Python Interview Question

1. Difference between ‘is’ and ‘==‘ in python

What will be the output of the following:

a = [1, 2 ,3]
b = [1, 2, 3]

print(a == b) #True logically both are same
print (a is b) #False because of different memory allocation

2. Count vowels in your name

```
def countVowel(name):
    vowels = "aeiouAEIOU" # Include uppercase vowels as well
    n = 0
    for char in name:
        if char in vowels:
            n += 1
    return n

# Test the function
print(countVowel("Abhirup")) # Output should be 3
```

3. Remove Duplicates from a list

```
def remove_duplicates(lst):
    seen = set()
    result = []
    for item in lst:
        if item not in seen:
            seen.add(item)
            result.append(item)
    return result

# Example usage:
my_list = [1, 2, 2, 3, 4, 1, 5]
print(remove_duplicates(my_list))  # Output: [1, 2, 3, 4, 5]

def remove_duplicates(lst):
    return list(dict.fromkeys(lst))

# Example usage:
my_list = [1, 2, 2, 3, 4, 1, 5]
print(remove_duplicates(my_list))  # Output: [1, 2, 3, 4, 5]
```

Merge two dictionaries:

```
merged_dict = dict1.update(dict2)
```

Reverse a string

```
def reverse_string(input_string):
    return input_string[::-1]
```

LST=[a,a,b,b,c,c]
Find count of occurrences in python and pyspark

```
def countLst(lst):
    count_dict = {}
    for item in lst:
        if item in count_dict:
            count_dict[item] += 1
        else:
            count_dict[item] = 1

    return count_dict

# Example usage:
LST = ['a', 'a', 'b', 'b', 'c', 'c']
counts = countLst(LST)

# Print the counts
for key, value in counts.items():
    print(f"{key}: {value}")
```

Input = [1, 2, [7, 9, [15, [12, 9,]] 18, 10]]
Output = [1, 2, 7, 9, 15, 12, 9, 18, 10]

```
def f_flatten(input_list):
    l = []
    for i in input_list:
        if(type(i) is list):
            l.extend(f_flatten(i))
        else:
            l.append(i)
    return l

print(f_flatten(input_list))
```

Given
arr =[3,3,3,3,4,0,4,4,4,7,7,7,7]
num = 7

Find first and last occurrence of num
output = [9,12]

```
def func(n, lst):
    indx = []
    for index, value in enumerate(lst):
        if value == n:
            indx.append(index)
    return [indx[0], indx[-1]]


lst1 = [3,3,3,3,4,0,4,4,4,7,7,7,7]
m = 7
print(func(m, lst1))

#Find Prime numbers from a range of values

def findPrime(n):
    prime = [1, 2]
    for i in range(n):
        if i not in prime and i%2 == 1:
            prime.append(i)
    print(prime)


# lst1 = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
findPrime(100)
```

Sort an Array with the inbuilt sort function

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]


arr = [64, 34, 25, 12, 22, 11, 90]

print(bubble_sort(arr))
```

Find second largest element from a list without sort function

```
def second_largest(lst):
```

```
    first_max = second_max = 0

    if len(lst) > 1:
        for num in lst:
            if num > first_max:
                second_max = first_max
                first_max = num
            elif num > second_max and num != first_max:
                second_max = num
        return second_max
    else:
        return "There is no second largest element."

# Example usage:
lst = [3, 2, 4, 5, 6, 8, 7, 19, 16, 6]
second_largest_element = second_largest(lst)
print("Second largest element:", second_largest_element)  #
Output: 16
```

Leetcode: https://leetcode.com/problems/best-time-to-buy-and-sell-stock/description/

```
prices = [7, 1, 5, 3, 6, 4]

def maxProfit(lst):
    buy = lst[0]
    max_profit = 0
    for i in range(1, len(lst)):
        if prices[i] < buy:
            buy = prices[i]
        profit = prices[i] - buy
        if profit > max_profit:
            max_profit = profit
    return max_profit

print(maxProfit(prices))
```

How to set task dependency in Airflow ?

Lists and Tuples:

1. **Find the maximum element in a list.**
2. **Calculate the sum of elements in a list.**
3. **Sort a list of integers in ascending order.**
4. **Reverse a list.**
5. **Check if a list is palindrome.**
6. **Count occurrences of an element in a list.**

7. **Remove duplicates from a list.**
8. **Find the second largest number in a list.**
9. **Merge two sorted lists into one sorted list.**
10. **Rotate elements in a list to the left by k positions.**

Strings:

11. **Reverse a string.**
12. **Check if a string is palindrome.**
13. **Count occurrences of each character in a string.**
14. **Check if two strings are anagrams.**
15. **Remove duplicate characters from a string.**
16. **Convert a string to lowercase/uppercase.**
17. **Find the first non-repeating character in a string.**
18. **Check if a string contains only digits.**
19. **Replace spaces in a string with %20 (URL encoding).**
20. **Reverse words in a string.**

Dictionaries:

21. **Merge two dictionaries.**
22. **Sort a dictionary by keys or values.**
23. **Check if a key exists in a dictionary.**
24. **Count frequency of each element in a list using a dictionary.**
25. **Remove a key from a dictionary.**
26. **Find keys with the maximum value in a dictionary.**
27. **Create a dictionary from two lists (keys and values).**
28. **Check if two dictionaries have the same keys.**
29. **Update values of one dictionary with values from another dictionary.**
30. **Convert a dictionary keys/values into a list.**

Sets:

31. **Check if two sets have any elements in common.**
32. **Find the intersection of two sets.**
33. **Find the union of two sets.**
34. **Check if a set is a subset of another set.**
35. **Remove duplicates from a list using a set.**
36. **Convert a list to a set.**
37. **Add elements to a set.**
38. **Remove elements from a set.**
39. **Clear all elements from a set.**
40. **Find the symmetric difference between two sets.**

Advanced:

41. **Implement a stack using a list (with push and pop operations).**
42. **Implement a queue using two stacks (with enqueue and dequeue operations).**
43. **Find the k-th smallest element in an unsorted list.**
44. **Implement binary search on a sorted list.**

45. **Calculate the Fibonacci sequence up to n terms.**
46. **Check if a given number is prime.**
47. **Calculate factorial of a number.**
48. **Implement bubble sort or selection sort on a list.**
49. **Find the longest common prefix in a list of strings.**
50. **Implement depth-first search (DFS) or breadth-first search (BFS) on a graph represented as an adjacency list.**