Assignment-2

Name: G.venkatapraveen

Reg.no: 192373023

Dept: Cse(D.s)

11. Container With Most Water

Program:

def max\_area(height):

max\_area = 0

left = 0

right = len(height) - 1

while left < right:

width = right - left

h = min(height[left], height[right])

max\_area = max(max\_area, width \* h)

if height[left] < height[right]:

left += 1

else:

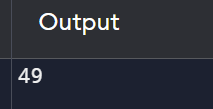
right -= 1

return max\_area

# Example

height = [1, 8, 6, 2, 5, 4, 8, 3, 7]

print(max\_area(height)) # Output: 49



12. Integer to Roman

def int\_to\_roman(num):

if not isinstance(num, int) or num <= 0 or num > 3999:

raise ValueError("Input must be a positive integer between 1 and 3999")

val = [1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1]

syms = ["M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"]

roman\_num = ''

i = 0

while num > 0:

for \_ in range(num // val[i]):

roman\_num += syms[i]

num -= val[i]

i += 1

return roman\_num

# Test the function

try:

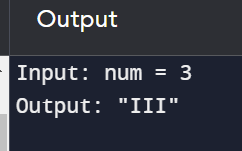
num = 3

roman\_numeral = int\_to\_roman(num)

print(f"Input: {num}\nOutput: {roman\_numeral}")

except ValueError as e:

print(f"Error: {e}")



13. Roman to Integer

def roman\_to\_int(s: str) -> int:

roman\_dict = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}

result = 0

prev\_value = 0

for char in s:

value = roman\_dict[char]

if value > prev\_value:

result += value - 2 \* prev\_value

else:

result += value

prev\_value = value

return result

# Test the function

input\_roman = "III"

output\_integer = roman\_to\_int(input\_roman)

print(f"Input: {input\_roman}")

print(f"Output: {output\_integer}")

A screenshot of a computer

Description automatically generated

|  |
| --- |
|  |

14. Longest Common Prefix

def longest\_common\_prefix(strs):

if not strs:

return ""

# Sort the list of strings

strs.sort()

# Find the common prefix between the first and last strings

prefix = ""

for i in range(len(strs[0])):

if strs[0][i] == strs[-1][i]:

prefix += strs[0][i]

else:

break

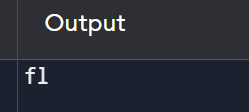
return prefix

# Test the function

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

output = longest\_common\_prefix(strs)

print(output) # Output: "fl".



15. 3Sum

def find\_triplets(nums):

nums.sort()

triplets = []

n = len(nums)

for i in range(n-2):

if i > 0 and nums[i] == nums[i-1]:

continue

left, right = i+1, n-1

while left < right:

total = nums[i] + nums[left] + nums[right]

if total < 0:

left += 1

elif total > 0:

right -= 1

else:

triplets.append([nums[i], nums[left], nums[right]])

while left < right and nums[left] == nums[left+1]:

left += 1

while left < right and nums[right] == nums[right-1]:

right -= 1

left += 1

right -= 1

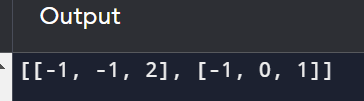
return triplets

# Example

nums = [-1, 0, 1, 2, -1, -4]

result = find\_triplets(nums)

print(result) # Output: [[-1, -1, 2], [-1, 0, 1]]



16. 3Sum Closest

def threeSumClosest(nums, target):

nums.sort()

n = len(nums)

closest\_sum = float('inf')

for i in range(n):

left, right = i + 1, n - 1

while left < right:

current\_sum = nums[i] + nums[left] + nums[right]

if abs(target - current\_sum) < abs(target - closest\_sum):

closest\_sum = current\_sum

if current\_sum < target:

left += 1

else:

right -= 1

return closest\_sum

# Example

nums = [-1, 2, 1, -4]

target = 1

result = threeSumClosest(nums, target)

print(result) # Output: 2

A screen shot of a computer

Description automatically generated

17. Letter Combinations of a Phone Number

from typing import List

def letter\_combinations(digits: str) -> List[str]:

if not digits:

return []

phone\_mapping = {

'2': 'abc',

'3': 'def',

'4': 'ghi',

'5': 'jkl',

'6': 'mno',

'7': 'pqrs',

'8': 'tuv',

'9': 'wxyz'

}

def backtrack(index, path):

if index == len(digits):

combinations.append(''.join(path))

return

for char in phone\_mapping[digits[index]]:

path.append(char)

backtrack(index + 1, path)

path.pop()

combinations = []

backtrack(0, [])

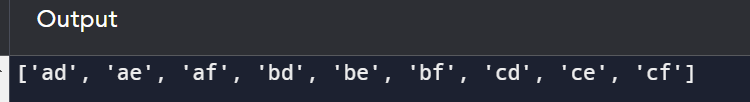
return combinations

# Test the function with the provided example

digits = "23"

output = letter\_combinations(digits)

print(output)



18. 4Sum

def fourSum(nums, target):

nums.sort()

result = []

n = len(nums)

for i in range(n - 3):

if i > 0 and nums[i] == nums[i - 1]:

continue

for j in range(i + 1, n - 2):

if j > i + 1 and nums[j] == nums[j - 1]:

continue

left, right = j + 1, n - 1

while left < right:

total = nums[i] + nums[j] + nums[left] + nums[right]

if total == target:

result.append([nums[i], nums[j], nums[left], nums[right]])

while left < right and nums[left] == nums[left + 1]:

left += 1

while left < right and nums[right] == nums[right - 1]:

right -= 1

left += 1

right -= 1

elif total < target:

left += 1

else:

right -= 1

return result

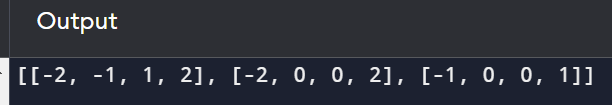
# Test the function with the provided example

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

target = 0

output = fourSum(nums, target)

print(output) # Output: [[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]



19. Remove Nth Node From End of List

class ListNode:

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

self.val = val

self.next = next

def removeNthFromEnd(head, n):

dummy = ListNode(0)

dummy.next = head

first = dummy

second = dummy

for \_ in range(n + 1):

first = first.next

while first is not None:

first = first.next

second = second.next

second.next = second.next.next

return dummy.next

# Example

head = ListNode(1, ListNode(2, ListNode(3, ListNode(4, ListNode(5))))

n = 2

result = removeNthFromEnd(head, n)

# Print the modified linked list

while result:

print(result.val, end=" ")

result = result.next

20. Valid Parentheses

def is\_valid(s: str) -> bool:

stack = []

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

for char in s:

if char in mapping:

top\_element = stack.pop() if stack else '#'

if mapping[char] != top\_element:

return False

else:

stack.append(char)

return not stack

# Test the function with the provided example

input\_string = "()"

print(is\_valid(input\_string)) # Output: True

A screenshot of a computer

Description automatically generated