* No code comments

A good code needs no comment. The variables, methods, and any other component of the code such as attributes should use easily identifiable and descriptive names. “Code comments are smell, remove them.”

* Dead comments or code should be deleted

Any unused piece of code or comments should be deleted. The best place to find them is in the version control systems.

* Incorrect behaviour at boundaries.

Boundaries should always be unit tested. No behaviour should be assumed.

* Positive conditionals

Positive conditionals are easier to read than negative conditionals.

* Standard architecture, coding, and design guidelines should be followed.

All the set standard architecture, coding, and design guidelines should be adhered to. Tools should also be used to ensure that this is accurate.

* Good and Consistent naming

Try to call variables, methods, classes to be understandable from the caller perspective. And it should be consistent.

* KISS principle should be applied

Needless complexity should be avoided. With increase in complexity, some design and codes in the system become useless. All designs and codes should be kept as simple as possible.

* Use of exceptions instead of return codes

In exceptional cases, an exception should be thrown, when the method fails to do its intended purpose. Null or return codes shouldn’t be used.

* Make things small. Class size and method size both matter

Typically, method should be small and compact in size. Ideally

* 1. Introduction to Python
* What is Python?
* Setting Up the Environment
* Your First Python Program
* 2. Basics of Python
* Variables and Data Types
* Operators
* Strings and Lists
* 3. Control Structures
* Conditional Statements
* Loops
* Error Handling
* 4. Functions and Modules
* Defining Functions
* Function Arguments
* Modules and Packages
* 5. Advanced Topics
* Object-Oriented Programming
* File Handling
* Working with Libraries
* 6. Data Management
* Introduction to Databases
* SQL Basics
* Data Analysis with Pandas
* 7. Web Development
* Introduction to Web Frameworks
* Building Web Applications with Flask/Django
* Working with APIs
* 8. Final Projects
* Building a Personal Project
* Contributing to Open Source
* Preparing for Job Interviews

# What is Python?

## Introduction to Python

Python is a high-level, interpreted programming language known for its simplicity and readability. Developed by Guido van Rossum and first released in 1991, Python emphasizes code readability with its use of significant indentation.

### Key Features of Python:

* **Easy to Learn and Use:** Python's syntax is clear and intuitive, making it an excellent choice for beginners.
* **Versatile:** Python can be used for a wide range of applications, including web development, data analysis, artificial intelligence, scientific computing, and more.
* **Extensive Libraries and Frameworks:** Python has a rich ecosystem of libraries and frameworks that facilitate various types of programming tasks.
* **Community Support:** Python has a large and active community, which contributes to a wealth of resources and documentation.
* **Cross-Platform:** Python is compatible with multiple operating systems, including Windows, macOS, and Linux.

### Popular Uses of Python:

* **Web Development:** Frameworks like Django and Flask are used for creating web applications.
* **Data Analysis:** Libraries like Pandas and NumPy are used for data manipulation and analysis.
* **Machine Learning and AI:** Tools such as TensorFlow and scikit-learn are used for building machine learning models.
* **Automation:** Python scripts are often used for automating repetitive tasks.

# Variables and Data Types: Questions

## Questions and Answers

### 1. What is the result of adding an integer and a string?

**Code:**

integer\_value = 10 string\_value = "20" result = integer\_value + int(string\_value) print(result)

**Hint:** You need to convert the string to an integer to perform the addition.

**Riddle:** I am not a number, but I become one when you change me. What am I?

**Short Story:** John was a novice coder who often struggled with combining different types of data. One day, he decided to add a number to a string, only to find himself in an unexpected situation. With a little conversion magic, John discovered that mixing data types could be both fun and educational.

### 2. How do you check the type of a variable in Python?

**Code:**

variable = 3.14 print(type(variable))

**Hint:** This function will reveal the type of the variable you pass to it.

**Riddle:** I tell you what something is without looking inside. What am I?

**Short Story:** Sarah was working on a project and needed to confirm the type of her variables. She learned that a simple function could reveal the exact nature of her data, making her debugging process much smoother.

### 3. What is the difference between a list and a tuple?

**Code:**

list\_example = [1, 2, 3] tuple\_example = (1, 2, 3) list\_example[1] = 4 # This works tuple\_example[1] = 4 # This will raise an error

**Hint:** Think about mutability when choosing between these two data structures.

**Riddle:** I am a collection that you can change. My twin is fixed and unchanging. Who am I?

**Short Story:** Emily was organizing her data and faced a dilemma: should she use a list or a tuple? By understanding their properties, she realized that lists were flexible, while tuples provided stability and immutability.

### 4. How can you concatenate two strings in Python?

**Code:**

string1 = "Hello" string2 = "World" concatenated\_string = string1 + " " + string2 print(concatenated\_string)

**Hint:** Use the operator that joins two strings together.

**Riddle:** I bring things together with a simple symbol. What am I?

**Short Story:** Alex wanted to create a greeting message by joining two strings. With a simple operator, he effortlessly combined them into a single message, showcasing how powerful string manipulation can be.

### 5. What is the output of the following code snippet?

**Code:**

number = 5 result = number \* 2 print(result)

**Hint:** Multiplying a number by 2 will double its value.

**Riddle:** If you take me and double me, I grow bigger. What am I?

**Short Story:** Mia was learning basic arithmetic operations in Python and discovered how multiplying a number could instantly change its value. This simple operation helped her understand fundamental concepts better.

### 6. How do you convert a float to an integer?

**Code:**

float\_value = 9.7 int\_value = int(float\_value) print(int\_value)

**Hint:** Use the function that truncates the decimal part of the float.

**Riddle:** I shorten a number by removing its fractional part. What am I?

**Short Story:** Liam needed to convert a floating-point number to an integer for his calculations. By using a conversion function, he successfully eliminated the decimal part and simplified his data.

### 7. What will be the result of the following operation?

**Code:**

text = "Python" number = 3 result = text \* number print(result)

**Hint:** Multiplying a string by a number will repeat the string.

**Riddle:** If you repeat me enough times, I’ll become longer. What am I?

**Short Story:** Olivia wanted to create a repetitive pattern using Python. By multiplying a string with a number, she was able to repeat the string as many times as needed, demonstrating the power of string operations.

### 8. How do you check if a variable is of type list?

**Code:**

variable = [1, 2, 3] is\_list = isinstance(variable, list) print(is\_list)

**Hint:** Use the function that checks if a variable matches a specified type.

**Riddle:** I can tell you if something belongs to a specific type. What am I?

**Short Story:** Daniel was working with different data types and needed to verify if a variable was a list. By using the right function, he could confidently check the type of his data and proceed with his work.

### 9. What is the difference between `==` and `is` in Python?

**Code:**

a = [1, 2, 3] b = [1, 2, 3] print(a == b) # True, because the contents are the same print(a is b) # False, because they are different objects

**Hint:** `==` checks for equality of values, while `is` checks for identity of objects.

**Riddle:** One checks if two things are the same inside, while the other checks if they are the same object. What are they?

**Short Story:** Laura encountered confusion while comparing objects in Python. She learned that `==` was used for value comparison, while `is` checked if two variables pointed to the same object in memory.

### 10. How do you create a dictionary in Python?

**Code:**

my\_dict = {"name": "Alice", "age": 25} print(my\_dict)

**Hint:** Use curly braces and key-value pairs to create a dictionary.

**Riddle:** I hold pairs of keys and values, like a small data store. What am I?

**Short Story:** Kevin needed a way to store related data together, like names and ages. He discovered dictionaries in Python, which allowed him to pair keys with corresponding values, making data management easier.

### 11. How do you convert a string to a float in Python?

**Code:**

string\_value = "3.14" float\_value = float(string\_value) print(float\_value)

**Hint:** Use the function that transforms a string representation of a number into a float.

**Riddle:** I turn a text number into a floating point. What am I?

**Short Story:** Emma needed to process some numeric data from a text file. She found out that converting the string representation of numbers to floats helped her perform accurate calculations.

### 12. What is the output of the following code?

**Code:**

x = 10 y = 20 print(x > y)

**Hint:** This code compares two numbers and checks if the first is greater than the second.

**Riddle:** I compare two values and show if one is greater. What am I?

**Short Story:** Mark was learning about comparison operators and used them to compare values. This simple comparison helped him understand how to make decisions based on numerical conditions.

### 13. How can you extract the integer part of a float?

**Code:**

float\_value = 7.89 integer\_part = int(float\_value) print(integer\_part)

**Hint:** Use a function to remove the decimal part of a float.

**Riddle:** I leave behind the decimal part and keep only the whole number. What am I?

**Short Story:** Alice was dealing with float numbers but only needed the integer part for her calculations. By using a conversion function, she was able to extract just what she needed.

### 14. What is the result of the following code?

**Code:**

x = "10" y = 5 result = int(x) + y print(result)

**Hint:** Convert the string to an integer before performing the addition.

**Riddle:** I mix a text number with a real number and add them together. What am I?

**Short Story:** Liam had a string that represented a number and wanted to add it to another integer. By converting the string to an integer, he managed to perform the addition without errors.

### 15. How do you create an empty list in Python?

**Code:**

empty\_list = [] print(empty\_list)

**Hint:** Use square brackets to initialize an empty list.

**Riddle:** I start with nothing but can hold many items. What am I?

**Short Story:** Sarah wanted to start a new project with a blank slate. By creating an empty list, she had a fresh container to add and organize her data as her project progressed.

### 16. What will the following code output?

**Code:**

value = "Hello" print(value \* 3)

**Hint:** Multiplying a string by a number will repeat the string.

**Riddle:** I can repeat myself multiple times when you ask. What am I?

**Short Story:** John found that repeating a string was useful for his text-based game. By multiplying the string, he quickly generated repeated patterns for his game’s interface.

### 17. How do you check if a variable is of type integer?

**Code:**

variable = 42 is\_integer = isinstance(variable, int) print(is\_integer)

**Hint:** Use a function to verify the type of the variable.

**Riddle:** I can tell you if something is an integer without guessing. What am I?

**Short Story:** Emma needed to verify data types to ensure compatibility in her program. Using the type-checking function, she confidently handled variables and avoided type-related bugs.

### 18. How do you access the first element of a list?

**Code:**

my\_list = [1, 2, 3, 4] first\_element = my\_list[0] print(first\_element)

**Hint:** List indices start from 0.

**Riddle:** I am the starting point of a sequence, located at position zero. What am I?

**Short Story:** Mike was iterating through a list and needed to access the first item. By knowing that list indexing starts at zero, he quickly retrieved the first element and used it in his application.

### 19. What is the difference between a string and a list in Python?

**Code:**

string\_example = "Hello" list\_example = [1, 2, 3] print(type(string\_example)) print(type(list\_example))

**Hint:** Strings are immutable sequences of characters, while lists are mutable sequences of items.

**Riddle:** One is a sequence of characters that can’t be changed, while the other holds items you can modify. What are they?

**Short Story:** Laura was working with different data types and needed to understand their differences. By exploring strings and lists, she learned that strings are fixed and lists can be modified, which helped her choose the right data structure for her tasks.

### 20. How do you remove an item from a list by its value?

**Code:**

my\_list = [1, 2, 3, 4] my\_list.remove(3) print(my\_list)

**Hint:** Use a method to remove the first occurrence of a value from the list.

**Riddle:** I can take away an item from a collection based on what it is, not where it is. What am I?

**Short Story:** Kevin wanted to clean up his list by removing a specific item. By using the `remove` method, he easily eliminated the value he no longer needed, keeping his data organized.

### 21. How do you find the type of a variable in Python?

**Code:**

variable = 3.14 print(type(variable))

**Hint:** Use a built-in function to check the type of a variable.

**Riddle:** I reveal what kind of data you're dealing with. What am I?

**Short Story:** Sam needed to debug his code and was unsure of the types of some variables. By using a built-in function, he quickly identified the types and fixed the issues.

### 22. What will the following code output?

**Code:**

x = [1, 2, 3] y = x y.append(4) print(x)

**Hint:** When you assign one list to another, they both refer to the same object.

**Riddle:** If I change a shared item, everyone sees the change. What am I?

**Short Story:** Alex was working with lists and realized that changes to one list affected another list that was pointing to the same data. Understanding this helped him manage data more effectively.

### 23. How do you create a tuple with a single element?

**Code:**

single\_element\_tuple = (5,) print(single\_element\_tuple)

**Hint:** A trailing comma is needed to create a tuple with only one element.

**Riddle:** I’m a collection with one member, but I need a special mark to be recognized. What am I?

**Short Story:** Julia was learning about tuples and needed a way to represent a single item in a tuple. By adding a comma, she successfully created a one-element tuple for her project.

### 24. What is the output of this code?

**Code:**

x = "Hello" y = "World" print(x + " " + y)

**Hint:** Concatenate the strings to form a complete message.

**Riddle:** I join two text fragments into a single line. What am I?

**Short Story:** Lily wanted to display a welcoming message. By concatenating strings, she crafted a greeting that combined different parts into a single, friendly message.

### 25. How do you check if a variable is a string?

**Code:**

variable = "Sample" is\_string = isinstance(variable, str) print(is\_string)

**Hint:** Use a function to verify if the variable is of type `str`.

**Riddle:** I determine if a value is composed of characters. What am I?

**Short Story:** Mark needed to ensure that user input was a string. By checking the variable type, he confirmed the input type and handled it appropriately in his program.

### 26. How do you convert a list to a string?

**Code:**

list\_example = ['a', 'b', 'c'] string\_example = ''.join(list\_example) print(string\_example)

**Hint:** Use a method to concatenate all elements of the list into a single string.

**Riddle:** I combine list elements into one continuous text. What am I?

**Short Story:** Anna wanted to combine a list of characters into a single string. By using a join method, she easily transformed the list into a cohesive text.

### 27. What is the output of the following code?

**Code:**

x = 10 y = 2.5 result = x / y print(result)

**Hint:** Division between an integer and a float results in a float.

**Riddle:** I divide a whole number by a fractional number to give a result with decimals. What am I?

**Short Story:** Emma was dividing different types of numbers in her calculations. She learned that dividing an integer by a float results in a float, which helped her manage numerical precision in her work.

### 28. How do you check if a variable is a number?

**Code:**

variable = 42 is\_number = isinstance(variable, (int, float)) print(is\_number)

**Hint:** Use `isinstance` with a tuple of types to check for numbers.

**Riddle:** I determine if a value can be counted or measured. What am I?

**Short Story:** Jack needed to check if a variable was a number for his calculations. By using `isinstance`, he was able to confirm the variable’s type and avoid errors in his math operations.

### 29. How do you convert an integer to a string?

**Code:**

integer\_value = 123 string\_value = str(integer\_value) print(string\_value)

**Hint:** Use a function to change an integer into its string representation.

**Riddle:** I turn numbers into text. What am I?

**Short Story:** Lisa wanted to create a report that included numbers and text. By converting integers to strings, she was able to format her report with both types of data seamlessly.

### 30. What will the following code output?

**Code:**

x = 1 y = 2 z = x + y print(z \* 2)

**Hint:** First, calculate the sum, then multiply the result.

**Riddle:** I sum two numbers and then double the result. What am I?

**Short Story:** Paul was working on a budgeting application. He needed to perform calculations involving sums and multiplications. By understanding how to sequence operations, he managed his budget accurately.

### 31. How do you create a dictionary with integer keys and string values?

**Code:**

my\_dict = {1: 'one', 2: 'two', 3: 'three'} print(my\_dict)

**Hint:** Use curly braces to define key-value pairs.

**Riddle:** I hold pairs of keys and values, and my keys are numbers. What am I?

**Short Story:** Nora needed a way to map integer keys to their string equivalents. By creating a dictionary, she efficiently stored and accessed the data she needed for her application.

### 32. What will be the output of the following code?

**Code:**

a = "Python" b = a[:3] print(b)

**Hint:** String slicing allows you to extract parts of a string.

**Riddle:** I extract a portion of a text using indices. What am I?

**Short Story:** Maya was working on text processing and needed to extract a substring. By using slicing, she obtained the necessary part of the string for her analysis.

### 33. How do you check if a variable is of type float?

**Code:**

variable = 3.14 is\_float = isinstance(variable, float) print(is\_float)

**Hint:** Use `isinstance` to verify the type of a variable.

**Riddle:** I determine if a value has decimals. What am I?

**Short Story:** Jake needed to ensure that his calculations were done with floating-point numbers. By checking the variable type, he confirmed it was a float and avoided potential errors.

### 34. What is the output of this code?

**Code:**

x = "Hello" y = x \* 3 print(y)

**Hint:** Multiplying a string replicates it multiple times.

**Riddle:** I repeat a string multiple times by using a multiplier. What am I?

**Short Story:** Olivia wanted to create a repeated pattern in her text-based game. By multiplying a string, she easily generated the repeated content she needed.

### 35. How do you convert a list of numbers to a list of strings?

**Code:**

numbers = [1, 2, 3] strings = list(map(str, numbers)) print(strings)

**Hint:** Use the `map` function to apply a conversion to each element.

**Riddle:** I transform each number into text, one by one. What am I?

**Short Story:** Emma had a list of numbers and needed them as strings for display purposes. By using the `map` function, she converted the list efficiently.

### 36. What will be the result of the following code?

**Code:**

x = [1, 2, [3, 4]] y = x[2] y.append(5) print(x)

**Hint:** Modifying a nested list will affect the original list.

**Riddle:** I alter an inner structure which reflects in the outer one. What am I?

**Short Story:** Rachel was working with nested lists and noticed that changes to an inner list affected the outer list. Understanding this helped her manage complex data structures more effectively.

### 37. How do you convert a string to an integer?

**Code:**

string\_value = "123" integer\_value = int(string\_value) print(integer\_value)

**Hint:** Use a function to change a string to its integer equivalent.

**Riddle:** I turn text into numbers for mathematical operations. What am I?

**Short Story:** Chris needed to perform calculations with user input, which was in string form. By converting the string to an integer, he was able to perform arithmetic operations.

### 38. What will be the output of the following code?

**Code:**

x = 5 y = 10 z = x \* y / 2 print(z)

**Hint:** Follow the order of operations to calculate the result.

**Riddle:** I perform multiplication and division, following the rules of precedence. What am I?

**Short Story:** Sarah was working on a formula and needed to ensure correct calculations. By understanding the order of operations, she accurately computed the result for her project.

### 39. How do you find the length of a string?

**Code:**

string\_value = "Hello, World!" length = len(string\_value) print(length)

**Hint:** Use a built-in function to get the number of characters.

**Riddle:** I measure how many characters are in a string. What am I?

**Short Story:** Kevin needed to determine the length of user input for validation. By using the `len` function, he quickly obtained the length and applied the necessary checks.

### 40. What is the result of this code?

**Code:**

x = 3.14 y = round(x) print(y)

**Hint:** Use a function to round a floating-point number to the nearest integer.

**Riddle:** I take a number with decimals and round it to the nearest whole number. What am I?

**Short Story:** Lisa needed to round a floating-point number to display a cleaner result. By using the `round` function, she simplified the number for user-friendly output.

### 41. How do you concatenate two strings?

**Code:**

str1 = "Hello" str2 = "World" result = str1 + " " + str2 print(result)

**Hint:** Use the `+` operator to join strings together.

**Riddle:** I join two pieces of text together, separated by a space. What am I?

**Short Story:** Emma needed to combine two strings for a greeting message. By concatenating them with a space, she created a personalized welcome message for her application.

### 42. What will be the output of this code?

**Code:**

a = 10 b = 3 result = a % b print(result)

**Hint:** The `%` operator gives the remainder of division.

**Riddle:** I reveal what is left after dividing one number by another. What am I?

**Short Story:** Alex was working on a project where he needed to find the remainder of a division operation. By using the `%` operator, he efficiently obtained the result he needed.

### 43. How do you convert a list of strings to a list of integers?

**Code:**

string\_list = ["1", "2", "3"] int\_list = list(map(int, string\_list)) print(int\_list)

**Hint:** Use `map` to apply `int` conversion to each string in the list.

**Riddle:** I turn each text item in a list into numbers. What am I?

**Short Story:** Mia had a list of numeric strings and needed them as integers for processing. By using `map`, she converted the strings to integers for further calculations.

### 44. What is the output of the following code?

**Code:**

x = [1, 2, 3] x.append(4) print(x)

**Hint:** The `append` method adds an element to the end of a list.

**Riddle:** I add an item to the end of a collection. What am I?

**Short Story:** Daniel was managing a list of numbers and needed to add a new element. By using the `append` method, he updated his list with the additional number.

### 45. How do you create a tuple with a single element?

**Code:**

single\_element\_tuple = (1,) print(single\_element\_tuple)

**Hint:** Include a comma after the single element to define a tuple.

**Riddle:** I hold just one item, but need a comma to be recognized. What am I?

**Short Story:** Julia needed a tuple with only one element for a function parameter. By using a trailing comma, she correctly created the single-element tuple for her needs.

### 46. What will be the result of this code?

**Code:**

x = 2 y = 5 z = x \*\* y print(z)

**Hint:** The `\*\*` operator is used for exponentiation.

**Riddle:** I perform the operation of raising one number to the power of another. What am I?

**Short Story:** Sara was calculating exponential growth for her data analysis. By using the `\*\*` operator, she computed the power and obtained the result needed for her study.

### 47. How do you check if a value is of type boolean?

**Code:**

value = True is\_boolean = isinstance(value, bool) print(is\_boolean)

**Hint:** Use `isinstance` to check if a value is a boolean.

**Riddle:** I verify if a value represents truth or falsehood. What am I?

**Short Story:** Emma was dealing with conditional logic and needed to ensure a variable was a boolean. By using `isinstance`, she confirmed the type to prevent logical errors in her code.

### 48. What is the result of this code?

**Code:**

x = "Hello" y = "World" result = f"{x} {y}" print(result)

**Hint:** Use an f-string to format the output with variables.

**Riddle:** I embed variables into a string using a special format. What am I?

**Short Story:** Michael needed to format a greeting message with variables. By using an f-string, he easily inserted the variables into the string and displayed the desired output.

### 49. How do you create a set in Python?

**Code:**

my\_set = {1, 2, 3} print(my\_set)

**Hint:** Use curly braces to define a set.

**Riddle:** I contain unique items and don't care about order. What am I?

**Short Story:** Lisa needed a collection of unique items for her application. By using a set, she ensured that each item appeared only once and maintained the required data structure.

### 50. What is the output of the following code?

**Code:**

x = [1, 2, 3] x.extend([4, 5]) print(x)

**Hint:** The `extend` method adds elements from another iterable to a list.

**Riddle:** I add elements from one list to another, merging them. What am I?

**Short Story:** John was merging two lists for data processing. By using the `extend` method, he combined the lists and obtained the complete dataset he needed.

### 51. How do you create a dictionary with default values?

**Code:**

from collections import defaultdict my\_dict = defaultdict(int) my\_dict['a'] += 1 print(my\_dict)

**Hint:** Use `defaultdict` from the `collections` module to set default values.

**Riddle:** I provide a default value for missing keys in a dictionary. What am I?

**Short Story:** Alex needed a dictionary where missing keys had default values. By using `defaultdict`, he avoided errors and ensured every key had an initial value.

### 52. How do you remove a key-value pair from a dictionary?

**Code:**

my\_dict = {'a': 1, 'b': 2} del my\_dict['a'] print(my\_dict)

**Hint:** Use the `del` statement to remove a key-value pair from a dictionary.

**Riddle:** I erase a specific entry from a collection of key-value pairs. What am I?

**Short Story:** Emma had a dictionary and needed to remove an obsolete entry. By using `del`, she cleaned up the dictionary and maintained relevant data.

### 53. What is the result of this code?

**Code:**

x = {1, 2, 3} y = {3, 4, 5} result = x & y print(result)

**Hint:** The `&` operator is used to find the intersection of two sets.

**Riddle:** I find common elements between two groups. What am I?

**Short Story:** Michael needed to determine the shared elements between two sets for analysis. By using the `&` operator, he efficiently found the common values.

### 54. How do you format a string with variable values?

**Code:**

name = "Alice" age = 30 formatted\_string = f"My name is {name} and I am {age} years old." print(formatted\_string)

**Hint:** Use an f-string to embed variables directly within a string.

**Riddle:** I embed values within a message to personalize it. What am I?

**Short Story:** Sarah wanted to create a message that included user-specific details. By using an f-string, she personalized the message with the user's name and age.

### 55. What will be the output of this code?

**Code:**

x = [1, 2, 3] x.remove(2) print(x)

**Hint:** The `remove` method deletes the first occurrence of a specified value from a list.

**Riddle:** I take away a specified item from a list. What am I?

**Short Story:** Daniel needed to remove a specific item from a list of numbers. By using the `remove` method, he deleted the unwanted element from the list.

### 56. How do you create a list of squares of numbers from 1 to 5?

**Code:**

squares = [x\*\*2 for x in range(1, 6)] print(squares)

**Hint:** Use list comprehension to generate squares of numbers.

**Riddle:** I create a list where each item is the square of a number. What am I?

**Short Story:** Lisa wanted a list of squared numbers for a mathematical computation. Using list comprehension, she quickly generated the list of squares.

### 57. What is the result of the following code?

**Code:**

x = "Python" result = x[1:4] print(result)

**Hint:** Use slicing to get a substring from a string.

**Riddle:** I extract a portion of a string using indices. What am I?

**Short Story:** Michael needed to extract a specific part of a string for processing. By using slicing, he obtained the substring he needed.

### 58. How do you create a list from a string where each character is an element?

**Code:**

my\_string = "hello" char\_list = list(my\_string) print(char\_list)

**Hint:** Convert the string to a list to get each character as an element.

**Riddle:** I turn each character of a string into an individual list item. What am I?

**Short Story:** Emma needed to process each character of a string individually. By converting the string to a list, she easily handled each character as a separate element.

### 59. What will be the output of this code?

**Code:**

x = 10 y = 20 x, y = y, x print(x, y)

**Hint:** Tuple unpacking allows for swapping variables.

**Riddle:** I swap the values of two variables. What am I?

**Short Story:** Sarah needed to exchange the values of two variables. By using tuple unpacking, she efficiently swapped their values without needing a temporary variable.

### 60. How do you create an empty set?

**Code:**

empty\_set = set() print(empty\_set)

**Hint:** Use the `set` constructor to create an empty set.

**Riddle:** I am a collection with no items initially. What am I?

**Short Story:** Emma started with an empty set to collect unique values. By using the `set` constructor, she prepared a clean slate for her data collection.

### 61. How do you check if a variable is of type integer?

**Code:**

value = 42 is\_integer = isinstance(value, int) print(is\_integer)

**Hint:** Use `isinstance` to verify the type of a variable.

**Riddle:** I confirm if a variable holds whole numbers. What am I?

**Short Story:** Alex was verifying data types to ensure correct operations. By using `isinstance`, he confirmed that a variable was indeed an integer.

### 62. How do you concatenate two strings?

**Code:**

str1 = "Hello" str2 = "World" result = str1 + " " + str2 print(result)

**Hint:** Use the `+` operator to join strings.

**Riddle:** I combine two text strings into one. What am I?

**Short Story:** Lisa needed to create a complete greeting message. By concatenating two strings with the `+` operator, she formed a cohesive message.

### 63. What will be the result of the following code?

**Code:**

x = [1, 2, 3] x.append(4) print(x)

**Hint:** The `append` method adds an element to the end of a list.

**Riddle:** I add a new item to the end of a collection. What am I?

**Short Story:** Daniel needed to extend a list of numbers. By using the `append` method, he added a new item to the end of the list, expanding his data collection.

### 64. How do you create a tuple with a single element?

**Code:**

single\_element\_tuple = (42,) print(single\_element\_tuple)

**Hint:** Use a trailing comma to create a single-element tuple.

**Riddle:** I hold just one item and use a special syntax to denote it. What am I?

**Short Story:** Emma needed to create a tuple with a single value. By adding a trailing comma, she ensured the tuple was correctly defined with one item.

### 65. How do you convert a string to an integer?

**Code:**

str\_number = "123" int\_number = int(str\_number) print(int\_number)

**Hint:** Use the `int` function to convert a string to an integer.

**Riddle:** I change a text representation of a number into an integer. What am I?

**Short Story:** Sarah had a string that represented a number and needed to perform arithmetic operations. By using the `int` function, she converted the string into an integer.

### 66. What is the output of this code?

**Code:**

x = {1, 2, 3} y = {2, 3, 4} result = x | y print(result)

**Hint:** The `|` operator is used to find the union of two sets.

**Riddle:** I combine all unique elements from two sets into one. What am I?

**Short Story:** Michael needed a combined list of unique items from two sets. By using the `|` operator, he found the union of both sets efficiently.

### 67. How do you create a list of numbers from 1 to 10?

**Code:**

numbers = list(range(1, 11)) print(numbers)

**Hint:** Use `range` and convert it to a list to get a sequence of numbers.

**Riddle:** I generate a sequence of numbers within a specified range. What am I?

**Short Story:** Lisa wanted a list of numbers for a loop iteration. By using `range` and converting it to a list, she quickly obtained the sequence of numbers.

### 68. How do you check if a value is in a list?

**Code:**

value = 5 my\_list = [1, 2, 3, 4, 5] is\_in\_list = value in my\_list print(is\_in\_list)

**Hint:** Use the `in` keyword to check for membership in a list.

**Riddle:** I determine if a specific value exists within a collection. What am I?

**Short Story:** Emma needed to verify if a number was in a list of values. By using the `in` keyword, she quickly checked for the value's presence in the list.

### 69. How do you convert a list of characters into a string?

**Code:**

char\_list = ['H', 'e', 'l', 'l', 'o'] string = ''.join(char\_list) print(string)

**Hint:** Use the `join` method to concatenate characters into a string.

**Riddle:** I combine characters from a list into a single text string. What am I?

**Short Story:** Sarah needed to merge a list of characters into a single string. By using the `join` method, she efficiently concatenated the characters into a coherent message.

### 70. What is the output of this code?

**Code:**

x = "5" y = "10" result = int(x) + int(y) print(result)

**Hint:** Convert strings to integers before performing arithmetic operations.

**Riddle:** I perform arithmetic on text representations of numbers by first converting them. What am I?

**Short Story:** Alex had numbers stored as strings and needed to add them. By converting the strings to integers, he was able to perform the arithmetic operation and obtain the correct result.

### 71. How do you find the type of a variable?

**Code:**

variable = 3.14 print(type(variable))

**Hint:** Use the `type` function to check the type of a variable.

**Riddle:** I reveal the type of a variable, showing what kind of data it holds. What am I?

**Short Story:** Sarah wanted to confirm the data type of a variable for type-specific operations. By using the `type` function, she easily identified the variable's type.

### 72. How do you create a list of tuples with paired elements?

**Code:**

list\_of\_tuples = [(1, 'a'), (2, 'b'), (3, 'c')] print(list\_of\_tuples)

**Hint:** Use parentheses to define tuples and list brackets for the list.

**Riddle:** I organize paired elements into tuples within a list. What am I?

**Short Story:** Michael needed a structured list of paired elements for processing. By creating a list of tuples, he effectively grouped the data for further use.

### 73. How do you check the length of a string?

**Code:**

text = "Hello World" length = len(text) print(length)

**Hint:** Use the `len` function to find the length of a string.

**Riddle:** I measure the number of characters in a text. What am I?

**Short Story:** Emma needed to know the length of a string for formatting purposes. By using the `len` function, she easily obtained the number of characters in the text.

### 74. How do you remove duplicates from a list?

**Code:**

my\_list = [1, 2, 2, 3, 4, 4, 5] unique\_list = list(set(my\_list)) print(unique\_list)

**Hint:** Convert the list to a set to remove duplicates and then back to a list.

**Riddle:** I eliminate repeated values from a collection, leaving only unique items. What am I?

**Short Story:** Lisa needed to clean up a list with duplicate entries. By converting the list to a set and then back to a list, she removed all duplicate values efficiently.

### 75. How do you sort a list in ascending order?

**Code:**

numbers = [4, 2, 9, 1, 5] numbers.sort() print(numbers)

**Hint:** Use the `sort` method to order elements in a list.

**Riddle:** I arrange the elements of a list from smallest to largest. What am I?

**Short Story:** Alex needed to order a list of numbers for analysis. By using the `sort` method, he arranged the numbers in ascending order for better readability.

### 76. How do you find the maximum value in a list?

**Code:**

numbers = [1, 3, 7, 2, 5] max\_value = max(numbers) print(max\_value)

**Hint:** Use the `max` function to get the largest number from a list.

**Riddle:** I identify the highest value among a collection of numbers. What am I?

**Short Story:** Sarah needed to find the highest number in a list for her report. By using the `max` function, she quickly identified the maximum value among the numbers.

### 77. How do you split a string into a list of words?

**Code:**

sentence = "Python is fun" words = sentence.split() print(words)

**Hint:** Use the `split` method to divide a string into words.

**Riddle:** I break a sentence into individual words. What am I?

**Short Story:** Emma needed to process a sentence by separating it into words. By using the `split` method, she obtained a list of words from the original sentence.

### 78. How do you count the occurrences of a specific character in a string?

**Code:**

text = "hello" count = text.count('l') print(count)

**Hint:** Use the `count` method to find the number of occurrences of a character.

**Riddle:** I determine how many times a specific character appears in a string. What am I?

**Short Story:** Lisa needed to count how many times a character appeared in a string. By using the `count` method, she found the occurrences of the character efficiently.

### 79. How do you reverse a list?

**Code:**

numbers = [1, 2, 3, 4, 5] numbers.reverse() print(numbers)

**Hint:** Use the `reverse` method to invert the order of elements in a list.

**Riddle:** I change the order of items in a list to its opposite. What am I?

**Short Story:** Michael needed to reverse the order of items in a list for his analysis. By using the `reverse` method, he quickly inverted the order of the list's elements.

### 80. How do you find the index of a specific item in a list?

**Code:**

my\_list = ['apple', 'banana', 'cherry'] index = my\_list.index('banana') print(index)

**Hint:** Use the `index` method to locate the position of an item in a list.

**Riddle:** I determine the position of an item within a list. What am I?

**Short Story:** Sarah needed to find the position of an item in a list. By using the `index` method, she quickly identified the item's location in the list.

### 81. How do you create a dictionary with keys and values?

**Code:**

my\_dict = {'name': 'Alice', 'age': 25} print(my\_dict)

**Hint:** Use curly braces to define a dictionary with key-value pairs.

**Riddle:** I store data in key-value pairs for easy access. What am I?

**Short Story:** Lisa needed to organize information with keys and values. By creating a dictionary, she structured the data for efficient access and manipulation.

### 82. How do you get a list of all the keys in a dictionary?

**Code:**

my\_dict = {'name': 'Alice', 'age': 25} keys = my\_dict.keys() print(keys)

**Hint:** Use the `keys` method to retrieve the dictionary's keys.

**Riddle:** I provide access to the keys stored in a dictionary. What am I?

**Short Story:** Michael needed to view all the keys in a dictionary. By using the `keys` method, he obtained a list of all the keys in the dictionary.

### 83. How do you get a list of all the values in a dictionary?

**Code:**

my\_dict = {'name': 'Alice', 'age': 25} values = my\_dict.values() print(values)

**Hint:** Use the `values` method to retrieve the dictionary's values.

**Riddle:** I provide access to the values stored in a dictionary. What am I?

**Short Story:** Sarah needed to view all the values in a dictionary. By using the `values` method, she obtained a list of all the values stored in the dictionary.

### 84. How do you remove a key-value pair from a dictionary?

**Code:**

my\_dict = {'name': 'Alice', 'age': 25} del my\_dict['age'] print(my\_dict)

**Hint:** Use the `del` statement to remove a specific key-value pair from a dictionary.

**Riddle:** I remove a specific item from a dictionary. What am I?

**Short Story:** Emma needed to delete a key-value pair from a dictionary. By using the `del` statement, she efficiently removed the unwanted item from the dictionary.

### 85. How do you merge two dictionaries?

**Code:**

dict1 = {'name': 'Alice', 'age': 25} dict2 = {'city': 'Wonderland'} merged\_dict = {\*\*dict1, \*\*dict2} print(merged\_dict)

**Hint:** Use the `\*\*` operator to merge dictionaries.

**Riddle:** I combine the items of two dictionaries into one. What am I?

**Short Story:** Lisa needed to combine two dictionaries for her project. By using the `\*\*` operator, she merged the dictionaries into a single comprehensive dictionary.

### 86. How do you check if a key exists in a dictionary?

**Code:**

my\_dict = {'name': 'Alice', 'age': 25} exists = 'name' in my\_dict print(exists)

**Hint:** Use the `in` operator to check for a key's existence in a dictionary.

**Riddle:** I determine whether a key is present in a dictionary. What am I?

**Short Story:** Michael needed to verify if a key was present in a dictionary. By using the `in` operator, he quickly checked for the key's existence.

### 87. How do you iterate over a dictionary's keys and values?

**Code:**

my\_dict = {'name': 'Alice', 'age': 25} for key, value in my\_dict.items(): print(key, value)

**Hint:** Use the `items` method to iterate over key-value pairs in a dictionary.

**Riddle:** I help you access both keys and values in a dictionary simultaneously. What am I?

**Short Story:** Emma needed to process both keys and values in a dictionary. By using the `items` method, she iterated over the key-value pairs and accessed each item effectively.

### 88. How do you create a set in Python?

**Code:**

my\_set = {1, 2, 3, 4, 5} print(my\_set)

**Hint:** Use curly braces to define a set with unique elements.

**Riddle:** I store unique items in an unordered collection. What am I?

**Short Story:** Sarah needed to store unique values without any particular order. By creating a set, she efficiently managed the unique elements in her collection.

### 89. How do you add an element to a set?

**Code:**

my\_set = {1, 2, 3} my\_set.add(4) print(my\_set)

**Hint:** Use the `add` method to include an element in a set.

**Riddle:** I help you include a new item in a collection of unique elements. What am I?

**Short Story:** Lisa needed to add a new item to her set of unique elements. By using the `add` method, she efficiently included the new item in the set.

### 90. How do you remove an element from a set?

**Code:**

my\_set = {1, 2, 3} my\_set.remove(2) print(my\_set)

**Hint:** Use the `remove` method to delete an element from a set.

**Riddle:** I help you delete an item from a collection of unique elements. What am I?

**Short Story:** Michael needed to remove an item from his set of unique values. By using the `remove` method, he efficiently deleted the unwanted item.

### 91. How do you check if a set contains a specific element?

**Code:**

my\_set = {1, 2, 3} contains = 2 in my\_set print(contains)

**Hint:** Use the `in` operator to check if an element is present in a set.

**Riddle:** I verify the presence of an item in a unique collection. What am I?

**Short Story:** Emma needed to check if an item was part of a set. By using the `in` operator, she quickly confirmed the presence of the element.

### 92. How do you clear all elements from a set?

**Code:**

my\_set = {1, 2, 3} my\_set.clear() print(my\_set)

**Hint:** Use the `clear` method to remove all elements from a set.

**Riddle:** I remove every item from a unique collection. What am I?

**Short Story:** Sarah needed to empty a set for a fresh start. By using the `clear` method, she removed all elements from the set.

### 93. How do you find the length of a set?

**Code:**

my\_set = {1, 2, 3} length = len(my\_set) print(length)

**Hint:** Use the `len` function to get the number of elements in a set.

**Riddle:** I measure the number of unique items in a collection. What am I?

**Short Story:** Michael needed to know how many items were in a set. By using the `len` function, he quickly determined the set's length.

### 94. How do you perform a union operation on two sets?

**Code:**

set1 = {1, 2, 3} set2 = {3, 4, 5} union\_set = set1.union(set2) print(union\_set)

**Hint:** Use the `union` method to combine elements from two sets.

**Riddle:** I combine the elements of two unique collections into one. What am I?

**Short Story:** Emma needed to combine two sets for a comprehensive list. By using the `union` method, she efficiently merged the elements of both sets.

### 95. How do you perform an intersection operation on two sets?

**Code:**

set1 = {1, 2, 3} set2 = {3, 4, 5} intersection\_set = set1.intersection(set2) print(intersection\_set)

**Hint:** Use the `intersection` method to find common elements between two sets.

**Riddle:** I identify the elements shared by two unique collections. What am I?

**Short Story:** Michael needed to find common items between two sets. By using the `intersection` method, he quickly determined the shared elements.

### 96. How do you perform a difference operation on two sets?

**Code:**

set1 = {1, 2, 3} set2 = {3, 4, 5} difference\_set = set1.difference(set2) print(difference\_set)

**Hint:** Use the `difference` method to find elements in one set but not in another.

**Riddle:** I find the items present in one unique collection but not in another. What am I?

**Short Story:** Emma needed to determine which items were unique to a set compared to another. By using the `difference` method, she identified the unique elements.

### 97. How do you perform a symmetric difference operation on two sets?

**Code:**

set1 = {1, 2, 3} set2 = {3, 4, 5} symmetric\_difference\_set = set1.symmetric\_difference(set2) print(symmetric\_difference\_set)

**Hint:** Use the `symmetric\_difference` method to find elements that are in either set but not both.

**Riddle:** I find the items that are unique to each collection, excluding those shared. What am I?

**Short Story:** Michael needed to identify items that were unique to each of two sets. By using the `symmetric\_difference` method, he determined the elements present in one set but not the other.

### 98. How do you check if one set is a subset of another?

**Code:**

set1 = {1, 2, 3} set2 = {1, 2, 3, 4, 5} is\_subset = set1.issubset(set2) print(is\_subset)

**Hint:** Use the `issubset` method to check if all elements of one set are in another set.

**Riddle:** I determine if one unique collection is entirely contained within another. What am I?

**Short Story:** Sarah needed to check if a set was completely contained within another set. By using the `issubset` method, she confirmed whether all elements of the first set were present in the second set.

### 99. How do you check if one set is a superset of another?

**Code:**

set1 = {1, 2, 3, 4, 5} set2 = {1, 2, 3} is\_superset = set1.issuperset(set2) print(is\_superset)

**Hint:** Use the `issuperset` method to check if a set contains all elements of another set.

**Riddle:** I determine if a unique collection contains all elements of another collection. What am I?

**Short Story:** Emma needed to check if one set contained all elements of another set. By using the `issuperset` method, she confirmed that the first set was indeed a superset of the second.

### 100. How do you remove duplicates from a list while preserving the original order?

**Code:**

def remove\_duplicates(lst): seen = set() result = [] for item in lst: if item not in seen: result.append(item) seen.add(item) return result my\_list = [1, 2, 2, 3, 4, 4, 5] unique\_list = remove\_duplicates(my\_list) print(unique\_list)

**Hint:** Use a set to track seen elements while iterating through the list to preserve order.

**Riddle:** I help you clean up a list by removing duplicates and keeping the order intact. What am I?

**Short Story:** Lisa needed to clean up a list with repeated elements. By using a set to track seen items and preserving the original order, she effectively removed duplicates from the list.

### 101. How do you create a list with a fixed number of elements initialized to zero?

**Code:**

fixed\_size\_list = [0] \* 10 print(fixed\_size\_list)

**Hint:** Use multiplication to create a list with repeated elements.

**Riddle:** I help you create a list of a certain size, filled with zeros. What am I?

**Short Story:** Emma needed a list with ten elements, all set to zero. By multiplying a list with a single zero, she easily created the list she needed.

### 102. How do you find the index of the first occurrence of an element in a list?

**Code:**

my\_list = [10, 20, 30, 20, 40] index = my\_list.index(20) print(index)

**Hint:** Use the `index` method to find the position of an element in a list.

**Riddle:** I help you find where an item first appears in a list. What am I?

**Short Story:** Michael needed to locate the position of the first occurrence of an element. By using the `index` method, he quickly found the element's position in the list.

### 103. How do you remove an element by value from a list?

**Code:**

my\_list = [1, 2, 3, 4] my\_list.remove(3) print(my\_list)

**Hint:** Use the `remove` method to delete an element by its value.

**Riddle:** I help you delete a specific item from a list based on its value. What am I?

**Short Story:** Sarah needed to remove a specific item from a list. By using the `remove` method, she successfully deleted the item by its value.

### 104. How do you insert an element at a specific position in a list?

**Code:**

my\_list = [1, 2, 4, 5] my\_list.insert(2, 3) print(my\_list)

**Hint:** Use the `insert` method to add an element at a specified index.

**Riddle:** I help you place an item at a precise location within a list. What am I?

**Short Story:** Emma needed to add an item to a specific position in a list. By using the `insert` method, she placed the item exactly where she wanted it.

### 105. How do you reverse the order of elements in a list?

**Code:**

my\_list = [1, 2, 3, 4] my\_list.reverse() print(my\_list)

**Hint:** Use the `reverse` method to change the order of elements in a list.

**Riddle:** I help you flip the order of items in a list. What am I?

**Short Story:** Michael wanted to reverse the order of elements in a list. By using the `reverse` method, he achieved the desired result.

### 106. How do you sort a list in ascending order?

**Code:**

my\_list = [4, 2, 3, 1] my\_list.sort() print(my\_list)

**Hint:** Use the `sort` method to arrange elements in ascending order.

**Riddle:** I help you arrange items from smallest to largest. What am I?

**Short Story:** Emma needed to sort a list of numbers in ascending order. By using the `sort` method, she easily organized the elements from smallest to largest.

### 107. How do you create a tuple with elements?

**Code:**

my\_tuple = (1, 2, 3, 4) print(my\_tuple)

**Hint:** Use parentheses to create a tuple with multiple elements.

**Riddle:** I am a collection that holds items in a fixed order. What am I?

**Short Story:** Sarah wanted to create a collection of items that couldn't be changed. By using parentheses, she created a tuple with the elements she needed.

### 108. How do you access an element in a tuple by index?

**Code:**

my\_tuple = (1, 2, 3, 4) element = my\_tuple[2] print(element)

**Hint:** Use square brackets and an index to access an element in a tuple.

**Riddle:** I allow you to retrieve an item from a fixed collection based on its position. What am I?

**Short Story:** Michael needed to access a specific item in a tuple. By using square brackets and the appropriate index, he retrieved the element he needed.

### 109. How do you concatenate two tuples?

**Code:**

tuple1 = (1, 2, 3) tuple2 = (4, 5, 6) concatenated\_tuple = tuple1 + tuple2 print(concatenated\_tuple)

**Hint:** Use the `+` operator to join two tuples together.

**Riddle:** I help you merge two fixed collections into one. What am I?

**Short Story:** Emma wanted to combine two tuples into a single collection. By using the `+` operator, she successfully merged the tuples.

### 110. How do you create a dictionary with keys and values?

**Code:**

my\_dict = {'key1': 'value1', 'key2': 'value2'} print(my\_dict)

**Hint:** Use curly braces to define a dictionary with key-value pairs.

**Riddle:** I hold data in pairs, associating keys with values. What am I?

**Short Story:** Sarah needed to create a collection of key-value pairs. By using curly braces, she built a dictionary with the data she required.

### 111. How do you access a value in a dictionary by its key?

**Code:**

my\_dict = {'key1': 'value1', 'key2': 'value2'} value = my\_dict['key1'] print(value)

**Hint:** Use square brackets with the key to retrieve the corresponding value from a dictionary.

**Riddle:** I allow you to find the value associated with a specific key. What am I?

**Short Story:** Michael needed to retrieve a value based on its key from a dictionary. By using square brackets with the key, he accessed the value he needed.

### 112. How do you add a new key-value pair to an existing dictionary?

**Code:**

my\_dict = {'key1': 'value1'} my\_dict['key2'] = 'value2' print(my\_dict)

**Hint:** Assign a value to a new key to add it to the dictionary.

**Riddle:** I help you expand a collection by adding new key-value pairs. What am I?

**Short Story:** Emma wanted to add new information to her dictionary. By assigning a value to a new key, she successfully added the new key-value pair.

### 113. How do you remove a key-value pair from a dictionary?

**Code:**

my\_dict = {'key1': 'value1', 'key2': 'value2'} del my\_dict['key1'] print(my\_dict)

**Hint:** Use the `del` statement to remove a key-value pair from a dictionary.

**Riddle:** I assist you in removing a specific item from a collection of key-value pairs. What am I?

**Short Story:** Sarah needed to remove a specific entry from her dictionary. By using the `del` statement, she successfully deleted the key-value pair.

### 114. How do you get all the keys from a dictionary?

**Code:**

my\_dict = {'key1': 'value1', 'key2': 'value2'} keys = my\_dict.keys() print(keys)

**Hint:** Use the `keys` method to retrieve all keys from a dictionary.

**Riddle:** I help you retrieve all the identifiers from a collection of key-value pairs. What am I?

**Short Story:** Michael needed to get a list of all the keys in his dictionary. By using the `keys` method, he obtained the identifiers he needed.

### 115. How do you get all the values from a dictionary?

**Code:**

my\_dict = {'key1': 'value1', 'key2': 'value2'} values = my\_dict.values() print(values)

**Hint:** Use the `values` method to retrieve all values from a dictionary.

**Riddle:** I help you access all the data associated with keys in a collection. What am I?

**Short Story:** Emma wanted to retrieve all the values from her dictionary. By using the `values` method, she obtained all the data associated with the keys.

### 116. How do you check if a key exists in a dictionary?

**Code:**

my\_dict = {'key1': 'value1', 'key2': 'value2'} exists = 'key1' in my\_dict print(exists)

**Hint:** Use the `in` operator to check for the presence of a key in a dictionary.

**Riddle:** I help you verify if a specific identifier is part of a collection of key-value pairs. What am I?

**Short Story:** Michael needed to confirm whether a particular key was in his dictionary. By using the `in` operator, he checked the presence of the key efficiently.

### 117. How do you create an empty dictionary?

**Code:**

empty\_dict = {} print(empty\_dict)

**Hint:** Use empty curly braces to create an empty dictionary.

**Riddle:** I am a collection ready to hold key-value pairs, but currently, I am empty. What am I?

**Short Story:** Sarah needed an empty dictionary to start her new project. By using empty curly braces, she created a dictionary ready to be filled with data.

### 118. How do you merge two dictionaries into one?

**Code:**

dict1 = {'key1': 'value1'} dict2 = {'key2': 'value2'} merged\_dict = {\*\*dict1, \*\*dict2} print(merged\_dict)

**Hint:** Use the unpacking operator to combine two dictionaries into one.

**Riddle:** I help you combine two collections of key-value pairs into a single one. What am I?

**Short Story:** Michael needed to merge two dictionaries into one. By using the unpacking operator, he successfully combined the two collections.

### 119. How do you create a set in Python?

**Code:**

my\_set = {1, 2, 3, 4} print(my\_set)

**Hint:** Use curly braces or the `set` constructor to create a set.

**Riddle:** I am a collection of unique items with no duplicates. What am I?

**Short Story:** Emma wanted to create a collection of unique items. By using curly braces, she created a set with distinct elements.

### 120. How do you add an element to a set?

**Code:**

my\_set = {1, 2, 3} my\_set.add(4) print(my\_set)

**Hint:** Use the `add` method to insert a new element into a set.

**Riddle:** I assist you in including a new item into a collection of unique elements. What am I?

**Short Story:** Sarah needed to add a new element to her set. By using the `add` method, she successfully included the item in her collection of unique elements.

### 121. How do you remove an element from a set?

**Code:**

my\_set = {1, 2, 3, 4} my\_set.remove(3) print(my\_set)

**Hint:** Use the `remove` method to delete an element from a set.

**Riddle:** I assist you in removing an item from a collection of unique elements. What am I?

**Short Story:** Emma needed to remove a specific number from her set. By using the `remove` method, she successfully deleted the item from her collection.

### 122. How do you check if an element exists in a set?

**Code:**

my\_set = {1, 2, 3, 4} exists = 3 in my\_set print(exists)

**Hint:** Use the `in` operator to check for the presence of an element in a set.

**Riddle:** I help you verify whether a specific item is part of a collection of unique elements. What am I?

**Short Story:** Michael needed to confirm if a particular number was in his set. By using the `in` operator, he checked the presence of the item efficiently.

### 123. How do you get the length of a set?

**Code:**

my\_set = {1, 2, 3, 4} length = len(my\_set) print(length)

**Hint:** Use the `len` function to find the number of elements in a set.

**Riddle:** I help you determine the number of unique items in a collection. What am I?

**Short Story:** Sarah wanted to find out how many elements were in her set. By using the `len` function, she quickly obtained the count of unique items.

### 124. How do you clear all elements from a set?

**Code:**

my\_set = {1, 2, 3, 4} my\_set.clear() print(my\_set)

**Hint:** Use the `clear` method to remove all elements from a set.

**Riddle:** I assist you in emptying a collection of unique items. What am I?

**Short Story:** Emma needed to remove all elements from her set. By using the `clear` method, she emptied the collection completely.

### 125. How do you find the intersection of two sets?

**Code:**

set1 = {1, 2, 3} set2 = {2, 3, 4} intersection = set1 & set2 print(intersection)

**Hint:** Use the `&` operator to find common elements between two sets.

**Riddle:** I help you find the shared items between two collections of unique elements. What am I?

**Short Story:** Michael wanted to identify common elements between two sets. By using the `&` operator, he successfully found the intersection of the sets.

### 126. How do you find the union of two sets?

**Code:**

set1 = {1, 2, 3} set2 = {2, 3, 4} union = set1 | set2 print(union)

**Hint:** Use the `|` operator to combine all elements from two sets.

**Riddle:** I help you combine all items from two collections of unique elements. What am I?

**Short Story:** Sarah needed to merge two sets into one. By using the `|` operator, she combined all unique items from both sets.

### 127. How do you find the difference between two sets?

**Code:**

set1 = {1, 2, 3} set2 = {2, 3, 4} difference = set1 - set2 print(difference)

**Hint:** Use the `-` operator to find elements that are in one set but not in the other.

**Riddle:** I help you identify items present in one collection but absent in another. What am I?

**Short Story:** Michael wanted to find elements unique to one set compared to another. By using the `-` operator, he determined the difference between the two sets.

### 128. How do you find the symmetric difference between two sets?

**Code:**

set1 = {1, 2, 3} set2 = {2, 3, 4} symmetric\_difference = set1 ^ set2 print(symmetric\_difference)

**Hint:** Use the `^` operator to find elements that are in either of the sets but not in both.

**Riddle:** I help you find items that are unique to each of two collections, excluding those that are common. What am I?

**Short Story:** Emma needed to identify elements that were unique to each of two sets. By using the `^` operator, she successfully found the symmetric difference between them.

### 129. How do you create a set from a list?

**Code:**

my\_list = [1, 2, 2, 3, 4] my\_set = set(my\_list) print(my\_set)

**Hint:** Use the `set` constructor to convert a list into a set.

**Riddle:** I help you eliminate duplicates and create a collection of unique items from a list. What am I?

**Short Story:** Sarah wanted to convert her list into a set to remove duplicates. By using the `set` constructor, she successfully created a set from the list.

### 130. How do you check if a set is a subset of another set?

**Code:**

set1 = {1, 2} set2 = {1, 2, 3, 4} is\_subset = set1.issubset(set2) print(is\_subset)

**Hint:** Use the `issubset` method to check if all elements of one set are contained in another.

**Riddle:** I help you determine if all items from one collection are included in another. What am I?

**Short Story:** Michael needed to verify if one set was completely contained within another. By using the `issubset` method, he efficiently performed the check.

### 131. How do you convert a string to an integer?

**Code:**

number\_str = "42" number\_int = int(number\_str) print(number\_int)

**Hint:** Use the `int()` function to convert a string to an integer.

**Riddle:** I transform a sequence of characters into a whole number. What am I?

**Short Story:** Jane had a string representing a number and needed to perform arithmetic operations. She used the `int()` function to convert the string to an integer.

### 132. What is the result of dividing an integer by a float in Python?

**Code:**

result = 5 // 2.0 print(result)

**Hint:** Integer division with a float results in a float.

**Riddle:** When dividing a whole number by a decimal, what type of result do you get?

**Short Story:** Michael was curious about the result of dividing an integer by a float. He found that the result was a float, even though he started with an integer.

### 133. How do you create a list of squares of numbers from 1 to 10?

**Code:**

squares = [x\*\*2 for x in range(1, 11)] print(squares)

**Hint:** Use list comprehension to generate a list of squares.

**Riddle:** I help you generate a list where each item is the square of numbers in a range. What am I?

**Short Story:** Sarah wanted to create a list of squares for a math problem. By using list comprehension, she quickly generated the list.

### 134. How do you use the modulus operator in Python?

**Code:**

remainder = 10 % 3 print(remainder)

**Hint:** Use the `%` operator to find the remainder of a division.

**Riddle:** I give you the leftover part after dividing two numbers. What am I?

**Short Story:** Emma wanted to find out the remainder of a division. She used the modulus operator to get the result.

### 135. How do you use the power operator in Python?

**Code:**

power = 2 \*\* 3 print(power)

**Hint:** Use the `\*\*` operator to raise a number to a power.

**Riddle:** I help you raise a number to an exponent. What am I?

**Short Story:** Michael needed to compute powers for a calculation. By using the `\*\*` operator, he obtained the desired result.

### 136. How do you concatenate two strings in Python?

**Code:**

str1 = "Hello" str2 = "World" result = str1 + " " + str2 print(result)

**Hint:** Use the `+` operator to join two strings together.

**Riddle:** I help you join two sequences of characters into one. What am I?

**Short Story:** Sarah wanted to combine two words into a greeting. She used the `+` operator to concatenate the strings.

### 137. How do you find the length of a list in Python?

**Code:**

my\_list = [1, 2, 3, 4] length = len(my\_list) print(length)

**Hint:** Use the `len()` function to get the number of items in a list.

**Riddle:** I help you count the number of items in a collection. What am I?

**Short Story:** Emma needed to determine how many items were in her list. By using the `len()` function, she got the count.

## Control Structures

### 138. How do you use an `if` statement to check if a number is positive?

**Code:**

number = 10 if number > 0: print("Positive") else: print("Not Positive")

**Hint:** Use the `if` keyword to perform a check and the `else` keyword to handle other cases.

**Riddle:** I help you decide actions based on a condition. What am I?

**Short Story:** Michael wanted to classify a number as positive or not. By using an `if` statement, he was able to check the condition and print the result.

### 139. How do you use an `if-else` statement to determine if a number is even or odd?

**Code:**

number = 7 if number % 2 == 0: print("Even") else: print("Odd")

**Hint:** Use the modulus operator `%` to check for even or odd numbers.

**Riddle:** I help you figure out whether a number is divisible by 2 without a remainder. What am I?

**Short Story:** Sarah wanted to determine if a number was even or odd. By using an `if-else` statement and the modulus operator, she got the result.

### 140. How do you use a `for` loop to iterate over a list?

**Code:**

my\_list = [1, 2, 3, 4] for item in my\_list: print(item)

**Hint:** Use the `for` keyword to loop over each item in a list.

**Riddle:** I help you process each item in a collection one by one. What am I?

**Short Story:** Emma needed to print each item in a list. By using a `for` loop, she efficiently iterated over the list.

### 141. How do you use a `while` loop to print numbers from 1 to 5?

**Code:**

number = 1 while number <= 5: print(number) number += 1

**Hint:** Use the `while` keyword to repeat actions while a condition is true.

**Riddle:** I keep repeating an action until a condition is no longer met. What am I?

**Short Story:** Michael wanted to print numbers from 1 to 5. By using a `while` loop, he achieved this task with ease.

### 142. How do you handle errors using `try` and `except`?

**Code:**

try: result = 10 / 0 except ZeroDivisionError: print("You can't divide by zero!")

**Hint:** Use `try` to attempt a block of code and `except` to handle specific errors.

**Riddle:** I help you manage exceptions and errors in your code gracefully. What am I?

**Short Story:** Sarah needed to handle a division by zero error. By using `try` and `except`, she managed to handle the error without crashing her program.

### 143. How do you use `finally` to ensure code executes regardless of an error?

**Code:**

try: file = open("example.txt", "r") content = file.read() finally: file.close() print("File closed.")

**Hint:** Use `finally` to ensure code runs after `try` and `except` blocks, regardless of whether an exception was raised.

**Riddle:** I ensure that certain code runs whether an exception occurs or not. What am I?

**Short Story:** Michael wanted to make sure a file was closed properly regardless of whether an error occurred. He used `finally` to ensure the file was closed.

## Functions and Modules

### 144. How do you define a function in Python?

**Code:**

def greet(name): return f"Hello, {name}!" print(greet("Alice"))

**Hint:** Use the `def` keyword to define a function.

**Riddle:** I help you group a set of statements under a name to perform a task. What am I?

**Short Story:** Sarah wanted to create a reusable block of code to greet users. By defining a function, she was able to achieve this efficiently.

### 145. How do you return multiple values from a function?

**Code:**

def min\_max(numbers): return min(numbers), max(numbers) result = min\_max([1, 2, 3, 4, 5]) print(result)

**Hint:** Use a tuple to return multiple values from a function.

**Riddle:** I allow you to return more than one value from a function. What am I?

**Short Story:** Michael needed to get both the minimum and maximum values from a list. By returning a tuple from his function, he obtained both values.

### 146. How do you use default arguments in a function?

**Code:**

def greet(name, greeting="Hello"): return f"{greeting}, {name}!" print(greet("Alice")) print(greet("Bob", "Hi"))

**Hint:** Set default values for function arguments to make them optional.

**Riddle:** I allow you to provide optional values for function parameters. What am I?

**Short Story:** Emma wanted to have a default greeting message but also allow custom messages. By using default arguments, she achieved this flexibility.

### 147. How do you use keyword arguments in a function?

**Code:**

def introduce(name, age): return f"My name is {name} and I am {age} years old." print(introduce(age=25, name="Alice"))

**Hint:** Use keyword arguments to specify which parameter you are providing a value for.

**Riddle:** I allow you to specify arguments by their names in a function call. What am I?

**Short Story:** Michael wanted to specify arguments in any order when calling his function. By using keyword arguments, he made his code more readable and flexible.

### 148. How do you import a module in Python?

**Code:**

import math print(math.sqrt(16))

**Hint:** Use the `import` keyword to include a module in your code.

**Riddle:** I help you bring in external code libraries into your program. What am I?

**Short Story:** Sarah needed to use mathematical functions in her program. She imported the `math` module and utilized its functions effectively.

### 149. How do you create your own module in Python?

**Code:**

# my\_module.py def add(a, b): return a + b # main.py import my\_module result = my\_module.add(2, 3) print(result)

**Hint:** Save your functions in a `.py` file and import it using the `import` keyword.

**Riddle:** I allow you to create reusable code that can be imported into other scripts. What am I?

**Short Story:** Michael wanted to organize his functions into a reusable module. He created a file with his functions and imported it into his main script.

## Advanced Topics

### 150. How do you define a class in Python?

**Code:**

class Dog: def \_\_init\_\_(self, name): self.name = name def bark(self): return f"{self.name} says Woof!" dog = Dog("Buddy") print(dog.bark())

**Hint:** Use the `class` keyword to define a new class and `\_\_init\_\_` to initialize attributes.

**Riddle:** I help you create new types that bundle data and behavior. What am I?

**Short Story:** Sarah wanted to model a real-world entity in code. By defining a class, she was able to create a blueprint for her objects.

### 151. How do you inherit from a parent class in Python?

**Code:**

class Animal: def speak(self): return "Animal sound" class Cat(Animal): def meow(self): return "Meow!" cat = Cat() print(cat.speak()) print(cat.meow())

**Hint:** Use parentheses to specify the parent class when defining a subclass.

**Riddle:** I allow a class to inherit attributes and methods from another class. What am I?

**Short Story:** Michael wanted his `Cat` class to reuse functionality from the `Animal` class. By inheriting from `Animal`, he achieved this efficiently.

### 152. How do you read from a file in Python?

**Code:**

with open("example.txt", "r") as file: content = file.read() print(content)

**Hint:** Use the `open` function with the `'r'` mode to read from a file.

**Riddle:** I allow you to access the contents of a file for reading. What am I?

**Short Story:** Sarah needed to read the contents of a file. By using the `open` function, she was able to access and print the file's contents.

### 153. How do you write to a file in Python?

**Code:**

with open("example.txt", "w") as file: file.write("Hello, World!")

**Hint:** Use the `open` function with the `'w'` mode to write to a file.

**Riddle:** I allow you to save data to a file. What am I?

**Short Story:** Michael wanted to save some text to a file. By using the `open` function with write mode, he was able to write the text to the file.

### 154. How do you install and use third-party libraries in Python?

**Code:**

# Install with pip # pip install requests import requests response = requests.get("https://api.example.com") print(response.json())

**Hint:** Use `pip` to install libraries and `import` to include them in your code.

**Riddle:** I help you include and use external libraries in your project. What am I?

**Short Story:** Sarah needed to make HTTP requests in her program. She installed the `requests` library using `pip` and used it to fetch data from an API.

## Data Management

### 155. How do you connect to a SQLite database in Python?

**Code:**

import sqlite3 connection = sqlite3.connect("example.db") cursor = connection.cursor() cursor.execute("CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY, name TEXT)") connection.commit() connection.close()

**Hint:** Use the `sqlite3` module to connect and interact with a SQLite database.

**Riddle:** I help you connect to and interact with a SQLite database. What am I?

**Short Story:** Sarah wanted to create a SQLite database. She used the `sqlite3` module to connect to the database and create a table.

### 156. How do you execute a SQL query using Python?

**Code:**

import sqlite3 connection = sqlite3.connect("example.db") cursor = connection.cursor() cursor.execute("INSERT INTO users (name) VALUES ('Alice')") connection.commit() connection.close()

**Hint:** Use a cursor object to execute SQL queries and commit changes.

**Riddle:** I help you run SQL commands to interact with a database. What am I?

**Short Story:** Michael needed to insert data into his database. He used a cursor object to execute an SQL query and committed the changes.

### 157. What is the SQL command to select all columns from a table?

**Code:**

SELECT \* FROM users;

**Hint:** Use the `SELECT` keyword followed by `\*` to select all columns.

**Riddle:** I help you retrieve all columns from a table in SQL. What am I?

**Short Story:** Sarah wanted to see all the data in her `users` table. She used the `SELECT \*` command to retrieve all columns.

### 158. How do you filter records in SQL using a `WHERE` clause?

**Code:**

SELECT \* FROM users WHERE name = 'Alice';

**Hint:** Use the `WHERE` clause to specify conditions for filtering records.

**Riddle:** I help you filter records based on a condition in SQL. What am I?

**Short Story:** Michael needed to find records where the name was 'Alice'. He used the `WHERE` clause to filter the results.

### 159. How do you load a CSV file into a Pandas DataFrame?

**Code:**

import pandas as pd df = pd.read\_csv("data.csv") print(df.head())

**Hint:** Use the `pd.read\_csv` function to load a CSV file into a DataFrame.

**Riddle:** I help you read CSV files into a structured DataFrame for analysis. What am I?

**Short Story:** Sarah needed to analyze data from a CSV file. She used Pandas to load the file into a DataFrame and inspected the first few rows.

### 160. How do you calculate the mean of a column in a Pandas DataFrame?

**Code:**

import pandas as pd df = pd.read\_csv("data.csv") mean\_value = df['column\_name'].mean() print(mean\_value)

**Hint:** Use the `mean` method on a DataFrame column to calculate the average value.

**Riddle:** I help you find the average value of a column in a DataFrame. What am I?

**Short Story:** Michael wanted to find the average value of a specific column in his DataFrame. By using the `mean` method, he calculated the average easily.

## Web Development

### 161. How do you create a simple Flask application?

**Code:**

from flask import Flask app = Flask(\_\_name\_\_) @app.route('/') def hello\_world(): return "Hello, World!" if \_\_name\_\_ == '\_\_main\_\_': app.run()

**Hint:** Use Flask's `Flask` class to create an application instance and define routes with the `@app.route` decorator.

**Riddle:** I help you create a basic web application with Flask. What am I?

**Short Story:** Sarah wanted to build a web application. By creating a simple Flask app, she was able to serve "Hello, World!" at the root URL.

### 162. How do you add a new route in Flask?

**Code:**

from flask import Flask app = Flask(\_\_name\_\_) @app.route('/') def home(): return "Home Page" @app.route('/about') def about(): return "About Page" if \_\_name\_\_ == '\_\_main\_\_': app.run()

**Hint:** Use the `@app.route` decorator to add new routes and define their corresponding view functions.

**Riddle:** I allow you to define multiple endpoints in a Flask application. What am I?

**Short Story:** Michael needed to create multiple pages for his Flask app. By adding new routes, he was able to handle different URLs and return different content.

## Variables and Data Types

### 165. What are the basic data types in Python?

**Code:**

# Basic data types in Python integer = 42 floating\_point = 3.14 string = "Hello, World!" boolean = True print(type(integer)) print(type(floating\_point)) print(type(string)) print(type(boolean))

**Hint:** Common data types include integers, floating points, strings, and booleans.

**Riddle:** I represent numbers, text, and truth values in Python. What am I?

**Short Story:** Sarah was learning about Python's data types. She found that integers, floats, strings, and booleans were essential for her programs.

### 166. How do you convert a string to an integer in Python?

**Code:**

string\_number = "123" integer\_number = int(string\_number) print(integer\_number)

**Hint:** Use the `int()` function to convert a string to an integer.

**Riddle:** I help you transform a string of digits into a number. What am I?

**Short Story:** Michael needed to perform mathematical operations on a number stored as a string. By converting it to an integer, he could do calculations easily.

## Operators

### 167. What is the result of using the `+` operator with strings?

**Code:**

string1 = "Hello, " string2 = "World!" result = string1 + string2 print(result)

**Hint:** The `+` operator concatenates strings.

**Riddle:** I join two strings into one. What am I?

**Short Story:** Sarah wanted to create a welcoming message. By using the `+` operator, she concatenated "Hello, " and "World!" into a single message.

### 168. How do you find the length of a list in Python?

**Code:**

my\_list = [1, 2, 3, 4, 5] length = len(my\_list) print(length)

**Hint:** Use the `len()` function to find the length of a list.

**Riddle:** I tell you how many items are in a list. What am I?

**Short Story:** Michael needed to know how many elements were in his list. By using `len()`, he quickly found the number of items.

## Strings and Lists

### 169. How do you access the first character of a string in Python?

**Code:**

my\_string = "Python" first\_char = my\_string[0] print(first\_char)

**Hint:** Use indexing to access characters in a string.

**Riddle:** I allow you to retrieve the character at a specific position in a string. What am I?

**Short Story:** Sarah needed the first letter of the word "Python". By indexing the string with `[0]`, she accessed the first character efficiently.

### 170. How do you create a list of the first 10 even numbers?

**Code:**

even\_numbers = [i for i in range(0, 20, 2)] print(even\_numbers)

**Hint:** Use list comprehension to generate a list of even numbers.

**Riddle:** I help you generate a sequence of even numbers. What am I?

**Short Story:** Michael wanted a list of even numbers. By using list comprehension, he quickly generated the first 10 even numbers in a concise manner.

## Conditional Statements

### 171. How do you write an `if` statement in Python?

**Code:**

x = 10 if x > 5: print("x is greater than 5") else: print("x is 5 or less")

**Hint:** Use `if` followed by a condition to execute code based on that condition.

**Riddle:** I allow you to execute code only if a certain condition is true. What am I?

**Short Story:** Sarah wanted to print a message based on the value of `x`. By using an `if` statement, she checked if `x` was greater than 5 and printed the appropriate message.

### 172. How do you write an `if-else` statement in Python?

**Code:**

y = 20 if y % 2 == 0: print("y is even") else: print("y is odd")

**Hint:** Use `if-else` to handle two possible outcomes based on a condition.

**Riddle:** I help you choose between two paths based on a condition. What am I?

**Short Story:** Michael needed to determine if a number was even or odd. By using `if-else`, he was able to check the condition and print the result accordingly.

## Loops

### 173. How do you write a `for` loop in Python?

**Code:**

for i in range(5): print(i)

**Hint:** Use `for` followed by a variable and an iterable to repeat code a specified number of times.

**Riddle:** I help you repeat a block of code multiple times. What am I?

**Short Story:** Sarah wanted to print numbers from 0 to 4. By using a `for` loop, she iterated through a range of numbers and printed each one.

### 174. How do you write a `while` loop in Python?

**Code:**

count = 0 while count < 5: print(count) count += 1

**Hint:** Use `while` followed by a condition to repeat code until the condition becomes false.

**Riddle:** I allow you to execute code as long as a condition remains true. What am I?

**Short Story:** Michael needed to print numbers from 0 to 4 using a condition. By using a `while` loop, he was able to repeatedly print and increment the count until it reached 5.

## Error Handling

### 175. How do you handle exceptions in Python?

**Code:**

try: x = 1 / 0 except ZeroDivisionError: print("You can't divide by zero!") finally: print("This will execute no matter what.")

**Hint:** Use `try`, `except`, and optionally `finally` to handle exceptions.

**Riddle:** I help you manage errors and ensure that some code always runs. What am I?

**Short Story:** Sarah was worried about a division by zero error. By using `try-except`, she was able to catch the error and provide a meaningful message, while `finally` ensured some code always executed.

## Functions and Modules

### 176. How do you define a function in Python?

**Code:**

def greet(name): return f"Hello, {name}!" print(greet("Alice"))

**Hint:** Use the `def` keyword to define a function and specify its parameters.

**Riddle:** I allow you to create reusable blocks of code that perform specific tasks. What am I?

**Short Story:** Michael wanted to greet users by name. By defining a `greet` function, he was able to reuse the greeting code and pass different names.

### 177. How do you import a module in Python?

**Code:**

import math print(math.sqrt(16))

**Hint:** Use the `import` statement to bring in external modules.

**Riddle:** I let you use code from other files and libraries. What am I?

**Short Story:** Sarah wanted to use mathematical functions from a module. By using `import math`, she accessed functions like `sqrt` to perform calculations.

## Object-Oriented Programming

### 178. How do you define a class in Python?

**Code:**

class Dog: def \_\_init\_\_(self, name): self.name = name def bark(self): return f"{self.name} says Woof!" my\_dog = Dog("Rex") print(my\_dog.bark())

**Hint:** Use the `class` keyword to define a class and `def` to define methods.

**Riddle:** I help you create objects and organize code into reusable blueprints. What am I?

**Short Story:** Michael wanted to model a dog. By defining a `Dog` class, he created a blueprint with attributes and methods to represent a dog in his program.

### 179. How do you create an instance of a class in Python?

**Code:**

class Car: def \_\_init\_\_(self, model): self.model = model my\_car = Car("Toyota") print(my\_car.model)

**Hint:** Call the class name with arguments to create an instance.

**Riddle:** I represent a specific object created from a class. What am I?

**Short Story:** Sarah needed a specific car object. By creating an instance of the `Car` class, she instantiated a `Toyota` car with its own model attribute.

## File Handling

### 180. How do you read a file in Python?

**Code:**

with open('file.txt', 'r') as file: content = file.read() print(content)

**Hint:** Use the `open()` function with the `'r'` mode to read a file.

**Riddle:** I let you access the contents of a file for reading. What am I?

**Short Story:** Michael needed to read from a file. By using `open('file.txt', 'r')`, he was able to read the file's contents and print them to the console.

### 181. How do you write to a file in Python?

**Code:**

with open('file.txt', 'w') as file: file.write("Hello, World!")

**Hint:** Use the `open()` function with the `'w'` mode to write to a file.

**Riddle:** I help you save data to a file. What am I?

**Short Story:** Sarah needed to save some text to a file. By using `open('file.txt', 'w')`, she wrote "Hello, World!" into the file, creating or overwriting it.

## Introduction to Databases

### 182. What is a database in Python?

**Code:**

import sqlite3 # Connect to a database (or create it if it doesn't exist) connection = sqlite3.connect('example.db') cursor = connection.cursor() # Create a table cursor.execute(''' CREATE TABLE IF NOT EXISTS users ( id INTEGER PRIMARY KEY, name TEXT ) ''') # Commit and close connection.commit() connection.close()

**Hint:** Use SQLite for simple database tasks in Python.

**Riddle:** I store data in a structured way and allow you to perform queries. What am I?

**Short Story:** Michael wanted to store user information. By using SQLite, he created a database and a table to keep track of users in his application.

### 183. How do you perform a simple SQL query in Python?

**Code:**

import sqlite3 # Connect to the database connection = sqlite3.connect('example.db') cursor = connection.cursor() # Insert a record cursor.execute("INSERT INTO users (name) VALUES ('Alice')") # Query the records cursor.execute("SELECT \* FROM users") rows = cursor.fetchall() for row in rows: print(row) # Commit and close connection.commit() connection.close()

**Hint:** Use `execute()` to perform SQL queries and `fetchall()` to retrieve results.

**Riddle:** I help you interact with a database using SQL commands. What am I?

**Short Story:** Sarah wanted to add and retrieve user records. By executing SQL queries with SQLite, she inserted a record and retrieved all users from the database.

## Control Structures

### 184. How do you use `if` statements in Python?

**Code:**

x = 10 if x > 5: print("x is greater than 5") elif x == 5: print("x is equal to 5") else: print("x is less than 5")

**Hint:** Use `if`, `elif`, and `else` to handle multiple conditions.

**Riddle:** I help you make decisions in your code based on conditions. What am I?

**Short Story:** Sarah wanted to check if a number was greater than, equal to, or less than 5. By using `if` statements, she could handle each case appropriately.

### 185. How do you use `for` loops in Python?

**Code:**

for i in range(5): print(i)

**Hint:** Use `for` to iterate over a sequence of elements.

**Riddle:** I allow you to iterate over a sequence and execute code multiple times. What am I?

**Short Story:** Michael needed to print numbers from 0 to 4. By using a `for` loop with `range`, he could efficiently iterate through the numbers and print them.

### 186. How do you handle errors using `try-except`?

**Code:**

try: result = 10 / 0 except ZeroDivisionError: print("Cannot divide by zero!")

**Hint:** Use `try` to execute code that may cause an error and `except` to handle specific errors.

**Riddle:** I help you catch and manage errors that occur during program execution. What am I?

**Short Story:** Sarah wanted to prevent her program from crashing when dividing by zero. By using `try-except`, she caught the error and displayed a user-friendly message.

## Functions and Modules

### 187. How do you pass arguments to a function in Python?

**Code:**

def add(a, b): return a + b result = add(3, 4) print(result)

**Hint:** Define parameters in the function definition and provide arguments when calling the function.

**Riddle:** I allow you to send values into a function for it to work with. What am I?

**Short Story:** Michael wanted to add two numbers. By defining a function `add` that accepts parameters `a` and `b`, he was able to pass values and get the sum.

### 188. How do you use modules in Python?

**Code:**

import math print(math.pi) print(math.sqrt(25))

**Hint:** Import modules using the `import` statement to use their functions and variables.

**Riddle:** I allow you to use code from external files and libraries. What am I?

**Short Story:** Sarah wanted to use mathematical functions in her program. By importing the `math` module, she could access constants like `pi` and functions like `sqrt`.

## Object-Oriented Programming

### 189. How do you define a class in Python with attributes?

**Code:**

class Person: def \_\_init\_\_(self, name, age): self.name = name self.age = age person1 = Person("Alice", 30) print(person1.name) print(person1.age)

**Hint:** Define attributes in the `\_\_init\_\_` method and use `self` to refer to them.

**Riddle:** I help you define a blueprint with characteristics for objects. What am I?

**Short Story:** Michael wanted to create person objects with specific attributes. By defining a `Person` class with `name` and `age` attributes, he instantiated and accessed the details of the person objects.

### 190. How do you create a method within a class in Python?

**Code:**

class Car: def \_\_init\_\_(self, make, model): self.make = make self.model = model def display\_info(self): return f"Car: {self.make} {self.model}" my\_car = Car("Toyota", "Camry") print(my\_car.display\_info())

**Hint:** Define methods within the class using `def` and use `self` to access attributes.

**Riddle:** I allow you to define functions that belong to a class and operate on its attributes. What am I?

**Short Story:** Sarah wanted her car objects to display their details. By creating a `display\_info` method in the `Car` class, she enabled each car object to show its make and model.

## File Handling

### 191. How do you handle file paths in Python?

**Code:**

import os # Get the absolute path abs\_path = os.path.abspath('file.txt') print(abs\_path)

**Hint:** Use the `os.path` module to handle file paths and directories.

**Riddle:** I help you work with file paths and directories. What am I?

**Short Story:** Michael needed to find the absolute path of a file. By using the `os.path.abspath` function, he retrieved the full path to `file.txt`.

### 192. How do you read a file line by line in Python?

**Code:**

with open('file.txt', 'r') as file: for line in file: print(line.strip())

**Hint:** Use a `with` statement to open the file and a `for` loop to iterate through lines.

**Riddle:** I help you read each line of a file without loading the entire file into memory. What am I?

**Short Story:** Sarah needed to process a large file line by line. By using a `with` statement and `for` loop, she efficiently read and processed each line from `file.txt`.

## Introduction to Databases

### 193. How do you connect to a SQLite database in Python?

**Code:**

import sqlite3 connection = sqlite3.connect('database.db') cursor = connection.cursor()

**Hint:** Use `sqlite3.connect` to connect to a database file.

**Riddle:** I allow you to establish a connection to a database for executing SQL queries. What am I?

**Short Story:** Michael needed to interact with a SQLite database. By using `sqlite3.connect`, he connected to `database.db` and prepared to execute SQL queries.

### 194. How do you create a table in a SQLite database?

**Code:**

import sqlite3 connection = sqlite3.connect('database.db') cursor = connection.cursor() cursor.execute(''' CREATE TABLE users ( id INTEGER PRIMARY KEY, username TEXT, email TEXT ) ''') connection.commit() connection.close()

**Hint:** Use the `CREATE TABLE` SQL statement to define the table structure.

**Riddle:** I help you define the structure of data in a database. What am I?

**Short Story:** Sarah wanted to set up a new table in her database to store user information. By executing the `CREATE TABLE` statement, she established a `users` table with `id`, `username`, and `email` columns.

## SQL Basics

### 195. How do you insert data into a SQLite table?

**Code:**

import sqlite3 connection = sqlite3.connect('database.db') cursor = connection.cursor() cursor.execute(''' INSERT INTO users (username, email) VALUES ('johndoe', 'johndoe@example.com') ''') connection.commit() connection.close()

**Hint:** Use the `INSERT INTO` SQL statement to add new records to a table.

**Riddle:** I allow you to add new rows of data to a table in your database. What am I?

**Short Story:** Michael needed to add a new user record to the `users` table. By executing the `INSERT INTO` statement, he added `johndoe` with an email address to the table.

### 196. How do you query data from a SQLite table?

**Code:**

import sqlite3 connection = sqlite3.connect('database.db') cursor = connection.cursor() cursor.execute('SELECT \* FROM users') rows = cursor.fetchall() for row in rows: print(row) connection.close()

**Hint:** Use the `SELECT` SQL statement to retrieve data from a table.

**Riddle:** I allow you to retrieve specific data from a database table. What am I?

**Short Story:** Sarah wanted to view all the records in the `users` table. By executing a `SELECT \*` query, she retrieved and printed all user data from the table.

## Data Analysis with Pandas

### 197. How do you read a CSV file into a Pandas DataFrame?

**Code:**

import pandas as pd df = pd.read\_csv('data.csv') print(df.head())

**Hint:** Use the `pd.read\_csv` function to load CSV data into a DataFrame.

**Riddle:** I help you load and analyze tabular data from CSV files. What am I?

**Short Story:** Michael needed to analyze data from a CSV file. By using `pd.read\_csv`, he loaded the data into a Pandas DataFrame and displayed the first few rows.

### 198. How do you filter data in a Pandas DataFrame?

**Code:**

import pandas as pd df = pd.read\_csv('data.csv') filtered\_df = df[df['age'] > 30] print(filtered\_df)

**Hint:** Use conditional statements to filter rows in a DataFrame.

**Riddle:** I help you extract specific rows of data based on conditions. What am I?

**Short Story:** Sarah wanted to filter out users older than 30 from her dataset. By applying a condition on the `age` column, she retrieved the relevant records from the DataFrame.

## Introduction to Web Frameworks

### 199. What is Flask used for in web development?

**Code:**

from flask import Flask app = Flask(\_\_name\_\_) @app.route('/') def home(): return "Hello, Flask!" if \_\_name\_\_ == '\_\_main\_\_': app.run()

**Hint:** Flask is a micro web framework for building web applications in Python.

**Riddle:** I help you create web applications with ease and flexibility. What am I?

**Short Story:** Michael wanted to create a simple web application. By using Flask, he set up a basic web server and defined a route that returned a greeting message.

### 200. How do you make a GET request to an API in Python?

**Code:**

import requests response = requests.get('https://api.example.com/data') data = response.json() print(data)

**Hint:** Use the `requests` library to make HTTP requests and handle responses.

**Riddle:** I allow you to fetch data from a web service using HTTP. What am I?

**Short Story:** Sarah needed to retrieve data from an online API. By using the `requests` library, she sent a GET request and processed the JSON response to access the data.

## Introduction to Web APIs

### 201. How do you post data to an API using Python?

**Code:**

import requests data = {'key': 'value'} response = requests.post('https://api.example.com/submit', json=data) print(response.status\_code) print(response.json())

**Hint:** Use the `requests.post` method to send data to an API endpoint.

**Riddle:** I help you send data to a web service for processing or storage. What am I?

**Short Story:** Sarah needed to send user data to a web service. By using `requests.post`, she sent the data in JSON format and checked the response from the server.

### 202. How do you handle JSON data in Python?

**Code:**

import json # Convert JSON string to Python dictionary json\_data = '{"name": "Alice", "age": 25}' data = json.loads(json\_data) print(data) # Convert Python dictionary to JSON string dict\_data = {"name": "Bob", "age": 30} json\_string = json.dumps(dict\_data) print(json\_string)

**Hint:** Use the `json` module to parse and generate JSON data.

**Riddle:** I allow you to convert between JSON strings and Python objects. What am I?

**Short Story:** Michael needed to work with JSON data in his application. By using the `json` module, he easily converted JSON strings to dictionaries and vice versa.

## Web Development Basics

### 203. What is a RESTful API?

**Answer:**

A RESTful API (Representational State Transfer) is a web service that uses HTTP requests to perform CRUD (Create, Read, Update, Delete) operations. It typically uses JSON or XML to transmit data and adheres to principles such as statelessness, client-server architecture, and uniform interface.

**Hint:** RESTful APIs are commonly used for web services and applications to communicate over the internet.

**Riddle:** I enable communication between web services using standard HTTP methods and data formats. What am I?

**Short Story:** Sarah was developing a web service and chose to use a RESTful API. This allowed her application to interact with other services over HTTP, using methods like GET and POST to manage resources.

### 204. How do you handle form data in a Flask web application?

**Code:**

from flask import Flask, request, render\_template app = Flask(\_\_name\_\_) @app.route('/submit', methods=['GET', 'POST']) def submit(): if request.method == 'POST': name = request.form['name'] email = request.form['email'] return f"Name: {name}, Email: {email}" return render\_template('form.html') if \_\_name\_\_ == '\_\_main\_\_': app.run()

**Hint:** Use the `request.form` object to access form data submitted via POST requests.

**Riddle:** I allow you to retrieve user input from HTML forms in a web application. What am I?

**Short Story:** Michael created a web form for user submissions. By using Flask’s `request.form`, he accessed the submitted data and displayed it back to the user.

## Advanced Topics

### 205. How do you implement a class in Python?

**Code:**

class Person: def \_\_init\_\_(self, name, age): self.name = name self.age = age def greet(self): return f"Hello, my name is {self.name} and I am {self.age} years old." # Creating an instance of the Person class person = Person('Alice', 30) print(person.greet())

**Hint:** Use the `class` keyword to define a class and the `\_\_init\_\_` method to initialize attributes.

**Riddle:** I allow you to create custom objects with attributes and methods. What am I?

**Short Story:** Sarah wanted to model real-world entities in her application. By defining a `Person` class, she created objects with `name` and `age` attributes and a method to greet.

### 206. How do you handle exceptions in Python?

**Code:**

try: result = 10 / 0 except ZeroDivisionError: print("Cannot divide by zero!") finally: print("This will always execute.")

**Hint:** Use `try`, `except`, and `finally` blocks to manage exceptions and ensure code execution.

**Riddle:** I help you manage errors and ensure that cleanup code runs regardless of exceptions. What am I?

**Short Story:** Michael needed to handle a potential division by zero error. By using `try` and `except`, he managed the error gracefully and ensured that certain code executed no matter what.

### 207. How do you work with environment variables in Python?

**Code:**

import os # Get an environment variable database\_url = os.getenv('DATABASE\_URL', 'default\_value') print(f"Database URL: {database\_url}") # Set an environment variable os.environ['NEW\_VAR'] = 'some\_value'

**Hint:** Use the `os` module to access and modify environment variables.

**Riddle:** I allow you to retrieve and set system-level configuration settings in your application. What am I?

**Short Story:** Sarah needed to configure her application with environment-specific settings. By using the `os` module, she accessed and set environment variables for her application’s configuration.

### 208. How do you perform file I/O operations in Python?

**Code:**

# Writing to a file with open('example.txt', 'w') as file: file.write('Hello, world!') # Reading from a file with open('example.txt', 'r') as file: content = file.read() print(content)

**Hint:** Use the `open` function with modes like `'r'` and `'w'` to read and write files.

**Riddle:** I allow you to create and access files on your filesystem. What am I?

**Short Story:** Michael needed to read and write data to a file. Using Python’s file handling capabilities, he efficiently managed file operations with context managers to ensure proper resource handling.

### 209. How do you connect to a database in Python?

**Code:**

import sqlite3 # Connect to a SQLite database conn = sqlite3.connect('example.db') cursor = conn.cursor() # Create a table cursor.execute('''CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY, name TEXT)''') # Insert a record cursor.execute('''INSERT INTO users (name) VALUES ('Alice')''') # Commit the changes and close the connection conn.commit() conn.close()

**Hint:** Use a database connector library such as `sqlite3` for SQLite databases or `psycopg2` for PostgreSQL.

**Riddle:** I help you manage data within a database, creating tables and performing operations. What am I?

**Short Story:** Sarah was developing an application that required database interaction. By using `sqlite3`, she created a table, inserted records, and managed database operations efficiently.

## Data Management

### 210. What is SQL and how is it used with Python?

**Answer:**

SQL (Structured Query Language) is a standard language for managing and manipulating databases. In Python, SQL can be used with libraries like `sqlite3`, `SQLAlchemy`, and `pymysql` to interact with databases. SQL statements are used to query, insert, update, and delete data within a database.

**Hint:** Use database libraries in Python to execute SQL queries and handle database interactions.

**Riddle:** I help you interact with databases using a specific language for querying and manipulation. What am I?

**Short Story:** Sarah needed to retrieve and manage data in her application. By using SQL with Python libraries, she could execute queries and handle database operations efficiently.

### 211. How do you use Pandas for data analysis in Python?

**Code:**

import pandas as pd # Create a DataFrame data = {'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [25, 30, 35]} df = pd.DataFrame(data) # Perform data analysis print(df.describe()) print(df['Name'])

**Hint:** Use the `pandas` library to create and manipulate DataFrames for data analysis.

**Riddle:** I allow you to work with data in tabular form and perform various data analysis operations. What am I?

**Short Story:** Michael needed to analyze data for his project. By using Pandas, he created DataFrames and performed statistical analysis to gain insights from the data.

## Web Development

### 212. What is Flask and how do you use it to create a web application?

**Code:**

from flask import Flask, render\_template app = Flask(\_\_name\_\_) @app.route('/') def home(): return render\_template('index.html') if \_\_name\_\_ == '\_\_main\_\_': app.run()

**Hint:** Flask is a micro web framework for Python. Use it to define routes and render templates for web applications.

**Riddle:** I am a lightweight web framework that helps you build web applications quickly and easily. What am I?

**Short Story:** Sarah decided to use Flask for her new web application. By defining routes and rendering HTML templates, she created a simple yet effective web app.

### 213. How do you handle user authentication in Django?

**Code:**

from django.contrib.auth import authenticate, login from django.http import HttpResponse from django.shortcuts import render def login\_view(request): if request.method == 'POST': username = request.POST['username'] password = request.POST['password'] user = authenticate(request, username=username, password=password) if user is not None: login(request, user) return HttpResponse("Logged in successfully.") else: return HttpResponse("Invalid credentials.") return render(request, 'login.html')

**Hint:** Use Django’s authentication system to manage user login and authentication.

**Riddle:** I help you manage user access and verify credentials in a web application. What am I?

**Short Story:** Michael needed to add user authentication to his Django application. By using Django’s built-in authentication system, he managed user login and credential verification effectively.

### 214. How do you make a REST API request in Python using the `requests` library?

**Code:**

import requests response = requests.get('https://api.example.com/data') if response.status\_code == 200: data = response.json() print(data) else: print(f"Failed to retrieve data: {response.status\_code}")

**Hint:** Use the `requests` library to send HTTP requests and handle API responses.

**Riddle:** I allow you to interact with web services and APIs by sending and receiving HTTP requests. What am I?

**Short Story:** Sarah needed to retrieve data from an external API for her application. By using the `requests` library, she made a GET request and processed the response to get the data she needed.

### 215. What is the purpose of a virtual environment in Python?

**Answer:**

A virtual environment is a self-contained directory that contains a Python installation for a particular version of Python, along with several additional packages. It allows you to manage dependencies for different projects separately and avoid conflicts between packages.

**Hint:** Use virtual environments to isolate project dependencies and manage package versions independently.

**Riddle:** I help you create isolated environments for Python projects to manage dependencies separately. What am I?

**Short Story:** Michael was working on multiple Python projects. By using virtual environments, he isolated dependencies for each project, preventing version conflicts and ensuring consistent behavior across projects.

### 216. How do you use `pip` to install packages?

**Code:**

# Install a package pip install requests # Install packages from a requirements file pip install -r requirements.txt

**Hint:** Use `pip` to manage Python packages and dependencies by installing them from the Python Package Index (PyPI) or from a requirements file.

**Riddle:** I am a package manager for Python that helps you install and manage libraries and dependencies. What am I?

**Short Story:** Sarah needed to add new libraries to her project. By using `pip`, she installed packages from PyPI and managed her project’s dependencies efficiently.

### 217. How do you create a simple API with Flask?

**Code:**

from flask import Flask, jsonify app = Flask(\_\_name\_\_) @app.route('/api/data') def get\_data(): return jsonify({'message': 'Hello, World!'}) if \_\_name\_\_ == '\_\_main\_\_': app.run()

**Hint:** Use Flask’s `jsonify` function to return JSON responses from your API endpoints.

**Riddle:** I allow you to create endpoints that provide data in JSON format through a web application framework. What am I?

**Short Story:** Michael wanted to build an API for his application. By using Flask, he created an endpoint that returned data in JSON format, making it easy to integrate with other services.

### 218. How do you debug a Python application?

**Answer:**

To debug a Python application, you can use various tools and techniques, such as:

* Using print statements to output variable values and track code execution.
* Using the built-in `pdb` module to set breakpoints and step through code.
* Using integrated development environment (IDE) debuggers for advanced debugging features.

**Hint:** Use debugging tools to inspect and troubleshoot issues in your code effectively.

**Riddle:** I help you examine and fix issues in your code by allowing you to step through execution and inspect variables. What am I?

**Short Story:** Sarah was struggling with bugs in her application. By using debugging tools and techniques, she was able to identify and fix issues, ensuring her application ran smoothly.

## Advanced Topics

### 219. What is Object-Oriented Programming (OOP) and how is it implemented in Python?

**Answer:**

Object-Oriented Programming (OOP) is a programming paradigm based on the concept of "objects," which can contain data in the form of fields (attributes) and code in the form of procedures (methods). Python supports OOP through its class system. Classes are used to create objects, and you can define methods and attributes within these classes.

**Code:**

class Animal: def \_\_init\_\_(self, name): self.name = name def speak(self): return f"{self.name} makes a sound" class Dog(Animal): def speak(self): return f"{self.name} barks" dog = Dog("Rex") print(dog.speak())

**Hint:** Use classes to define objects and inheritance to extend functionality in Python.

**Riddle:** I am a paradigm that uses classes and objects to organize code and manage complexity. What am I?

**Short Story:** Sarah was designing a simulation game and used OOP principles to create different types of animals, each with its own behaviors and attributes, making her code more organized and modular.

### 220. How do you handle file operations in Python?

**Code:**

# Writing to a file with open('example.txt', 'w') as file: file.write("Hello, world!") # Reading from a file with open('example.txt', 'r') as file: content = file.read() print(content)

**Hint:** Use the `open` function with modes like `'w'` for writing and `'r'` for reading. Use context managers (`with` statement) to handle files safely.

**Riddle:** I help you create, read, and write files in your application. What am I?

**Short Story:** Michael needed to store user data in a file. By using Python’s file handling functions, he could easily write data to files and read it back when needed.

### 221. How do you work with external libraries in Python?

**Answer:**

To work with external libraries in Python, you typically install them using a package manager like `pip`. After installation, you import and use these libraries in your code. Libraries provide pre-written code to perform common tasks, which saves development time and effort.

**Code:**

# Install an external library using pip # pip install requests import requests response = requests.get('https://api.example.com/data') print(response.status\_code)

**Hint:** Use `pip` to install libraries and `import` to use them in your Python code.

**Riddle:** I help you extend the functionality of Python by adding pre-built packages to your project. What am I?

**Short Story:** Sarah wanted to use a library to simplify her HTTP requests. By installing and importing the `requests` library, she could make API calls effortlessly.

## Data Management

### 222. How do you perform basic SQL queries using SQLite in Python?

**Code:**

import sqlite3 # Connect to a database (or create it if it doesn't exist) conn = sqlite3.connect('example.db') cursor = conn.cursor() # Create a table cursor.execute('''CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY, name TEXT)''') # Insert a record cursor.execute('''INSERT INTO users (name) VALUES ('Alice')''') # Query records cursor.execute('''SELECT \* FROM users''') rows = cursor.fetchall() for row in rows: print(row) conn.commit() conn.close()

**Hint:** Use `sqlite3` to connect to an SQLite database, execute SQL commands, and manage data.

**Riddle:** I help you create and manage a lightweight database using SQL queries in Python. What am I?

**Short Story:** Michael needed a simple way to store and retrieve user information. By using SQLite with Python, he could manage data efficiently without needing a full-fledged database system.

### 223. What are some common libraries used for data analysis in Python?

**Answer:**

Common libraries used for data analysis in Python include:

* **Pandas:** Provides data structures and data analysis tools.
* **Numpy:** Supports large, multi-dimensional arrays and matrices.
* **Matplotlib:** Used for creating static, animated, and interactive visualizations.
* **Seaborn:** Builds on Matplotlib and provides a high-level interface for drawing attractive statistical graphics.

**Hint:** Explore libraries like Pandas, Numpy, and Matplotlib for effective data analysis and visualization.

**Riddle:** I offer tools and libraries for handling data, performing numerical computations, and creating visualizations in Python. What am I?

**Short Story:** Sarah was analyzing a large dataset and used libraries like Pandas and Matplotlib to clean, process, and visualize the data, gaining valuable insights quickly.

## Web Development

### 224. How do you set up a basic Django project?

**Code:**

# Install Django # pip install django # Create a new Django project django-admin startproject myproject # Navigate into the project directory cd myproject # Start the development server python manage.py runserver

**Hint:** Use Django’s command-line tools to create a project and start the development server.

**Riddle:** I help you set up a web project with a built-in development server and a structured directory for your code. What am I?

**Short Story:** Michael wanted to build a web application quickly. By setting up a Django project, he had a ready-to-use framework to start building his application.

### 225. How do you make an API request using the `requests` library?

**Code:**

import requests response = requests.get('https://api.example.com/data') if response.status\_code == 200: data = response.json() print(data) else: print('Failed to retrieve data')

**Hint:** Use the `requests` library to send HTTP requests and handle responses.

**Riddle:** I enable you to fetch data from web services by sending HTTP requests and processing the responses in Python. What am I?

**Short Story:** Sarah needed to integrate her application with an external service. By using the `requests` library, she could easily make API requests and handle the responses.

## Data Management

### 226. How do you perform data analysis with Pandas?

**Code:**

import pandas as pd # Load data from a CSV file df = pd.read\_csv('data.csv') # Display the first few rows of the dataframe print(df.head()) # Calculate basic statistics print(df.describe()) # Filter data based on a condition filtered\_df = df[df['column\_name'] > value] print(filtered\_df)

**Hint:** Use Pandas functions like `read\_csv`, `describe`, and filtering operations to analyze and manipulate data.

**Riddle:** I help you analyze and manipulate data efficiently using dataframes and series. What am I?

**Short Story:** Sarah used Pandas to analyze her sales data. By loading the data into a dataframe, she could easily compute statistics and filter the information to gain insights.

### 227. What are SQL basics you need to know?

**Answer:**

Some SQL basics include:

* **SELECT:** Retrieve data from a database.
* **INSERT:** Add new records to a table.
* **UPDATE:** Modify existing records.
* **DELETE:** Remove records from a table.
* **JOIN:** Combine data from multiple tables.
* **WHERE:** Filter records based on conditions.

**Hint:** Familiarize yourself with basic SQL commands to interact with databases.

**Riddle:** I am a language used to manage and manipulate relational databases. What am I?

**Short Story:** Michael was learning to interact with databases and found that understanding SQL basics was crucial for querying and managing his data effectively.

### 228. How do you connect to a SQL database using Python?

**Code:**

import sqlite3 # Connect to a SQLite database conn = sqlite3.connect('example.db') cursor = conn.cursor() # Execute a SQL command cursor.execute('''SELECT \* FROM users''') rows = cursor.fetchall() for row in rows: print(row) conn.close()

**Hint:** Use the `sqlite3` module to connect to SQLite databases and perform SQL operations.

**Riddle:** I enable Python to interact with SQL databases by establishing connections and executing queries. What am I?

**Short Story:** Sarah needed to connect her Python application to a SQLite database. By using the `sqlite3` module, she was able to query and manage her data effectively.

## Web Development

### 229. What is the purpose of web frameworks like Flask and Django?

**Answer:**

Web frameworks like Flask and Django are designed to simplify web development by providing tools and libraries for common tasks, such as routing, handling requests, and interacting with databases. They help developers build web applications quickly and efficiently by offering a structured approach and reusable components.

**Hint:** Use web frameworks to streamline the development of web applications and manage common tasks easily.

**Riddle:** I provide a set of tools and libraries to help you build web applications efficiently, handling tasks like routing and data management. What am I?

**Short Story:** Michael wanted to develop a web application but needed a structured approach. By choosing Flask for a lightweight solution and Django for a more feature-rich framework, he could build his applications more efficiently.

### 230. How do you handle user authentication in a Flask application?

**Code:**

from flask import Flask, request, redirect, url\_for, session from werkzeug.security import generate\_password\_hash, check\_password\_hash app = Flask(\_\_name\_\_) app.secret\_key = 'your\_secret\_key' # Dummy user database users = {'user1': generate\_password\_hash('password123')} @app.route('/login', methods=['GET', 'POST']) def login(): if request.method == 'POST': username = request.form['username'] password = request.form['password'] if username in users and check\_password\_hash(users[username], password): session['user'] = username return redirect(url\_for('welcome')) else: return 'Invalid credentials' return ''' Username: Password: ''' @app.route('/welcome') def welcome(): if 'user' in session: return f'Welcome, {session["user"]}!' return redirect(url\_for('login')) if \_\_name\_\_ == '\_\_main\_\_': app.run()

**Hint:** Use `werkzeug.security` for hashing passwords and Flask’s session management for user authentication.

**Riddle:** I help you manage user sessions and secure authentication processes in your web application. What am I?

**Short Story:** Sarah was building a login system for her Flask application. By using session management and password hashing, she ensured that user authentication was secure and efficient.

## Data Management (Continued)

### 231. How do you perform basic SQL operations in a database?

**Code:**

import sqlite3 # Connect to a database conn = sqlite3.connect('example.db') cursor = conn.cursor() # Create a table cursor.execute('''CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY, name TEXT, age INTEGER)''') # Insert a record cursor.execute('''INSERT INTO users (name, age) VALUES (?, ?)''', ('Alice', 30)) # Query records cursor.execute('''SELECT \* FROM users''') print(cursor.fetchall()) # Update a record cursor.execute('''UPDATE users SET age = ? WHERE name = ?''', (31, 'Alice')) # Delete a record cursor.execute('''DELETE FROM users WHERE name = ?''', ('Alice',)) conn.commit() conn.close()

**Hint:** Use SQL commands for creating tables, inserting, updating, and deleting records in a database.

**Riddle:** I allow you to manage and manipulate data in a relational database by performing operations such as creating, inserting, updating, and deleting. What am I?

**Short Story:** Michael was working on a project that required a database. By performing basic SQL operations, he managed to create tables, insert records, and handle data efficiently.

### 232. How do you handle errors in a Python program?

**Code:**

try: # Code that might raise an exception result = 10 / 0 except ZeroDivisionError: print('Error: Division by zero is not allowed.') except Exception as e: print(f'An unexpected error occurred: {e}') finally: print('This block is always executed.')

**Hint:** Use `try`, `except`, and `finally` blocks to handle exceptions and ensure your program runs smoothly even when errors occur.

**Riddle:** I help you manage unexpected issues in your code by allowing you to catch and handle exceptions gracefully. What am I?

**Short Story:** Sarah's application crashed due to a division by zero error. By implementing error handling with `try` and `except`, she ensured her program could handle such issues without crashing.

## Functions and Modules

### 233. How do you define a function in Python?

**Code:**

def greet(name): return f'Hello, {name}!' # Call the function print(greet('Alice'))

**Hint:** Use the `def` keyword to define a function and specify parameters and return values.

**Riddle:** I am a reusable block of code that performs a specific task and can be called with different arguments. What am I?

**Short Story:** Michael needed a way to greet users in his application. By defining a function named `greet`, he could easily return a personalized greeting message.

### 234. How do you import and use modules in Python?

**Code:**

import math # Use functions from the math module print(math.sqrt(16)) # Output: 4.0 print(math.pi) # Output: 3.141592653589793

**Hint:** Use the `import` statement to bring in modules and access their functions and variables.

**Riddle:** I provide additional functionality by allowing you to use pre-written code from external files or libraries. What am I?

**Short Story:** Sarah needed to perform mathematical operations in her code. By importing the `math` module, she could use its functions and constants to simplify her calculations.

### 235. How do you create a module in Python?

**Code:**

# Save this code in a file named mymodule.py def add(a, b): return a + b def subtract(a, b): return a - b

**Hint:** Create a new Python file with functions or variables, and import it into other scripts to use its functionality.

**Riddle:** I am a file containing Python code that can be imported and used in other Python scripts. What am I?

**Short Story:** Michael needed to reuse some functions across different projects. By creating a module named `mymodule`, he could easily import and use those functions whenever needed.

### 236. How do you define a function with default arguments?

**Code:**

def greet(name='Guest'): return f'Hello, {name}!' # Call the function with and without an argument print(greet()) # Output: Hello, Guest! print(greet('Alice')) # Output: Hello, Alice!

**Hint:** Specify default values for parameters when defining a function, so they can be omitted when calling the function.

**Riddle:** I allow you to provide default values for function parameters, making them optional when calling the function. What am I?

**Short Story:** Sarah wanted to create a function that greeted users. By setting a default argument, she made it possible to greet guests if no specific name was provided.

## Advanced Topics

### 237. What is Object-Oriented Programming (OOP) in Python?

**Answer:**

Object-Oriented Programming (OOP) is a programming paradigm that uses objects and classes to structure code. In Python, OOP allows you to define classes that encapsulate data and methods, promoting code reusability and organization. Key concepts include encapsulation, inheritance, and polymorphism.

**Hint:** OOP helps organize code into objects and classes, making it more modular and reusable.

**Riddle:** I am a programming paradigm that organizes code into objects, classes, and methods, helping to model real-world entities. What am I?

**Short Story:** Michael was building a complex application and found that OOP allowed him to organize his code into classes and objects, making it easier to manage and extend his application.

### 238. How do you implement inheritance in Python?

**Code:**

class Animal: def speak(self): return 'Animal sound' class Dog(Animal): def speak(self): return 'Woof' # Create instances animal = Animal() dog = Dog() print(animal.speak()) # Output: Animal sound print(dog.speak()) # Output: Woof

**Hint:** Use parentheses to inherit from a parent class and override methods in the child class.

**Riddle:** I allow you to create a new class based on an existing class, inheriting its properties and methods. What am I?

**Short Story:** Sarah wanted to create a class for dogs that extended the functionality of a base animal class. By using inheritance, she was able to define specific behaviors for dogs while reusing common features from the animal class.

### 239. What is polymorphism in Python?

**Answer:**

Polymorphism is a concept in programming that allows methods to do different things based on the object it is acting upon. In Python, polymorphism allows methods to be overridden in child classes, providing different implementations for the same method name. It enables objects of different classes to be treated as objects of a common superclass.

**Hint:** Polymorphism allows objects to use methods that are named the same but behave differently depending on the object.

**Riddle:** I allow a method to have different implementations based on the type of object that invokes it. What am I?

**Short Story:** Michael's application needed different behaviors for different types of objects. By implementing polymorphism, he was able to use the same method name for different objects while providing specific functionality for each type.

### 240. How do you implement encapsulation in Python?

**Code:**

class Person: def \_\_init\_\_(self, name): self.\_\_name = name # Private attribute def get\_name(self): return self.\_\_name def set\_name(self, name): self.\_\_name = name # Create an instance person = Person('Alice') print(person.get\_name()) # Output: Alice # Modify name using setter method person.set\_name('Bob') print(person.get\_name()) # Output: Bob

**Hint:** Use double underscores to define private attributes and provide public methods to access or modify them.

**Riddle:** I help you hide the internal details of an object and only expose necessary methods for interaction. What am I?

**Short Story:** Sarah wanted to ensure that the internal details of her `Person` class were not exposed directly. By implementing encapsulation, she controlled access to the class's attributes through getter and setter methods.

## Project 1: Basic Calculator

### Question:

Create a basic calculator that performs addition, subtraction, multiplication, and division.

**Code:**

def calculator(): operation = input('Enter operation (+, -, \*, /): ') num1 = float(input('Enter first number: ')) num2 = float(input('Enter second number: ')) if operation == '+': print(f'The result is {num1 + num2}') elif operation == '-': print(f'The result is {num1 - num2}') elif operation == '\*': print(f'The result is {num1 \* num2}') elif operation == '/': if num2 != 0: print(f'The result is {num1 / num2}') else: print('Error: Division by zero!') else: print('Invalid operation!') calculator()

**Hint:** Use conditional statements to handle different operations.

**Riddle:** I take two numbers and an operation as input and provide the result based on the operation. What am I?

**Short Story:** Alice wanted to create a tool for quick calculations. By implementing a basic calculator, she could easily perform arithmetic operations and solve math problems quickly.

## Project 2: To-Do List

### Question:

Create a simple to-do list application where users can add, remove, and view tasks.

**Code:**

tasks = [] def show\_tasks(): if tasks: print('To-Do List:') for i, task in enumerate(tasks, 1): print(f'{i}. {task}') else: print('No tasks in the list.') def add\_task(): task = input('Enter a task: ') tasks.append(task) print('Task added!') def remove\_task(): show\_tasks() try: task\_index = int(input('Enter the task number to remove: ')) - 1 if 0 <= task\_index < len(tasks): removed\_task = tasks.pop(task\_index) print(f'Task "{removed\_task}" removed!') else: print('Invalid task number!') except ValueError: print('Invalid input!') while True: print('\n1. Show tasks\n2. Add task\n3. Remove task\n4. Quit') choice = input('Enter your choice: ') if choice == '1': show\_tasks() elif choice == '2': add\_task() elif choice == '3': remove\_task() elif choice == '4': break else: print('Invalid choice!')

**Hint:** Use a list to store tasks and provide options to add, remove, and display tasks.

**Riddle:** I help you manage your tasks by allowing you to add, remove, and view them. What am I?

**Short Story:** Sarah wanted to keep track of her daily tasks. By creating a to-do list application, she was able to stay organized and manage her time effectively.

## Project 3: Number Guessing Game

### Question:

Create a number guessing game where the user has to guess a randomly generated number within a certain range.

**Code:**

import random def guess\_number(): number = random.randint(1, 100) attempts = 0 while True: guess = int(input('Guess a number between 1 and 100: ')) attempts += 1 if guess < number: print('Too low!') elif guess > number: print('Too high!') else: print(f'Congratulations! You guessed the number in {attempts} attempts.') break guess\_number()

**Hint:** Use the `random` module to generate a random number and provide feedback based on the user’s guess.

**Riddle:** I challenge you to guess a hidden number with the help of hints about whether your guess is too high or too low. What am I?

**Short Story:** Michael wanted to create a fun game for his friends. By developing a number guessing game, he added an element of excitement and competition to their gatherings.

## Project 4: Simple Password Generator

### Question:

Create a password generator that creates a random password with a specified length.

**Code:**

import random import string def generate\_password(length): characters = string.ascii\_letters + string.digits + string.punctuation return ''.join(random.choice(characters) for \_ in range(length)) length = int(input('Enter the password length: ')) print('Generated password:', generate\_password(length))

**Hint:** Use the `random` and `string` modules to generate a random combination of characters for the password.

**Riddle:** I create a secure combination of letters, digits, and symbols to protect your information. What am I?

**Short Story:** Sarah needed a secure password for her new accounts. By building a password generator, she ensured that her passwords were strong and unique.

## Project 5: BMI Calculator

### Question:

Create a BMI (Body Mass Index) calculator that takes user weight and height as input and computes the BMI.

**Code:**

def calculate\_bmi(weight, height): return weight / (height \*\* 2) weight = float(input('Enter your weight (kg): ')) height = float(input('Enter your height (m): ')) bmi = calculate\_bmi(weight, height) print(f'Your BMI is: {bmi:.2f}') if bmi < 18.5: print('You are underweight.') elif 18.5 <= bmi < 24.9: print('You have a normal weight.') elif 25 <= bmi < 29.9: print('You are overweight.') else: print('You are obese.')

**Hint:** BMI is calculated as weight divided by the square of height.

**Riddle:** I use your weight and height to tell you if you are underweight, normal weight, overweight, or obese. What am I?

**Short Story:** John wanted to monitor his health. By creating a BMI calculator, he could easily assess his weight status and make informed decisions about his diet and exercise.

## Project 6: Prime Number Checker

### Question:

Write a program to check if a given number is a prime number.

**Code:**

def is\_prime(number): if number <= 1: return False for i in range(2, int(number \*\* 0.5) + 1): if number % i == 0: return False return True number = int(input('Enter a number: ')) if is\_prime(number): print(f'{number} is a prime number.') else: print(f'{number} is not a prime number.')

**Hint:** Check divisibility from 2 up to the square root of the number.

**Riddle:** I am a number greater than 1 that is only divisible by 1 and itself. What am I?

**Short Story:** Emily was learning about number theory and wanted to identify prime numbers. By writing a prime checker, she could easily test numbers and understand the properties of primes.

## Project 7: Currency Converter

### Question:

Create a currency converter that converts an amount from one currency to another based on given exchange rates.

**Code:**

def convert\_currency(amount, rate): return amount \* rate amount = float(input('Enter amount in USD: ')) rate = float(input('Enter exchange rate to EUR: ')) converted\_amount = convert\_currency(amount, rate) print(f'Amount in EUR: {converted\_amount:.2f}')

**Hint:** Multiply the amount by the exchange rate to get the converted value.

**Riddle:** I take an amount and a rate to convert it into another currency. What am I?

**Short Story:** Daniel was traveling to Europe and needed to convert his dollars to euros. By creating a currency converter, he could quickly see how much he would have in the local currency.

## Project 8: Text File Reader

### Question:

Write a program to read and display the contents of a text file.

**Code:**

filename = input('Enter the filename: ') try: with open(filename, 'r') as file: contents = file.read() print('File contents:') print(contents) except FileNotFoundError: print('File not found.') except IOError: print('Error reading the file.')

**Hint:** Use the `open` function in read mode and handle exceptions if the file does not exist or cannot be read.

**Riddle:** I allow you to view the contents of a file stored on your computer. What am I?

**Short Story:** Laura needed to review a document she had saved earlier. By creating a text file reader, she could easily open and display the content of her files without needing additional software.

## Project 9: Email Validator

### Question:

Create a program to validate if an input string is a properly formatted email address.

**Code:**

import re def is\_valid\_email(email): pattern = r'^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$' return re.match(pattern, email) is not None email = input('Enter an email address: ') if is\_valid\_email(email): print('Valid email address.') else: print('Invalid email address.')

**Hint:** Use regular expressions to match the email pattern.

**Riddle:** I check if an email address is formatted correctly with an @ symbol and a domain name. What am I?

**Short Story:** Mike needed to validate user email addresses for his application. By writing an email validator, he could ensure that all inputted addresses were in the correct format before processing them further.

## Project 10: Rock, Paper, Scissors Game

### Question:

Implement the classic game of Rock, Paper, Scissors where the user plays against the computer.

**Code:**

import random def play\_rps(): choices = ['rock', 'paper', 'scissors'] user\_choice = input('Enter rock, paper, or scissors: ').lower() if user\_choice not in choices: print('Invalid choice!') return computer\_choice = random.choice(choices) print(f'Computer chose: {computer\_choice}') if user\_choice == computer\_choice: print('It\'s a tie!') elif (user\_choice == 'rock' and computer\_choice == 'scissors') or \ (user\_choice == 'paper' and computer\_choice == 'rock') or \ (user\_choice == 'scissors' and computer\_choice == 'paper'): print('You win!') else: print('You lose!') play\_rps()

**Hint:** Use conditional statements to determine the winner based on the choices made by the user and the computer.

**Riddle:** I am a game where you choose rock, paper, or scissors to play against the computer. What am I?

**Short Story:** Alex wanted to create a fun and interactive game for friends. By developing Rock, Paper, Scissors, he was able to provide entertainment and practice programming skills at the same time.

## Project 11: Simple Quiz

### Question:

Develop a simple quiz program where users answer multiple-choice questions, and the program scores their answers.

**Code:**

def ask\_question(question, options, correct\_option): print(question) for idx, option in enumerate(options, 1): print(f'{idx}. {option}') answer = int(input('Enter the number of your choice: ')) return answer == correct\_option score = 0 questions = [ {"question": "What is the capital of France?", "options": ["Berlin", "Madrid", "Paris", "Lisbon"], "correct\_option": 3}, {"question": "What is 2 + 2?", "options": ["3", "4", "5", "6"], "correct\_option": 2} ] for q in questions: if ask\_question(q["question"], q["options"], q["correct\_option"]): print('Correct!') score += 1 else: print('Wrong!') print(f'Your score: {score}/{len(questions)}')

**Hint:** Use a loop to iterate through questions and keep track of the user's score.

**Riddle:** I ask questions and check if your answers are correct. What am I?

**Short Story:** Sarah wanted to create a fun quiz for her friends. By writing a simple quiz program, she could challenge their knowledge and keep track of their scores.

## Project 12: Number Guessing Game

### Question:

Write a number guessing game where the user tries to guess a randomly generated number within a certain range.

**Code:**

import random def guess\_number(): number = random.randint(1, 100) attempts = 0 while True: guess = int(input('Guess the number (between 1 and 100): ')) attempts += 1 if guess < number: print('Too low!') elif guess > number: print('Too high!') else: print(f'Congratulations! You guessed the number in {attempts} attempts.') break guess\_number()

**Hint:** Use a loop to allow multiple guesses and provide feedback based on whether the guess is too high or too low.

**Riddle:** I am a game where you guess a number and receive hints until you find the correct one. What am I?

**Short Story:** Mark enjoyed challenging his friends with guessing games. By creating a number guessing game, he added a fun element to his programming projects.

## Project 13: Contact Book

### Question:

Create a simple contact book where you can add, view, and delete contacts. Each contact should have a name and phone number.

**Code:**

contacts = {} def add\_contact(name, phone): contacts[name] = phone def view\_contacts(): for name, phone in contacts.items(): print(f'Name: {name}, Phone: {phone}') def delete\_contact(name): if name in contacts: del contacts[name] else: print('Contact not found.') while True: action = input('Choose an action: add, view, delete, or quit: ').lower() if action == 'add': name = input('Enter contact name: ') phone = input('Enter phone number: ') add\_contact(name, phone) elif action == 'view': view\_contacts() elif action == 'delete': name = input('Enter contact name to delete: ') delete\_contact(name) elif action == 'quit': break else: print('Invalid action.')

**Hint:** Use a dictionary to store contacts and provide options for adding, viewing, and deleting them.

**Riddle:** I help you keep track of names and phone numbers, letting you add, view, or remove them as needed. What am I?

**Short Story:** Lisa wanted to manage her contacts more effectively. By creating a contact book, she could easily keep track of important phone numbers and information.

## Project 14: To-Do List

### Question:

Build a to-do list application that allows users to add, view, and mark tasks as completed.

**Code:**

tasks = [] def add\_task(task): tasks.append({"task": task, "completed": False}) def view\_tasks(): for idx, task in enumerate(tasks): status = 'Completed' if task['completed'] else 'Not Completed' print(f'{idx + 1}. {task["task"]} - {status}') def complete\_task(index): if 0 <= index < len(tasks): tasks[index]['completed'] = True else: print('Task not found.') while True: action = input('Choose an action: add, view, complete, or quit: ').lower() if action == 'add': task = input('Enter the task: ') add\_task(task) elif action == 'view': view\_tasks() elif action == 'complete': index = int(input('Enter task number to mark as completed: ')) - 1 complete\_task(index) elif action == 'quit': break else: print('Invalid action.')

**Hint:** Use a list to store tasks and a dictionary to keep track of their completion status.

**Riddle:** I help you manage tasks by allowing you to add, view, and complete them. What am I?

**Short Story:** Tom wanted to organize his daily tasks better. By developing a to-do list application, he could keep track of what needed to be done and monitor his progress.

## Project 15: Basic Calculator

### Question:

Develop a basic calculator that can perform addition, subtraction, multiplication, and division based on user input.

**Code:**

def calculate(a, b, operator): if operator == '+': return a + b elif operator == '-': return a - b elif operator == '\*': return a \* b elif operator == '/': if b != 0: return a / b else: return 'Error: Division by zero.' else: return 'Invalid operator.' a = float(input('Enter first number: ')) b = float(input('Enter second number: ')) operator = input('Enter operator (+, -, \*, /): ') result = calculate(a, b, operator) print(f'Result: {result}')

**Hint:** Use conditional statements to perform the operation based on the chosen operator.

**Riddle:** I can perform basic arithmetic operations including addition, subtraction, multiplication, and division. What am I?

**Short Story:** Rachel needed a simple tool for performing calculations quickly. By creating a basic calculator, she could easily perform mathematical operations without needing a separate application.

## Project 16: Temperature Converter

### Question:

Create a program that converts temperatures between Celsius, Fahrenheit, and Kelvin.

**Code:**

def celsius\_to\_fahrenheit(celsius): return (celsius \* 9/5) + 32 def fahrenheit\_to\_celsius(fahrenheit): return (fahrenheit - 32) \* 5/9 def celsius\_to\_kelvin(celsius): return celsius + 273.15 def kelvin\_to\_celsius(kelvin): return kelvin - 273.15 temp = float(input('Enter temperature: ')) unit = input('Enter unit (C/F/K): ').upper() if unit == 'C': print(f'Fahrenheit: {celsius\_to\_fahrenheit(temp)}') print(f'Kelvin: {celsius\_to\_kelvin(temp)}') elif unit == 'F': print(f'Celsius: {fahrenheit\_to\_celsius(temp)}') elif unit == 'K': print(f'Celsius: {kelvin\_to\_celsius(temp)}') else: print('Invalid unit.')

**Hint:** Use functions to convert between units and prompt the user for input.

**Riddle:** I convert temperatures from one scale to another, ensuring you can work with Celsius, Fahrenheit, and Kelvin. What am I?

**Short Story:** Emily needed to compare temperatures in different scales for her science project. By creating a temperature converter, she could quickly switch between Celsius, Fahrenheit, and Kelvin.

## Project 17: Palindrome Checker

### Question:

Write a program that checks if a given string is a palindrome (a word that reads the same backward as forward).

**Code:**

def is\_palindrome(s): s = s.lower() return s == s[::-1] string = input('Enter a string: ') if is\_palindrome(string): print('The string is a palindrome.') else: print('The string is not a palindrome.')

**Hint:** Check if the string reads the same forward and backward by reversing it.

**Riddle:** I am a word or phrase that reads the same forwards and backwards. What am I?

**Short Story:** John was fascinated by palindromes and wanted to check if a word or phrase met this criteria. By writing a palindrome checker, he could explore various palindromic sequences.

## Project 18: Simple Bank Account

### Question:

Create a simple bank account program where users can deposit, withdraw, and check their balance.

**Code:**

class BankAccount: def \_\_init\_\_(self): self.balance = 0 def deposit(self, amount): self.balance += amount print(f'Deposited ${amount}. New balance: ${self.balance}') def withdraw(self, amount): if amount <= self.balance: self.balance -= amount print(f'Withdrew ${amount}. New balance: ${self.balance}') else: print('Insufficient funds.') def check\_balance(self): print(f'Current balance: ${self.balance}') account = BankAccount() while True: action = input('Choose an action: deposit, withdraw, check, or quit: ').lower() if action == 'deposit': amount = float(input('Enter amount to deposit: ')) account.deposit(amount) elif action == 'withdraw': amount = float(input('Enter amount to withdraw: ')) account.withdraw(amount) elif action == 'check': account.check\_balance() elif action == 'quit': break else: print('Invalid action.')

**Hint:** Use a class to encapsulate account details and methods for deposit, withdrawal, and balance checking.

**Riddle:** I manage your money, allowing you to deposit, withdraw, and check your balance. What am I?

**Short Story:** Olivia wanted to keep track of her finances more effectively. By creating a simple bank account program, she could manage her deposits, withdrawals, and balance with ease.

## Project 19: Countdown Timer

### Question:

Build a countdown timer that counts down from a specified number of seconds and alerts the user when time is up.

**Code:**

import time def countdown\_timer(seconds): while seconds: mins, secs = divmod(seconds, 60) timeformat = '{:02d}:{:02d}'.format(mins, secs) print(timeformat, end='\r') time.sleep(1) seconds -= 1 print('Time\'s up!') seconds = int(input('Enter the number of seconds for the countdown: ')) countdown\_timer(seconds)

**Hint:** Use a loop and `time.sleep()` to create a countdown effect and display the remaining time.

**Riddle:** I count down from a given number of seconds, showing the time left and notifying you when it reaches zero. What am I?

**Short Story:** David needed a way to time his cooking. By creating a countdown timer, he could easily monitor the cooking time and receive a notification when it was over.

## Project 20: Random Password Generator

### Question:

Write a program that generates a random password with a mix of letters, numbers, and special characters.

**Code:**

import random import string def generate\_password(length): characters = string.ascii\_letters + string.digits + string.punctuation return ''.join(random.choice(characters) for i in range(length)) length = int(input('Enter the length of the password: ')) print(f'Generated password: {generate\_password(length)}')

**Hint:** Use the `random` module to select characters from a set of letters, digits, and symbols to generate a secure password.

**Riddle:** I generate a secure password using a combination of letters, numbers, and special characters. What am I?

**Short Story:** Sarah wanted to ensure her online accounts were secure. By creating a random password generator, she could easily generate strong and unique passwords for each of her accounts.

## Project 21: Contact Book

### Question:

Create a contact book application that allows users to add, delete, and search for contacts.

**Code:**

class ContactBook: def \_\_init\_\_(self): self.contacts = {} def add\_contact(self, name, phone): self.contacts[name] = phone print(f'Contact {name} added.') def delete\_contact(self, name): if name in self.contacts: del self.contacts[name] print(f'Contact {name} deleted.') else: print('Contact not found.') def search\_contact(self, name): return self.contacts.get(name, 'Contact not found.') book = ContactBook() while True: action = input('Choose an action: add, delete, search, or quit: ').lower() if action == 'add': name = input('Enter name: ') phone = input('Enter phone number: ') book.add\_contact(name, phone) elif action == 'delete': name = input('Enter name: ') book.delete\_contact(name) elif action == 'search': name = input('Enter name: ') print(f'Phone number: {book.search\_contact(name)}') elif action == 'quit': break else: print('Invalid action.')

**Hint:** Use a dictionary to store contact names and phone numbers, and implement methods for adding, deleting, and searching contacts.

**Riddle:** I store names and phone numbers, allowing you to add, delete, and search for contacts. What am I?

**Short Story:** Lisa wanted a way to keep track of her friends' phone numbers. By creating a contact book application, she could easily manage her contacts and find their numbers whenever needed.

## Project 22: Unit Converter

### Question:

Build a unit converter that can convert between different units of measurement, such as length, weight, and volume.

**Code:**

def convert\_length(value, unit\_from, unit\_to): conversion\_factors = { 'm\_to\_km': 0.001, 'km\_to\_m': 1000, 'm\_to\_cm': 100, 'cm\_to\_m': 0.01 } key = f'{unit\_from}\_to\_{unit\_to}' return value \* conversion\_factors.get(key, 1) value = float(input('Enter value to convert: ')) unit\_from = input('Enter from unit (m, km, cm): ') unit\_to = input('Enter to unit (m, km, cm): ') print(f'Converted value: {convert\_length(value, unit\_from, unit\_to)} {unit\_to}')

**Hint:** Create conversion factors for different units and apply the appropriate conversion based on user input.

**Riddle:** I transform values between various units of measurement, helping you switch between meters, kilometers, and centimeters. What am I?

**Short Story:** Mark needed to convert measurements for his DIY project. By creating a unit converter, he could easily switch between different units of length, weight, and volume.

## Project 23: Todo List Application

### Question:

Create a to-do list application where users can add, remove, and view tasks.

**Code:**

class TodoList: def \_\_init\_\_(self): self.tasks = [] def add\_task(self, task): self.tasks.append(task) print(f'Task "{task}" added.') def remove\_task(self, task): if task in self.tasks: self.tasks.remove(task) print(f'Task "{task}" removed.') else: print('Task not found.') def view\_tasks(self): print('Tasks:') for task in self.tasks: print(f'- {task}') todo\_list = TodoList() while True: action = input('Choose an action: add, remove, view, or quit: ').lower() if action == 'add': task = input('Enter task: ') todo\_list.add\_task(task) elif action == 'remove': task = input('Enter task: ') todo\_list.remove\_task(task) elif action == 'view': todo\_list.view\_tasks() elif action == 'quit': break else: print('Invalid action.')

**Hint:** Use a list to store tasks and implement methods for adding, removing, and viewing tasks.

**Riddle:** I keep track of your tasks, allowing you to add, remove, and view them whenever you like. What am I?

**Short Story:** Jessica wanted to stay organized and manage her tasks effectively. By creating a to-do list application, she could keep track of her daily activities and never miss a deadline.

## Project 24: Basic Calculator

### Question:

Develop a basic calculator that can perform addition, subtraction, multiplication, and division.

**Code:**

def add(x, y): return x + y def subtract(x, y): return x - y def multiply(x, y): return x \* y def divide(x, y): if y != 0: return x / y else: return 'Error: Division by zero.' print('Select operation:') print('1. Add') print('2. Subtract') print('3. Multiply') print('4. Divide') choice = input('Enter choice (1/2/3/4): ') num1 = float(input('Enter first number: ')) num2 = float(input('Enter second number: ')) if choice == '1': print(f'Result: {add(num1, num2)}') elif choice == '2': print(f'Result: {subtract(num1, num2)}') elif choice == '3': print(f'Result: {multiply(num1, num2)}') elif choice == '4': print(f'Result: {divide(num1, num2)}') else: print('Invalid input.')

**Hint:** Define functions for each arithmetic operation and use user input to select the operation and numbers.

**Riddle:** I perform basic arithmetic operations, allowing you to add, subtract, multiply, and divide numbers. What am I?

**Short Story:** Alex needed a simple tool for performing basic arithmetic operations. By creating a basic calculator, he could quickly perform calculations and solve math problems.

## Project 25: Weather App

### Question:

Build a weather application that fetches and displays the current weather for a given location using a weather API.

**Code:**

import requests def get\_weather(city): api\_key = 'your\_api\_key\_here' base\_url = 'http://api.openweathermap.org/data/2.5/weather' params = {'q': city, 'appid': api\_key, 'units': 'metric'} response = requests.get(base\_url, params=params) data = response.json() if data['cod'] == 200: temperature = data['main']['temp'] weather\_description = data['weather'][0]['description'] print(f'Temperature: {temperature}°C') print(f'Weather: {weather\_description}') else: print('City not found.') city = input('Enter city name: ') get\_weather(city)

**Hint:** Use the `requests` library to fetch weather data from an API and parse the response to display the temperature and weather description.

**Riddle:** I provide current weather information for any city using data from an online source. What am I?

**Short Story:** Maria wanted to know the current weather before heading out. By creating a weather app, she could easily check the weather for any city and plan her day accordingly.

## Project 26: Quiz Game

### Question:

Develop a quiz game where users answer multiple-choice questions and receive a score at the end.

**Code:**

questions = [ {'question': 'What is the capital of France?', 'choices': ['Paris', 'London', 'Rome', 'Berlin'], 'answer': 'Paris'}, {'question': 'What is 2 + 2?', 'choices': ['3', '4', '5', '6'], 'answer': '4'}, {'question': 'What is the largest planet in our solar system?', 'choices': ['Earth', 'Mars', 'Jupiter', 'Saturn'], 'answer': 'Jupiter'} ] score = 0 for q in questions: print(q['question']) for i, choice in enumerate(q['choices']): print(f'{i + 1}. {choice}') answer = input('Enter your choice (1/2/3/4): ') if q['choices'][int(answer) - 1] == q['answer']: score += 1 print('Correct!') else: print('Wrong!') print(f'Your final score is {score}/{len(questions)}')

**Hint:** Create a list of questions and choices, and keep track of the score based on user responses.

**Riddle:** I test your knowledge with questions and choices, giving you a score at the end. What am I?

**Short Story:** John wanted a fun way to test his knowledge. By building a quiz game, he could challenge himself with questions and keep track of his score.

## Project 27: Expense Tracker

### Question:

Create an expense tracker application that records expenses and displays a summary of total spending.

**Code:**

expenses = [] def add\_expense(amount, category): expenses.append({'amount': amount, 'category': category}) print(f'Expense of ${amount} added to category {category}.') def show\_summary(): total = sum(expense['amount'] for expense in expenses) print(f'Total spending: ${total}') for category in set(expense['category'] for expense in expenses): category\_total = sum(expense['amount'] for expense in expenses if expense['category'] == category) print(f'{category}: ${category\_total}') while True: action = input('Choose an action: add, summary, or quit: ').lower() if action == 'add': amount = float(input('Enter amount: ')) category = input('Enter category: ') add\_expense(amount, category) elif action == 'summary': show\_summary() elif action == 'quit': break else: print('Invalid action.')

**Hint:** Store expenses in a list of dictionaries and provide methods to add expenses and display a summary.

**Riddle:** I keep track of your spending and categorize your expenses, showing you a summary of your total costs. What am I?

**Short Story:** Emma wanted to keep her finances in check. By developing an expense tracker, she could easily record her expenses and monitor her spending habits.

## Project 28: Digital Diary

### Question:

Build a digital diary application where users can write and save journal entries.

**Code:**

entries = [] def add\_entry(title, content): entries.append({'title': title, 'content': content}) print(f'Entry "{title}" added.') def view\_entries(): for entry in entries: print(f'Title: {entry["title"]}') print(f'Content: {entry["content"]}') print('---') while True: action = input('Choose an action: add, view, or quit: ').lower() if action == 'add': title = input('Enter title: ') content = input('Enter content: ') add\_entry(title, content) elif action == 'view': view\_entries() elif action == 'quit': break else: print('Invalid action.')

**Hint:** Use a list to store journal entries, and implement methods to add and view entries.

**Riddle:** I store your thoughts and memories in written form, allowing you to view them whenever you wish. What am I?

**Short Story:** Sarah wanted a private place to record her thoughts. By creating a digital diary application, she could easily jot down her experiences and reflect on them later.

## Project 29: Password Generator

### Question:

Develop a password generator that creates strong, random passwords of varying lengths.

**Code:**

import random import string def generate\_password(length): characters = string.ascii\_letters + string.digits + string.punctuation return ''.join(random.choice(characters) for \_ in range(length)) length = int(input('Enter password length: ')) print(f'Generated password: {generate\_password(length)}')

**Hint:** Use the `random` and `string` libraries to generate a random password with letters, digits, and punctuation.

**Riddle:** I create secure and random passwords of any length, making it harder for anyone to guess. What am I?

**Short Story:** Alex wanted a way to generate strong passwords for his online accounts. By building a password generator, he could create secure passwords quickly and easily.

## Project 30: Number Guessing Game

### Question:

Build a number guessing game where users try to guess a randomly generated number within a certain number of attempts.

**Code:**

import random def number\_guessing\_game(): number = random.randint(1, 100) attempts = 0 while True: guess = int(input('Guess the number (between 1 and 100): ')) attempts += 1 if guess < number: print('Too low!') elif guess > number: print('Too high!') else: print(f'Congratulations! You guessed the number in {attempts} attempts.') break number\_guessing\_game()

**Hint:** Generate a random number and give feedback to the user based on their guesses until they find the correct number.

**Riddle:** I challenge you to guess a number with hints, and reward you when you find the right answer. What am I?

**Short Story:** Emma enjoyed playing guessing games and wanted to create her own. By developing a number guessing game, she could test her intuition and improve her guessing skills.

## Project 31: Currency Converter

### Question:

Create a currency converter that converts an amount from one currency to another using real-time exchange rates from an API.

**Code:**

import requests def convert\_currency(amount, from\_currency, to\_currency): api\_key = 'your\_api\_key\_here' base\_url = 'https://api.exchangerate-api.com/v4/latest/' response = requests.get(base\_url + from\_currency) data = response.json() rate = data['rates'].get(to\_currency, 1) return amount \* rate amount = float(input('Enter amount: ')) from\_currency = input('Enter from currency code (e.g., USD): ') to\_currency = input('Enter to currency code (e.g., EUR): ') converted\_amount = convert\_currency(amount, from\_currency, to\_currency) print(f'{amount} {from\_currency} is equal to {converted\_amount:.2f} {to\_currency}')

**Hint:** Use an API to get real-time exchange rates and perform the currency conversion based on user input.

**Riddle:** I help you convert money between different currencies, providing up-to-date rates for accurate conversion. What am I?

**Short Story:** Tom needed to travel abroad and wanted to know how much his money would be worth in different countries. By building a currency converter, he could easily calculate the conversion rates and plan his budget.

## Project 32: Task Manager

### Question:

Develop a task manager application that allows users to add, view, and delete tasks from a to-do list.

**Code:**

tasks = [] def add\_task(task): tasks.append(task) print(f'Task "{task}" added.') def view\_tasks(): print('Tasks:') for i, task in enumerate(tasks): print(f'{i + 1}. {task}') def delete\_task(index): if 0 <= index < len(tasks): removed\_task = tasks.pop(index) print(f'Task "{removed\_task}" deleted.') else: print('Invalid index.') while True: action = input('Choose an action: add, view, delete, or quit: ').lower() if action == 'add': task = input('Enter task: ') add\_task(task) elif action == 'view': view\_tasks() elif action == 'delete': index = int(input('Enter task number to delete: ')) - 1 delete\_task(index) elif action == 'quit': break else: print('Invalid action.')

**Hint:** Use a list to store tasks and implement methods to add, view, and delete tasks from the list.

**Riddle:** I keep track of your to-do items, letting you add, view, and remove tasks from your list. What am I?

**Short Story:** Jane had a lot of tasks to manage and needed a way to stay organized. By creating a task manager, she could keep track of her to-do list and ensure nothing important was forgotten.

## Project 33: Calendar Application

### Question:

Build a calendar application that allows users to add, view, and remove events on a monthly calendar view.

**Code:**

import calendar events = {} def add\_event(day, event): if day in events: events[day].append(event) else: events[day] = [event] print(f'Event "{event}" added for day {day}.') def view\_calendar(year, month): cal = calendar.month(year, month) print(cal) for day, event\_list in events.items(): print(f'Events on {day}:') for event in event\_list: print(f' - {event}') def remove\_event(day, event): if day in events and event in events[day]: events[day].remove(event) if not events[day]: del events[day] print(f'Event "{event}" removed from day {day}.') else: print('Event not found.') year = int(input('Enter year: ')) month = int(input('Enter month (1-12): ')) while True: action = input('Choose an action: add, view, remove, or quit: ').lower() if action == 'add': day = int(input('Enter day: ')) event = input('Enter event: ') add\_event(day, event) elif action == 'view': view\_calendar(year, month) elif action == 'remove': day = int(input('Enter day: ')) event = input('Enter event to remove: ') remove\_event(day, event) elif action == 'quit': break else: print('Invalid action.')

**Hint:** Use a dictionary to store events by day, and provide methods to add, view, and remove events from the calendar.

**Riddle:** I let you manage your events on a monthly calendar, making sure you never forget an important date. What am I?

**Short Story:** Alice wanted to keep track of her appointments and events. By developing a calendar application, she could easily organize her schedule and stay on top of her commitments.

## Project 34: Recipe Book

### Question:

Create a recipe book application where users can add, view, and search for recipes based on ingredients.

**Code:**

recipes = {} def add\_recipe(name, ingredients, instructions): recipes[name] = {'ingredients': ingredients, 'instructions': instructions} print(f'Recipe "{name}" added.') def view\_recipe(name): recipe = recipes.get(name) if recipe: print(f'Recipe: {name}') print('Ingredients:') for ingredient in recipe['ingredients']: print(f' - {ingredient}') print('Instructions:') print(recipe['instructions']) else: print('Recipe not found.') def search\_recipes(ingredient): found\_recipes = [name for name, recipe in recipes.items() if ingredient in recipe['ingredients']] if found\_recipes: print('Recipes containing', ingredient, ':') for name in found\_recipes: print(f' - {name}') else: print('No recipes found containing', ingredient) while True: action = input('Choose an action: add, view, search, or quit: ').lower() if action == 'add': name = input('Enter recipe name: ') ingredients = input('Enter ingredients (comma-separated): ').split(',') instructions = input('Enter instructions: ') add\_recipe(name, ingredients, instructions) elif action == 'view': name = input('Enter recipe name to view: ') view\_recipe(name) elif action == 'search': ingredient = input('Enter ingredient to search: ') search\_recipes(ingredient) elif action == 'quit': break else: print('Invalid action.')

**Hint:** Store recipes in a dictionary and provide methods to add, view, and search recipes based on ingredients.

**Riddle:** I help you organize your favorite recipes and find new ones based on ingredients you have. What am I?

**Short Story:** Lily loved cooking and wanted to keep track of her favorite recipes. By building a recipe book application, she could easily store and search for recipes based on ingredients.

## Project 35: Alarm Clock

### Question:

Develop an alarm clock application that allows users to set alarms and get notified when the time arrives.

**Code:**

import time from datetime import datetime def set\_alarm(alarm\_time): print(f'Alarm set for {alarm\_time}') while True: current\_time = datetime.now().strftime('%H:%M') if current\_time == alarm\_time: print('Alarm ringing!') break time.sleep(30) alarm\_time = input('Enter alarm time (HH:MM): ') set\_alarm(alarm\_time)

**Hint:** Use the `datetime` module to check the current time and compare it with the alarm time to trigger notifications.

**Riddle:** I wake you up at the exact time you set, helping you start your day on schedule. What am I?

**Short Story:** Mark wanted to create an alarm clock application to wake him up at specific times. By developing this app, he could ensure he never missed an important meeting or event.

## Project 36: Weather Dashboard

### Question:

Build a weather dashboard that displays current weather information for a given city using an API.

**Code:**

import requests def get\_weather(city): api\_key = 'your\_api\_key\_here' url = f'http://api.openweathermap.org/data/2.5/weather?q={city}&appid={api\_key}' response = requests.get(url) data = response.json() if data['cod'] == 200: weather = data['weather'][0]['description'] temp = data['main']['temp'] - 273.15 # Convert from Kelvin to Celsius print(f'Weather in {city}: {weather}') print(f'Temperature: {temp:.2f}°C') else: print('City not found.') city = input('Enter city name: ') get\_weather(city)

**Hint:** Use the OpenWeatherMap API to fetch and display current weather data for a specified city.

**Riddle:** I show you the current weather and temperature for any city, helping you plan your day accordingly. What am I?

**Short Story:** Emily wanted to know the weather before heading out. By building a weather dashboard, she could quickly check the weather conditions for any city and plan her day.

## Project 37: Personal Budget Tracker

### Question:

Create a personal budget tracker that allows users to record their income and expenses and view their current balance.

**Code:**

budget = {'income': 0, 'expenses': 0} def record\_income(amount): budget['income'] += amount print(f'Income recorded: {amount}') def record\_expense(amount): budget['expenses'] += amount print(f'Expense recorded: {amount}') def view\_balance(): balance = budget['income'] - budget['expenses'] print(f'Current balance: {balance}') while True: action = input('Choose an action: income, expense, balance, or quit: ').lower() if action == 'income': amount = float(input('Enter income amount: ')) record\_income(amount) elif action == 'expense': amount = float(input('Enter expense amount: ')) record\_expense(amount) elif action == 'balance': view\_balance() elif action == 'quit': break else: print('Invalid action.')

**Hint:** Maintain a dictionary to keep track of income and expenses and calculate the current balance based on user inputs.

**Riddle:** I help you keep track of your finances by recording your income and expenses, showing your current balance. What am I?

**Short Story:** Sarah wanted to manage her personal finances better. By creating a budget tracker, she could keep track of her income and expenses, helping her stay on top of her financial situation.

## Project 38: Habit Tracker

### Question:

Develop a habit tracker application that allows users to record their daily habits and view their progress over time.

**Code:**

import datetime habits = {} def add\_habit(habit): habits[habit] = [] def record\_habit(habit, date): if habit in habits: habits[habit].append(date) print(f'Habit "{habit}" recorded for {date}.') else: print('Habit not found.') def view\_progress(): today = datetime.date.today().strftime('%Y-%m-%d') print(f'Progress for {today}:') for habit, dates in habits.items(): print(f'{habit}: {len(dates)} times') while True: action = input('Choose an action: add, record, view, or quit: ').lower() if action == 'add': habit = input('Enter habit to add: ') add\_habit(habit) elif action == 'record': habit = input('Enter habit to record: ') date = input('Enter date (YYYY-MM-DD): ') record\_habit(habit, date) elif action == 'view': view\_progress() elif action == 'quit': break else: print('Invalid action.')

**Hint:** Use a dictionary to track habits and the dates they were performed. Implement functions to add, record, and view progress on habits.

**Riddle:** I help you keep track of your daily habits, letting you record and review your progress over time. What am I?

**Short Story:** Emma wanted to build better habits and monitor her progress. By developing a habit tracker, she could easily keep track of her daily habits and stay motivated.

## Project 39: Simple Chatbot

### Question:

Create a simple chatbot that can respond to basic questions and have a conversation with the user.

**Code:**

def chatbot\_response(user\_input): responses = { 'hi': 'Hello! How can I help you today?', 'how are you': 'I am just a bot, but I am doing well. How about you?', 'bye': 'Goodbye! Have a great day!', } return responses.get(user\_input.lower(), 'I am not sure how to respond to that.') while True: user\_input = input('You: ') response = chatbot\_response(user\_input) print(f'Bot: {response}') if user\_input.lower() == 'bye': break

**Hint:** Use a dictionary to store predefined responses and match user input to provide appropriate replies.

**Riddle:** I am a digital assistant that responds to your messages with pre-set answers, helping you with basic queries. What am I?

**Short Story:** Kevin wanted a way to interact with his computer using natural language. By creating a simple chatbot, he could have basic conversations and get responses to common questions.

## Project 40: Markdown to HTML Converter

### Question:

Develop a program that converts Markdown text into HTML format.

**Code:**

import markdown def convert\_markdown\_to\_html(markdown\_text): return markdown.markdown(markdown\_text) markdown\_text = '''# Sample Markdown \*\*Bold Text\*\* \*Italic Text\* - List item 1 - List item 2''' html = convert\_markdown\_to\_html(markdown\_text) print(html)

**Hint:** Use the `markdown` library to convert Markdown text to HTML.

**Riddle:** I transform text written in Markdown into structured HTML format, making it suitable for web pages. What am I?

**Short Story:** Laura wanted a tool to convert her Markdown notes into HTML for her blog. By creating this converter, she could easily publish her formatted content online.

## Project 41: Password Generator

### Question:

Write a program to generate strong, random passwords of a given length.

**Code:**

import random import string def generate\_password(length): characters = string.ascii\_letters + string.digits + string.punctuation return ''.join(random.choice(characters) for \_ in range(length)) length = int(input('Enter the desired length of the password: ')) print(f'Generated Password: {generate\_password(length)}')

**Hint:** Use the `random` and `string` modules to generate a password with a mix of letters, digits, and punctuation.

**Riddle:** I create strong, random strings of characters to help you secure your accounts. What am I?

**Short Story:** Tom needed a secure password for his online accounts. By building a password generator, he could easily create strong passwords to enhance his security.

## Project 42: Contact Book

### Question:

Build a contact book application that allows users to add, view, and search for contacts.

**Code:**

contacts = {} def add\_contact(name, phone): contacts[name] = phone print(f'Contact "{name}" added.') def view\_contacts(): for name, phone in contacts.items(): print(f'Name: {name}, Phone: {phone}') def search\_contact(name): phone = contacts.get(name) if phone: print(f'Contact found: {name} - {phone}') else: print('Contact not found.') while True: action = input('Choose an action: add, view, search, or quit: ').lower() if action == 'add': name = input('Enter contact name: ') phone = input('Enter phone number: ') add\_contact(name, phone) elif action == 'view': view\_contacts() elif action == 'search': name = input('Enter contact name to search: ') search\_contact(name) elif action == 'quit': break else: print('Invalid action.')

**Hint:** Store contacts in a dictionary and provide methods to add, view, and search for contacts.

**Riddle:** I keep track of your friends and family’s phone numbers, allowing you to add, view, and search for contacts. What am I?

**Short Story:** Sarah wanted a way to manage her contacts. By creating a contact book application, she could easily add, view, and search for contact information.

## Project 43: Expense Tracker

### Question:

Design an expense tracker that allows users to record their daily expenses and view a summary of their spending.

**Code:**

expenses = [] def add\_expense(amount, description): expenses.append({'amount': amount, 'description': description}) print(f'Expense of {amount} added: {description}') def view\_summary(): total = sum(expense['amount'] for expense in expenses) print(f'Total Expenses: {total}') for expense in expenses: print(f'{expense["amount"]}: {expense["description"]}') while True: action = input('Choose an action: add, summary, or quit: ').lower() if action == 'add': amount = float(input('Enter expense amount: ')) description = input('Enter expense description: ') add\_expense(amount, description) elif action == 'summary': view\_summary() elif action == 'quit': break else: print('Invalid action.')

**Hint:** Use a list to store expenses and implement functions to add and view a summary of expenses.

**Riddle:** I help you manage your daily spending by recording your expenses and summarizing your total spending. What am I?

**Short Story:** Lily wanted to keep track of her expenses and manage her budget better. By creating an expense tracker, she could record and review her daily spending efficiently.

## Project 44: Online Quiz

### Question:

Create an online quiz application that presents users with multiple-choice questions and calculates their score based on their answers.

**Code:**

questions = { 'What is the capital of France?': ['Paris', 'London', 'Berlin', 'Rome'], 'Which planet is known as the Red Planet?': ['Mars', 'Venus', 'Earth', 'Jupiter'] } def ask\_questions(): score = 0 for question, options in questions.items(): print(question) for i, option in enumerate(options): print(f'{i + 1}. {option}') answer = int(input('Enter your answer (1-4): ')) - 1 if options[answer] == 'Paris' or options[answer] == 'Mars': # Correct answers score += 1 print(f'Your score: {score}/{len(questions)}') ask\_questions()

**Hint:** Use a dictionary to store questions and multiple-choice options, and calculate the score based on user answers.

**Riddle:** I present you with questions and options, then calculate your score based on your answers. What am I?

**Short Story:** John wanted to test his knowledge and have some fun. By creating an online quiz, he could answer questions and see how well he did.

## Project 45: To-Do List Manager

### Question:

Develop a to-do list manager that allows users to add, view, and mark tasks as completed.

**Code:**

tasks = {} def add\_task(task): tasks[task] = False print(f'Task "{task}" added.') def mark\_completed(task): if task in tasks: tasks[task] = True print(f'Task "{task}" marked as completed.') else: print('Task not found.') def view\_tasks(): for task, completed in tasks.items(): status = 'Completed' if completed else 'Pending' print(f'{task}: {status}') while True: action = input('Choose an action: add, complete, view, or quit: ').lower() if action == 'add': task = input('Enter task description: ') add\_task(task) elif action == 'complete': task = input('Enter task to mark as completed: ') mark\_completed(task) elif action == 'view': view\_tasks() elif action == 'quit': break else: print('Invalid action.')

**Hint:** Use a dictionary to manage tasks and their completion status, with functions to add, complete, and view tasks.

**Riddle:** I help you keep track of your tasks, allowing you to add, complete, and review them. What am I?

**Short Story:** Mark wanted a way to stay organized and keep track of his tasks. By building a to-do list manager, he could efficiently manage his daily activities.

## Project 46: URL Shortener

### Question:

Create a URL shortener that converts long URLs into shorter, more manageable ones.

**Code:**

import hashlib urls = {} def shorten\_url(long\_url): short\_hash = hashlib.md5(long\_url.encode()).hexdigest()[:6] short\_url = f'http://short.url/{short\_hash}' urls[short\_url] = long\_url return short\_url def get\_long\_url(short\_url): return urls.get(short\_url, 'URL not found.') long\_url = input('Enter long URL to shorten: ') short\_url = shorten\_url(long\_url) print(f'Shortened URL: {short\_url}')

**Hint:** Use hashing to create unique short URLs and store the mapping between long and short URLs.

**Riddle:** I transform long, unwieldy URLs into short, easy-to-share links. What am I?

**Short Story:** Alex needed a way to shorten long URLs for sharing on social media. By creating a URL shortener, he could generate compact links for easier distribution.

## Project 47: Currency Converter

### Question:

Develop a currency converter that allows users to convert amounts from one currency to another.

**Code:**

exchange\_rates = { 'USD': 1.0, 'EUR': 0.85, 'JPY': 110.0, } def convert\_currency(amount, from\_currency, to\_currency): if from\_currency in exchange\_rates and to\_currency in exchange\_rates: converted\_amount = amount \* (exchange\_rates[to\_currency] / exchange\_rates[from\_currency]) return converted\_amount else: return 'Currency not supported.' amount = float(input('Enter amount to convert: ')) from\_currency = input('Enter from currency (USD, EUR, JPY): ') to\_currency = input('Enter to currency (USD, EUR, JPY): ') result = convert\_currency(amount, from\_currency, to\_currency) print(f'Converted Amount: {result}')

**Hint:** Use a dictionary to store exchange rates and implement a function to convert between currencies based on user input.

**Riddle:** I help you convert amounts between different currencies, making it easier to handle foreign exchange. What am I?

**Short Story:** Maria needed to convert currencies for her travels. By creating a currency converter, she could easily convert amounts and manage her money in different countries.

## Project 48: Calendar Event Organizer

### Question:

Create a calendar event organizer that allows users to add, view, and manage their events.

**Code:**

events = {} def add\_event(date, event): if date in events: events[date].append(event) else: events[date] = [event] print(f'Event "{event}" added on {date}.') def view\_events(date): if date in events: for event in events[date]: print(f'Event on {date}: {event}') else: print('No events on this date.') while True: action = input('Choose an action: add, view, or quit: ').lower() if action == 'add': date = input('Enter event date (YYYY-MM-DD): ') event = input('Enter event description: ') add\_event(date, event) elif action == 'view': date = input('Enter date to view events (YYYY-MM-DD): ') view\_events(date) elif action == 'quit': break else: print('Invalid action.')

**Hint:** Use a dictionary to store events by date and provide functions to add and view events.

**Riddle:** I help you organize and keep track of your calendar events, letting you add and review events for specific dates. What am I?

**Short Story:** Jane needed a way to manage her appointments and events. By creating a calendar event organizer, she could easily add, view, and track her important dates.

## Project 49: Simple Email Sender

### Question:

Write a program that can send an email with a subject and body to a specified recipient.

**Code:**

import smtplib from email.mime.text import MIMEText from email.mime.multipart import MIMEMultipart def send\_email(subject, body, to\_email): from\_email = 'your\_email@example.com' password = 'your\_password' msg = MIMEMultipart() msg['From'] = from\_email msg['To'] = to\_email msg['Subject'] = subject msg.attach(MIMEText(body, 'plain')) with smtplib.SMTP('smtp.example.com', 587) as server: server.starttls() server.login(from\_email, password) server.send\_message(msg) print('Email sent successfully.') subject = input('Enter email subject: ') body = input('Enter email body: ') to\_email = input('Enter recipient email address: ') send\_email(subject, body, to\_email)

**Hint:** Use the `smtplib` library to send emails and the `email` library to create the email content.

**Riddle:** I send emails with a subject and body to a recipient's address, using your email account. What am I?

**Short Story:** Alice needed to send emails programmatically. By building a simple email sender, she could automate sending messages with a subject and content to any recipient.

## Project 50: Basic Calculator

### Question:

Develop a basic calculator that performs addition, subtraction, multiplication, and division.

**Code:**

def calculator(): while True: operation = input('Enter operation (+, -, \*, /) or "quit" to exit: ').strip() if operation == 'quit': break if operation in ('+', '-', '\*', '/'): try: num1 = float(input('Enter first number: ')) num2 = float(input('Enter second number: ')) if operation == '+': result = num1 + num2 elif operation == '-': result = num1 - num2 elif operation == '\*': result = num1 \* num2 elif operation == '/': result = num1 / num2 print(f'Result: {result}') except ValueError: print('Invalid input. Please enter numbers only.') except ZeroDivisionError: print('Error: Division by zero is not allowed.') else: print('Invalid operation.') calculator()

**Hint:** Implement basic arithmetic operations and handle errors such as invalid input and division by zero.

**Riddle:** I help you perform basic arithmetic operations like addition, subtraction, multiplication, and division. What am I?

**Short Story:** John wanted a simple way to perform calculations. By creating a basic calculator, he could easily add, subtract, multiply, and divide numbers on the go.

## Project 51: Binary Search Algorithm

### Question:

Implement a binary search algorithm to find an element in a sorted list.

**Code:**

def binary\_search(arr, target): left, right = 0, len(arr) - 1 while left <= right: mid = (left + right) // 2 if arr[mid] == target: return mid elif arr[mid] < target: left = mid + 1 else: right = mid - 1 return -1 arr = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19] target = int(input('Enter number to search: ')) result = binary\_search(arr, target) print(f'Element found at index: {result}' if result != -1 else 'Element not found.')

**Hint:** The binary search algorithm divides the search interval in half repeatedly until the target value is found.

**Riddle:** I find elements in a sorted list by repeatedly dividing the search space in half. What am I?

**Short Story:** Laura needed to search for numbers in a large list quickly. By implementing a binary search algorithm, she could efficiently locate the elements she was interested in.

## Project 52: Simple Password Generator

### Question:

Create a simple password generator that generates a random password with a specified length.

**Code:**

import random import string def generate\_password(length): characters = string.ascii\_letters + string.digits + string.punctuation password = ''.join(random.choice(characters) for i in range(length)) return password length = int(input('Enter password length: ')) print(f'Generated Password: {generate\_password(length)}')

**Hint:** Use the `random` and `string` libraries to generate a password with a mix of letters, digits, and special characters.

**Riddle:** I create strong, random passwords with letters, digits, and symbols to keep your accounts secure. What am I?

**Short Story:** Sarah wanted to generate strong passwords for her accounts. By creating a password generator, she could easily create secure passwords for her online profiles.

## Project 53: Sudoku Solver

### Question:

Write a Sudoku solver that fills in the empty cells of a Sudoku puzzle.

**Code:**

def print\_board(board): for row in board: print(' '.join(str(num) if num != 0 else '.' for num in row)) def is\_valid(board, row, col, num): for i in range(9): if board[row][i] == num or board[i][col] == num: return False start\_row, start\_col = 3 \* (row // 3), 3 \* (col // 3) for i in range(start\_row, start\_row + 3): for j in range(start\_col, start\_col + 3): if board[i][j] == num: return False return True def solve\_sudoku(board): empty = find\_empty\_location(board) if not empty: return True row, col = empty for num in range(1, 10): if is\_valid(board, row, col, num): board[row][col] = num if solve\_sudoku(board): return True board[row][col] = 0 return False def find\_empty\_location(board): for i in range(9): for j in range(9): if board[i][j] == 0: return (i, j) return None # Example Sudoku Board board = [ [5, 3, 0, 0, 7, 0, 0, 0, 0], [6, 0, 0, 1, 9, 5, 0, 0, 0], [0, 9, 8, 0, 0, 0, 0, 6, 0], [8, 0, 0, 0, 6, 0, 0, 0, 3], [4, 0, 0, 8, 0, 3, 0, 0, 1], [7, 0, 0, 0, 2, 0, 0, 0, 6], [0, 6, 0, 0, 0, 0, 2, 8, 0], [0, 0, 0, 4, 1, 9, 0, 0, 5], [0, 0, 0, 0, 8, 0, 0, 7, 9] ] if solve\_sudoku(board): print\_board(board) else: print('No solution exists.')

**Hint:** Use backtracking to solve the Sudoku puzzle by trying possible numbers in each empty cell and checking if they are valid.

**Riddle:** I solve puzzles by filling in the empty cells with numbers, ensuring no repeats in rows, columns, or sub-grids. What am I?

**Short Story:** Tom enjoyed solving Sudoku puzzles but wanted an automated way to solve them. By creating a Sudoku solver, he could easily find solutions to even the most challenging puzzles.

## Project 54: Markdown to HTML Converter

### Question:

Create a simple converter that converts Markdown text to HTML.

**Code:**

import markdown def convert\_markdown\_to\_html(markdown\_text): html = markdown.markdown(markdown\_text) return html markdown\_text = ''' # Heading 1 ## Heading 2 \*\*Bold Text\*\* \*Italic Text\* - List item 1 - List item 2 ''' print('Converted HTML:') print(convert\_markdown\_to\_html(markdown\_text))

**Hint:** Use the `markdown` library to convert Markdown text to HTML format.

**Riddle:** I transform Markdown text into HTML format, enabling formatted text and structure. What am I?

**Short Story:** Emily wanted to publish her Markdown notes as HTML. By creating a Markdown to HTML converter, she could easily convert her formatted text into web-ready HTML.

## Project 55: Quiz Application

### Question:

Build a simple quiz application that asks multiple-choice questions and keeps track of the score.

**Code:**

questions = { "What is the capital of France?": ["a) Berlin", "b) Madrid", "c) Paris", "d) Rome"], "What is 2 + 2?": ["a) 3", "b) 4", "c) 5", "d) 6"] } def ask\_question(question, options): print(question) for option in options: print(option) answer = input('Enter your answer (e.g., a, b, c, d): ') return answer def run\_quiz(questions): score = 0 for question, options in questions.items(): answer = ask\_question(question, options) if answer == 'c': # assuming 'c' is the correct answer for simplicity score += 1 print(f'Your score: {score}/{len(questions)}') run\_quiz(questions)

**Hint:** Use a dictionary to store questions and multiple-choice options, then check the user's answer to keep track of their score.

**Riddle:** I test your knowledge by asking questions and keeping track of your score. What am I?

**Short Story:** Alex wanted to create a fun quiz for his friends. By building a quiz application, he could challenge them with questions and see who scored the highest.

## Project 56: Weather Information Fetcher

### Question:

Create a program that fetches and displays the current weather information for a given city using a weather API.

**Code:**

import requests def get\_weather(city): api\_key = 'your\_api\_key' # replace with your API key url = f'http://api.openweathermap.org/data/2.5/weather?q={city}&appid={api\_key}&units=metric' response = requests.get(url) data = response.json() if data['cod'] == 200: temp = data['main']['temp'] description = data['weather'][0]['description'] return f'Temperature: {temp}°C\nDescription: {description}' else: return 'City not found.' city = input('Enter city name: ') print(get\_weather(city))

**Hint:** Use the `requests` library to make an API call and fetch weather information. Remember to handle errors for invalid city names.

**Riddle:** I provide you with the current temperature and weather description for any city you choose. What am I?

**Short Story:** Maria wanted to check the weather before her trip. By creating a weather information fetcher, she could easily get current weather updates for any city.

## Project 57: Expense Tracker

### Question:

Develop an expense tracker application that records expenses and displays a summary of total expenses.

**Code:**

def add\_expense(expenses, amount, description): expenses.append({'amount': amount, 'description': description}) def display\_expenses(expenses): total = sum(expense['amount'] for expense in expenses) print(f'Total Expenses: {total}') for expense in expenses: print(f"Description: {expense['description']}, Amount: {expense['amount']}") expenses = [] while True: amount = float(input('Enter expense amount: ')) description = input('Enter expense description: ') add\_expense(expenses, amount, description) more = input('Add more expenses? (yes/no): ') if more.lower() != 'yes': break display\_expenses(expenses)

**Hint:** Store expenses in a list of dictionaries and calculate the total expenses using a loop.

**Riddle:** I help you keep track of your spending and show you how much you've spent in total. What am I?

**Short Story:** Lisa wanted to manage her finances better. By building an expense tracker, she could easily record and review her spending habits.

## Project 58: Simple Chatbot

### Question:

Build a simple chatbot that responds to user input with predefined responses.

**Code:**

def chatbot\_response(user\_input): responses = { 'hello': 'Hi there!', 'how are you?': 'I am good, thank you!', 'bye': 'Goodbye!' } return responses.get(user\_input.lower(), 'Sorry, I did not understand that.') def chatbot(): print('Chatbot: Hello! Type "bye" to exit.') while True: user\_input = input('You: ') response = chatbot\_response(user\_input) print(f'Chatbot: {response}') if user\_input.lower() == 'bye': break chatbot()

**Hint:** Use a dictionary to map user inputs to predefined responses. Ensure the chatbot can handle various inputs.

**Riddle:** I converse with you using fixed responses and can say "hello" and "goodbye". What am I?

**Short Story:** Dave wanted a fun project to interact with users. By creating a simple chatbot, he could have engaging conversations with his friends.

## Project 59: Email Sender

### Question:

Create a program to send an email using the SMTP protocol.

**Code:**

import smtplib from email.mime.text import MIMEText from email.mime.multipart import MIMEMultipart def send\_email(subject, body, to\_email): from\_email = 'your\_email@example.com' password = 'your\_password' msg = MIMEMultipart() msg['From'] = from\_email msg['To'] = to\_email msg['Subject'] = subject msg.attach(MIMEText(body, 'plain')) with smtplib.SMTP('smtp.example.com', 587) as server: server.starttls() server.login(from\_email, password) server.sendmail(from\_email, to\_email, msg.as\_string()) subject = 'Test Email' body = 'This is a test email sent from Python.' to\_email = 'recipient@example.com' send\_email(subject, body, to\_email)

**Hint:** Use the `smtplib` and `email` libraries to send emails. Replace placeholders with your actual email details.

**Riddle:** I help you send emails programmatically using Python. What am I?

**Short Story:** Alice needed to automate sending emails for her work. By creating an email sender program, she could easily dispatch emails with minimal effort.

## Project 60: URL Shortener

### Question:

Build a simple URL shortener that converts long URLs into short, shareable links.

**Code:**

import hashlib def shorten\_url(url): return hashlib.md5(url.encode()).hexdigest()[:6] long\_url = input('Enter a URL to shorten: ') short\_url = shorten\_url(long\_url) print(f'Short URL: {short\_url}')

**Hint:** Use hashing to generate a short version of the URL. Ensure that the shortened URL is unique.

**Riddle:** I transform long URLs into short, easy-to-share links. What am I?

**Short Story:** Michael wanted to share links easily on social media. By creating a URL shortener, he could convert long URLs into concise, shareable links.

## Project 61: Personal Diary

### Question:

Create a personal diary application where users can write, read, and delete diary entries.

**Code:**

diary = [] def add\_entry(entry): diary.append(entry) def view\_entries(): for i, entry in enumerate(diary, start=1): print(f'Entry {i}: {entry}') def delete\_entry(index): if 0 <= index < len(diary): diary.pop(index) else: print('Invalid index.') while True: action = input('What would you like to do? (add/view/delete/quit): ') if action == 'add': entry = input('Write your diary entry: ') add\_entry(entry) elif action == 'view': view\_entries() elif action == 'delete': index = int(input('Enter the index of the entry to delete: ')) - 1 delete\_entry(index) elif action == 'quit': break else: print('Invalid action.')

**Hint:** Use a list to store diary entries and provide options to add, view, or delete entries.

**Riddle:** I keep your thoughts and memories, allowing you to write, read, and delete as you wish. What am I?

**Short Story:** Emma wanted a private space to record her thoughts. By creating a personal diary application, she could easily manage her entries and reflect on her experiences.

## Project 62: Random Password Generator

### Question:

Develop a program to generate a random, secure password with a specified length.

**Code:**

import random import string def generate\_password(length): characters = string.ascii\_letters + string.digits + string.punctuation return ''.join(random.choice(characters) for i in range(length)) length = int(input('Enter password length: ')) print(f'Generated Password: {generate\_password(length)}')

**Hint:** Use the `random` module and `string` constants to create a secure password with a mix of letters, digits, and punctuation.

**Riddle:** I create a secure, random combination of characters to protect your accounts. What am I?

**Short Story:** Jake wanted to ensure his online accounts were secure. By developing a random password generator, he could create strong passwords to keep his information safe.

## Project 63: Unit Converter

### Question:

Build a unit converter that can convert between different units of length, weight, and temperature.

**Code:**

def convert\_length(value, unit\_from, unit\_to): conversion\_factors = { 'm\_to\_cm': 100, 'cm\_to\_m': 0.01, 'km\_to\_m': 1000, 'm\_to\_km': 0.001 } return value \* conversion\_factors.get(f'{unit\_from}\_to\_{unit\_to}', 1) def convert\_weight(value, unit\_from, unit\_to): conversion\_factors = { 'kg\_to\_g': 1000, 'g\_to\_kg': 0.001, 'lb\_to\_kg': 0.453592, 'kg\_to\_lb': 2.20462 } return value \* conversion\_factors.get(f'{unit\_from}\_to\_{unit\_to}', 1) def convert\_temperature(value, unit\_from, unit\_to): if unit\_from == 'C' and unit\_to == 'F': return (value \* 9/5) + 32 elif unit\_from == 'F' and unit\_to == 'C': return (value - 32) \* 5/9 elif unit\_from == 'C' and unit\_to == 'K': return value + 273.15 elif unit\_from == 'K' and unit\_to == 'C': return value - 273.15 elif unit\_from == 'F' and unit\_to == 'K': return (value - 32) \* 5/9 + 273.15 elif unit\_from == 'K' and unit\_to == 'F': return (value - 273.15) \* 9/5 + 32 else: return value value = float(input('Enter value to convert: ')) unit\_from = input('Enter unit from (length: m, cm, km; weight: kg, g, lb; temperature: C, F, K): ') unit\_to = input('Enter unit to (length: m, cm, km; weight: kg, g, lb; temperature: C, F, K): ') if unit\_from in ['m', 'cm', 'km'] and unit\_to in ['m', 'cm', 'km']: print(f'Converted value: {convert\_length(value, unit\_from, unit\_to)}') elif unit\_from in ['kg', 'g', 'lb'] and unit\_to in ['kg', 'g', 'lb']: print(f'Converted value: {convert\_weight(value, unit\_from, unit\_to)}') elif unit\_from in ['C', 'F', 'K'] and unit\_to in ['C', 'F', 'K']: print(f'Converted value: {convert\_temperature(value, unit\_from, unit\_to)}') else: print('Invalid units.')

**Hint:** Use dictionaries to define conversion factors for different units and provide appropriate formulas for temperature conversion.

**Riddle:** I transform measurements from one unit to another, be it length, weight, or temperature. What am I?

**Short Story:** Sarah needed a tool to convert various units for her research. By creating a unit converter, she could easily switch between different measurements in her projects.

## Project 64: Tic-Tac-Toe Game

### Question:

Develop a simple text-based Tic-Tac-Toe game where two players can play against each other.

**Code:**

def print\_board(board): for row in board: print(' | '.join(row)) print('-' \* 5) def check\_winner(board, player): # Check rows, columns, and diagonals for i in range(3): if all([cell == player for cell in board[i]]) or all([board[j][i] == player for j in range(3)]): return True if board[0][0] == board[1][1] == board[2][2] == player or board[0][2] == board[1][1] == board[2][0] == player: return True return False def tic\_tac\_toe(): board = [[' ' for \_ in range(3)] for \_ in range(3)] players = ['X', 'O'] turn = 0 while True: print\_board(board) player = players[turn % 2] row = int(input(f'Player {player}, enter row (0-2): ')) col = int(input(f'Player {player}, enter column (0-2): ')) if board[row][col] == ' ': board[row][col] = player if check\_winner(board, player): print\_board(board) print(f'Player {player} wins!') break elif all(cell != ' ' for row in board for cell in row): print\_board(board) print('It\'s a draw!') break turn += 1 else: print('Cell already occupied. Try again.') tic\_tac\_toe()

**Hint:** Create a board as a list of lists and implement functions to check for a winner or a draw.

**Riddle:** I am a game where two players take turns marking spaces on a 3x3 grid. What am I?

**Short Story:** Tom and Jerry wanted to play a game during their lunch break. They created a simple Tic-Tac-Toe game to enjoy their time together and challenge each other.

## Project 65: Email Sender

### Question:

Create a Python script to send an email with an attachment using the `smtplib` and `email` libraries.

**Code:**

import smtplib from email.mime.multipart import MIMEMultipart from email.mime.text import MIMEText from email.mime.base import MIMEBase from email import encoders def send\_email(subject, body, to\_email, attachment\_path=None): from\_email = 'your\_email@example.com' password = 'your\_password' msg = MIMEMultipart() msg['From'] = from\_email msg['To'] = to\_email msg['Subject'] = subject msg.attach(MIMEText(body, 'plain')) if attachment\_path: attachment = open(attachment\_path, 'rb') part = MIMEBase('application', 'octet-stream') part.set\_payload(attachment.read()) encoders.encode\_base64(part) part.add\_header('Content-Disposition', f'attachment; filename= {attachment\_path}') msg.attach(part) with smtplib.SMTP('smtp.example.com', 587) as server: server.starttls() server.login(from\_email, password) server.send\_message(msg) subject = 'Test Email' body = 'This is a test email with an attachment.' to\_email = 'recipient@example.com' attachment\_path = 'path/to/attachment.txt' send\_email(subject, body, to\_email, attachment\_path)

**Hint:** Use `smtplib` to send the email and `email` to construct the email message and attachments.

**Riddle:** I allow you to send messages and files over the internet. What am I?

**Short Story:** Jessica needed to send an important document to her colleague. She developed a script to send emails with attachments efficiently, saving her time and effort.

## Project 66: URL Shortener

### Question:

Create a URL shortener that converts long URLs into shorter ones and keeps track of the original URL.

**Code:**

import random import string url\_mapping = {} def shorten\_url(original\_url): short\_url = ''.join(random.choices(string.ascii\_letters + string.digits, k=6)) url\_mapping[short\_url] = original\_url return short\_url def get\_original\_url(short\_url): return url\_mapping.get(short\_url, 'URL not found.') original\_url = input('Enter the URL to shorten: ') short\_url = shorten\_url(original\_url) print(f'Short URL: {short\_url}') short\_url\_input = input('Enter short URL to retrieve original URL: ') print(f'Original URL: {get\_original\_url(short\_url\_input)}')

**Hint:** Use a dictionary to map short URLs to original URLs and generate random strings for short URLs.

**Riddle:** I turn long URLs into short, memorable links. What am I?

**Short Story:** Alex wanted a way to shorten URLs for easier sharing. By creating a URL shortener, he could quickly convert long links into short, easy-to-remember ones.

## Project 67: Expense Tracker

### Question:

Create a simple expense tracker that allows users to input expenses and view a summary of their spending.

**Code:**

expenses = [] def add\_expense(amount, description): expenses.append({'amount': amount, 'description': description}) def view\_expenses(): total = sum(expense['amount'] for expense in expenses) print('Expense Summary:') for expense in expenses: print(f"{expense['description']}: ${expense['amount']:.2f}") print(f'Total: ${total:.2f}') while True: action = input('What would you like to do? (add/view/quit): ') if action == 'add': amount = float(input('Enter expense amount: ')) description = input('Enter expense description: ') add\_expense(amount, description) elif action == 'view': view\_expenses() elif action == 'quit': break else: print('Invalid action.')

**Hint:** Use a list to store expense records and provide options to add or view them.

**Riddle:** I help you track and summarize your spending to manage your budget. What am I?

**Short Story:** Laura wanted to keep track of her expenses more effectively. She created an expense tracker to manage her spending and see how much she was saving each month.

## Project 68: Currency Converter

### Question:

Build a currency converter that converts an amount from one currency to another using predefined conversion rates.

**Code:**

conversion\_rates = { 'USD\_to\_EUR': 0.85, 'EUR\_to\_USD': 1.18, 'USD\_to\_GBP': 0.76, 'GBP\_to\_USD': 1.32 } def convert\_currency(amount, from\_currency, to\_currency): key = f'{from\_currency}\_to\_{to\_currency}' rate = conversion\_rates.get(key) if rate: return amount \* rate else: return 'Conversion rate not available.' amount = float(input('Enter amount: ')) from\_currency = input('Enter currency to convert from (USD, EUR, GBP): ') to\_currency = input('Enter currency to convert to (USD, EUR, GBP): ') print(f'Converted amount: {convert\_currency(amount, from\_currency, to\_currency)}')

**Hint:** Use a dictionary to define conversion rates between currencies and perform the conversion based on user input.

**Riddle:** I convert money from one currency to another, helping you understand its value in different countries. What am I?

**Short Story:** David often traveled abroad and needed a way to convert currencies easily. He built a currency converter to quickly translate amounts between different currencies.

## Project 69: Stopwatch

### Question:

Create a simple stopwatch that can start, stop, and reset the time.

**Code:**

import time def stopwatch(): start\_time = None elapsed\_time = 0 while True: action = input('Enter action (start/stop/reset/quit): ') if action == 'start': start\_time = time.time() print('Stopwatch started.') elif action == 'stop': if start\_time: elapsed\_time += time.time() - start\_time print(f'Stopwatch stopped. Time elapsed: {elapsed\_time:.2f} seconds.') start\_time = None else: print('Stopwatch is not running.') elif action == 'reset': start\_time = None elapsed\_time = 0 print('Stopwatch reset.') elif action == 'quit': break else: print('Invalid action.') stopwatch()

**Hint:** Use the `time` module to measure elapsed time and provide options to start, stop, or reset the stopwatch.

**Riddle:** I measure time intervals and can be started, stopped, or reset at your command. What am I?

**Short Story:** Mike wanted to keep track of his workout durations precisely. He built a stopwatch to monitor his exercise times and improve his performance.

## Project 70: Alarm Clock

### Question:

Create an alarm clock application that allows users to set an alarm for a specific time.

**Code:**

import time from datetime import datetime def set\_alarm(alarm\_time): while True: now = datetime.now().strftime('%H:%M:%S') if now == alarm\_time: print('Alarm ringing!') break time.sleep(1) alarm\_time = input('Enter alarm time in HH:MM:SS format: ') set\_alarm(alarm\_time)

**Hint:** Use the `datetime` module to get the current time and compare it with the alarm time to trigger the alarm.

**Riddle:** I ring at a specific time to wake you up or remind you of an event. What am I?

**Short Story:** Emily needed a way to wake up on time for her early meetings. She developed an alarm clock application that would alert her at the set time.

## Project 71: To-Do List

### Question:

Create a to-do list application where users can add, view, and delete tasks.

**Code:**

tasks = [] def add\_task(task): tasks.append(task) def view\_tasks(): if tasks: print('To-Do List:') for i, task in enumerate(tasks, start=1): print(f'{i}. {task}') else: print('No tasks in the list.') def delete\_task(index): if 0 <= index < len(tasks): tasks.pop(index) else: print('Invalid index.') while True: action = input('What would you like to do? (add/view/delete/quit): ') if action == 'add': task = input('Enter task: ') add\_task(task) elif action == 'view': view\_tasks() elif action == 'delete': index = int(input('Enter the index of the task to delete: ')) - 1 delete\_task(index) elif action == 'quit': break else: print('Invalid action.')

**Hint:** Use a list to store tasks and provide options to add, view, or delete them from the list.

**Riddle:** I help you organize and manage tasks that need to be completed. What am I?

**Short Story:** John wanted a way to keep track of his daily tasks. He created a to-do list application to manage his responsibilities and stay organized.

## Project 72: Simple Calculator

### Question:

Develop a basic calculator that can perform addition, subtraction, multiplication, and division.

**Code:**

def calculator(): while True: operation = input('Enter operation (+, -, \*, /) or quit to exit: ') if operation == 'quit': break if operation in ['+', '-', '\*', '/']: num1 = float(input('Enter first number: ')) num2 = float(input('Enter second number: ')) if operation == '+': print(f'Result: {num1 + num2}') elif operation == '-': print(f'Result: {num1 - num2}') elif operation == '\*': print(f'Result: {num1 \* num2}') elif operation == '/': if num2 != 0: print(f'Result: {num1 / num2}') else: print('Error: Division by zero.') else: print('Invalid operation.') calculator()

**Hint:** Use conditional statements to perform calculations based on user input and handle division by zero errors.

**Riddle:** I perform arithmetic operations like addition and multiplication. What am I?

**Short Story:** Sarah wanted a simple tool for performing quick calculations. She built a basic calculator to help with her everyday math problems.

## Project 73: Weather App

### Question:

Create a weather application that fetches and displays the current weather for a given location using an API.

**Code:**

import requests def get\_weather(location): api\_key = 'your\_api\_key' url = f'http://api.weatherapi.com/v1/current.json?key={api\_key}&q={location}' response = requests.get(url) data = response.json() if 'error' in data: return data['error']['message'] else: temp = data['current']['temp\_c'] condition = data['current']['condition']['text'] return f'Temperature: {temp}°C, Condition: {condition}' location = input('Enter location: ') print(get\_weather(location))

**Hint:** Use the `requests` library to fetch weather data from an API and parse the JSON response.

**Riddle:** I provide current weather information for any location when asked. What am I?

**Short Story:** Lisa wanted to know the weather conditions before planning her day. She built a weather app that retrieves and displays the current weather for any location.

## Project 74: Quiz Game

### Question:

Design a quiz game where users answer multiple-choice questions and get a score at the end.

**Code:**

questions = [ {'question': 'What is the capital of France?', 'options': ['A. Paris', 'B. London', 'C. Berlin'], 'answer': 'A'}, {'question': 'What is 2 + 2?', 'options': ['A. 3', 'B. 4', 'C. 5'], 'answer': 'B'} ] def quiz\_game(): score = 0 for q in questions: print(q['question']) for option in q['options']: print(option) answer = input('Your answer (A, B, C): ').strip().upper() if answer == q['answer']: score += 1 print(f'Your score: {score}/{len(questions)}') quiz\_game()

**Hint:** Use a list of dictionaries to store questions, options, and correct answers. Iterate through the list to present questions and check answers.

**Riddle:** I ask you questions and keep track of your answers to calculate a score. What am I?

**Short Story:** James enjoyed testing his knowledge with quizzes. He developed a quiz game to challenge his friends and see who scored the highest.

## Project 75: Simple Chatbot

### Question:

Build a basic chatbot that can respond to a few predefined questions.

**Code:**

responses = { 'hi': 'Hello!', 'how are you?': 'I am good, thank you!', 'bye': 'Goodbye!' } def chatbot(): while True: user\_input = input('You: ').lower() if user\_input in responses: print('Bot:', responses[user\_input]) elif user\_input == 'quit': break else: print('Bot: I do not understand that.') chatbot()

**Hint:** Use a dictionary to store predefined responses and match user input to provide appropriate replies.

**Riddle:** I can hold simple conversations and answer specific questions based on preset responses. What am I?

**Short Story:** Sarah wanted a fun way to interact with her computer. She built a simple chatbot to have quick conversations and provide basic responses.

## Project 76: Fibonacci Sequence Generator

### Question:

Create a program that generates the Fibonacci sequence up to a given number.

**Code:**

def fibonacci(n): a, b = 0, 1 while a < n: print(a, end=' ') a, b = b, a + b print() limit = int(input('Enter the upper limit for Fibonacci sequence: ')) fibonacci(limit)

**Hint:** Use a loop to generate and print the Fibonacci sequence up to the specified limit.

**Riddle:** I produce a sequence where each number is the sum of the two preceding ones. What am I?

**Short Story:** Oliver wanted to explore mathematical sequences. He created a Fibonacci sequence generator to see how numbers in the sequence grow.

## Project 77: Palindrome Checker

### Question:

Develop a function that checks if a given string is a palindrome.

**Code:**

def is\_palindrome(s): return s == s[::-1] string = input('Enter a string: ') if is\_palindrome(string): print('The string is a palindrome.') else: print('The string is not a palindrome.')

**Hint:** Use string slicing to reverse the string and compare it with the original.

**Riddle:** I am a word or phrase that reads the same backward as forward. What am I?

**Short Story:** Emma was intrigued by palindromes and wanted to check if a string was one. She wrote a function to verify if a given string is a palindrome.

## Project 78: Prime Number Checker

### Question:

Create a program to check if a given number is a prime number.

**Code:**

def is\_prime(n): if n <= 1: return False for i in range(2, int(n \*\* 0.5) + 1): if n % i == 0: return False return True number = int(input('Enter a number: ')) if is\_prime(number): print('The number is a prime number.') else: print('The number is not a prime number.')

**Hint:** Check divisibility up to the square root of the number to determine if it is prime.

**Riddle:** I am a number greater than 1 that cannot be divided evenly by any number other than 1 and itself. What am I?

**Short Story:** Lucas was curious about prime numbers and wanted to identify them easily. He developed a prime number checker to find out if numbers were prime.

## Project 79: Word Frequency Counter

### Question:

Write a program that counts the frequency of each word in a given text.

**Code:**

from collections import Counter def word\_frequency(text): words = text.split() return Counter(words) text = input('Enter text: ') freq = word\_frequency(text) for word, count in freq.items(): print(f'{word}: {count}')

**Hint:** Use the `Counter` class from the `collections` module to count word frequencies.

**Riddle:** I tell you how many times each word appears in a text. What am I?

**Short Story:** Mia needed to analyze text data for her project. She wrote a program to count the frequency of each word in a text to understand its content better.

## Project 80: Simple To-Do List

### Question:

Create a simple to-do list application where users can add, view, and remove tasks.

**Code:**

tasks = [] def display\_tasks(): if tasks: for idx, task in enumerate(tasks, 1): print(f'{idx}. {task}') else: print('No tasks.') def add\_task(task): tasks.append(task) def remove\_task(index): if 0 < index <= len(tasks): tasks.pop(index - 1) else: print('Invalid task number.') while True: print('1. View tasks') print('2. Add task') print('3. Remove task') print('4. Exit') choice = input('Enter your choice: ') if choice == '1': display\_tasks() elif choice == '2': task = input('Enter task: ') add\_task(task) elif choice == '3': index = int(input('Enter task number to remove: ')) remove\_task(index) elif choice == '4': break else: print('Invalid choice.')

**Hint:** Use a list to store tasks and provide options to view, add, and remove tasks.

**Riddle:** I help you keep track of your tasks and manage them effectively. What am I?

**Short Story:** Alex wanted a simple way to manage his daily tasks. He developed a to-do list application to organize and keep track of his tasks easily.

## Project 81: Simple Bank Account

### Question:

Create a simple bank account simulation that allows users to deposit, withdraw, and check their balance.

**Code:**

class BankAccount: def \_\_init\_\_(self): self.balance = 0 def deposit(self, amount): self.balance += amount print(f'Deposited: ${amount}') def withdraw(self, amount): if amount <= self.balance: self.balance -= amount print(f'Withdrew: ${amount}') else: print('Insufficient funds.') def check\_balance(self): print(f'Current balance: ${self.balance}') account = BankAccount() while True: print('1. Deposit') print('2. Withdraw') print('3. Check balance') print('4. Exit') choice = input('Enter your choice: ') if choice == '1': amount = float(input('Enter amount to deposit: ')) account.deposit(amount) elif choice == '2': amount = float(input('Enter amount to withdraw: ')) account.withdraw(amount) elif choice == '3': account.check\_balance() elif choice == '4': break else: print('Invalid choice.')

**Hint:** Create a class to manage bank account operations and allow users to interact with it through a menu.

**Riddle:** I manage your money by keeping track of deposits, withdrawals, and balance. What am I?

**Short Story:** Mia wanted to manage her finances better. She created a simple bank account simulation to track her deposits and withdrawals.

## Project 82: Basic Todo List with File I/O

### Question:

Develop a to-do list application that saves and loads tasks from a file.

**Code:**

def load\_tasks(): try: with open('tasks.txt', 'r') as file: return [line.strip() for line in file] except FileNotFoundError: return [] def save\_tasks(tasks): with open('tasks.txt', 'w') as file: for task in tasks: file.write(task + '\n') tasks = load\_tasks() while True: print('1. View tasks') print('2. Add task') print('3. Remove task') print('4. Exit') choice = input('Enter your choice: ') if choice == '1': if tasks: for idx, task in enumerate(tasks, 1): print(f'{idx}. {task}') else: print('No tasks.') elif choice == '2': task = input('Enter task: ') tasks.append(task) save\_tasks(tasks) elif choice == '3': index = int(input('Enter task number to remove: ')) if 0 < index <= len(tasks): tasks.pop(index - 1) save\_tasks(tasks) else: print('Invalid task number.') elif choice == '4': break else: print('Invalid choice.')

**Hint:** Use file I/O to save and load tasks, so they persist between program runs.

**Riddle:** I help you keep track of tasks and remember them even after you close the program. What am I?

**Short Story:** Emma wanted her to-do list to be persistent. She created a to-do list application that saves tasks to a file so they are available the next time she opens the app.

## Project 83: Simple Budget Tracker

### Question:

Create a budget tracker that allows users to record income and expenses and see their current balance.

**Code:**

class BudgetTracker: def \_\_init\_\_(self): self.balance = 0 def add\_income(self, amount): self.balance += amount print(f'Added income: ${amount}') def add\_expense(self, amount): self.balance -= amount print(f'Added expense: ${amount}') def check\_balance(self): print(f'Current balance: ${self.balance}') tracker = BudgetTracker() while True: print('1. Add income') print('2. Add expense') print('3. Check balance') print('4. Exit') choice = input('Enter your choice: ') if choice == '1': amount = float(input('Enter income amount: ')) tracker.add\_income(amount) elif choice == '2': amount = float(input('Enter expense amount: ')) tracker.add\_expense(amount) elif choice == '3': tracker.check\_balance() elif choice == '4': break else: print('Invalid choice.')

**Hint:** Use a class to manage income and expenses and keep track of the balance.

**Riddle:** I help you manage your finances by tracking your income and expenses. What am I?

**Short Story:** John wanted to keep track of his spending and saving. He built a budget tracker to monitor his finances and manage his money effectively.

## Project 84: Number Guessing Game

### Question:

Create a number guessing game where the user tries to guess a randomly generated number within a certain number of attempts.

**Code:**

import random def number\_guessing\_game(): number = random.randint(1, 100) attempts = 10 while attempts > 0: guess = int(input('Guess a number between 1 and 100: ')) if guess < number: print('Too low!') elif guess > number: print('Too high!') else: print('Congratulations! You guessed the number.') return attempts -= 1 print(f'Attempts remaining: {attempts}') print('Sorry, you ran out of attempts. The number was', number) number\_guessing\_game()

**Hint:** Use the `random` module to generate a random number and provide feedback based on the user's guesses.

**Riddle:** I challenge you to guess a hidden number within a limited number of tries. What am I?

**Short Story:** Jane wanted a fun game to play. She created a number guessing game where players try to guess a randomly generated number before running out of attempts.

## Project 85: Basic Contact Book

### Question:

Develop a contact book that allows users to add, view, and delete contacts.

**Code:**

contacts = {} def add\_contact(name, phone): contacts[name] = phone def view\_contacts(): if contacts: for name, phone in contacts.items(): print(f'{name}: {phone}') else: print('No contacts.') def delete\_contact(name): if name in contacts: del contacts[name] print(f'Contact {name} deleted.') else: print('Contact not found.') while True: print('1. View contacts') print('2. Add contact') print('3. Delete contact') print('4. Exit') choice = input('Enter your choice: ') if choice == '1': view\_contacts() elif choice == '2': name = input('Enter contact name: ') phone = input('Enter contact phone number: ') add\_contact(name, phone) elif choice == '3': name = input('Enter contact name to delete: ') delete\_contact(name) elif choice == '4': break else: print('Invalid choice.')

**Hint:** Use a dictionary to store contact names and phone numbers, and provide options to manage the contact list.

**Riddle:** I help you keep track of people’s names and phone numbers. What am I?

**Short Story:** Tom wanted an easy way to manage his contacts. He built a contact book application to add, view, and delete contacts as needed.

## Project 86: Currency Converter

### Question:

Create a currency converter that converts an amount from one currency to another using a fixed conversion rate.

**Code:**

def convert\_currency(amount, rate): return amount \* rate rate = 1.1 # Example conversion rate from USD to EUR amount = float(input('Enter amount in USD: ')) converted\_amount = convert\_currency(amount, rate) print(f'Amount in EUR: {converted\_amount:.2f}')

**Hint:** Define a conversion rate and use it to convert the amount from one currency to another.

**Riddle:** I transform money from one currency to another using a conversion rate. What am I?

**Short Story:** Alice wanted to convert her US dollars to euros. She created a currency converter to easily perform the conversion based on a fixed rate.

## Project 87: Simple Calculator

### Question:

Build a simple calculator that can perform addition, subtraction, multiplication, and division.

**Code:**

def calculator(): while True: print('1. Addition') print('2. Subtraction') print('3. Multiplication') print('4. Division') print('5. Exit') choice = input('Enter your choice: ') if choice == '5': break num1 = float(input('Enter first number: ')) num2 = float(input('Enter second number: ')) if choice == '1': print(f'Result: {num1 + num2}') elif choice == '2': print(f'Result: {num1 - num2}') elif choice == '3': print(f'Result: {num1 \* num2}') elif choice == '4': if num2 != 0: print(f'Result: {num1 / num2}') else: print('Error: Division by zero.') else: print('Invalid choice.') calculator()

**Hint:** Use conditional statements to perform the chosen arithmetic operation based on user input.

**Riddle:** I can add, subtract, multiply, and divide numbers. What am I?

**Short Story:** Sarah needed a tool for basic arithmetic operations. She built a simple calculator to perform addition, subtraction, multiplication, and division.

## Project 88: Prime Number Checker

### Question:

Create a program to check if a number is prime or not.

**Code:**

def is\_prime(n): if n <= 1: return False for i in range(2, int(n \*\* 0.5) + 1): if n % i == 0: return False return True number = int(input('Enter a number: ')) if is\_prime(number): print(f'{number} is a prime number.') else: print(f'{number} is not a prime number.')

**Hint:** A prime number is only divisible by 1 and itself. Use a loop to check divisibility.

**Riddle:** I can tell you whether a number is only divisible by itself and 1. What am I?

**Short Story:** David was interested in prime numbers. He wrote a program to check if a given number was prime or not for his mathematical studies.

## Project 89: Temperature Converter

### Question:

Develop a temperature converter that can convert between Celsius and Fahrenheit.

**Code:**

def celsius\_to\_fahrenheit(celsius): return (celsius \* 9/5) + 32 def fahrenheit\_to\_celsius(fahrenheit): return (fahrenheit - 32) \* 5/9 temp = float(input('Enter temperature: ')) unit = input('Enter unit (C for Celsius, F for Fahrenheit): ') if unit.upper() == 'C': print(f'Temperature in Fahrenheit: {celsius\_to\_fahrenheit(temp):.2f}') elif unit.upper() == 'F': print(f'Temperature in Celsius: {fahrenheit\_to\_celsius(temp):.2f}') else: print('Invalid unit.')

**Hint:** Use the conversion formulas for Celsius to Fahrenheit and vice versa to perform the conversion.

**Riddle:** I can convert temperatures between Celsius and Fahrenheit. What am I?

**Short Story:** Lisa wanted to convert temperatures for her weather report. She built a temperature converter to switch between Celsius and Fahrenheit easily.

## Project 90: Simple Quiz Game

### Question:

Create a simple quiz game that asks the user a series of questions and provides feedback on their answers.

**Code:**

questions = { 'What is the capital of France?': 'Paris', 'What is 2 + 2?': '4', 'What is the largest planet in our solar system?': 'Jupiter' } score = 0 for question, answer in questions.items(): user\_answer = input(question + ' ') if user\_answer.lower() == answer.lower(): print('Correct!') score += 1 else: print('Wrong! The correct answer is', answer) print(f'Your final score is {score}/{len(questions)}')

**Hint:** Use a dictionary to store questions and answers, and compare user input with the correct answers.

**Riddle:** I test your knowledge with questions and tell you how many you got right. What am I?

**Short Story:** Emily wanted to create a fun quiz game. She designed a simple quiz game that asks questions and keeps track of the score based on the user's answers.

## Project 91: Simple Stopwatch

### Question:

Build a simple stopwatch that can start, stop, and reset the timer.

**Code:**

import time def stopwatch(): start\_time = None while True: print('1. Start') print('2. Stop') print('3. Reset') print('4. Exit') choice = input('Enter your choice: ') if choice == '1': if start\_time is None: start\_time = time.time() print('Stopwatch started.') else: print('Stopwatch is already running.') elif choice == '2': if start\_time is not None: elapsed\_time = time.time() - start\_time print(f'Stopwatch stopped. Time elapsed: {elapsed\_time:.2f} seconds') start\_time = None else: print('Stopwatch is not running.') elif choice == '3': start\_time = None print('Stopwatch reset.') elif choice == '4': break else: print('Invalid choice.') stopwatch()

**Hint:** Use the `time` module to measure elapsed time and provide options to start, stop, and reset the stopwatch.

**Riddle:** I measure how much time has passed and can be started, stopped, or reset. What am I?

**Short Story:** Lucas wanted a tool to track time during his experiments. He built a simple stopwatch that could start, stop, and reset to measure elapsed time accurately.

## Project 92: Basic To-Do List with Persistence

### Question:

Create a to-do list application that allows adding, viewing, and removing tasks, and saves the list to a file so it persists between runs.

**Code:**

import json def load\_tasks(filename): try: with open(filename, 'r') as file: return json.load(file) except FileNotFoundError: return [] def save\_tasks(filename, tasks): with open(filename, 'w') as file: json.dump(tasks, file) def todo\_list(): tasks = load\_tasks('tasks.json') while True: print('1. Add Task') print('2. View Tasks') print('3. Remove Task') print('4. Exit') choice = input('Enter your choice: ') if choice == '1': task = input('Enter task: ') tasks.append(task) save\_tasks('tasks.json', tasks) elif choice == '2': if tasks: for i, task in enumerate(tasks, 1): print(f'{i}. {task}') else: print('No tasks to display.') elif choice == '3': if tasks: for i, task in enumerate(tasks, 1): print(f'{i}. {task}') task\_num = int(input('Enter task number to remove: ')) if 1 <= task\_num <= len(tasks): tasks.pop(task\_num - 1) save\_tasks('tasks.json', tasks) else: print('Invalid task number.') else: print('No tasks to remove.') elif choice == '4': break else: print('Invalid choice.') todo\_list()

**Hint:** Use JSON to store tasks so that the to-do list persists even after the program is closed.

**Riddle:** I keep track of tasks and save them so that they persist between sessions. What am I?

**Short Story:** Michael needed a way to manage his daily tasks. He developed a to-do list application that saved tasks to a file so he could keep track of them even after closing the app.

## Project 93: Simple Number Guessing Game

### Question:

Create a number guessing game where the user has to guess a randomly generated number within a certain range.

**Code:**

import random def number\_guessing\_game(): number\_to\_guess = random.randint(1, 100) attempts = 0 while True: guess = int(input('Guess the number (between 1 and 100): ')) attempts += 1 if guess < number\_to\_guess: print('Too low!') elif guess > number\_to\_guess: print('Too high!') else: print(f'Congratulations! You guessed the number in {attempts} attempts.') break number\_guessing\_game()

**Hint:** Use the `random` module to generate a random number and compare user guesses with the generated number.

**Riddle:** I generate a random number and let you guess it, providing feedback on whether your guess is too high or too low. What am I?

**Short Story:** Emma wanted a fun way to challenge her friends. She created a number guessing game where users guessed a random number, and she provided hints until they got it right.

## Project 94: Simple File Reader

### Question:

Write a program to read the contents of a file and display them to the user.

**Code:**

def read\_file(filename): try: with open(filename, 'r') as file: content = file.read() print(content) except FileNotFoundError: print('File not found.') filename = input('Enter the filename: ') read\_file(filename)

**Hint:** Use the `open` function to read the contents of a file and handle exceptions if the file does not exist.

**Riddle:** I read the content of a file and display it to you. What am I?

**Short Story:** Liam needed to view the contents of a file for his project. He wrote a simple file reader to open and display file contents to make his work easier.

## Project 95: Basic Unit Converter

### Question:

Create a unit converter that can convert between different units of length (e.g., meters to feet, centimeters to inches).

**Code:**

def meters\_to\_feet(meters): return meters \* 3.28084 def centimeters\_to\_inches(centimeters): return centimeters / 2.54 length = float(input('Enter length: ')) unit = input('Enter unit (M for meters, CM for centimeters): ') if unit.upper() == 'M': print(f'Length in feet: {meters\_to\_feet(length):.2f}') elif unit.upper() == 'CM': print(f'Length in inches: {centimeters\_to\_inches(length):.2f}') else: print('Invalid unit.')

**Hint:** Implement conversion functions for different units and allow users to choose the unit they want to convert from.

**Riddle:** I convert measurements from one unit to another, such as meters to feet. What am I?

**Short Story:** Jack wanted to convert lengths between different units. He created a unit converter that could easily switch between meters, feet, centimeters, and inches.

## Project 96: Simple Expense Tracker

### Question:

Develop a basic expense tracker that allows users to add and view expenses, and saves the data to a file.

**Code:**

import json def load\_expenses(filename): try: with open(filename, 'r') as file: return json.load(file) except FileNotFoundError: return [] def save\_expenses(filename, expenses): with open(filename, 'w') as file: json.dump(expenses, file) def expense\_tracker(): expenses = load\_expenses('expenses.json') while True: print('1. Add Expense') print('2. View Expenses') print('3. Exit') choice = input('Enter your choice: ') if choice == '1': name = input('Enter expense name: ') amount = float(input('Enter amount: ')) expenses.append({'name': name, 'amount': amount}) save\_expenses('expenses.json', expenses) elif choice == '2': if expenses: for expense in expenses: print(f'{expense["name"]}: ${expense["amount"]:.2f}') else: print('No expenses to display.') elif choice == '3': break else: print('Invalid choice.') expense\_tracker()

**Hint:** Use JSON to store and load expense data, and provide options to add and view expenses.

**Riddle:** I track and record expenses and save the data to a file. What am I?

**Short Story:** Nora needed a way to manage her spending. She created a simple expense tracker that allowed her to add and view expenses, keeping track of her spending efficiently.

## Project 97: Simple Interest Calculator

### Question:

Write a program to calculate simple interest given the principal amount, rate of interest, and time period.

**Code:**

def calculate\_simple\_interest(principal, rate, time): return principal \* rate \* time / 100 principal = float(input('Enter principal amount: ')) rate = float(input('Enter rate of interest: ')) time = float(input('Enter time period (in years): ')) interest = calculate\_simple\_interest(principal, rate, time) print(f'Simple Interest: ${interest:.2f}')

**Hint:** Use the formula for simple interest: `principal \* rate \* time / 100` to perform the calculation.

**Riddle:** I calculate the interest earned on a principal amount over a period of time. What am I?

**Short Story:** Liam wanted to know how much interest he would earn on his savings. He wrote a simple interest calculator to quickly compute the interest based on his principal, interest rate, and time.

## Project 98: Basic Temperature Converter

### Question:

Create a temperature converter that converts between Celsius and Fahrenheit.

**Code:**

def celsius\_to\_fahrenheit(celsius): return (celsius \* 9/5) + 32 def fahrenheit\_to\_celsius(fahrenheit): return (fahrenheit - 32) \* 5/9 temperature = float(input('Enter temperature: ')) unit = input('Enter unit (C for Celsius, F for Fahrenheit): ') if unit.upper() == 'C': print(f'Temperature in Fahrenheit: {celsius\_to\_fahrenheit(temperature):.2f}') elif unit.upper() == 'F': print(f'Temperature in Celsius: {fahrenheit\_to\_celsius(temperature):.2f}') else: print('Invalid unit.')

**Hint:** Implement conversion functions for Celsius and Fahrenheit, and allow users to choose the unit they want to convert from.

**Riddle:** I convert temperatures between Celsius and Fahrenheit. What am I?

**Short Story:** Sarah needed to convert temperatures for her weather research. She created a temperature converter that allowed her to switch between Celsius and Fahrenheit easily.

## Project 99: Simple Calculator with GUI

### Question:

Develop a simple calculator with a graphical user interface (GUI) that performs basic arithmetic operations like addition, subtraction, multiplication, and division.

**Code:**

import tkinter as tk def add(): try: num1 = float(entry1.get()) num2 = float(entry2.get()) result.set(num1 + num2) except ValueError: result.set('Error') def subtract(): try: num1 = float(entry1.get()) num2 = float(entry2.get()) result.set(num1 - num2) except ValueError: result.set('Error') def multiply(): try: num1 = float(entry1.get()) num2 = float(entry2.get()) result.set(num1 \* num2) except ValueError: result.set('Error') def divide(): try: num1 = float(entry1.get()) num2 = float(entry2.get()) if num2 != 0: result.set(num1 / num2) else: result.set('Error') except ValueError: result.set('Error') app = tk.Tk() app.title('Simple Calculator') entry1 = tk.Entry(app) entry1.pack() entry2 = tk.Entry(app) entry2.pack() result = tk.StringVar() result\_label = tk.Label(app, textvariable=result) result\_label.pack() tk.Button(app, text='+', command=add).pack() tk.Button(app, text='-', command=subtract).pack() tk.Button(app, text='\*', command=multiply).pack() tk.Button(app, text='/', command=divide).pack() app.mainloop()

**Hint:** Use the `tkinter` library to create the GUI and handle basic arithmetic operations.

**Riddle:** I perform basic arithmetic operations and display the results in a graphical interface. What am I?

**Short Story:** Jessica wanted to make basic calculations easier. She built a simple calculator with a GUI using tkinter that allowed her to add, subtract, multiply, and divide numbers with ease.

## Project 100: Simple Contact Book

### Question:

Create a contact book application that allows users to add, view, and search for contacts, and saves the contacts to a file.

**Code:**

import json def load\_contacts(filename): try: with open(filename, 'r') as file: return json.load(file) except FileNotFoundError: return [] def save\_contacts(filename, contacts): with open(filename, 'w') as file: json.dump(contacts, file) def contact\_book(): contacts = load\_contacts('contacts.json') while True: print('1. Add Contact') print('2. View Contacts') print('3. Search Contact') print('4. Exit') choice = input('Enter your choice: ') if choice == '1': name = input('Enter contact name: ') phone = input('Enter phone number: ') contacts.append({'name': name, 'phone': phone}) save\_contacts('contacts.json', contacts) elif choice == '2': if contacts: for contact in contacts: print(f'{contact["name"]}: {contact["phone"]}') else: print('No contacts to display.') elif choice == '3': search\_name = input('Enter name to search: ') found = False for contact in contacts: if contact['name'].lower() == search\_name.lower(): print(f'{contact["name"]}: {contact["phone"]}') found = True break if not found: print('Contact not found.') elif choice == '4': break else: print('Invalid choice.') contact\_book()

**Hint:** Store contacts in a JSON file to persist data, and provide options to add, view, and search for contacts.

**Riddle:** I keep track of names and phone numbers, allowing you to add, view, and search for contacts. What am I?

**Short Story:** Olivia wanted an easy way to manage her contacts. She developed a simple contact book application that saved contact details to a file and allowed her to search for contacts quickly.