I Already Know Python

Bonus Exercises & Resources
for Advanced Practitioners

Learn More

FreeCodeCamp

For those of you already familiar with the Python concepts we're discussing in the course, and looking for more in-depth material or advanced topics, I recommend checking out www.freecodecamp.org. This site offers a wealth of Python resources that go beyond the basics, allowing you to explore more complex projects, algorithms, and applications in Python. It's a great way to deepen your understanding and challenge yourself further in your Python programming journey.

Here are a few courses that you might be interested about:

- Beginner Python Course - A video from Harvard University that introduces Python's most basic concepts.

- Intermediate Python Course - It initiates with basic syntax like printing "Hello, World!", followed by explanations of variables, data types, and handling user input. Control structures such as if-else-elif statements, match-case, and loops are covered. It dives into collections, detailing strings, lists, tuples, and dictionaries, before moving onto functions. Each topic is supported by code snippets and thorough explanations, aiming to provide a practical understanding of Python's basic programming constructs.

- Object Oriented Programming with Python - Core principles like Polymorphism, Inheritance, Encapsulation, and Abstraction are detailed, explaining how they contribute to organized and efficient code. The articles further delve into the practical implementation of OOP through the use of classes, objects, and methods in Python, illustrating how they promote code reusability and manageability. By explaining the goals of OOP and distinguishing between methods and functions, the articles provide a comprehensive understanding of OOP's significance in writing clean, understandable, and extendable code.

- Data Analysis with Python *(+ 12-hour video)* - Data Analysis has been around for a long time. But up until a few years ago, developers practiced it using expensive, closed-source tools like Tableau. But recently, Python, SQL, and other open libraries have changed Data Analysis forever.In the Data Analysis with Python Certification, you'll learn the fundamentals of data analysis with Python. By the end of this certification, you'll know how to read data from sources like CSVs and SQL, and how to use libraries like Numpy, Pandas, Matplotlib, and Seaborn to process and visualize data.

RealPython

Graphical User Interface

GUI in Python refers to the use of libraries or frameworks to create applications with graphical elements like windows, buttons, and text boxes, rather than just text-based interfaces. Common Python libraries for building GUIs include Tkinter and PyQt.

User Interfaces - The learning path on the provided link is designed to enhance your Python GUI programming skills, offering tutorials and courses on different GUI frameworks like Tkinter, PySimpleGUI, PyQt, wxPython, and Kivy. It covers creating various applications, ranging from basic ones like a temperature converter and text editor, to more complex projects like a desktop calculator and a cross-platform mobile application.

Flask

Flask is a lightweight web framework for Python, designed to be easy to use and extend. It provides tools, libraries, and technologies to build web applications, but remains simple and doesn't require specific tools or libraries. It's often chosen for small to medium projects or microservices due to its simplicity and flexibility.

Building a Flask Application - This link guides you on transitioning from a local Python script to a deployed Flask web application. It covers the basics of web applications, converting a Python script into a Flask web application, enhancing user experience with HTML, and deploying the application on Google App Engine.

Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It's known for its "batteries-included" philosophy, providing an extensive set of features out of the box, such as an Object-Relational Mapping, authentication support, an admin interface, and more. It's particularly well-suited for building robust and scalable web applications, including complex database-driven websites.

Django - This tutorial guides you on building a portfolio web application using Django, a Python web framework. It covers understanding Django's advantages, its site architecture, setting up a project with multiple apps, building models and views, connecting Django templates, and uploading images to the site.

Web Scraping

Web scraping is the process of extracting data from websites. This is typically done through automated scripts or programs that simulate a user browsing the web, access web pages, and then extract useful information from these pages, such as text, images, or other data. In Python, libraries like Beautiful Soup, Scrapy, and Selenium are commonly used for web scraping tasks. The process is widely used for data gathering, analysis, and automation of web-based tasks.

Web Scraping - This tutorial offers a practical introduction to web scraping in Python, covering three main areas:

- parsing website data using string methods and regular expressions,
- utilizing an HTML parser to parse website data,
- interacting with forms and other components of a website.

Build a CV and Train for Interviews

LeetCode is an online platform that provides a vast collection of coding challenges and problems aimed at improving programming and algorithmic skills. It's widely used by individuals preparing for technical job interviews in the software industry.

Regular practice on LeetCode demonstrates an ability to tackle complex algorithmic challenges, a skill highly valued in many tech roles. Additionally, Many companies use LeetCode-style questions in their technical interviews. Being adept at these questions signals readiness for such challenges. Thus, you should actively maintain a LeetCode profile and regularly solve problems and participate in contests to improve your ranking.

It is a good idea to **integrate LeetCode into your CV** by solving LeetCode problems and pushing your well-commented, clean code solutions to a **public GitHub repository**. Link this repository in your CV. Mention the number of problems solved, especially if it's significant, the difficulty levels tackled, and your ranking or percentile in contests. Highlight any specific, challenging problems that you solved which demonstrate deep algorithmic understanding or unique Pythonic solutions.

In Python you may want to showcase your skills in one of the three code problem categories:

- Algorithms
- Database
- Data Analysis (pandas)

There are 3 levels of difficulty, start with the easiest ones and then gradually move up to the hard code problems.

Python Projects

Birthday Paradox Simulation

The Birthday Paradox is a famous problem in probability theory. It states that in a group of just 23 people, there's about a 50% chance that at least two people will have the same birthday. This probability surprisingly increases to 99.9% for a group of 70 people. The paradoxical part is how intuitively we might expect the probability of shared birthdays to be much lower.

Why is it Surprising ? Well, intuitively, we might think that since there are 365 days in a year, the chance of two people sharing the same birthday would be quite low in a small group. However, the Birthday Paradox reveals the non-intuitive nature of probability, especially when dealing with large sets and combinations.

Here is a nice video that explains the concept.

Guideline:

1. Write a function to generate birthdays:
    a. Use a loop to generate a specified number of random birthdays.
    b. Store birthdays as datetime.date objects.

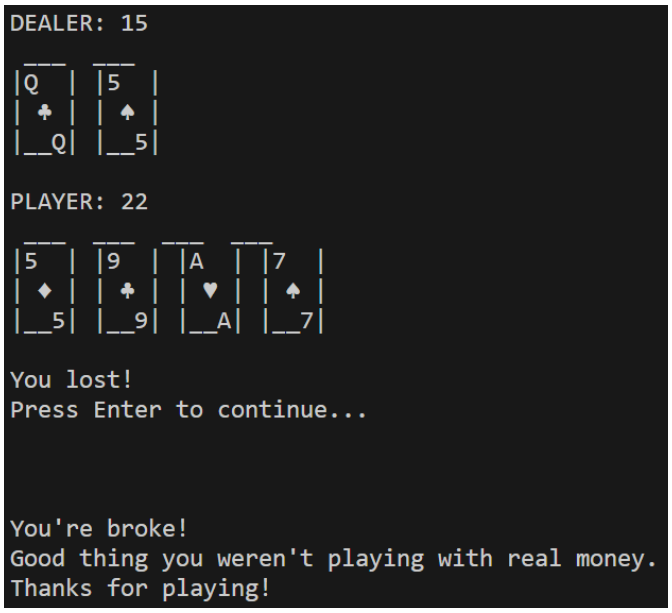
2. Write a function to find a match:
    a. Write a function to check if there's at least one pair of matching birthdays in the provided list.
    b. Consider the efficiency of your algorithm: comparing each birthday to every other can be computationally expensive with a large list.

3. Simulate the paradox:
    a. Repeatedly generate groups of birthdays and check for matches.
    b. Keep track of how many groups had at least one match.
    c. Calculate the percentage of groups with matches.

4. Implement a simple user interaction:
    a. Allow the user to specify the number of birthdays to simulate.
    b. Display the results in an easy-to-understand format.

BlackJack

Blackjack is a card game where each player competes against the dealer. The goal is to get the total card value as close to 21 as possible without exceeding it. Face cards (King, Queen, Jack) are worth 10 points, Aces can be 1 or 11, and other cards are worth their face value.The player can "Hit" to take another card, "Stand" to end their turn, or "Double Down" to double their bet with one more card.



```
DEALER: 15

 ___   ___
|Q  | |5  |
| ♣ | | ♠ |
|__Q| |__5|

PLAYER: 22

 ___   ___   ___   ___
|5  | |9  | |A  | |7  |
| ♦ | | ♣ | | ♥ | | ♠ |
|__5| |__9| |__A| |__7|

You lost!
Press Enter to continue...



You're broke!
Good thing you weren't playing with real money.
Thanks for playing!
```

1. Create something to control the flow of the game (start, rounds, end). At the start of each round, deal two cards to both the player and the dealer from the deck.

2. Decide how you'll represent cards and suits. Tuples can be used for cards such as ('A', 'Hearts') and so on. Write a function to display the cards.

3. Create functions to build and shuffle the deck.

4. Create functions to handle player actions, calculate hand values, and determine game outcomes. Allow the player to Hit, Stand, or Double Down. After both player and dealer stand, compare the hand values to determine the winner.