Python Programming Practice Questions

100 Comprehensive Programming Problems

Difficulty Levels: Beginner to Advanced
Topics Covered: Basic Python, OOP, Data Structures, Algorithms, File Handling, Modules, Web Scraping, Data Science

---

1. Basic Python Programs (Questions 1-25)

1.1. Basic Input/Output and Operations

Question 1: Write a Python program that takes two numbers as input and displays their sum, difference, product, and quotient with proper exception handling.

Question 2: Create a program that reads a character and displays whether it's a vowel, consonant, digit, or special character.

Question 3: Implement a program that converts temperature between Celsius and Fahrenheit with menu-driven approach.

Question 4: Write a program to swap two numbers using a third variable and without using a third variable.

Question 5: Create a program that calculates simple and compound interest with user inputs.

Question 6: Write a program to find the largest among three numbers using conditional statements.

Question 7: Implement a program that checks if a year is a leap year or not.

Question 8: Create a calculator using if-elif that handles basic arithmetic operations.

Question 9: Write a program that determines the type of triangle based on sides (equilateral, isosceles, scalene).

Question 10: Implement a program that calculates electricity bill based on units consumed with slab rates.

1.2. Looping Structures

Question 11: Write a program to print all prime numbers between 1 and 100.

Question 12: Create a program to find the sum of all even numbers between 1 and n.

Question 13: Implement a program to print various patterns (pyramid, diamond, number patterns).

Question 14: Write a program to find the GCD and LCM of two numbers using Euclidean algorithm.

Question 15: Create a program that reverses a number and checks if it's a palindrome.

Question 16: Implement a program to find all Armstrong numbers between 1 and 1000.

Question 17: Write a program to calculate the sum of series: 1 + 1/2 + 1/3 + ... + 1/n.

Question 18: Create a program to convert decimal to binary, octal, and hexadecimal.

Question 19: Implement a program to find the factorial of a number using recursion.

Question 20: Write a program to generate Fibonacci series up to n terms using both iteration and recursion.

Question 21: Create a program to find the sum of digits of a number until it becomes single digit.

Question 22: Implement a program to print multiplication table for a given number.

Question 23: Write a program to find all perfect numbers in a given range.

Question 24: Create a program to calculate the sum of squares of first n natural numbers.

Question 25: Implement a program to check if a string is palindrome or not.

---

2. Functions and Data Structures (Questions 26-45)

2.1. Functions

Question 26: Write a function that checks if a number is prime and use it to find all prime numbers in a range.

Question 27: Create a function to calculate power of a number using recursion.

Question 28: Implement a function that returns the reverse of a number.

Question 29: Write a function with default arguments and demonstrate its usage.

Question 30: Create a recursive function to find the sum of digits of a number.

Question 31: Implement a function that takes variable-length arguments (*args and **kwargs).

Question 32: Write a lambda function to find square and cube of a number.

Question 33: Create a decorator function to measure execution time of other functions.

Question 34: Implement a generator function to yield Fibonacci numbers.

Question 35: Write a function that uses map(), filter(), and reduce().

2.2. Lists and Tuples

Question 36: Write a program to find the largest and smallest element in a list.

Question 37: Implement a program to sort a list using bubble sort algorithm.

Question 38: Create a program to search an element in a list using linear and binary search.

Question 39: Write a program to merge two sorted lists into one sorted list.

Question 40: Implement a program to find the frequency of each element in a list.

Question 41: Create a program to rotate a list by k positions.

Question 42: Write a program to find the second largest element in a list.

Question 43: Implement a program to remove duplicates from a list.

Question 44: Create a program to find the union and intersection of two lists.

Question 45: Write a program to implement list comprehension for various operations.

---

3. Dictionaries and Sets (Questions 46-55)

Question 46: Create a program to count word frequency in a text using dictionary.

Question 47: Implement a phone directory using dictionary with CRUD operations.

Question 48: Write a program to merge two dictionaries.

Question 49: Create a program to find common elements in multiple lists using sets.

Question 50: Implement a program to remove duplicates using sets.

Question 51: Write a program to demonstrate set operations (union, intersection, difference).

Question 52: Create a program to find the most frequent element in a list using dictionary.

Question 53: Implement a program to group similar items using dictionary.

Question 54: Write a program to sort a dictionary by key and by value.

Question 55: Create a program to invert a dictionary (swap keys and values).

---

4. Object-Oriented Programming (Questions 56-70)

4.1. Classes and Objects

Question 56: Create a Student class with attributes like name, roll number, marks, and methods to calculate grade and display information.

Question 57: Implement a Bank Account class with deposit, withdraw, and balance inquiry functionality.

Question 58: Write a Circle class with methods to calculate area, circumference, and diameter.

Question 59: Create a Rectangle class with methods to calculate area, perimeter, and diagonal.

Question 60: Implement a Complex Number class with methods for arithmetic operations.

4.2. Advanced OOP

Question 61: Create a Book class with appropriate constructors and demonstrate inheritance.

Question 62: Implement a Date class with validation for day, month, year and methods to compare dates.

Question 63: Write a Stack class with push, pop, and peek operations.

Question 64: Create a base class Shape with derived classes Circle, Rectangle, Triangle implementing area calculation.

Question 65: Implement a Person class with Student and Teacher as derived classes.

Question 66: Write a program demonstrating method overloading and operator overloading.

Question 67: Create an abstract base class using ABC module.

Question 68: Implement multiple inheritance with practical example.

Question 69: Write a program demonstrating property decorators and getters/setters.

Question 70: Create a class with class methods and static methods.

---

5. File Handling (Questions 71-80)

Question 71: Write a program to create a file and write student information to it.

Question 72: Implement a program to read from a file and display its contents.

Question 73: Create a program to copy contents from one file to another.

Question 74: Write a program to count words, lines, and characters in a file.

Question 75: Implement a student record system using file handling (add, search, delete records).

Question 76: Create a program to merge two files into a third file.

Question 77: Write a program to find and replace specific text in a file.

Question 78: Implement a program that handles CSV files for data storage.

Question 79: Create a program to maintain inventory using JSON file.

Question 80: Write a program that demonstrates reading and writing binary files.

---

6. Exception Handling (Questions 81-85)

Question 81: Write a program to handle division by zero and value errors.

Question 82: Implement custom exception classes for banking application.

Question 83: Create a program that handles file not found exception.

Question 84: Write a program demonstrating try-except-else-finally block.

Question 85: Implement a program with multiple exception handlers.

---

7. Modules and Packages (Questions 86-90)

Question 86: Create a custom module for mathematical operations and import it.

Question 87: Write a program that uses built-in modules like math, random, datetime.

Question 88: Implement a package with multiple modules and demonstrate its usage.

Question 89: Create a program that uses third-party packages (install using pip).

Question 90: Write a program demonstrating the use of **name** == "**main**".

---

8. Advanced Python Concepts (Questions 91-100)

Question 91: Implement multithreading for parallel processing.

Question 92: Create a program using multiprocessing for CPU-intensive tasks.

Question 93: Write a program demonstrating context managers using with statement.

Question 94: Implement a program using regular expressions for pattern matching.

Question 95: Create a simple web scraper using requests and BeautifulSoup.

Question 96: Write a program to connect to database using SQLite.

Question 97: Implement a REST API client using requests module.

Question 98: Create a program using pandas for data analysis.

Question 99: Write a program using matplotlib for data visualization.

Question 100: Implement a complete student management system with all OOP concepts, file handling, exception handling, and database connectivity.