Python Lists - Notes, Interview Questions, and Exercises

Notes on Python Lists

1. Introduction to Lists

- A list is a mutable, ordered sequence of elements.
- Elements can be of any data type, including other lists (nested lists).
- Lists are defined using square brackets `[]`, and elements are separated by commas.

2. Creating Lists

- **Empty List**: `empty_list = []`
- **List with Elements**: `fruits = ['apple', 'banana', 'cherry']`

3. Accessing List Elements

- **Indexing**: Access elements using their index, e.g., `fruits[0]` returns `'apple'`.
- **Negative Indexing**: Access elements from the end using negative indices, e.g., `fruits[-1]` returns `'cherry'`.
- **Slicing**: Extract sublists using slicing, e.g., `fruits[1:3]` returns `['banana', 'cherry']`.

4. Modifying Lists

- **Changing Elements**: `fruits[1] = 'orange'` changes the second element to `'orange'`.
- **Adding Elements**:
  - `append()`: Adds a single element to the end, e.g., `fruits.append('grape')`.
  - `insert()`: Adds an element at a specific index, e.g., `fruits.insert(1, 'kiwi')`.
  - `extend()`: Adds multiple elements, e.g., `fruits.extend(['mango', 'pineapple'])`.
- **Removing Elements**:
  - `remove()`: Removes the first occurrence of an element, e.g., `fruits.remove('banana')`.
  - `pop()`: Removes an element at a specific index, e.g., `fruits.pop(1)`.
  - `clear()`: Removes all elements from the list.

5. List Operations

- **Concatenation**: Combine lists using the `+` operator, e.g., `fruits + ['pear', 'peach']`.
- **Repetition**: Repeat lists using the `*` operator, e.g., `fruits * 2`.
- **Membership**: Check if an element exists using `in`, e.g., `'apple' in fruits`.
- **Length**: Get the number of elements using `len(fruits)`.

6. List Methods

- `sort()`: Sorts the list in ascending order.
- `reverse()`: Reverses the order of the list.
- `index()`: Returns the index of the first occurrence of an element.

- `count()`: Returns the number of occurrences of an element.

7. List Comprehensions

- A concise way to create lists, e.g., `[x**2 for x in range(10)]` creates a list of squares.

8. Nested Lists

- Lists within lists, e.g., `matrix = [[1, 2], [3, 4], [5, 6]]`.
- Accessing elements in nested lists, e.g., `matrix[0][1]` returns `2`.

9. Copying Lists

- **Shallow Copy**: Using slicing `[:]` or `list.copy()`.
- **Deep Copy**: Using the `copy` module's `deepcopy()`.

10. Iterating Through Lists

- Using loops to iterate over elements, e.g., `for fruit in fruits: print(fruit)`.

Interview Questions on Python Lists

Basic Questions

- What are lists in Python, and how do they differ from arrays?
- How do you add and remove elements from a list in Python?
- Explain list slicing with examples.

Intermediate Questions

- How would you find the second largest number in a list?
- What is list comprehension? Provide examples.
- How would you flatten a nested list?

Advanced Questions

- Explain the difference between shallow and deep copy in lists.
- How can you remove duplicates from a list without using sets?
- Discuss the time complexity of common list operations.

Exercises on Python Lists

Basic Exercises

- Create a list of the first 10 natural numbers.
- Find the sum of all elements in a list.
- Remove all occurrences of a specific element from a list.

Intermediate Exercises

- Write a program to rotate a list by n elements.
- Create a list of the squares of the numbers from 1 to 10 using list comprehension.
- Merge two sorted lists into a single sorted list.

Advanced Exercises

- Given a list of tuples representing pairs, sort the list by the second element in each tuple.
- Implement a function to find the longest common prefix in a list of strings.

- Write a program to find all unique triplets in a list that sum up to zero.
