**CSA0835-Python Programming**

**DAY-4**

**PROGRAMS AND OUTPUTS**

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

**Test Case 1: Input: nums = [2,7,11,15], target = 9**

**Output: [0,1]**

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

**Output: [1,2]**

def two\_sum(nums, target):

for i in range(len(nums)):

for j in range(i+1, len(nums)):

if nums[i] + nums[j] == target:

return [i, j]

nums = [2, 7, 11, 15]

target = 9

print(two\_sum(nums, target))

**2. You are given two integer arrays, nums1, and nums2, sorted in non-decreasing order, and two integers, m, and n, representing the number of elements in nums1 and nums2, respectively. Merge nums1 and nums2 into a single array, num1 sorted in non-decreasing order.**

**Test Case 1: Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3**

**Output: [1,2,2,3,5,6]**

**Test Case 2: Input: nums1 = [1], m = 1, nums2 = [], n = 0**

**Output: [1]**

def merge(nums1, m, nums2, n):

return sorted(nums1[:m] + nums2[:n])

nums1 = [1, 2, 3, 0, 0, 0]

m = 3

nums2 = [2, 5, 6]

n = 3

print(merge(nums1, m, nums2, n))

**3. Write a function to find the longest common prefix string amongst an array of strings. If there is no common prefix, return an empty string ””.**

**Test Case 1: Input: strs = [“flower”,“flow”,“flight”]**

**Output: ”fl”**

**Test Case 2: Input: strs = [“dog”,“racecar”,“car”]**

**Output: ””**

def longest\_common\_prefix(strs):

if not strs:

return ""

shortest\_str = min(strs, key=len)

for i, char in enumerate(shortest\_str):

for other in strs:

if other[i] != char:

return shortest\_str[:i]

return shortest\_str

strs = ["flower", "flow", "flight"]

print(longest\_common\_prefix(strs))

**4. Given a string s containing just the characters ’(’, ’)’, ’’, ’’, ’[’ and ’]’, determine if the input string is valid. An input string is valid if: Open brackets must be closed by the same type of brackets. Open brackets must be closed in the correct order. Every close bracket has a corresponding open bracket of the same type.**

**Test Case 1: Input: s = ”()[]”**

**Output: true**

**Test Case 2: Input: s = ”(]”**

**Output: false**

def isValid(s):

stack = []

pairs = {')': '(', '}': '{', ']': '['}

for char in s:

if char in pairs.values():

stack.append(char)

elif char in pairs:

if not stack or stack.pop() != pairs[char]:

return False

return not stack

s = "()"

print(isValid(s))

**5. You are given the two sorted lists, list 1 and list 2. Merge the two lists into one sorted list. The list should be made by splicing together the elements of the first two lists. Return the merged list.**

**Test Case 1: Input: list1 = [1,2,4], list2 = [1,3,4]**

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

**Test Case 2: Input: list1 = [], list2 = [0]**

**Output: [0]**

def merge\_sorted\_lists(list1, list2):

return sorted(list1 + list2)

list1 = [1, 2, 4]

list2 = [1, 3, 4]

print(merge\_sorted\_lists(list1, list2))

**6. Given an integer array of nums and an integer value, remove all occurrences of val in nums in place. The order of the elements may be changed. Then, return the number of elements in nums that are not equal to value. Consider the number of elements in nums which are not equal to val be k, to get accepted, you need to do the following things: Change the array nums such that the first k elements of nums contain the elements which are not equal to val. The remaining elements of nums are not essential, nor is the size of nums. Return k.**

**Test Case 1: Input: nums = [3,2,2,3], val = 3**

**Output: 2, nums = [2,2, , ]**

**Test Case 2: Input: nums = [0,1,2,2,3,0,4,2], val = 2**

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

def removeElement(nums, val):

k = 0

for i in range(len(nums)):

if nums[i] != val:

nums[k] = nums[i]

k += 1

return k

nums1 = [3, 2, 2, 3]

val1 = 3

k1 = removeElement(nums1, val1)

print(k1,nums1[:k1])

**7. Given two strings, needle and haystack, return the index of the first occurrence of needle in a haystack, or -1 if the needle is not part of the haystack.**

**Test Case 1: Input: haystack = “sadbutsad”, needle = “sad”**

**Output: 0**

**Test Case 2: Input: haystack = “leetcode”, needle = “leeto”**

**Output: -1**

def strStr(haystack, needle):

if needle in haystack:

return haystack.index(needle)

else:

return -1

haystack = "sadbutsad"

needle = "sad"

print(strStr(haystack,needle))

**8. Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if inserted in order. You must write an algorithm with O(log n) runtime complexity.**

**Test Case 1: Input: nums = [1,3,5,6], target = 5**

**Output: 2**

**Test Case 2: Input: nums = [1,3,5,6], target = 2**

**Output: 1**

def searchInsert(nums, target):

left, right = 0, len(nums) - 1

while left <= right:

mid = (left + right) // 2

if nums[mid] == target:

return mid

elif nums[mid] < target:

left = mid + 1

else:

right = mid - 1

return left

nums = [1, 3, 5, 6]

target = 2

print(searchInsert(nums, target))

**9. Given a string s consisting of words and spaces, return the length of the last word in the string. A word is a maximal substring consisting of non-space characters only.**

**Test Case 1: Input: s = “Hello World”**

**Output: 5**

**Test Case 2: Input: s = “ fly me to the moon ”**

**Output: 4**

s=input("enter the string;")

words=s.split()

print(len(words[-1]))

**10. You are given a large integer represented as an integer array of digits, where each digit [i] is the ith digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0s. Increment the large integer by one and return the resulting array of digits.**

**Test Case 1: Input: digits = [1,2,3]**

**Output: [1,2,4]**

**Test Case 2: Input: digits = [4,3,2,1]**

**Output: [4,3,2,2]**

def plusOne(digits):

for i in range(len(digits) - 1, -1, -1):

if digits[i] == 9:

digits[i] = 0

else:

digits[i] += 1

return digits

return [1] + digits

digits = [1, 2, 3]

print(plusOne(digits))

**11. Given two binary strings a and b, return their sum as a binary string.**

**Test Case 1: Input: a = “11”, b = “1”**

**Output: “100”**

**Test Case 2: Input: a = “1010”, b = “1011”**

**Output: “10101”**

bin1=input("enter the binary number1 : ")

bin2=input("enter the binary number2 : ")

bin1=int(bin1,2)

bin2=int(bin2,2)

result=bin1+bin2

print("addition : ",bin(result)[2:])

**12. Given a non-negative integer x, return the square root of x rounded down to the nearest integer. The returned integer should be non-negative as well. It would help if you did not use any built-in exponent function or operator.**

**Test Case 1: Input: x = 4**

**Output: 2**

**Test Case 2: Input: x = 8**

**Output: 2**

def mySqrt(x):

i = 0

while i \* i <= x:

i += 1

return i - 1

x = 4

print(mySqrt(x))

**13. Given a string s, reverse only all the vowels in the string and return it. The vowels are ’a’, ’e’, ’i’, ’o’, and ’u’, and they can appear in lower and upper cases, more than once.**

**Test Case 1: Input: s = ”hello”**

**Output: ”holle”**

**Test Case 2: Input: s = ”leetcode”**

**Output: ”leotcede**

def reverse\_vowels(s):

vowels = set('aeiouAEIOU')

s\_list = list(s)

i, j = 0, len(s\_list) - 1

while i < j:

if s\_list[i] in vowels and s\_list[j] in vowels:

s\_list[i], s\_list[j] = s\_list[j], s\_list[i]

i, j = i + 1, j - 1

if s\_list[i] not in vowels: i += 1

if s\_list[j] not in vowels: j -= 1

return ''.join(s\_list)

print(reverse\_vowels("hello"))

**14. Given two integer arrays nums1 and nums2, return an array of their intersection. Each element in the result must be unique and you may return the result in any order.**

**Test Case 1: Input: nums1 = [1,2,2,1], nums2 = [2,2]**

**Output: [2]**

**Test Case 2: Input: nums1 = [4,9,5], nums2 = [9,4,9,8,4]**

**Output: [9,4]**

def intersection(nums1, nums2):

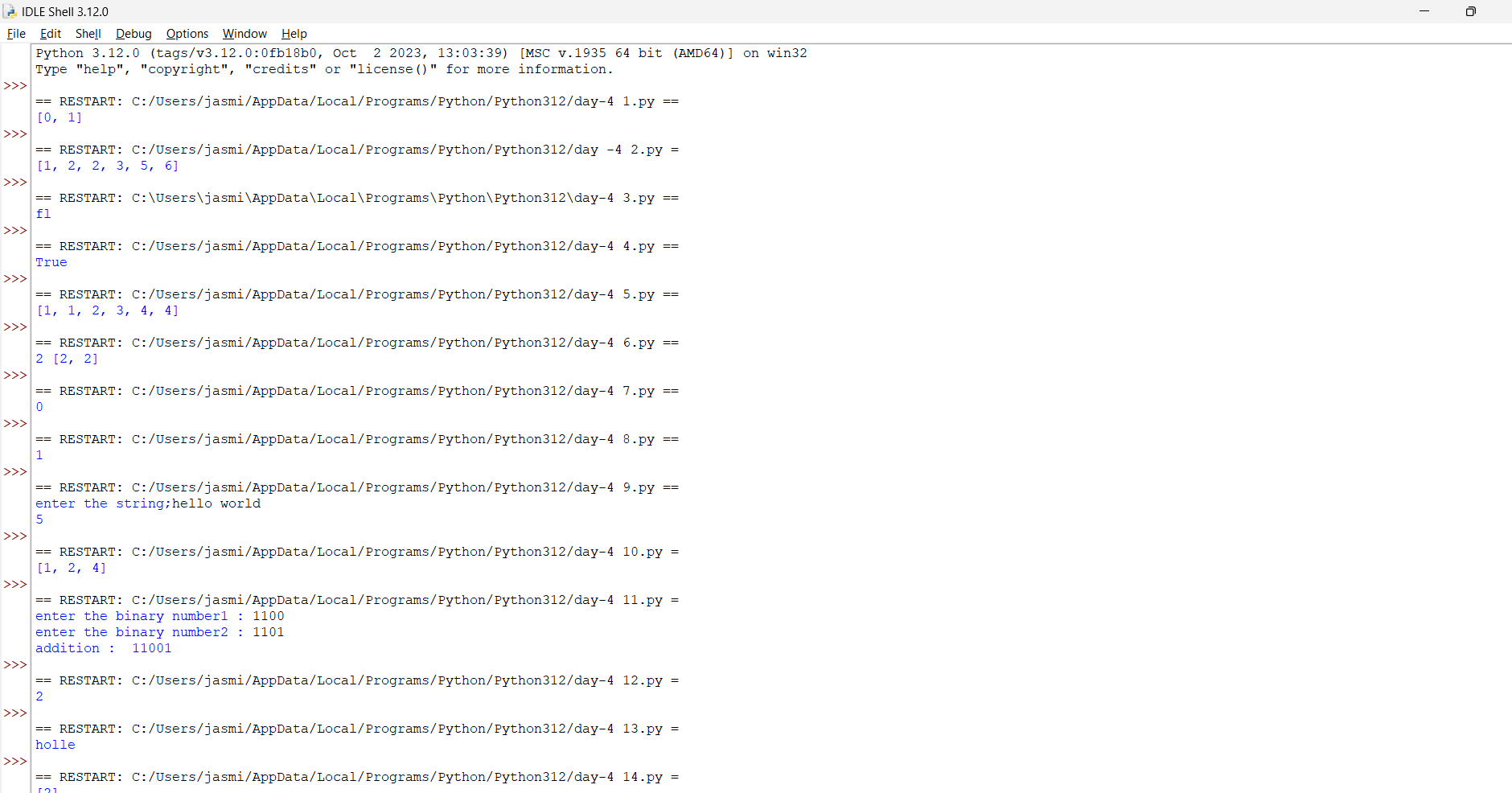
return list(set(nums1) & set(nums2))

nums1 = [1, 2, 2, 1]

nums2 = [2, 2]

print(intersection(nums1, nums2))

**OUTPUT FOR THE ABOVE ALL CODES:**

****