**[1. Two Sum](https://leetcode.com/problems/two-sum/)**

Given an array of integers nums and an integer target, return *indices of the two numbers such that they add up to target*.

You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

You can return the answer in any order.

**Example 1:**

**Input:** nums = [2,7,11,15], target = 9

**Output:** [0,1]

**Explanation:** Because nums[0] + nums[1] == 9, we return [0, 1].

**Example 2:**

**Input:** nums = [3,2,4], target = 6

**Output:** [1,2]

**Example 3:**

**Input:** nums = [3,3], target = 6

**Output:** [0,1]

**Constraints:**

* 2 <= nums.length <= 104
* -109 <= nums[i] <= 109
* -109 <= target <= 109
* **Only one valid answer exists.**

ANS：

class Solution:

    def twoSum(self, nums: List[int], target: int) -> List[int]:

        num\_dict = {}

        for i, num in enumerate(nums):

            complement = target - num

            if complement in num\_dict:

                return [num\_dict[complement], i]

            num\_dict[num] = i

**2.** [**Add Two Numbers**](https://leetcode.com/problems/add-two-numbers/)

Medium

Topics

Companies

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

**Example 1:**

一張含有 圓形, 行, 圖表, 螢幕擷取畫面 的圖片

自動產生的描述

**Input:** l1 = [2,4,3], l2 = [5,6,4]

**Output:** [7,0,8]

**Explanation:** 342 + 465 = 807.

**Example 2:**

**Input:** l1 = [0], l2 = [0]

**Output:** [0]

**Example 3:**

**Input:** l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9]

**Output:** [8,9,9,9,0,0,0,1]

ANS：

# Definition for singly-linked list.

# class ListNode:

#     def \_\_init\_\_(self, val=0, next=None):

#         self.val = val

#         self.next = next

class Solution:

    def addTwoNumbers(self, l1: Optional[ListNode], l2: Optional[ListNode]) -> Optional[ListNode]:

        head = ListNode()

        current = head

        carry = 0

        while (l1 != None or l2 != None or carry != 0):

            l1\_value = l1.val if l1 else 0

            l2\_value = l2.val if l2 else 0

            total = l1\_value + l2\_value + carry

            current.next = ListNode(total % 10)

            carry = total // 10

            # Move list pointers forward

            l1 = l1.next if l1 else None

            l2 = l2.next if l2 else None

            current = current.next

        return head.next

**3. DataFrame: employees**

+-------------+--------+

| Column Name | Type |

+-------------+--------+

| employee\_id | int |

| name | object |

| department | object |

| salary | int |

+-------------+--------+

Write a solution to display the **first 3**rowsof this DataFrame.

ANS：

import pandas as pd

def selectFirstRows(employees: pd.DataFrame) -> pd.DataFrame:

    return employees.head(3)

**4.** [**Score of a String**](https://leetcode.com/problems/score-of-a-string/)

You are given a string s. The **score** of a string is defined as the sum of the absolute difference between the **ASCII** values of adjacent characters.

Return the **score** ofs.

**Example 1:**

**Input:** s = "hello"

**Output:** 13

**Explanation:**

The **ASCII** values of the characters in s are: 'h' = 104, 'e' = 101, 'l' = 108, 'o' = 111. So, the score of s would be |104 - 101| + |101 - 108| + |108 - 108| + |108 - 111| = 3 + 7 + 0 + 3 = 13.

**Example 2:**

**Input:** s = "zaz"

**Output:** 50

**Explanation:**

The **ASCII** values of the characters in s are: 'z' = 122, 'a' = 97. So, the score of s would be |122 - 97| + |97 - 122| = 25 + 25 = 50.

ANS：

class Solution:

    def scoreOfString(self, s: str) -> int:

        score = 0

        for i in range(len(s) - 1):

            score += abs(ord(s[i]) - ord(s[i + 1]))

        return score

**5.** [**Modify Columns**](https://leetcode.com/problems/modify-columns/)

DataFrame employees

+-------------+--------+

| Column Name | Type |

+-------------+--------+

| name | object |

| salary | int |

+-------------+--------+

A company intends to give its employees a pay rise.

Write a solution to **modify** the salary column by multiplying each salary by 2.

The result format is in the following example.

**Example 1:**

**Input:**

DataFrame employees

+---------+--------+

| name | salary |

+---------+--------+

| Jack | 19666 |

| Piper | 74754 |

| Mia | 62509 |

| Ulysses | 54866 |

+---------+--------+

**Output:**

+---------+--------+

| name | salary |

+---------+--------+

| Jack | 39332 |

| Piper | 149508 |

| Mia | 125018 |

| Ulysses | 109732 |

+---------+--------+

**Explanation:**

Every salary has been doubled.

ANS：

def modifySalaryColumn(employees: pd.DataFrame) -> pd.DataFrame:

    employees['salary'] =  employees['salary'].apply(lambda x: x\*2)

    return employees

**6.** [**Reshape Data: Concatenate**](https://leetcode.com/problems/reshape-data-concatenate/)

DataFrame df1

+-------------+--------+

| Column Name | Type |

+-------------+--------+

| student\_id | int |

| name | object |

| age | int |

+-------------+--------+

DataFrame df2

+-------------+--------+

| Column Name | Type |

+-------------+--------+

| student\_id | int |

| name | object |

| age | int |

+-------------+--------+

Write a solution to concatenate these two DataFrames **vertically** into one DataFrame.

The result format is in the following example.

**Example 1:**

**Input:**

**df1**

+------------+---------+-----+

| student\_id | name | age |

+------------+---------+-----+

| 1 | Mason | 8 |

| 2 | Ava | 6 |

| 3 | Taylor | 15 |

| 4 | Georgia | 17 |

+------------+---------+-----+

**df2**

+------------+------+-----+

| student\_id | name | age |

+------------+------+-----+

| 5 | Leo | 7 |

| 6 | Alex | 7 |

+------------+------+-----+

**Output:**

+------------+---------+-----+

| student\_id | name | age |

+------------+---------+-----+

| 1 | Mason | 8 |

| 2 | Ava | 6 |

| 3 | Taylor | 15 |

| 4 | Georgia | 17 |

| 5 | Leo | 7 |

| 6 | Alex | 7 |

+------------+---------+-----+

**Explanation:**

The two DataFramess are stacked vertically, and their rows are combined.

ANS：

import pandas as pd

def concatenateTables(df1: pd.DataFrame, df2: pd.DataFrame) -> pd.DataFrame:

    df = pd.concat([df1, df2], axis=0)

    return df

**7.** [**Concatenation of Array**](https://leetcode.com/problems/concatenation-of-array/)

Given an integer array nums of length n, you want to create an array ans of length 2n where ans[i] == nums[i] and ans[i + n] == nums[i] for 0 <= i < n (**0-indexed**).

Specifically, ans is the **concatenation** of two nums arrays.

Return *the array*ans.

**Example 1:**

**Input:** nums = [1,2,1]

**Output:** [1,2,1,1,2,1]

**Explanation:** The array ans is formed as follows:

- ans = [nums[0],nums[1],nums[2],nums[0],nums[1],nums[2]]

- ans = [1,2,1,1,2,1]

**Example 2:**

**Input:** nums = [1,3,2,1]

**Output:** [1,3,2,1,1,3,2,1]

**Explanation:** The array ans is formed as follows:

- ans = [nums[0],nums[1],nums[2],nums[3],nums[0],nums[1],nums[2],nums[3]]

- ans = [1,3,2,1,1,3,2,1]

ANS：

class Solution:

    def getConcatenation(self, nums: List[int]) -> List[int]:

        ans\_list = []

        for time in range(2):

            for i in range(len(nums)):

                ans\_list.append(nums[i])

        return ans\_list

**8.** [**Create a New Column**](https://leetcode.com/problems/create-a-new-column/)

DataFrame employees

+-------------+--------+

| Column Name | Type. |

+-------------+--------+

| name | object |

| salary | int. |

+-------------+--------+

A company plans to provide its employees with a bonus.

Write a solution to create a new column name bonus that contains the **doubled values** of the salary column.

The result format is in the following example.

**Example 1:**

**Input:**

DataFrame employees

+---------+--------+

| name | salary |

+---------+--------+

| Piper | 4548 |

| Grace | 28150 |

| Georgia | 1103 |

| Willow | 6593 |

| Finn | 74576 |

| Thomas | 24433 |

+---------+--------+

**Output:**

+---------+--------+--------+

| name | salary | bonus |

+---------+--------+--------+

| Piper | 4548 | 9096 |

| Grace | 28150 | 56300 |

| Georgia | 1103 | 2206 |

| Willow | 6593 | 13186 |

| Finn | 74576 | 149152 |

| Thomas | 24433 | 48866 |

+---------+--------+--------+

**Explanation:**

A new column bonus is created by doubling the value in the column salary.

ANS：

import pandas as pd

def createBonusColumn(employees: pd.DataFrame) -> pd.DataFrame:

    employees['bonus'] = employees['salary'].apply(lambda x: x\*2)

    return employees

**9.** [**Build Array from Permutation**](https://leetcode.com/problems/build-array-from-permutation/)

Easy

Topics

Companies

Hint

Given a **zero-based permutation** nums (**0-indexed**), build an array ans of the **same length** where ans[i] = nums[nums[i]] for each 0 <= i < nums.length and return it.

A **zero-based permutation** nums is an array of **distinct** integers from 0 to nums.length - 1 (**inclusive**).

**Example 1:**

**Input:** nums = [0,2,1,5,3,4]

**Output:** [0,1,2,4,5,3]

**Explanation:** The array ans is built as follows:

ans = [nums[nums[0]], nums[nums[1]], nums[nums[2]], nums[nums[3]], nums[nums[4]], nums[nums[5]]]

= [nums[0], nums[2], nums[1], nums[5], nums[3], nums[4]]

= [0,1,2,4,5,3]

**Example 2:**

**Input:** nums = [5,0,1,2,3,4]

**Output:** [4,5,0,1,2,3]

**Explanation:** The array ans is built as follows:

ans = [nums[nums[0]], nums[nums[1]], nums[nums[2]], nums[nums[3]], nums[nums[4]], nums[nums[5]]]

= [nums[5], nums[0], nums[1], nums[2], nums[3], nums[4]]

= [4,5,0,1,2,3]

ANS：

class Solution:

    def buildArray(self, nums: List[int]) -> List[int]:

        ans\_list = []

        for i in range(len(nums)):

            ans\_list.append(nums[nums[i]])

        return ans\_list

**10**[**. Permutation Difference between Two Strings**](https://leetcode.com/problems/permutation-difference-between-two-strings/)

You are given two strings s and t such that every character occurs at most once in s and t is a permutation of s.

The **permutation difference** between s and t is defined as the **sum** of the absolute difference between the index of the occurrence of each character in s and the index of the occurrence of the same character in t.

Return the **permutation difference** between s and t.

**Example 1:**

**Input:** s = "abc", t = "bac"

**Output:** 2

**Explanation:**

For s = "abc" and t = "bac", the permutation difference of s and t is equal to the sum of:

* The absolute difference between the index of the occurrence of "a" in s and the index of the occurrence of "a" in t.
* The absolute difference between the index of the occurrence of "b" in s and the index of the occurrence of "b" in t.
* The absolute difference between the index of the occurrence of "c" in s and the index of the occurrence of "c" in t.

That is, the permutation difference between s and t is equal to |0 - 1| + |2 - 2| + |1 - 0| = 2.

**Example 2:**

**Input:** s = "abcde", t = "edbac"

**Output:** 12

**Explanation:** The permutation difference between s and t is equal to |0 - 3| + |1 - 2| + |2 - 4| + |3 - 1| + |4 - 0| = 12.

ANS：

class Solution:

    def findPermutationDifference(self, s: str, t: str) -> int:

        difference = 0

        for i in range(len(s)):

            difference += abs(i - t.index(s[i]))

        return difference