1. Python Questions

23. Write a program to assign a string value to a variable and print it.

24. Write a program to assign values to multiple variables in a single line and print them.

25. Write a program to create a variable x with an integer value and cast it to a float.

26. Write a program to define a variable inside a function and print it.

27. Write a program to assign the same value to three variables and print all of them.

28. Write a program to show the difference between a global and local variable with the same name.

29. Write a program to update a global variable inside a function using the global keyword.

30. Write a program to cast a float to an integer and observe the output.

31. Write a program to assign multiple values to fewer variables and handle the error.

32. Write a program that uses a variable before declaring it inside a function (without global) and explain the error.

33. Write a program to take three user inputs in a single line and assign them to three variables, then print them.

34. Write a program that has a global variable and a function that modifies it using both local and global scope.

35. Write a program to cast a string containing numbers into integers and perform arithmetic operations.

36. Write a program that dynamically changes variable types (int → str → float) and prints each type.

37. Write a program that shows unpacking of values using multiple variable assignment, including the use of \* operator.

38. Write a program to define a variable globally, then shadow it locally and print both inside and outside the function.

39. Write a program that takes a list of strings representing numbers, casts them to integers, sums them, and prints the total.

40. Write a program to demonstrate the use of global inside nested functions (function inside function).

41. Write a program where multiple assignments are used with different types and one type is changed dynamically during execution.

42. Write a program to simulate an error caused by assigning values to more variables than provided and fix it using \* unpacking.

43. Strings:

44. Write a program to convert a string to lowercase using .lower().

45. Write a program to convert a string to uppercase using .upper().

46. Write a program to remove spaces from the beginning and end of a string using .strip().

47. Write a program to replace "a" with "@" using .replace().

48. Write a program to split a string into words using .split().

49. Write a program to check if a string starts with "Hello" using .startswith().

50. Write a program to check if a string ends with "World" using .endswith().

51. Write a program to count how many times "e" appears in a string using .count().

52. Write a program to find the index of "o" using .find() in a string.

53. Write a program to center-align a string with "\*" padding using .center(20, "\*").

54. Write a program to check if all characters in a string are alphabetic using .isalpha().

55. Write a program to check if a string is all digits using .isdigit().

56. Write a program to swap the case of all characters using .swapcase().

57. Write a program to title-case a string using .title().

58. Write a program to find "Python" using .index() and handle error if not found.

59. Write a program to join a list ["Python", "is", "fun"] into a single string using .join().

60. Write a program to check if a string is all lowercase using .islower().

61. Write a program to check if a string is all uppercase using .isupper().

62. Write a program to check if a string has only whitespace using .isspace().

63. Write a program to convert a string with tab spacing using .expandtabs(4).

64. Write a program to right-align a string using .rjust(10, ".").

65. Write a program to validate if a string is a valid identifier using .isidentifier().

66. Write a program to check if all characters in a string are printable using .isprintable().

67. Write a program that takes user input and checks if it’s numeric using .isnumeric().

68. Write a program to remove trailing characters (like periods) using .rstrip(".").

69. Write a program to take a user's full name as input and print initials in uppercase followed by last name.

70. Write a program to check if a user-input sentence is a palindrome, ignoring spaces and cases.

71. Write a program to take a string and print every 2nd character in reverse order.

72. Write a program to input a sentence and count how many vowels it contains.

73. Write a program to replace all vowels in a string with \* using replace() and loop.

74. Write a program to input two strings and compare them after trimming, lowering, and removing extra spaces.

75. Write a program to input a string and count how many words are longer than 4 characters.

76. 33.Write a program to input a word and print it centered in a 30-character wide line using \* as filler.

77. 34.Write a program to validate if a user input is a valid Python identifier.

Write a program to input a multi-line string and print the total number of lines.

Write a program to input a string and print only the alphanumeric characters.

Write a program to input a sentence and find the longest word.

Write a program to take two inputs and print a formatted table of name and score.

. Write a program to input a string and format it as: "Welcome, <Name>!"

Write a program to check if a number is positive, negative, or zero.

Write a program to find the largest of three numbers.

Write a program to check if a number is even or odd.

Write a program to check if a year is a leap year.

Write a program to assign grades based on marks.

Write a program to print day name based on user input (1–7).

Write a program to create a simple calculator using match-case.

Write a program to identify vowels using match-case.

. Write a program to print the season based on month number.

10. Write a program to display traffic light instruction using match-case.

Write a program to print numbers 1 to 10 using while loop.

Write a program to find the factorial of a number using while loop.

Write a program to sum digits of a number using while loop.

Write a program to find how many digits are in a number.

Write a program to print multiplication table of a number.. Write a program to print even numbers from 1 to 20.

. Write a program to print characters in a string.

Write a program to find the sum of numbers from 1 to n.

Write a program to print a triangle pattern of stars.

Write a program to print only vowels from a string.

Write a program to count vowels in a string.

Write a program to find square of numbers 1 to 10.

. Write a program to print only odd numbers between 50 and 100.

Write a program to count how many times a character appears in a string.

Write a program to check if a number is prime using for and else.

Write a program to print a diamond pattern using for and if.

Write a program to simulate a simple calculator using match-case.

Write a program to find the factorial using a while loop.

Write a program to find the number of vowels in a sentence.

Write a program to print Fibonacci numbers up to N using while.

Write a program to check if a number is Armstrong or not.

Write a program to count how many times each vowel appears in a sentence.

Write a program to validate a password using conditions.

Write a program to print a number pyramid pattern using nested loops.

Write a program to insert a new item in the 2nd position of a list.

Write a program to remove all occurrences of the number 5 in a list.

Write a program to print only even numbers from a list using list comprehension.

Write a program to copy a list and prove it's not linked to the original.

Write a program to count how many times 'apple' appears in a list.

Write a program to create a list of squares from 1 to 10 using list comprehension.

Write a program to find and remove duplicates from a list.

Write a program to check if a user-entered value exists in the list.

Write a program to filter names longer than 4 characters.

Write a program to replace all negative numbers in a list with 0.

Write a program to add user inputs to a list until they enter 'stop'.

. Write a program to extract all even numbers from a nested list.

Write a program to sort a list of dictionaries by 'age'.

Write a program to remove duplicates from a list while keeping order.

Write a program to find the common elements in two lists.

. Write a program to count how many sublists are in a list.

Write a program to flatten a list of arbitrary depth (only 2-levels).

Write a program to filter students who passed (score ≥ 50) from a list of dicts.

Write a program to reverse a list without using built-in reverse().

Write a program to count frequency of each item in a list.

Write a program to zip two lists into a dictionary

Write a program to create a tuple from user input of 5 names.

Write a program to access the last two elements of a tuple.

1. Python Questions

23. Write a program to assign a string value to a variable and print it.

24. Write a program to assign values to multiple variables in a single line and print them.

25. Write a program to create a variable x with an integer value and cast it to a float.

26. Write a program to define a variable inside a function and print it.

27. Write a program to assign the same value to three variables and print all of them.

28. Write a program to show the difference between a global and local variable with the same name.

29. Write a program to update a global variable inside a function using the global keyword.

30. Write a program to cast a float to an integer and observe the output.

31. Write a program to assign multiple values to fewer variables and handle the error.

32. Write a program that uses a variable before declaring it inside a function (without global) and explain the error.

33. Write a program to take three user inputs in a single line and assign them to three variables, then print them.

34. Write a program that has a global variable and a function that modifies it using both local and global scope.

35. Write a program to cast a string containing numbers into integers and perform arithmetic operations.

36. Write a program that dynamically changes variable types (int → str → float) and prints each type.

37. Write a program that shows unpacking of values using multiple variable assignment, including the use of \* operator.

38. Write a program to define a variable globally, then shadow it locally and print both inside and outside the function.

39. Write a program that takes a list of strings representing numbers, casts them to integers, sums them, and prints the total.

40. Write a program to demonstrate the use of global inside nested functions (function inside function).

41. Write a program where multiple assignments are used with different types and one type is changed dynamically during execution.

42. Write a program to simulate an error caused by assigning values to more variables than provided and fix it using \* unpacking.

43. Strings:

44. Write a program to convert a string to lowercase using .lower().

45. Write a program to convert a string to uppercase using .upper().

46. Write a program to remove spaces from the beginning and end of a string using .strip().

47. Write a program to replace "a" with "@" using .replace().

48. Write a program to split a string into words using .split().

49. Write a program to check if a string starts with "Hello" using .startswith().

50. Write a program to check if a string ends with "World" using .endswith().

51. Write a program to count how many times "e" appears in a string using .count().

52. Write a program to find the index of "o" using .find() in a string.

53. Write a program to center-align a string with "\*" padding using .center(20, "\*").

54. Write a program to check if all characters in a string are alphabetic using .isalpha().

55. Write a program to check if a string is all digits using .isdigit().

56. Write a program to swap the case of all characters using .swapcase().

57. Write a program to title-case a string using .title().

58. Write a program to find "Python" using .index() and handle error if not found.

59. Write a program to join a list ["Python", "is", "fun"] into a single string using .join().

60. Write a program to check if a string is all lowercase using .islower().

61. Write a program to check if a string is all uppercase using .isupper().

62. Write a program to check if a string has only whitespace using .isspace().

63. Write a program to convert a string with tab spacing using .expandtabs(4).

64. Write a program to right-align a string using .rjust(10, ".").

65. Write a program to validate if a string is a valid identifier using .isidentifier().

66. Write a program to check if all characters in a string are printable using .isprintable().

67. Write a program that takes user input and checks if it’s numeric using .isnumeric().

68. Write a program to remove trailing characters (like periods) using .rstrip(".").

69. Write a program to take a user's full name as input and print initials in uppercase followed by last name.

70. Write a program to check if a user-input sentence is a palindrome, ignoring spaces and cases.

71. Write a program to take a string and print every 2nd character in reverse order.

72. Write a program to input a sentence and count how many vowels it contains.

73. Write a program to replace all vowels in a string with \* using replace() and loop.

74. Write a program to input two strings and compare them after trimming, lowering, and removing extra spaces.

75. Write a program to input a string and count how many words are longer than 4 characters.

76. 33.Write a program to input a word and print it centered in a 30-character wide line using \* as filler.

77. 34.Write a program to validate if a user input is a valid Python identifier.

Write a program to input a multi-line string and print the total number of lines.

Write a program to input a string and print only the alphanumeric characters.

Write a program to input a sentence and find the longest word.

Write a program to take two inputs and print a formatted table of name and score.

. Write a program to input a string and format it as: "Welcome, <Name>!"

Write a program to check if a number is positive, negative, or zero.

Write a program to find the largest of three numbers.

Write a program to check if a number is even or odd.

Write a program to check if a year is a leap year.

Write a program to assign grades based on marks.

Write a program to print day name based on user input (1–7).

Write a program to create a simple calculator using match-case.

Write a program to identify vowels using match-case.

. Write a program to print the season based on month number.

10. Write a program to display traffic light instruction using match-case.

Write a program to print numbers 1 to 10 using while loop.

Write a program to find the factorial of a number using while loop.

Write a program to sum digits of a number using while loop.

Write a program to find how many digits are in a number.

Write a program to print multiplication table of a number.. Write a program to print even numbers from 1 to 20.

. Write a program to print characters in a string.

Write a program to find the sum of numbers from 1 to n.

Write a program to print a triangle pattern of stars.

Write a program to print only vowels from a string.

Write a program to count vowels in a string.

Write a program to find square of numbers 1 to 10.

. Write a program to print only odd numbers between 50 and 100.

Write a program to count how many times a character appears in a string.

Write a program to check if a number is prime using for and else.

Write a program to print a diamond pattern using for and if.

Write a program to simulate a simple calculator using match-case.

Write a program to find the factorial using a while loop.

Write a program to find the number of vowels in a sentence.

Write a program to print Fibonacci numbers up to N using while.

Write a program to check if a number is Armstrong or not.

Write a program to count how many times each vowel appears in a sentence.

Write a program to validate a password using conditions.

Write a program to print a number pyramid pattern using nested loops.

Write a program to insert a new item in the 2nd position of a list.

Write a program to remove all occurrences of the number 5 in a list.

Write a program to print only even numbers from a list using list comprehension.

Write a program to copy a list and prove it's not linked to the original.

Write a program to count how many times 'apple' appears in a list.

Write a program to create a list of squares from 1 to 10 using list comprehension.

Write a program to find and remove duplicates from a list.

Write a program to check if a user-entered value exists in the list.

Write a program to filter names longer than 4 characters.

Write a program to replace all negative numbers in a list with 0.

Write a program to add user inputs to a list until they enter 'stop'.

. Write a program to extract all even numbers from a nested list.

Write a program to sort a list of dictionaries by 'age'.

Write a program to remove duplicates from a list while keeping order.

Write a program to find the common elements in two lists.

. Write a program to count how many sublists are in a list.

Write a program to flatten a list of arbitrary depth (only 2-levels).

Write a program to filter students who passed (score ≥ 50) from a list of dicts.

Write a program to reverse a list without using built-in reverse().

Write a program to count frequency of each item in a list.

Write a program to zip two lists into a dictionary

Write a program to create a tuple from user input of 5 names.

Write a program to access the last two elements of a tuple.

Write a program to check if a number exists in a tuple.

Write a program to unpack a tuple of 3 values into variables.

Write a program to count how many times a value appears in a tuple.

Write a program to join two tuples entered by the user.

Write a program to find the index of a value in a tuple.

Write a program to create a nested tuple and access a specific element.

Write a program to sort values from a tuple (by converting it to a list).

Write a program to find the largest and smallest numbers in a tuple.

Write a program to remove duplicates from a tuple.

Write a program to reverse a tuple.

Write a program to accept a list of items from the user and store them as a tuple. Then print only the elements at even indices.

Input:

Enter items separated by comma: apple,banana,cherry,date,fig

Write a program to unpack a tuple into three variables and print them in reverse order.

✅ 3. Write a program to convert a tuple of strings into a single comma-separated string.

✅ 4. Write a program to accept a tuple of numbers and find the average of only the odd numbers.

. Write a program to accept a tuple and remove duplicates while preserving order.

6. Write a program to accept a tuple of words and print the longest word.

Write a program to replace the 2nd element of a tuple by converting it to a list and back.

Write a program to sort a tuple of integers in descending order.

Write a program to check if a nested tuple contains a given element.

Write a program to sort a tuple of integers in descending order.

Write a program to rotate a tuple to the left by one position.

✅ 12. Write a program to reverse each word in a tuple of strings.

Write a program to print the element with the highest frequency in a tuple.

Write a program to print the element with the highest frequency in a tuple.

Write a program to flatten a nested tuple of 2-level depth.

Write a program to take two sets as input and print their symmetric difference.

Write a program to find common vowels in a given sentence.

Write a program to find if one set is a subset of another.

Write a program to remove all duplicate characters from a word using sets.

Write a program to add only non-existing elements from one set to another.

Write a program to count the number of unique words in a sentence.

Write a program to check if two sets are disjoint.

Write a program to remove elements from one set that exist in another.

Write a program to find all letters used in two input names.

Write a program to keep only common elements between two sets.

Write a program to remove a given item from a set only if it exists.

Write a program to take 3 sets and find common elements in all.

Write a program to take two sets and print only unique elements of the first.

Write a program to find union of multiple sets.

Write a program to find max and min from a numeric set.

**🧠 1. Write a program to group words by their first letter using a dictionary.**

**🧠 2. Write a program to count word frequencies in a sentence and print them sorted by frequency.**

**🧠 3. Write a program to check if two dictionaries are deeply equal (including nested ones).**

**🧠 4. Write a program to flatten a nested dictionary.**

**🧠 5. Write a program to create a dictionary where keys are numbers from 1 to n and values are their squares.**

**🧠 7. Write a program to merge dictionaries with duplicate keys by summing their values.**

**🧠 8. Write a program to create a nested dictionary from a list of tuples (hierarchical).**

**🧠 9. Write a program to find the key with the maximum value in a dictionary.**

**🧠 10. Write a program to update a nested dictionary key’s value safely.**

**1. Write a program to define a function that prints "Hello, World!"**

**2. Write a program to define a function that takes your name and prints a greeting.**

**3. Write a program to return the square of a number using a function.**

**4. Write a function that takes two numbers and returns their sum.**

**5. Write a program with a function that takes a list and prints all its elements.**

**6. Write a program with a function that returns whether a number is even.**

**7. Write a function that returns the length of a given string.**

**8. Write a program to create a default argument in a function.**

**9. Write a program to demonstrate keyword arguments.**

**10. Write a program that returns the factorial of a number using recursion.**

**11. Write a program using \*args to find the sum of any number of arguments.**

**12. Write a program using \*\*kwargs to print all key-value pairs.**

**13. Write a program to check if a string is a palindrome using a function.**

**14. Write a program to define a lambda function that multiplies 3 numbers.**

**15. Write a lambda function to return the square of a number and call it.**

**16. Write a program using map() and lambda to square a list of numbers.**

**17. Write a program using filter() and lambda to get only even numbers.**

**18. Write a program to use a lambda inside a function.**

**19. Write a recursive function to reverse a string.**

**20. Write a lambda function to sort a list of tuples by the second element.**

**21. Write a function that takes another function as an argument and calls it.**

**22. Write a program to find the max element using reduce() and lambda.**

**23. Write a program using lambda inside a list comprehension.**

**24. Write a higher-order function that returns a power function.**

**25. Write a lambda function with a condition to return "Even" or "Odd".**

1. **Write a Python program to create a simple class Student with a method show() that prints "Hello Student".**
2. **Write a program to initialize a class with name and age, and display them.**
3. **Write a class with a method that returns the square of a number.**
4. **Write a program with a class Car having attributes: brand and year. Print the brand.**
5. **Write a program to demonstrate self is used to access instance variables.**
6. **Write a program to check if a class method can be called multiple times.**
7. **Create a class Circle that calculates the area given the radius.**
8. **Write a program where you create two objects of the same class.**
9. **Create a class with a method that returns a string in uppercase.**
10. **Write a program to delete an attribute from an object.**
11. **Write a program to demonstrate single inheritance.**
12. **Demonstrate method overriding using inheritance.**
13. **Write a program to demonstrate super() for calling parent constructor.**
14. **Write a class that overrides the \_\_str\_\_ method.**
15. **Create an iterator that returns numbers from 1 to 3.**
16. **Write a program that demonstrates multiple inheritance.**
17. **Demonstrate polymorphism with two unrelated classes having same method.**

1. **Write a class with class variable and instance variable.**
2. **Write a program where a child class calls parent method and adds its own logic.**
3. **Write a class that supports iteration using \_\_iter\_\_ and \_\_next\_\_ manually.**

1. **Write a program for multilevel inheritance.**
2. **Demonstrate operator overloading for + using \_\_add\_\_.**
3. **Write a program to create a class method using @classmethod.**
4. **Use @staticmethod to create a utility function.**
5. **Create nested classes and access inner class.**
6. **Create an abstract base class using abc module.**
7. **Override \_\_len\_\_ in a custom class.**
8. **Implement duck typing with different classes.**
9. **Create a class that acts like a list using magic methods.**
10. **Write a metaclass that prints when a class is created.**

Great! Based on the file handling topics from W3Schools, here are **20 Python questions with answers and explanations**—starting from **basic**, then moving to **tricky and advanced levels**.

**🔹 BASIC LEVEL FILE HANDLING QUESTIONS**

**1. Write a program to open a file named demo.txt and read its content.**

f = open("demo.txt", "r")

print(f.read())

f.close()

**Explanation:**

* "r" mode opens file for reading.
* Always close file after reading using f.close().

**2. Write a program to create a new file named hello.txt and write "Hello World" into it.**

f = open("hello.txt", "w")

f.write("Hello World")

f.close()

**Explanation:**

* "w" mode creates the file or overwrites if it exists.

**3. Write a program to append "Python" to hello.txt.**

f = open("hello.txt", "a")

f.write("\nPython")

f.close()

**Explanation:**

* "a" mode appends data to the file without deleting existing content.

**4. Write a program to read only the first 5 characters of a file.**

f = open("demo.txt", "r")

print(f.read(5))

f.close()

**Explanation:**

* read(5) reads the first 5 characters only.

**5. Write a program to read a file line by line using a loop.**

f = open("demo.txt", "r")

for line in f:

print(line.strip())

f.close()

**Explanation:**

* Using for line in f reads file line-by-line.
* strip() removes \n at the end of each line.

**🔹 TRICKY LEVEL FILE HANDLING QUESTIONS**

**6. Write a program to check if a file named sample.txt exists.**

import os

if os.path.exists("sample.txt"):

print("File exists.")

else:

print("File does not exist.")

**Explanation:**

* os.path.exists() checks file existence.

**7. Write a program to delete a file named temp.txt if it exists.**

import os

if os.path.exists("temp.txt"):

os.remove("temp.txt")

print("Deleted.")

else:

print("File not found.")

**Explanation:**

* os.remove() deletes the file safely after checking.

**8. Write a program to create a file and write multiple lines into it.**

lines = ["Line 1", "Line 2", "Line 3"]

f = open("multi.txt", "w")

for line in lines:

f.write(line + "\n")

f.close()

**Explanation:**

* Writes each line with a newline character.

**9. Write a program to read a file and count the number of lines.**

f = open("demo.txt", "r")

lines = f.readlines()

print("Total lines:", len(lines))

f.close()

**Explanation:**

* readlines() returns a list of all lines.

**10. Write a program to read a file and print all words in reverse order.**

f = open("demo.txt", "r")

text = f.read()

words = text.split()

print(" ".join(reversed(words)))

f.close()

**Explanation:**

* Splits text into words, reverses them, and joins with space.

**🔹 ADVANCED LEVEL FILE HANDLING QUESTIONS**

**11. Write a program to copy content from one file to another.**

f1 = open("source.txt", "r")

f2 = open("destination.txt", "w")

f2.write(f1.read())

f1.close()

f2.close()

**Explanation:**

* Reads from one file and writes into another.

**12. Write a program to count occurrences of a specific word in a file.**

word\_to\_find = "Python"

f = open("demo.txt", "r")

text = f.read()

print("Occurrences:", text.count(word\_to\_find))

f.close()

**Explanation:**

* str.count() counts the given word.

**13. Write a program to display only lines that contain the word "data" from a file.**

f = open("demo.txt", "r")

for line in f:

if "data" in line:

print(line.strip())

f.close()

**Explanation:**

* Checks and prints only lines containing "data".

**14. Write a program to write a list of names into a file, one name per line.**

names = ["Alice", "Bob", "Charlie"]

f = open("names.txt", "w")

f.writelines(name + "\n" for name in names)

f.close()

**Explanation:**

* writelines() takes an iterable of strings.

**15. Write a program to read a file and replace every occurrence of "old" with "new".**

f = open("demo.txt", "r")

content = f.read()

f.close()

content = content.replace("old", "new")

f = open("demo.txt", "w")

f.write(content)

f.close()

**Explanation:**

* First reads content, modifies it, then writes it back.

**16. Write a program to read the last line of a file.**

f = open("demo.txt", "r")

lines = f.readlines()

print(lines[-1].strip())

f.close()

**Explanation:**

* Accesses last element from the list of lines.

**17. Write a program to read only the second line of a file.**

f = open("demo.txt", "r")

lines = f.readlines()

if len(lines) >= 2:

print(lines[1].strip())

else:

print("Less than 2 lines.")

f.close()

**Explanation:**

* Index 1 refers to the second line (0-based indexing).

**18. Write a program to open a file safely using with block.**

with open("demo.txt", "r") as f:

print(f.read())

**Explanation:**

* with block auto-closes the file even if an error occurs.

**19. Write a program to get the file size in bytes.**

import os

size = os.path.getsize("demo.txt")

print("File size:", size, "bytes")

**Explanation:**

* getsize() gives size in bytes.

**20. Write a program that reads a file and prints only unique words.**

f = open("demo.txt", "r")

words = f.read().split()

unique\_words = set(words)

print(unique\_words)

f.close()

**Explanation:**

* Converting list of words to a set removes duplicates.

Would you like these questions as a PDF later, or should I continue with **error handling questions** next?

Great! Here are **2 different Python questions and answers** based on:

**✅ 1. Variable Scope (Local, Global)**

**Question:**  
Write a program to demonstrate the difference between a local and a global variable in Python.

**Answer:**

x = 50 # Global variable

def my\_function():

x = 10 # Local variable

print("Local x inside function:", x)

my\_function()

print("Global x outside function:", x)

**Output:**

Local x inside function: 10

Global x outside function: 50

**Explanation:**

* The global variable x = 50 exists outside any function.
* Inside the function, a **local variable** x = 10 is created.
* Changing x inside the function does not affect the global x.

**✅ 2. Error Handling with Try-Except**

**Question:**  
Write a program to handle a division by zero error using try-except.

**Answer:**

try:

a = int(input("Enter a number: "))

result = 100 / a

print("Result:", result)

except ZeroDivisionError:

print("You cannot divide by zero!")

except ValueError:

print("Please enter a valid number.")

**Example Output:**

Enter a number: 0

You cannot divide by zero!

**Explanation:**

* If the user inputs 0, ZeroDivisionError is caught.
* If the user inputs something like "abc", ValueError is caught.
* Using try-except helps avoid program crashes due to runtime errors.

Would you like more questions on scope, error types (finally, else, raise), or more tricky cases?

Perfect! Here are **4 more different questions** (2 on **scope** and 2 on **try-except**) ranging from **basic to tricky/advanced**, based on the W3Schools topics:

**🔵 Scope Questions**

**✅ 1. *Tricky Scope with Global Keyword***

**Question:**  
Write a program where a function modifies a global variable using the global keyword.

**Answer:**

x = 5

def update():

global x

x = x + 10

print("Inside function:", x)

update()

print("Outside function:", x)

**Output:**

Inside function: 15

Outside function: 15

**Explanation:**

* Using global x allows modification of the global variable inside the function.
* Without global, Python would create a new local variable x.

**✅ 2. *Nested Function with nonlocal Keyword***

**Question:**  
Write a program to show how nonlocal works in nested functions.

**Answer:**

def outer():

x = "outer value"

def inner():

nonlocal x

x = "changed in inner"

print("Inner:", x)

inner()

print("Outer:", x)

outer()

**Output:**

Inner: changed in inner

Outer: changed in inner

**Explanation:**

* nonlocal allows inner() to modify the x from outer()’s scope.

**🔴 Try-Except Questions**

**✅ 3. *Multiple Exceptions Handling***

**Question:**  
Write a program that catches both ValueError and ZeroDivisionError.

**Answer:**

try:

num = int(input("Enter a number: "))

print(10 / num)

except ValueError:

print("Not a number!")

except ZeroDivisionError:

print("Cannot divide by zero!")

**Input Example:**

abc

**Output:**

Not a number!

**✅ 4. *Try-Except with else and finally***

**Question:**  
Write a program that uses else and finally blocks along with try-except.

**Answer:**

try:

x = int(input("Enter a positive number: "))

if x < 0:

raise ValueError("Negative number not allowed")

except ValueError as ve:

print("Error:", ve)

else:

print("You entered:", x)

finally:

print("Execution completed.")

**Output Example:**

Enter a positive number: -5

Error: Negative number not allowed

Execution completed.

Would you like:

* More **tricky/real-world examples**?
* A **small quiz** on scope and error handling?
* Or move to the next W3Schools topic?
* Great! Here are some classic Python program questions with answers and explanations, including factorial, Fibonacci, prime, Armstrong, and anagram:
* **1. Write a program to find the factorial of a number.**
* num = int(input("Enter a number: "))
* factorial = 1
* for i in range(1, num + 1):
* factorial \*= i
* print("Factorial:", factorial)
* ✅ **Explanation**: Multiplying numbers from 1 to num using a loop.
* **2. Write a program to print Fibonacci series up to n terms.**
* n = int(input("Enter number of terms: "))
* a, b = 0, 1
* for \_ in range(n):
* print(a, end=" ")
* a, b = b, a + b
* ✅ **Explanation**: Each number is the sum of the two preceding ones.
* **3. Write a program to check if a number is prime.**
* num = int(input("Enter a number: "))
* if num > 1:
* for i in range(2, num):
* if num % i == 0:
* print("Not a prime number")
* break
* else:
* print("Prime number")
* else:
* print("Not a prime number")
* ✅ **Explanation**: A prime number has only 1 and itself as divisors.
* **4. Write a program to check if a number is Armstrong (e.g., 153).**
* num = int(input("Enter a number: "))
* sum = 0
* temp = num
* n = len(str(num))
* while temp > 0:
* digit = temp % 10
* sum += digit \*\* n
* temp //= 10
* if num == sum:
* print("Armstrong number")
* else:
* print("Not an Armstrong number")
* ✅ **Explanation**: Sum of digits raised to power of number of digits.
* **5. Write a program to check if two strings are anagrams.**
* str1 = input("Enter first string: ")
* str2 = input("Enter second string: ")
* if sorted(str1.lower()) == sorted(str2.lower()):
* print("Anagram")
* else:
* print("Not an anagram")
* ✅ **Explanation**: Anagrams have the same characters in any order.
* Would you like more such questions (like palindrome, reverse number, pattern printing, etc.)?
* Great! Here are **10 classic logic-based Python programs** that take **user input**, with **answers and explanations**:
* **1. Factorial of a Number**
* num = int(input("Enter a number: "))
* factorial = 1
* for i in range(1, num + 1):
* factorial \*= i
* print("Factorial:", factorial)
* **Explanation**: Multiply all numbers from 1 to num.
* **2. Fibonacci Series**
* n = int(input("Enter number of terms: "))
* a, b = 0, 1
* for \_ in range(n):
* print(a, end=" ")
* a, b = b, a + b
* **Explanation**: Generate n terms of the Fibonacci sequence using a loop.
* **3. Prime Number Check**
* num = int(input("Enter a number: "))
* if num > 1:
* for i in range(2, num):
* if num % i == 0:
* print("Not Prime")
* break
* else:
* print("Prime")
* else:
* print("Not Prime")
* **Explanation**: A number is prime if it has no divisors other than 1 and itself.
* **4. Armstrong Number Check**
* num = int(input("Enter a number: "))
* sum = 0
* temp = num
* while temp > 0:
* digit = temp % 10
* sum += digit \*\* len(str(num))
* temp //= 10
* print("Armstrong" if sum == num else "Not Armstrong")
* **Explanation**: Armstrong number = sum of its digits raised to the power of the number of digits.
* **5. Anagram Checker**
* s1 = input("Enter first string: ").lower()
* s2 = input("Enter second string: ").lower()
* if sorted(s1) == sorted(s2):
* print("Anagram")
* else:
* print("Not Anagram")
* **Explanation**: Anagrams have the same letters in different order.
* **6. Palindrome Check**
* text = input("Enter a string: ")
* if text == text[::-1]:
* print("Palindrome")
* else:
* print("Not Palindrome")
* **Explanation**: A palindrome reads the same forwards and backwards.
* **7. Sum of Digits**
* num = int(input("Enter a number: "))
* total = 0
* while num > 0:
* total += num % 10
* num //= 10
* print("Sum of digits:", total)
* **Explanation**: Use modulo and division to extract and sum digits.
* **8. Reverse a Number**
* num = int(input("Enter a number: "))
* rev = 0
* while num > 0:
* rev = rev \* 10 + num % 10
* num //= 10
* print("Reversed number:", rev)
* **Explanation**: Extract digits and build the number in reverse.
* **9. LCM of Two Numbers**
* a = int(input("Enter first number: "))
* b = int(input("Enter second number: "))
* max\_num = max(a, b)
* while True:
* if max\_num % a == 0 and max\_num % b == 0:
* print("LCM:", max\_num)
* break
* max\_num += 1
* **Explanation**: Least Common Multiple = smallest number divisible by both.
* **10. GCD of Two Numbers**
* a = int(input("Enter first number: "))
* b = int(input("Enter second number: "))
* while b:
* a, b = b, a % b
* print("GCD:", a)
* **Explanation**: Euclidean algorithm for Greatest Common Divisor.
* Would you like **10 more logic-based problems** (like perfect number, magic number, Harshad number, etc.)?
* Here is a set of **Python logic-based programs with user input**, including:
* **✅ 1. Factorial of a number**
* num = int(input("Enter a number: "))
* fact = 1
* for i in range(1, num + 1):
* fact \*= i
* print("Factorial:", fact)
* **✅ 2. Fibonacci series up to N terms**
* n = int(input("Enter number of terms: "))
* a, b = 0, 1
* print("Fibonacci series:", end=" ")
* for \_ in range(n):
* print(a, end=" ")
* a, b = b, a + b
* **✅ 3. Prime number check**
* num = int(input("Enter a number: "))
* if num > 1:
* for i in range(2, num):
* if num % i == 0:
* print("Not a prime")
* break
* else:
* print("Prime number")
* else:
* print("Not a prime")
* **✅ 4. Armstrong number (3-digit)**
* num = int(input("Enter a 3-digit number: "))
* sum = 0
* temp = num
* while temp > 0:
* digit = temp % 10
* sum += digit \*\* 3
* temp //= 10
* if num == sum:
* print("Armstrong number")
* else:
* print("Not an Armstrong number")
* **✅ 5. Anagram check**
* str1 = input("Enter first string: ")
* str2 = input("Enter second string: ")
* if sorted(str1.lower()) == sorted(str2.lower()):
* print("Anagram")
* else:
* print("Not an Anagram")
* **✅ 6. Pattern - Right-Angled Triangle**
* rows = int(input("Enter number of rows: "))
* for i in range(1, rows + 1):
* print("\*" \* i)
* **✅ 7. Pattern - Pyramid**
* rows = int(input("Enter number of rows: "))
* for i in range(1, rows + 1):
* print(" " \* (rows - i) + "\* " \* i)
* **✅ 8. Reverse a number**
* num = int(input("Enter a number: "))
* rev = 0
* while num > 0:
* rev = rev \* 10 + num % 10
* num //= 10
* print("Reversed number:", rev)
* **✅ 9. Palindrome number check**
* num = int(input("Enter a number: "))
* original = num
* rev = 0
* while num > 0:
* rev = rev \* 10 + num % 10
* num //= 10
* if original == rev:
* print("Palindrome")
* else:
* print("Not a Palindrome")
* **✅ 10. Count vowels in a string**
* text = input("Enter a string: ").lower()
* vowels = "aeiou"
* count = sum(1 for char in text if char in vowels)
* print("Vowel count:", count)
* Would you like more logic-based programs (like patterns, conversions, etc.) or advanced puzzles next?
* Perfect! Here are **10 Python logic-based questions using functions with user input**, ideal for practice and interviews:
* **✅ 1. Function to check factorial of a number**
* def factorial(n):
* result = 1
* for i in range(1, n + 1):
* result \*= i
* return result
* num = int(input("Enter a number: "))
* print("Factorial:", factorial(num))
* **✅ 2. Function to generate Fibonacci series up to N terms**
* def fibonacci(n):
* a, b = 0, 1
* for \_ in range(n):
* print(a, end=" ")
* a, b = b, a + b
* terms = int(input("Enter number of terms: "))
* fibonacci(terms)
* **✅ 3. Function to check if a number is prime**
* def is\_prime(n):
* if n <= 1:
* return False
* for i in range(2, n):
* if n % i == 0:
* return False
* return True
* num = int(input("Enter a number: "))
* print("Prime" if is\_prime(num) else "Not Prime")
* **✅ 4. Function to check Armstrong number**
* def is\_armstrong(n):
* power = len(str(n))
* return n == sum(int(digit) \*\* power for digit in str(n))
* num = int(input("Enter a number: "))
* print("Armstrong" if is\_armstrong(num) else "Not Armstrong")
* **✅ 5. Function to check anagram**
* def is\_anagram(str1, str2):
* return sorted(str1.lower()) == sorted(str2.lower())
* s1 = input("Enter first string: ")
* s2 = input("Enter second string: ")
* print("Anagram" if is\_anagram(s1, s2) else "Not Anagram")
* **✅ 6. Function to print triangle pattern**
* def triangle\_pattern(rows):
* for i in range(1, rows + 1):
* print("\*" \* i)
* n = int(input("Enter number of rows: "))
* triangle\_pattern(n)
* **✅ 7. Function to reverse a number**
* def reverse\_number(n):
* rev = 0
* while n > 0:
* rev = rev \* 10 + n % 10
* n //= 10
* return rev
* num = int(input("Enter a number: "))
* print("Reversed:", reverse\_number(num))
* **✅ 8. Function to check palindrome number**
* def is\_palindrome(n):
* return str(n) == str(n)[::-1]
* num = int(input("Enter a number: "))
* print("Palindrome" if is\_palindrome(num) else "Not Palindrome")
* **✅ 9. Function to count vowels in a string**
* def count\_vowels(text):
* vowels = "aeiou"
* return sum(1 for char in text.lower() if char in vowels)
* string = input("Enter a string: ")
* print("Vowel count:", count\_vowels(string))
* **✅ 10. Function to calculate sum of digits**
* def sum\_of\_digits(n):
* total = 0
* while n > 0:
* total += n % 10
* n //= 10
* return total
* num = int(input("Enter a number: "))
* print("Sum of digits:", sum\_of\_digits(num))
* Would you like a set of **function-based pattern programs**, **number conversion**, or **recursion examples** next?
* Great! Here's a collection of **10 logic-based Python programs using recursion** that involve **user input**. These examples will help you understand how recursion works for problems like factorial, Fibonacci, palindrome, GCD, etc.
* **✅ 1. Recursive function to calculate factorial**
* def factorial(n):
* if n == 0 or n == 1:
* return 1
* return n \* factorial(n - 1)
* num = int(input("Enter a number: "))
* print("Factorial:", factorial(num))
* **✅ 2. Recursive Fibonacci**
* def fibonacci(n):
* if n <= 1:
* return n
* return fibonacci(n - 1) + fibonacci(n - 2)
* terms = int(input("Enter number of terms: "))
* for i in range(terms):
* print(fibonacci(i), end=" ")
* **✅ 3. Recursive function to reverse a string**
* def reverse\_string(s):
* if len(s) == 0:
* return s
* return reverse\_string(s[1:]) + s[0]
* text = input("Enter a string: ")
* print("Reversed:", reverse\_string(text))
* **✅ 4. Recursive function to calculate sum of digits**
* def sum\_digits(n):
* if n == 0:
* return 0
* return n % 10 + sum\_digits(n // 10)
* num = int(input("Enter a number: "))
* print("Sum of digits:", sum\_digits(num))
* **✅ 5. Recursive function to calculate power**
* def power(base, exp):
* if exp == 0:
* return 1
* return base \* power(base, exp - 1)
* b = int(input("Enter base: "))
* e = int(input("Enter exponent: "))
* print("Result:", power(b, e))
* **✅ 6. Recursive function to find GCD**
* def gcd(a, b):
* if b == 0:
* return a
* return gcd(b, a % b)
* x = int(input("Enter first number: "))
* y = int(input("Enter second number: "))
* print("GCD:", gcd(x, y))
* **✅ 7. Recursive function to check palindrome string**
* def is\_palindrome(s):
* if len(s) <= 1:
* return True
* if s[0] != s[-1]:
* return False
* return is\_palindrome(s[1:-1])
* word = input("Enter a string: ")
* print("Palindrome" if is\_palindrome(word) else "Not Palindrome")
* **✅ 8. Recursive function to count digits**
* def count\_digits(n):
* if n == 0:
* return 0
* return 1 + count\_digits(n // 10)
* num = int(input("Enter a number: "))
* print("Digit count:", count\_digits(num))
* **✅ 9. Recursive function to print numbers from N to 1**
* def print\_desc(n):
* if n == 0:
* return
* print(n, end=" ")
* print\_desc(n - 1)
* num = int(input("Enter a number: "))
* print\_desc(num)
* **✅ 10. Recursive function to convert decimal to binary**
* def decimal\_to\_binary(n):
* if n == 0:
* return ""
* return decimal\_to\_binary(n // 2) + str(n % 2)
* num = int(input("Enter a decimal number: "))
* binary = decimal\_to\_binary(num)
* print("Binary:", binary if binary else "0")
* Would you like more recursion questions, or should we move on to **pattern printing with recursion**, **list recursion**, or **advanced recursion puzzles**?

Perfect! Here's a **clear and complete set of OOP-based Python examples** that demonstrate:

* Class and Object
* Inheritance
* Polymorphism
* Encapsulation
* Abstraction

Each with a **simple user-friendly explanation and code example**.👇

**✅ 1. Class and Object**

class Student:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

def display(self):

print(f"Name: {self.name}, Age: {self.age}")

# Creating object

s1 = Student("Anitta", 21)

s1.display()

👉 **Explanation**: This defines a class Student and creates an object s1. The object uses the class method display().

**✅ 2. Inheritance**

class Animal:

def speak(self):

print("Animal speaks")

class Dog(Animal): # Dog inherits from Animal

def bark(self):

print("Dog barks")

d = Dog()

d.speak() # from parent

d.bark() # from child

👉 **Explanation**: Dog inherits the speak() method from Animal (parent), and adds its own method bark().

**✅ 3. Polymorphism**

class Bird:

def sound(self):

print("Bird sound")

class Sparrow(Bird):

def sound(self):

print("Chirp Chirp")

class Owl(Bird):

def sound(self):

print("Hoot Hoot")

def make\_sound(bird):

bird.sound()

b1 = Sparrow()

b2 = Owl()

make\_sound(b1)

make\_sound(b2)

👉 **Explanation**: Both Sparrow and Owl override sound() differently. The same method name works in multiple forms = **polymorphism**.

**✅ 4. Encapsulation**

class BankAccount:

def \_\_init\_\_(self, balance):

self.\_\_balance = balance # private variable

def deposit(self, amount):

self.\_\_balance += amount

def get\_balance(self):

return self.\_\_balance

acc = BankAccount(1000)

acc.deposit(500)

print("Balance:", acc.get\_balance())

# print(acc.\_\_balance) ❌ Error: Cannot access private variable

👉 **Explanation**: \_\_balance is private (encapsulation). It can only be accessed/modified via public methods like deposit() and get\_balance().

**✅ 5. Abstraction using ABC module**

from abc import ABC, abstractmethod

class Shape(ABC):

@abstractmethod

def area(self):

pass

class Circle(Shape):

def \_\_init\_\_(self, radius):

self.radius = radius

def area(self):

return 3.14 \* self.radius \* self.radius

c = Circle(5)

print("Area:", c.area())

👉 **Explanation**: Shape is an abstract class with an abstract method area(). It **forces** child class Circle to implement area() — this is **abstraction**.

**⚡ Summary Table:**

| **Concept** | **Purpose** | **Shown in Code** |
| --- | --- | --- |
| Class/Object | Creating reusable templates | Student example |
| Inheritance | Reusing code from parent class | Dog example |
| Polymorphism | Same method, different behavior | sound() example |
| Encapsulation | Protect data (private) | BankAccount |
| Abstraction | Hide complexity, enforce structure | Shape example |

Would you like:

* Real-world OOP example (like employee, hospital, ATM)?
* Or mini-projects using OOP (e.g., student management or quiz system)?

Here’s a rich collection of **OOP-based Python programs** showing **multiple combinations** of concepts like:

* Class + Object
* Inheritance + Polymorphism
* Encapsulation + Abstraction
* Recursion in classes
* Function + OOP mix
* Real-world inspired designs (ATM, Library, Student System)

**✅ 1. Student Grade System (Class + Object + Function)**

class Student:

def \_\_init\_\_(self, name, marks):

self.name = name

self.marks = marks

def grade(self):

if self.marks >= 90:

return "A"

elif self.marks >= 75:

return "B"

elif self.marks >= 60:

return "C"

else:

return "F"

s = Student("Anitta", 85)

print(f"{s.name}'s Grade: {s.grade()}")

**✅ 2. Bank System (Encapsulation + Inheritance)**

class Account:

def \_\_init\_\_(self, balance):

self.\_\_balance = balance

def deposit(self, amount):

self.\_\_balance += amount

def get\_balance(self):

return self.\_\_balance

class SavingsAccount(Account):

def add\_interest(self):

interest = self.get\_balance() \* 0.05

self.deposit(interest)

acc = SavingsAccount(1000)

acc.add\_interest()

print("Balance with interest:", acc.get\_balance())

**✅ 3. Calculator using Class and Recursion**

class Calculator:

def factorial(self, n):

if n == 0:

return 1

return n \* self.factorial(n - 1)

c = Calculator()

num = int(input("Enter number: "))

print("Factorial:", c.factorial(num))

**✅ 4. Polymorphism with Inheritance**

class Vehicle:

def fuel(self):

print("Generic Fuel")

class Car(Vehicle):

def fuel(self):

print("Petrol")

class Truck(Vehicle):

def fuel(self):

print("Diesel")

vehicles = [Car(), Truck()]

for v in vehicles:

v.fuel()

**✅ 5. Abstract Class with Polymorphism (Abstraction + Polymorphism)**

from abc import ABC, abstractmethod

class Employee(ABC):

@abstractmethod

def get\_salary(self):

pass

class Manager(Employee):

def get\_salary(self):

return 50000

class Developer(Employee):

def get\_salary(self):

return 40000

emp\_list = [Manager(), Developer()]

for emp in emp\_list:

print("Salary:", emp.get\_salary())

**✅ 6. Library Management (OOP Mini Project)**

class Book:

def \_\_init\_\_(self, title, author):

self.title = title

self.author = author

self.available = True

def display(self):

print(f"{self.title} by {self.author} - {'Available' if self.available else 'Not Available'}")

class Library:

def \_\_init\_\_(self):

self.books = []

def add\_book(self, b):

self.books.append(b)

def borrow\_book(self, title):

for book in self.books:

if book.title == title and book.available:

book.available = False

print(f"You borrowed '{title}'")

return

print("Book not available.")

# Usage

lib = Library()

lib.add\_book(Book("Python 101", "W3 Schools"))

lib.add\_book(Book("OOP in Python", "John Doe"))

lib.borrow\_book("Python 101")

lib.borrow\_book("Python 101")

**✅ 7. ATM Machine Simulation (Encapsulation + Menu + Object)**

class ATM:

def \_\_init\_\_(self, balance):

self.\_\_balance = balance

def withdraw(self, amount):

if amount > self.\_\_balance:

print("Insufficient balance")

else:

self.\_\_balance -= amount

print("Withdrawn:", amount)

def deposit(self, amount):

self.\_\_balance += amount

print("Deposited:", amount)

def show\_balance(self):

print("Balance:", self.\_\_balance)

atm = ATM(5000)

while True:

print("\n1. Deposit\n2. Withdraw\n3. Balance\n4. Exit")

ch = int(input("Enter choice: "))

if ch == 1:

amt = int(input("Amount to deposit: "))

atm.deposit(amt)

elif ch == 2:

amt = int(input("Amount to withdraw: "))

atm.withdraw(amt)

elif ch == 3:

atm.show\_balance()

else:

break

**✅ 8. Student Inheritance + Encapsulation + Function**

class Person:

def \_\_init\_\_(self, name, age):

self.\_name = name

self.\_age = age

class Student(Person):

def \_\_init\_\_(self, name, age, roll):

super().\_\_init\_\_(name, age)

self.\_\_roll = roll

def show(self):

print(f"Name: {self.\_name}, Age: {self.\_age}, Roll No: {self.\_\_roll}")

s = Student("Anitta", 21, "A123")

s.show()

Would you like:

* More **real-world OOP projects**?
* Or tricky **MCQ or theory questions** on OOP?

Let me know!

Great! Here are **10 tough or tricky Python questions** for each:

**🔷 List - Tough/Tricky Questions**

1. Write a program to rotate a list to the right by k steps (user input).
2. Write a program to flatten a nested list without using recursion.
3. Write a program to split a list into all possible combinations of two non-empty lists.
4. Write a program to remove duplicates from a list without using set().
5. Write a program to count frequency of elements and return the top 3 most frequent.
6. Write a program to rearrange a list so that positive and negative numbers alternate.
7. Write a program to find the longest increasing sublist in a list.
8. Write a program to chunk a list into sublists of size k.
9. Write a program to find all indices of a specific element in a list.
10. Write a program to reverse a list in-place using slicing logic only.

**🔷 Tuple - Tough/Tricky Questions**

1. Write a program to convert a list of tuples into a dictionary, ignoring duplicates.
2. Write a program to count the number of tuples having a common first element.
3. Write a program to sort a list of tuples based on the second value in descending order.
4. Write a program to remove tuples of length 1 from a list of tuples.
5. Write a program to find the tuple with the maximum product in a list.
6. Write a program to merge two tuples element-wise (assume equal length).
7. Write a program to check if a tuple can be rearranged to be sorted.
8. Write a program to unpack a nested tuple of unknown depth (using recursion).
9. Write a program to group tuples based on their first element.
10. Write a program to replace the nth value of each tuple in a list of tuples with a user input.

**🔷 Set - Tough/Tricky Questions**

1. Write a program to remove all elements from one set that are present in another set.
2. Write a program to count the number of unique elements across multiple sets.
3. Write a program to find symmetric difference across n sets.
4. Write a program to find the longest consecutive sequence in a set of integers.
5. Write a program to check if all characters in a string are unique using sets.
6. Write a program to find duplicate elements between two lists using sets.
7. Write a program to find sets that are subsets of a given main set from a list of sets.
8. Write a program to group a list of strings by anagram sets.
9. Write a program to create a set of characters that are vowels in multiple strings.
10. Write a program to check if two sets are disjoint without using isdisjoint().

**🔷 Dictionary - Tough/Tricky Questions**

1. Write a program to group a list of words by their first letter using a dictionary.
2. Write a program to invert a dictionary with non-unique values.
3. Write a program to merge multiple dictionaries and sum values for common keys.
4. Write a program to sort a dictionary by values in descending order.
5. Write a program to filter a dictionary based on a condition on its values.
6. Write a program to find the key with the longest string value.
7. Write a program to create a frequency dictionary from a list of words.
8. Write a program to flatten a nested dictionary into dot-separated keys.
9. Write a program to check if two dictionaries are deeply equal.
10. Write a program to build a dictionary from two lists using list comprehension.

Would you like full solutions with explanations for these?