper, Scissors**: You now know enough to rebuild your Rock, Paper, Scissors project! Your project should include the following:
     - Randomly selects rock, paper, or scissors for each player in the game. You should explore how to use Python's Random Module to achieve this.
     - Determines the winner of the game based on the random values that were selected.
     - **BONUS**: Explore how to accept user input through use of the `input()` method.
   - **Option #2 - Guess the Word**: Create a simple game in which the user tries to guess a randomly selected word from a list of words. The program should provide feedback to the user after each guess, indicating whether the guess was correct or not and allow the user to continue guessing until they correctly guess the word.

2. Intermediate Python

This section covers the following topics:

- Advanced Data Structures
- File Handling and Exceptions
- Unit Testing in Python

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

Tasks

1. Advanced Data Structures
   - ☐ Learn about sets and frozensets
   - ☐ Work with nested data structures (a list of lists, a list of dictionaries, etc.)
   - ☐ Find and manipulate an advanced dataset
     - Can't find a dataset? Ask ChatGPT to create a dataset for you!
2. 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
     - E.g. Write a Python program that reads a text file and counts the number of words in it. The program should handle file handling and exception handling to ensure that the file exists and can be accessed without errors.
3. Unit Testing in Python
   - ☐ Install the pytests unit testing library
   - ☐ Go back to the mini-project you made from the last section and create some tests for functions you you wrote.
   - ☐ Use a TDD approach to create some functions by writing the tests ***before*** creating the functions.
     - E.g. Write tests for a function to return the largest element of an array before writing the function itself.
4. Build Something!
   - **Option #1 - Word Frequency Analyzer**: Create a word frequency analyzer that reads a text file, process its contents, and generate a report of the frequency of each word in the file. Be sure to utilize exception handling and write unit tests.
   - **Option #2 - Movie Recommendation System**: Create a movie recommendation system that will suggest movies to users based on their preferences and previously watched movies. The program should read a file containing a list of previously watched movies and their genres and recommends movies based on the values in the file. Be sure to utilize exception handling and write unit tests.

3. Object Oriented Python

This section covers the following topics:

- Classes and Objects
- Inheritance
- Polymorphism
- 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. 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.
     - E.g. Create a `Person` class with a `name`, `age`, and `hometown` properties. The class should also have a `bio` method that prints on the persons information.
2. 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.
     - E.g. Create an `Animal` parent class with multiple animal subclasses that inherit information from the parent class.
3. Polymorphism
   - ☐ Create a program that uses polymorphism through method overriding and/or method overloading.
   - ☐ 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.
     - E.g. Create a shape calculator that calculates the area of different shapes such as rectangles, circles, and triangles each with a `area()` method.
4. 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.
     - E.g. Create a bank account management system. The program allows users to create bank accounts, deposit and withdraw money, and retrieve the account balance. The program should ensure that the internal details and operations of the bank account are hidden and can only be accessed through well-defined methods.
5. Build Something
   - **Option #1 - Scooter Application**: Recreate the Scooter Application from Bootcamp using Python.
   - **Option #2 - Inventory Management System**: Create an online store inventory that allows users to add, update, and view products in an store's inventory. Users should also be able to perform operations such

as checking stock availability, calculating total inventory value, and generating inventory reports.

4. Advanced Python

This section covers the following topics:

- Regular Expressions
- Web Scraping
- 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. 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.
     - E.g. Use regular expressions to validate that email addresses are in a valid format.
2. Web Scraping
   - ☐ Understand the basics of web scraping
   - ☐ Learn how to use the `requests` and `BeautifulSoup` libraries to scrape websites
   - ☐ Utilize the `requests` and/or `BeautifulSoup` libraries to create a program that uses web scraping
3. Data Visualization
   - ☐ Learn about data visualization libraries like Matplotlib and Seaborn
   - ☐ Understand the basics of creating plots and charts
   - ☐ Create a program that creates a data visualization using Matplotlib and/or Seaborn

5. Review

This section covers the following topics:

- Review and Practice
- Explore Additional Python Topics

Tasks

1. Review and Practice
   - ☐ 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

Build Something Using Python!

After completing your Python Learning Plan, pick one of the projects below and build it using your newly learned 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!