List

1. How do you create an empty list?

empty\_list = []

1. How do you add an element to a list using indexing?

To add an element at a specific index, we can use list slicing. Also appending an element, we use the append() method.

|  |
| --- |
| a=[1,2,3]  a.append(4)  a.insert(4,5)  print(a)  output:  [1, 2, 3, 4, 5] |

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

First\_element= a[0]

1. How can you replace an element at a specific index in a list?

my\_list[1] = 100 # Replace element at index 1 with 100

1. How can you access the last element of a list without knowing its length?

last\_element = my\_list[-1]

1. How do you access a slice of a list (e.g., the first three elements)?

first\_three\_elements = my\_list[:3]

1. How can you concatenate two lists?

|  |
| --- |
| list1 = [1, 2, 3]  list2 = [4, 5, 6]  concatenated\_list = list1 + list2 |

1. How can you check if an element is in a list?

if 3 in my\_list:

print("3 is in the list")

1. How can you count the number of elements in a list?

list\_length = len(my\_list)

1. How do you reverse a list using indexing?

reversed\_list = my\_list[::-1]

1. How can you remove an element from a list by index?

We can use the del statement or the pop() method to remove an element by index:

|  |
| --- |
| # Using del  del my\_list[2]  # Using pop (also returns the removed element)  removed\_element = my\_list.pop(2) |

1. How do you find the index of a particular element in a list?

By using the index() method we can get the index of a particular element.

Eg: index\_of\_element = my\_list.index(3)

1. How do you check if a list is empty?

We can check the length of the list or use a direct boolean check.

|  |
| --- |
| if len(my\_list) == 0:  print("List is empty")  # or  if not my\_list:  print("List is empty") |

1. How can you access every second element in a list?

We can use slicing with a step:

Eg: every\_second\_element = my\_list[::2]

1. How can you access a list in reverse order without modifying it?

We can use slicing to access the list in reverse order.

Eg: reversed\_list = my\_list[::-1]

1. How would you remove all occurrences of a specific element from a list?

We can use a list comprehension to create a new list without unwanted elements.

Eg: my\_list = [x for x in my\_list if x != 3]

Or we also can use .remove() function

Eg: while 3 in my\_list:

my\_list.remove(3)

1. Given two lists, how would you find the common elements between them?

We can use list comprehension, or convert the lists to sets and find the common elements between them.

|  |
| --- |
| # Using list comprehension  common\_elements = [x for x in list1 if x in list2]  # Using set intersection (faster for large lists)  common\_elements = list(set(list1) & set(list2)) |

1. How would you create a new list that contains only the unique elements from an existing list?

We can use the set() to get unique elements.

Eg: unique\_elements = list(set(my\_list))

Also we can use the loop as:

|  |
| --- |
| unique\_elements = []  for x in my\_list:  if x not in unique\_elements:  unique\_elements.append(x) |

1. How would you flatten a list of lists into a single list (without using any built-in functions)?

We can use a nested loop to flatten the list as:

|  |
| --- |
| list\_of\_lists = [[1, 2], [3, 4], [5, 6]]  flattened\_list = []  for sublist in list\_of\_lists:  for item in sublist:  flattened\_list.append(item)  print(falttened\_list)  output:  [1, 2, 3, 4, 5, 6] |

1. How would you split a list into two parts at a given index?

|  |
| --- |
| index = 3  list1 = my\_list[:index]  list2 = my\_list[index:] |

1. How would you rotate a list by n positions to the left?

We can rotate a list using slicing:

|  |
| --- |
| def rotate\_left(my\_list, n):  return my\_list[n:] + my\_list[:n]  # Example usage:  my\_list = [1, 2, 3, 4, 5]  print(rotate\_left(my\_list, 2))  Output:  [3, 4, 5, 1, 2] |

1. How would you check if a list is a palindrome?

|  |
| --- |
| def is\_palindrome(my\_list):  return my\_list == my\_list[::-1]  my\_list = [1, 2, 3, 2, 1]  print(is\_palindrome(my\_list))  Output:  True |

1. How would you sort a list without using any built-in sort functions (e.g., using bubble sort)?

By using bubble sort:

|  |
| --- |
| def bubble\_sort(my\_list):  n = len(my\_list)  for i in range(n):  for j in range(0, n - i - 1):  if my\_list[j] > my\_list[j + 1]:  my\_list[j], my\_list[j + 1] = my\_list[j + 1], my\_list[j]  return my\_list  my\_list = [64, 25, 12, 22, 11]  print(bubble\_sort(my\_list))  Output:  [11, 12, 22, 25, 64] |

1. How would you find the second-largest element in a list without using built-in sorting functions?

|  |
| --- |
| def find\_second\_largest(my\_list):  largest = second\_largest = float('-inf')  for num in my\_list:  if num > largest:  second\_largest = largest  largest = num  elif largest > num > second\_largest:  second\_largest = num  return second\_largest  my\_list = [10, 5, 20, 8]  print(find\_second\_largest(my\_list))  Output:  10 |

1. How would you find pairs of elements in a list that sum up to a specific number?

|  |
| --- |
| def find\_pairs(my\_list, target\_sum):  pairs = []  seen = set()  for num in my\_list:  complement = target\_sum - num  if complement in seen:  pairs.append((complement, num))  seen.add(num)  return pairs  my\_list = [2, 4, 3, 5, 7]  target\_sum = 7  print(find\_pairs(my\_list, target\_sum))  Output:  [(4, 3), (5, 2)] |

1. How would you remove duplicates from a list while preserving the original order?

|  |
| --- |
| def remove\_duplicates(my\_list):  seen = set()  return [x for x in my\_list if not (x in seen or seen.add(x))]  my\_list = [1, 2, 2, 3, 4, 3, 5]  print(remove\_duplicates(my\_list))  Output:  [1, 2, 3, 4, 5] |

1. How would you merge two sorted lists into a single sorted list?

|  |
| --- |
| def merge\_and\_sort(list1, list2):  merged\_list = list1 + list2 # Concatenate the two lists  return sorted(merged\_list) # Sort the merged list  list1 = [1, 3, 5, 7]  list2 = [2, 4, 6, 8]  print(merge\_and\_sort(list1, list2))  Output:  [1, 2, 3, 4, 5, 6, 7, 8] |

1. How would you check if one list is a subset of another list?

|  |
| --- |
| def is\_subset(list1, list2):  return set(list1).issubset(set(list2))  list1 = [1, 2]  list2 = [1, 2, 3, 4]  print(is\_subset(list1, list2))  Output:  True |

1. How would you find the longest consecutive sequence in a list of numbers?

|  |
| --- |
| def longest\_consecutive\_sequence(nums):  nums = set(nums)  longest\_streak = 0  for num in nums:  if num - 1 not in nums:  current\_num = num  current\_streak = 1  while current\_num + 1 in nums:  current\_num += 1  current\_streak += 1  longest\_streak = max(longest\_streak, current\_streak)  return longest\_streak  my\_list = [100, 4, 200, 1, 3, 2]  print(longest\_consecutive\_sequence(my\_list)) # Output: 4 (sequence is 1, 2, 3, 4) |

1. How would you find the majority element in a list (the element that appears more than half the time)?

|  |
| --- |
| def find\_majority\_element(nums):      candidate = None      count = 0        # Step 1: Find a candidate for the majority element      for num in nums:          if count == 0:  # If counter is 0, pick a new candidate              candidate = num          if num == candidate:              count += 1  # Increment counter if the current element matches the candidate          else:              count -= 1  # Otherwise, decrement the counter      if nums.count(candidate) > len(nums) // 2:          return candidate      else:          return "No majority element"  nums = [3, 3, 4, 2, 3, 3, 2, 3, 3]  print(find\_majority\_element(nums))  Output:  3 |