

ChatGPT: Your Python Coach

Mastering the Essentials in 100 Prompts

Sergio Rojas-Galeano

ChatGPT: Your Python Coach

Mastering the Essentials in 100 Prompts

Sergio Rojas-Galeano, PhD.

Bogotá, Colombia

About the Author

Sergio Rojas-Galeano is an enthusiastic computer scientist with a passion for the cross-roads between Artificial Intelligence, Machine Learning, Evolutionary Computation, Natural Language Processing, Software Engineering, and Scientific Edition. With professional experience from both academic and industry positions in Colombia and England, he is deeply committed to upholding ethical standards and advancing the field of computer science. Currently, he serves as a Full Professor at Universidad Distrital Francisco José de Caldas (Colombia), where he has taught programming and computational intelligence for over two decades. With a PhD in Computer Science from University College London (UCL) in the UK (awarded in 2009), he has also been a visiting scholar at UCL (in 2022) and Universidad de Málaga in Spain (in 2023). He is author or co-author of over 70 scientific papers and has previously published the book, *Models of Learning and Optimisation for Data Scientists* (LeanPub, 2019).

email: sargaleano@gmail.com

License

This document is licensed as:



Permission to use this document is subject to the License; any other use is strictly prohibited. The contents of this document are provided on an "AS IS" basis without warranties of any kind, either express or implied. The author assumes no responsibility or liability for any consequences arising from the use of this document, including but not limited to errors, omissions, or inaccuracies. The reader assumes full responsibility for the application of the concepts and techniques described in this book.

© 2023 by Sergio Rojas-Galeano.

Colophon

This document was typeset using [L^AT_EX](#) and the [kaobook](#) class. The author crafted the content with the help of OpenAI's [ChatGPT](#) content-generation system, based on his own ideas, purpose and style. Cover design was created by the author using templates and art available at the [Canva](#) graphic design platform. All product names, logos, brands, trademarks and registered trademarks are property of their respective owners. All company, product and service names, as well as celebrity, book, character, band, team or movie names mentioned in this book are used solely for identification and educational purposes. Use of these names, trademarks and brands does not imply endorsement.

First Edition

June 2023

Bogotá, Colombia

Preface

Welcome to *ChatGPT: Your Python Coach. Mastering the Essentials in 100 Prompts!* If you're reading this, it means you're about to embark on an exciting journey into the world of programming. But why Python, you may ask? Well, Python is one of the most widely used and versatile programming languages out there, with applications ranging from web development and data analysis to machine learning and artificial intelligence (AI).

In fact, there has been a growing interest in AI and its applications in recent years, including the development of language models like ChatGPT. While there are valid concerns regarding the ethical implications, data biases, and potential for generating fake content, it is worth noting that in the realm of programming, these concerns can be somewhat mitigated as the content generated by ChatGPT primarily consists of code, which can be easily evaluated for its validity and appropriateness by simply running it. This provides an advantage in minimising the risks associated with hallucination or inaccurate information, as the output can be verified through practical execution. As such, language models like ChatGPT have become valuable tools for programming tasks, offering developers enhanced assistance in their coding endeavours.

This guide has been carefully crafted to help you navigate the basic concepts of Python programming with ease, with ChatGPT as your trusty mentor. ChatGPT is a powerful language model designed to assist you with any questions or doubts you may have while learning Python. It will provide you with examples, explanations, and feedback, making your learning experience interactive and engaging.

In this guide, you'll learn the fundamentals of Python programming, including variables, data types, operators, flow control structures such as conditionals and loops, containers such as lists and dictionaries and modularisation and reutilisation concepts. You'll also get to work on fun and exciting challenges that will put your newfound skills to the test.

But this guide is more than just a collection of exercises and examples. It's an invitation to explore the power of programming and its potential impact on our lives, and to see how AI and language models like ChatGPT are expected to leverage the abilities and productivity of programmers to create powerful applications for a promising future. As you delve deeper into the world of Python, you'll witness how programming can be used to solve real-world problems, automate tedious tasks, and innovate products and services.

The majority of the content in this book, including this preface, was initially generated with the assistance of ChatGPT. However, it is important to note that the draft material served as a foundation which I then extensively revised, adjusted, and enriched. This process was driven by my educational expertise, with the goal of creating didactic and informative content for readers. While ChatGPT played a role in generating the initial material, the final content has been carefully curated and tailored to ensure its educational value and correctness.

To promote transparency in this innovative approach to AI-assisted writing, I have deliberately chosen to include a comprehensive list of the 100 prompts I conceived and utilised to feed ChatGPT, ensuring thorough coverage of the intended thematic scope of this guide. This list will be incorporated at the end of the book for easy reference. Furthermore, this *List of Prompts* may empower readers to reproduce a similar yet personalised version of the book that aligns with their individual preferences and requirements.

Besides, I will utilise endnotes to reference each prompt in its corresponding position on the list whenever it is used in the text. For instance, the first prompt employed to generate the initial draft of this preface is referred next¹.

Lastly, I've created a companion repository that houses a collection of Jupyter Notebooks that contain the Python code examples from the book, thoughtfully organised by chapters for convenient browsing and exploration:

[Link not available in sample book](#)

Moreover, starting from Chapter 4, I've added an exciting feature. For the more engaging code examples, I've included links to the [PythonTutor.com](#) website. These links allow you to visualise the step-by-step execution of the code, dynamically observing the program's memory state. Look for these links at the bottom of the code listings and output boxes. It's a fantastic way to enhance your understanding and bring the code to life!

So, if you are eager to learn programming and start creating programs with Python and ChatGPT, I hope you find this guide useful and enjoyable to read.

*Sergio Rojas-Galeano
Bogotá, June 23, 2023*

Contents

Preface	v
Contents	vii
Introduction	3
Is this the right book for you?	3
About Python	3
Programming cycle	4
The tools	5
Using Colab	6
Using PythonTutor	8
ChatGPT as a programming coach	10
Using ChatGPT	11
ChatGPT prompt template	14
1 Know your Basics	15
1.1 Hello, World!	15
1.2 Strings and string formatting	17
1.3 Hands-on time!	22
2 Data Building Blocks	27
2.1 Data types	27
2.2 Basic Input/Output	30
2.3 A coding buddy	32
2.4 Hands-on time!	36
3 The Operations Room	39
3.1 Arithmetic operators	39
3.2 Comparison operators	41
3.3 Logical operators	43
3.4 Bitwise operators	44
3.5 Assignment operators	48
3.6 Identity operators	49
3.7 Memberships operators	51
3.8 Exponentiation operator	52
3.9 Advanced math operators	53
3.10 Hands-on time!	55
4 The Control Tower	59
4.1 If/Else statements	59

4.2	Match-case statements	64
4.3	For loop statements	66
4.4	While loop statements	69
4.5	Nested control structures	74
4.6	Hands-on time!	81
5	The Containers Zone	91
5.1	Lists	91
5.2	Tuples	99
5.3	Sets	103
5.4	Strings	107
5.5	Dictionaries	113
5.6	Hands-on time!	126
6	Modular Affairs	133
6.1	Modularisation and Reusability	133
6.2	Functions	136
6.3	Classes and Objects	143
6.4	Hands-on time!	153
7	Finale	161
7.1	Digging Further	161
7.2	Suggested Readings	167
	List of the 100 prompts	169

Introduction

Is this the right book for you?

This book is designed to help newcomers like you learn the basics of programming with ease, specifically in the Python language, one of the most versatile and widely used programming languages out there.

But this book is more than just a typical programming guide. It's like having a dedicated coach by your side, guiding you through the learning journey with examples, explanations, and feedback. And that coach happens to be an AI language model - ChatGPT. With ChatGPT as your guide, you'll have access to a powerful tool that will help you understand the essential programming concepts. You will receive real-time feedback that will help you learn faster and more effectively, ultimately helping you to become a proficient programmer.²

This book serves as an introductory guide to fundamental concepts in Python, including algorithms, flow control structures, containers, and modularisation. Mastering these basics will provide you with a solid foundation for tackling more advanced topics in future volumes. So, embrace these fundamental concepts and gear up for the thrilling challenges that await you. In the meantime, this chapter introduces the tools that will be used throughout the book. If you are already familiar with these tools and want to dive straight into coding, feel free to skip this chapter and jump to Chapter 1.

About Python

Python is a programming language that's as easy as pie – or rather, as easy as writing down a recipe for your favourite dessert! It was created in the late 1980s by a Dutch programmer named Guido van Rossum, who named it after the Monty Python comedy troupe.³

Today, Python is used by everyone from NASA to Instagram to build everything from space rovers to photo filters. It's known for its simplicity, readability, and versatility. In fact, it's one of the most popular programming languages in the world, and for good reason. It's a language that's easy to learn but has powerful capabilities, making it ideal for beginners and experts alike.

Chapter contents

Is this book for you?.....	3
About Python.....	3
Programming cycle.....	4
The tools.....	5
Using Colab.....	6
Using PythonTutor.....	8
ChatGPT as a coach.....	10
Using ChatGPT.....	11
Prompt templates.....	13

Monty Python

Monty Python is a British comedy group known for their unique and absurdist humour. Their television show, Monty Python's Flying Circus, aired from 1969 to 1974 and gained a cult following. They also created several popular films, including Monty Python and the Holy Grail and Life of Brian. Their influence on comedy and pop culture has been significant and continues to be celebrated by fans worldwide.⁴

ChatGPT as a programming coach

Disclosure

This book was written with the assistance of ChatGPT. Throughout the book, pieces of automatically generated content served as a draft, which I then reviewed, verified, adjusted, remixed, and augmented until they reached their final form. For the sake of transparency, I've also included the prompts I used to feed the model to generate the drafts, which I labelled with a superscript number directly following the final text fragment where they were used. These numbers indicate their position in the *List of Prompts* provided at the end of the book.

ChatGPT is a language model created by OpenAI that is designed to communicate with humans using natural language. It is a type of artificial intelligence that can understand and respond to text input in a conversational way. It has been trained on a vast amount of data to be able to understand and generate natural language, including structured high-level computer languages, making it an excellent tool to help students learn programming in a conversational entertaining way.¹²

As a student learning programming, ChatGPT can serve as a helpful coach. You can ask it questions about programming concepts, ask for examples of code, or even get feedback on your own code. It can provide explanations and clarifications on programming topics and serve as a tool for practising coding skills.

Concretely, here are some ways ChatGPT can help you learn programming:

- ▶ It can provide explanations and examples of programming concepts and syntax, helping you to better understand them.
- ▶ It can answer coding questions, such as why a particular syntax rule is needed or how to carry out a specific task.
- ▶ It can help you debug your code by suggesting possible solutions and explaining why certain errors occur.
- ▶ It can assist you in learning how to write efficient and optimised code by providing feedback on your code's performance.
- ▶ It can suggest resources and learning materials to help you further your understanding of programming.
- ▶ It can help you document your code so that it is easier for other programmers to understand or for you to remember in future occasions.
- ▶ It can translate code between different programming languages to create new versions of your programs.
- ▶ It can provide contextual references about the meaning, history, or anecdotes related to the development of programming and computer science in general.

Although ChatGPT can be a valuable tool for learning programming, it should not be relied on as the sole source of information or feedback. It's important to use it responsibly, cross-check responses with other sources, and be aware of potential inaccuracies or biases in the data used to train this AI language model.

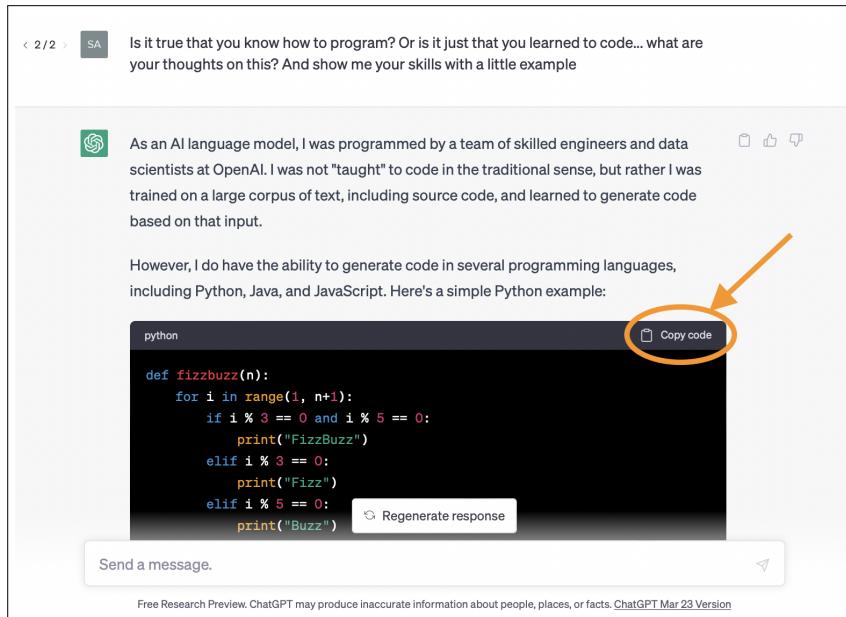


Figure 10: ChatGPT chat page.

In this book, I will use ChatGPT as described before. To make the material more readable, I will format the prompts, explanatory text, code and output of the response, using different coloured text boxes. For the sake of illustration, I have reproduced the above screenshot in the aforementioned formatted style:

Is it true that you know how to program? Or is it just that you learned to code... what are your thoughts on this? And show me your skills with a little example.

As an AI language model, I was programmed by a team of skilled engineers and data scientists at OpenAI. I was not "taught" to code in the traditional sense, but rather I was trained on a large corpus of text, including source code, and learned to generate code based on that input. Here's an example with its output:

```

1 def fizzbuzz(n):
2     for i in range(1, n+1):
3         if i % 3 == 0 and i % 5 == 0:
4             print("FizzBuzz")
5 ...

```

1
2
Fizz
4
Buzz
...

1 Know your Basics

1.1 Hello, World!

Explain the concept of the 'Hello, World!' program in a simple but colorful manner, without using any code. Additionally, include a relatable anecdote to provide beginners in programming with a warm introduction to the topic.¹⁴

Ah, the legendary "Hello, World!" program! This is the perfect way to kick off your journey into programming, and it's an experience that you'll always remember. Imagine yourself sitting at your computer, feeling a mix of excitement and nervousness. You're about to write your first Python program ever!

The "Hello, World!" program is the simplest program you can write, but it's a significant event. It's the equivalent of a newborn baby's first cry or a musician's first note on a new instrument. This program is the foundation of all programming languages, and it's the perfect starting point for anyone who wants to learn to code.

Now, you might be wondering, "Why on earth would I want to write a program that just says 'Hello, World!'?" Well, it's not about the message itself, but rather about the process of writing the program. This simple program is a great way to get familiar with a new language, overcome the fear of starting something new, and gain the confidence to continue learning and exploring.

But why the message "Hello World"? Who invented it? What's the meaning?¹⁵

The origin of the "Hello, World!" program is somewhat unclear, but it is believed to have originated from the Bell Labs in the early 1970s. The first recorded instance of the phrase being used in a programming context was in a 1974 book about the programming language C by Brian Kernighan and Dennis Ritchie.

As for the meaning behind the message, it is simply a way for programmers to test that their program is functioning properly and able to display output. It's a tradition and a rite of passage for beginners in the programming world, signalling the start of their journey towards becoming proficient programmers.

Well, enough with the words. Let's dive into the code and begin the ChatGPT coaching experience.

Ready, set, code!

Write the classic "Hello, World!" program in Python and explain it. Provide the output as well. I will be using Colab.¹⁶

Here's the classic "Hello, World!" program in Python:

```
1 print("Hello, world!")
```

The `print()` function is a built-in function in Python that allows you to output text or other data to the console, in this case the text (also known as *string*) "Hello, World!". The output should look like this:

Hello, world!

To run this program in Google Colab, follow these steps:

1. Click on the "Copy code" button on ChatGPT.
2. Paste it in the first cell of your notebook in Colab.
3. Click the "play" button next to the cell or press "Shift + Enter"

Explain what a syntax error is and demonstrate 3 possible syntax errors in the 'Hello, World!' program with their respective outputs.¹⁷

When writing a program, you need to follow the correct syntax of the programming language. A syntax error occurs when you make a mistake in the syntax of the code, such as missing a punctuation mark, using the wrong case for a keyword, or typing an incorrect operator. These errors prevent the code from being interpreted correctly. Here are three possible syntax errors that could have occurred in the "Hello, World!" program:

1. Missing Parenthesis:

```
print("Hello, World!"  
      File "<stdin>", line 1  
      print("Hello, World!"  
            ^  
      SyntaxError: unexpected EOF while parsing
```

The error message tells us that the unexpected end of file (EOF) was encountered while parsing the code.

2. Misspelled Keyword:

```
prnt("Hello, World!")  
      File "<stdin>", line 1  
      prnt("Hello, World!")  
            ^  
      NameError: name 'prnt' is not defined
```

Data Building Blocks 2

2.1 Data types

*What exactly is a program? How does it relate to data?*³⁰

A program is a set of instructions that tells a computer what to do. It is a sequence of statements that are executed by the computer in order to accomplish a specific task. Programs are written in a programming language, which is a set of syntax rules that define how the instructions are to be written.

Data is any kind of information that can be stored and processed by a computer. It can be numbers, text, images, audio, or video. Data is used by programs to perform their tasks. For example, a program that calculates the average of a set of numbers will need to read the numbers from a data file or ask the user to enter them.

The relationship between programs and data is that programs can be used to create, store, retrieve, and manipulate data. They can also be used to analyse data and to generate new information.

Here are some examples of how programs and data are related:

- ▶ A word processor uses data (text) to create documents.
- ▶ A spreadsheet uses data (numbers) to calculate values.
- ▶ A database uses data (records) to store information.
- ▶ A web browser uses data (web pages) to display information on a screen.
- ▶ A game uses data (characters, levels, etc.) to create an interactive experience.

*Ok. Explain with examples and outputs how data is represented in Python. Provide examples from comics characters to better understand the concept.*³¹

In Python, data is represented by assigning values to variables. These values can be of different types, such as strings, integers, and booleans. Here are some examples:

```
1 # Assigning a string to a variable
2 hero = "Spider-Man"
3 print("Character:", hero)
4 # Assigning an integer to a variable
5 age = 23
6 print("Age:", age)
7 # Assigning a boolean to a variable
8 is_villain = False
9 print("Is villain?", is_villain)
```

2.3 A coding buddy

ChatGPT is not only capable of suggesting coding exercises but can also solve them all by itself! This amazing feature opens up a whole new world of exciting learning possibilities for you! You can challenge yourself by asking ChatGPT to propose a coding problem. Once you've solved it using your own knowledge and skills, you can compare your solution with ChatGPT's. This is where the real fun begins! You can get feedback from ChatGPT on the differences between your solution and its own. You can inquire about which solution is more efficient, or learn more about new techniques and tricks that you didn't know before but ChatGPT's got covered! Just imagine how much fun you'll have with your new coding buddy, ChatGPT!

Let's see an example of how this works.

Hi there, ChatGPT! I'd like to practice my coding skills with an exercise that involves user input, data type conversion, and printing output, maybe even including a menu of options. Could you please suggest an exercise for me to work on? Once I have the statement, I'd also appreciate it if you could show me an example of the expected output. Thanks!³⁵

Hi, budding programmer! Let's practice your user input, data type conversion, and printing skills with an entertaining exercise. Write a program that asks the user for at least 5 pieces of personal information, such as name, age, and hobbies, etc. Then, offer the user a menu of two options: "1-Short Bio" and "2-Long Bio". Depending on the user's choice, your program should print either a short or long biography. Get creative with your output and have fun! Here's an example of the output:

Welcome to the Personal Bio Program!

Please enter your name: John
 Please enter your surname: Smith
 Please enter your age: 25
 Please enter your favorite hobby: books
 Please enter your favorite food: pizza
 Please enter your favorite color: red

Menu of options:

1. Short Bio
2. Long Bio

Enter your choice (1 or 2): 1

--- Short Bio ---

John, 25, is a programming student whose favourite hobby is reading books.

The Operations Room

3

List all of the Python operators and provide a brief description for each.⁴²

Here is a list of different types of Python operators and their definitions:

1. **Arithmetic operators:** used for performing mathematical operations, such as addition, subtraction, multiplication, division, etc.
2. **Comparison operators:** used for comparing values and returning a Boolean value (True or False), such as equal to, not equal to, greater than, less than, etc.
3. **Logical operators:** used for combining multiple conditions and returning a Boolean value, such as and, or, not.
4. **Bitwise operators:** used for performing operations on individual bits of binary numbers, such as AND, OR, XOR, etc.
5. **Assignment operators:** used for assigning values to variables, such as `=`, `+=`, `-=`, `*=`, `/=`, etc.
6. **Identity operators:** used for comparing the memory location of two objects, such as `is`, `is not`.
7. **Membership operators:** used for checking if a value is present in a sequence, such as `in`, `not in`.
8. **Exponentiation operator:** used for raising a number to a power, represented by `**`.

So, let's take a closer look at each of them.

3.1 Arithmetic operators

Explain the details of arithmetic operators in Python in a colourful tone and include a single code example to illustrate the use of all the operators. The example should compare entertaining statistics about Marvel vs X-men movies and box office.⁴³

In Python, arithmetic operators are used to perform basic mathematical operations such as addition, subtraction, multiplication, and division. These operators are used to manipulate numerical data types such as integers and floating-point numbers.

Here are the key arithmetic operators in Python:

- **Addition:** The plus sign (+) is used for addition. It adds two values together. For example, $2 + 2$ equals 4.
- **Subtraction:** The minus sign (-) is used for subtraction. It subtracts the second value from the first value. For example, $5 - 2$ equals 3.

The Control Tower 4

List all of the control structures in programming and provide a brief description for each.⁵⁹

In programming, there are several control structures that allow you to control the flow of execution in your code. These control structures can be categorized into three main types: branching structures, repetition structures, and nested structures.

1. Branching Structures (Conditionals):

- ▶ **If-Else Statement:** Executes a block of code if a certain condition is true. It can also include an optional `else` statement to provide an alternative block of code to execute if the condition is false.
- ▶ **Match-case Statement:** Evaluates an expression and matches it against multiple cases, executing the code block associated with the matched case.

2. Repetition Structures (Loops):

- ▶ **For Loop:** Executes a block of code for a specific number of iterations. It is often used when the number of iterations is known in advance.
- ▶ **While Loop:** Executes a block of code repeatedly as long as a specified condition is true. It is useful when the number of iterations is not known beforehand.
- ▶ **Do-While Loop:** Similar to the `while` loop, but it guarantees that the code block is executed at least once before checking the condition.

3. Nested Structures:

- ▶ **Nested Conditionals:** Allows you to have an `if-else` statement within another `if-else` statement. It provides more complex branching based on multiple conditions.
- ▶ **Nested Loops:** Involves placing one loop inside another loop. This allows for more intricate control over repetitive tasks.

These control structures give you the ability to make decisions, repeat actions, and conceive more intricate program logic. By combining these structures effectively, you can create powerful and flexible programs to meet various requirements.

So, let's take a closer look at each of them.

4.1 If/Else statements

Explain everything I need to know about `if/else` statements in Python in a fun and creative way, with playful examples.⁶⁰

Sure! Let's dive into the exciting world of conditionals statements in Python.

4.5 Nested control structures

What is a nested control structure and why is it useful?⁷⁰

A nested control structure is a control structure (such as an `if` statement, `for` loop, or `while` loop) that is placed within another control structure. It allows for multiple levels of branching or looping within a program.

Nested control structures are powerful tools in programming that allow for the handling of complex logic and the execution of advanced tasks. They enable the creation of intricate decision-making processes and iterative operations, providing flexibility and adaptability to programs. With nested control structures, you can design code that responds to multiple conditions, performs repetitive tasks with varying complexity, and implements sophisticated algorithms. This dynamic approach makes programs more efficient and capable of solving complex problems, as they can adapt to changing conditions, make decisions based on multiple factors, and execute iterative operations at different levels of depth.

However, it's important to use nested control structures judiciously and maintain code readability. As the depth of nesting increases, code can become harder to understand and debug. Therefore, it's crucial to strike a balance between using nested control structures to tackle complex scenarios and keeping the code maintainable and comprehensible.

Let us now dive into nesting control structures in Python. Show me how to use nested conditionals allowing for multiple levels of branching, nested loops including nested double for loops, double while loops and mixed while/for or for/while loops, as well as combinations of loops within conditionals and conditional within loops. Use examples from superheroes and comics to be creative and playful in your explanation.⁷¹

Sure! Let's dive into the exciting world of superheroes to exploring the concept of nested control structures in Python.

1. *Superhero vs Supervillain.* Here's an example program that asks the user to choose a superhero and supervillain, and then determines who beats who using two-level nested conditionals.

In this example, the nested conditionals allow you to handle various combinations of superhero and super-villain choices. Each nested block checks the selected superhero and super-villain, and based on the combination, it prints a corresponding "who beats who" message. If the chosen combination does not match any of the nested conditionals, it displays an appropriate error message.

Let's take a look at the code:

Explain what a container structure is, then list all of the container structures available in Python and provide a brief description for each, using real-life analogies and a friendly tone.⁷⁶

A container structure in Python is a data structure that is capable of holding multiple values or objects. It allows you to organise and store data in a structured manner. Each container structure has its own unique characteristics and is suitable for different purposes.

1. **Lists:** Think of a shopping bag where you can put various items. It is an ordered collection of elements that can be of different types.
2. **Tuples:** Imagine a sealed envelope that contains multiple pieces of information. A tuple is similar to a list, but it is immutable, meaning it cannot be modified once created.
3. **Sets:** Picture a drawer filled with unique items. A set is an unordered collection of unique elements.
4. **Strings:** Consider a string as a sentence or a sequence of characters. It is an immutable container that holds a collection of characters.
5. **Dictionaries:** Visualise a real-life dictionary where words and their definitions are stored. A dictionary is a key-value pair data structure.

These container structures provide flexibility in organising and manipulating data based on your specific needs. By choosing the right container structure, you can efficiently store and process data in your Python programs.

So, let's take a closer look at each of them.

5.1 Lists

Explain everything I need to know about lists in Python in a fun and creative way, using playful examples.⁷⁷

Sure! Let's embark on a playful adventure into the world of Python lists! Imagine you are a wizard with a magical backpack that can hold all sorts of fantastic items. In Python, lists are just like your magical backpack, where you can keep, add, remove and retrieve items at your convenience!

1. **Creating Lists:** To create a list, imagine it as your magical backpack. You can put anything you want inside, like shiny gems, powerful potions, or enchanted scrolls. Just use square brackets [] and separate the items with commas.

Ok. Now let's look at some code samples to see tuples in action! This time, why not use rock bands to illustrate the items you listed previously?⁸²

Sure! Let's dive into some code examples using rock bands to illustrate the different aspects of tuples. We'll explore each item previously mentioned in a fun and creative way.

1. **Immutable Lineup:** Say you are organising a rock concert, you can use a tuple to define the lineup of bands. Once the lineup is set, it remains fixed throughout the concert. Let's see an example:

```
1 lineup = ("The Beatles", "Led Zeppelin", "AC/DC", "Guns N' Roses", "Queen")
2 print(lineup)
3
4 # Uncommenting the line below will raise an error
5 # lineup[0] = "Coldplay"
```

Once the arrangement is set using a tuple, it cannot be changed. Attempting to modify an element of the tuple will result in an error, because tuples are immutable, just like trying to replace a band after the concert has started.

```
==== Immutable Lineup ====
('The Beatles', 'Led Zeppelin', 'AC/DC', "Guns N' Roses", 'Queen')
```

2. **Ordered Placement:** When planning a concert, you want to ensure that the bands perform in a specific order. Tuples preserve the order of elements. Let's see how it looks:

```
1 lineup = ("The Beatles", "Led Zeppelin", "AC/DC", "Guns N' Roses", "Queen")
2
3 for band in lineup:
4     print("Next band:", band)
```

In this example, the bands perform in the order defined by the tuple. The loop iterates over the lineup, and each band is printed in the correct order.

```
==== Ordered Placement ====
Next band: The Beatles
Next band: Led Zeppelin
Next band: AC/DC
Next band: Guns N' Roses
Next band: Queen
```

3. **Mix and Match:** Rock bands can have diverse lineups with members playing different roles. Tuples allow you to create complex structures by combining different types of data. Let's see an example with the Beatles:

6.1 Modularisation and Reusability

Explain briefly the concept of modularisation and reusability, as well as why they are useful in programming projects. Then, using real-world analogies, explain the various approaches for implementing these notions in a friendly but neat manner.⁹⁴

Reusability and modularisation are like secret tricks of the trade in software development, especially for big and complex projects. They come with a bunch of benefits that make life easier for programmers.

When you write reusable code, you don't have to start from scratch every time. You can save lots of time and effort by using existing solutions instead of reinventing the wheel. And with modularisation, you break down your code into smaller, manageable pieces. It's like having puzzle pieces that fit together perfectly. This makes it easier to understand, maintain, and fix any bugs that pop up. Plus, you can reuse those modular pieces in different parts of your project.

So, when it comes to tackling those larger projects, embracing reusability and modularisation is key. It helps you work smarter, collaborate seamlessly, and keeps your code shining bright for the long run. In Python, you have a bunch of cool techniques and practices to put these ideas into action:

- 1. Functions:** Functions are like specialised tools in a kitchen. They encapsulate specific tasks and can be reused across different recipes. Similarly, in programming, functions encapsulate specific operations, promoting code reusability. By utilising functions, programmers can write modular code that can be easily reused, making it easier to handle larger and more complex projects.
- 2. Classes and Object-Oriented Programming (OOP):** Classes in OOP are like specialised workers on a construction site. Each worker has a specific role and expertise. Similarly, classes encapsulate data and behaviour into objects, promoting code reuse. By leveraging classes and objects, programmers can create modular and scalable code, making it easier to manage complexity in bigger projects.
- 3. Modules and Packages:** Modules and packages are like toolboxes with labelled compartments for different tools. They provide a way to organise and package related code components, promoting code reusability. By using modules and packages, programmers can easily find and reuse code functionalities, allowing for efficient development and maintenance of larger projects.

```
20 for friend in filtered_friends:
21     print(friend)
```

Example 1 demonstrates the use of a lambda function as the key parameter in the `sorted()` function. The lambda function `lambda x: len(x)` takes a friend's name as input `x` and returns the length of the name using `len(x)`. This lambda function serves as the basis for sorting the friends list by the length of their names.

Example 2 showcases the use of a lambda function as the filtering condition in the `filter()` function. The lambda function `lambda x: x.startswith('H')` takes a friend's name as input `x` and returns a boolean value indicating whether the name starts with 'H', using `startswith('H')`. This lambda function is used to filter the friends list and include only those friends whose names start with 'H'.

```
==== Sorted Friends by Name Length ====
Ron Weasley (11)
Harry Potter (12)
Draco Malfoy (12)
Hermione Granger (16)

==== Friends with Names Starting with 'H' ====
Harry Potter
Hermione Granger
```

↗ Visualise a step-by-step execution of this code on PythonTutor: <https://tinyurl.com/3jkk8meu>

Lambda functions are compact and inline functions that are useful for simple, one-line operations. They provide a convenient way to define functions on the fly without the need for a formal `def` statement.

6.3 Classes and Objects

Ok. So let's continue with some code examples to see classes in action! Consider using classic movies and characters trivia to illustrate the items you listed previously, including random features.⁹⁷

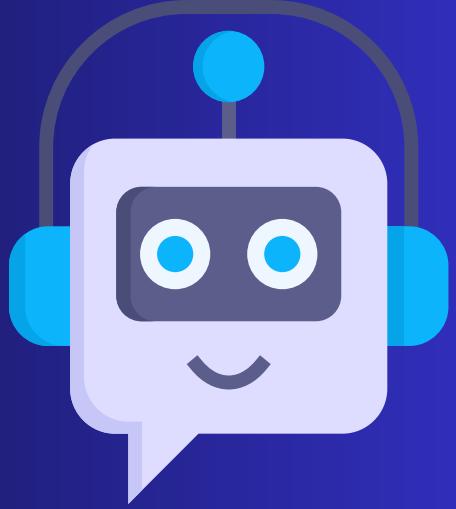
Certainly! Below are a few code examples that demonstrate the use of classes in the context of movies trivia.

- **Mighty Templates:** Classes are like mighty templates for creating objects with predefined structures. They define the overall structure and behaviour of objects. Once you have defined a class, you can use it to create as many instances or objects from that class as you need. Each object will have its own set of data and will behave according to the template defined by the class.

The End

List of the 100 Prompts

1. "ChatGPT: Your Python Coach. Mastering the Essentials in 100 Prompts" is the title of a book I'm writing to teach the fundamentals of Python. Provide a draft version of an engaging and thought-provoking Preface for it. Emphasise the impact AI, and language models in particular, are having in many areas, including programming, and how it is expected to leverage programmers' abilities and productivity to create powerful applications in the near future, mentioning in passing that the issues regarding fake content and hallucination that have been detected on these language models are alleviated in the area of programming because the generated code can be validated by simply running it. Comment also that the initial generated content has been carefully revised and enriched based on my educational expertise.
2. Explain with a colourful tone that the purpose of this book is to initiate a newcomer to the world of programming, in particular, to the Python language, and to help them learn the basic concepts with examples, explanations, and feedback, just like a teacher guides a student through their learning journey. Only that here, that teacher is ChatGPT. Emphasise the new possibilities and advantages of using modern developments in AI to assist the learning and development process of programmers.
3. Provide a witty description of the Python programming language that is suitable for a first-year college student. Make some notes on the language's history and current version, speculate on its future, and provide useful links to organisations or websites where the reader can obtain more technical information.
4. Write a short note on Monty Python.
5. Briefly describe the key differences between Python 2.x and Python 3.x versions. Highlight the latest version's advantages.
6. Create a scheme of the standard cycle of program development, using Latex smartdiagram package with the circular diagram type.
7. Explain to a first-year college reader the standard cycle of program development, including the steps of writing (coding), running, testing, and correcting (debugging) (do it with a bulleted list).
8. Clarify that Python is an interpreted computer language; explain how it differs from a compiled language (advantages and weaknesses). Mention that there are a variety of IDEs available, but in this book, rather than dealing with software installation hassles, I will rely on ready-to-use environments that meet the needs of this book, so we can focus on concepts and algorithms. Jupiter Notebooks, Colaboratory, PythonTutor, and ChatGPT will be used as tools. Make a bullet list of these tools, including a brief description, website, and the steps required to open an account if one is required.
9. Describe in detail how to create a new Jupiter Notebook in Google Colab.
10. Explain the two types of cells in Jupyter Notebook.
11. Explain in detail how to use PythonTutor.
12. Explain to a first-year college student what ChatGPT is, how it can help students learn programming, how to use it responsibly to achieve good learning goals, and whether there are any ethical concerns about using it or relying too heavily on it and how to address them.



This guide has been carefully crafted to help you navigate the basic concepts of Python programming with ease, with ChatGPT as your trusty mentor.

ChatGPT is a powerful language model designed to assist you with any questions or doubts you may have while learning Python. It will provide you with examples, explanations, and feedback, making your learning experience interactive and engaging.