100 Real-World Python Coding Questions (Fresher to Intermediate)

# 1. Python Basics & Setup

1. 1. Write a script to display "Welcome to Python World!" using print().
2. 2. Create a Python program to accept your name and age, then display them.
3. 3. Accept two numbers and print their sum, product, and average.
4. 4. Accept your system path using input() and print each folder name on a new line.
5. 5. Use command-line arguments to read a user's name and print a greeting.
6. 6. Write a Python program to find the area of a circle. Accept radius from user.
7. 7. Accept temperature in Celsius and convert it to Fahrenheit.
8. 8. Write a script to swap two numbers without using a third variable.
9. 9. Accept 3 numbers from the user and print the largest.
10. 10. Write a script to check whether Python is properly installed and display the version.

# 2. Data Types & Variables

1. 1. Accept 5 float numbers from the user and display their types.
2. 2. Write a program to demonstrate all types of literals with examples.
3. 3. Convert a string to integer, float, and boolean with examples.
4. 4. Write a script that accepts your name and age and prints a welcome message with formatted output.
5. 5. Accept two complex numbers and perform addition and multiplication.
6. 6. Create a program that calculates the EMI based on loan amount, interest, and tenure.
7. 7. Write a program to show variable scope (local vs global).
8. 8. Demonstrate the use of type(), id(), and isinstance() with suitable examples.
9. 9. Write a program to accept a hexadecimal number and convert it to decimal.
10. 10. Illustrate implicit and explicit type casting.

# 3. Conditional Statements

1. 1. Accept a number and check whether it is even or odd.
2. 2. Accept a year and check if it’s a leap year.
3. 3. Create a simple calculator using if-elif-else statements.
4. 4. Accept a character and check if it is a vowel or consonant.
5. 5. Check whether a number is positive, negative, or zero.
6. 6. Accept three side lengths and check whether they form a valid triangle.
7. 7. Accept a password and validate its strength based on length and characters.
8. 8. Write a program to check if a given year is a century year.
9. 9. Check whether a person is eligible to vote, drive, or both based on age.
10. 10. Accept marks for 5 subjects and calculate total, average, and grade.

# 4. Loops

1. 1. Print all even numbers between 1 to 100.
2. 2. Print the multiplication table of a number entered by the user.
3. 3. Accept a number and calculate its factorial.
4. 4. Check whether a number is prime.
5. 5. Print Fibonacci series up to n terms.
6. 6. Count the number of digits in an integer.
7. 7. Find the sum of digits of a given number.
8. 8. Write a program to reverse a number using while loop.
9. 9. Print the following pattern using nested loops:  
   \*  
   \* \*  
   \* \* \*  
   \* \* \* \*
10. 10. Print all prime numbers between 1 to 100.

# 5. Strings

1. 1. Accept a string and count vowels and consonants.
2. 2. Accept a sentence and reverse it.
3. 3. Accept a string and check if it’s a palindrome.
4. 4. Accept a string and remove all special characters.
5. 5. Accept two strings and check if they are anagrams.
6. 6. Find the frequency of each character in a string.
7. 7. Convert a string to title case without using built-in function.
8. 8. Count the number of words in a sentence.
9. 9. Replace all spaces with hyphens in a sentence.
10. 10. Remove duplicate characters from a string.

# 6. Lists & Tuples

1. 1. Accept 10 numbers from the user and store in a list.
2. 2. Find the maximum and minimum in a list without using max() or min().
3. 3. Remove duplicates from a list.
4. 4. Sort a list of integers without using sort().
5. 5. Count even and odd numbers in a list.
6. 6. Accept a list and find the second highest element.
7. 7. Create a nested list and access individual elements.
8. 8. Create a tuple with 5 elements and convert it to a list.
9. 9. Check whether a list contains any duplicate values.
10. 10. Accept n numbers and create a list comprehension that returns only squares of even numbers.

# 7. Dictionary & Sets

1. 1. Create a dictionary with student names and marks and display them.
2. 2. Accept a string and count the frequency of each word using a dictionary.
3. 3. Create a dictionary from two lists using zip().
4. 4. Find common and unique elements between two sets.
5. 5. Merge two dictionaries into one.
6. 6. Write a program to count frequency of each character using set and dictionary.
7. 7. Remove all duplicates from a list using a set.
8. 8. Accept dictionary input from the user and display sorted by keys.
9. 9. Create a nested dictionary to store employee info (name, age, salary).
10. 10. Accept a sentence and print word count using dictionary.

# 8. Functions & Modules

1. 1. Write a function to calculate factorial of a number.
2. 2. Write a recursive function to compute nth Fibonacci number.
3. 3. Create a function with default arguments.
4. 4. Demonstrate global vs local variable with an example.
5. 5. Write a function that accepts variable number of arguments and returns their sum.
6. 6. Write a program using lambda to sort a list of tuples based on second element.
7. 7. Use map() to square all elements in a list.
8. 8. Use filter() to remove all odd numbers from a list.
9. 9. Use reduce() to compute product of all elements in a list.
10. 10. Create a user-defined module for temperature conversion and import it.

# 9. OOP in Python

1. 1. Create a class Student with attributes name, age, and grade. Create methods to set and display them.
2. 2. Write a class Rectangle to compute area and perimeter.
3. 3. Create a class with private variables and access them using getter/setter.
4. 4. Create a base class Vehicle and a derived class Car with inheritance.
5. 5. Create a class that demonstrates method overloading using default arguments.
6. 6. Demonstrate the use of constructor and destructor.
7. 7. Write a class to demonstrate class vs instance variables.
8. 8. Implement a bank system where user can deposit, withdraw, and check balance.
9. 9. Implement polymorphism using method overriding.
10. 10. Demonstrate data abstraction using ABC (Abstract Base Class).

# 10. File Handling & Exception

1. 1. Write a program to read a text file and count words.
2. 2. Write a program to copy contents of one file to another.
3. 3. Append data to a file using Python.
4. 4. Read a file line by line and print each with line number.
5. 5. Handle file not found exception while reading a file.
6. 6. Write a program to search a word in a file and count its occurrences.
7. 7. Write a program to serialize and deserialize a dictionary using pickle.
8. 8. Create a log file that logs all user input from a script.
9. 9. Accept student data and store in a file.
10. 10. Demonstrate try-except-finally block with division operation.