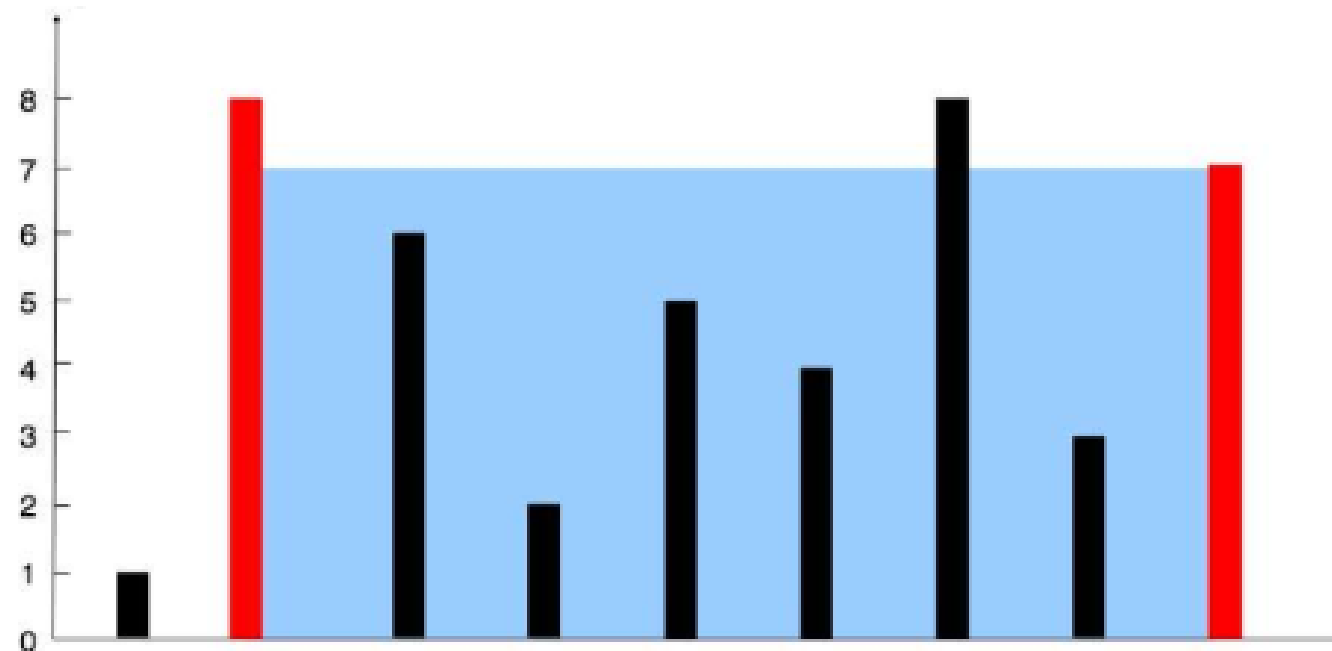


**11 .Container With Most Water** You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]). Find two lines that together with the x-axis form a container, such that the container contains the most water. Return the maximum amount of water a container can store. Notice that you may not slant the container.



Example 2:

Input: height = [1,8,6,2,5,4,8,3,7] Output: 49 Explanation: The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.

```
Container With Most Water.py - C:/Users/jayan/OneDrive/Documents/DAA/Container With Most Water.py (3.12.2)
File Edit Format Run Options Window Help

def max_area(height):
    left, right = 0, len(height) - 1
    max_water = 0
    while left < right:
        width = right - left
        min_height = min(height[left], height[right])
        area = width * min_height
        max_water = max(max_water, area)
        if height[left] < height[right]:
            left += 1
        else:
            right -= 1
    return max_water

height = [1, 8, 6, 2, 5, 4, 8, 3, 7]
max_water_area = max_area(height)
print(max_water_area)
```

```
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/jayan/OneDrive/Documents/DAA/Container With Most Water.py
49
>>> |
```

**12. Integer to Roman** Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M. Symbol Value I 1 V 5 X 10 L 50 C 100 D 500 M 1000 For example, 2 is written as II in Roman numeral, just two one's added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II. Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used: • I can be placed before V (5) and X (10) to make 4 and 9. • X can be placed before L (50) and C (100) to make 40 and 90. • C can be placed before D (500) and M (1000) to make 400 and 900. Given an integer, convert it to a roman numeral.

Example 1: Input: num = 3 Output: "III" Explanation: 3 is represented as 3 ones.

```
Integer to roman.py - C:/Users/jayan/OneDrive/Documents/DAA/Integer to roman.py (3.12.2)
File Edit Format Run Options Window Help
def int_to_roman(num):
    roman_dict = {
        1000: "M",
        900: "CM",
        500: "D",
        400: "CD",
        100: "C",
        90: "XC",
        50: "L",
        40: "XL",
        10: "X",
        9: "IX",
        5: "V",
        4: "IV",
        1: "I"
    }
    roman_numeral = ""
    for value, symbol in roman_dict.items():
        while num >= value:
            roman_numeral += symbol
            num -= value
    return roman_numeral

num = 3549
roman_num = int_to_roman(num)
print(roman_num)
```

```
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/jayan/OneDrive/Documents/DAA/Integer to roman.py
MMMDXLIX
>>>
```

**13. Roman to Integer** Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M. Symbol Value I 1 V 5 X 10 L 50 C 100 D 500 M 1000 For example, 2 is written as II in Roman numeral, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II. Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used: • I can be placed before V (5) and X (10) to make 4 and 9. • X can be placed before L (50)

and C (100) to make 40 and 90. • C can be placed before D (500) and M (1000) to make 400 and 900. Given a roman numeral, convert it to an integer.

Example 1: Input: s = "III" Output: 3 Explanation: III = 3.

```
Roman to integer.py - C:/Users/jayan/OneDrive/Documents/DAA/Roman to integer.py (3.12.2)
File Edit Format Run Options Window Help

def roman_to_int(s):
    roman_dict = {
        "I": 1,
        "V": 5,
        "X": 10,
        "L": 50,
        "C": 100,
        "D": 500,
        "M": 1000
    }
    integer_value = 0
    for i in range(len(s)):
        current_symbol = s[i]
        current_value = roman_dict[current_symbol]
        if i + 1 < len(s) and roman_dict[s[i + 1]] > current_value:
            integer_value -= current_value
        else:
            integer_value += current_value
    return integer_value

roman_num = "MMMDCXLIX"
integer_value = roman_to_int(roman_num)
print(integer_value)

IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
n32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/jayan/OneDrive/Documents/DAA/Roman to integer.py
3549
>>> |
```

**14. Longest Common Prefix** 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 "".

Example 1: Input: strs = ["flower","flow","flight"] Output: "fl"

Longest common prefix.py - C:/Users/jayan/OneDrive/Documents/DAA/Longest common prefix.py (3.12.2)

File Edit Format Run Options Window Help

```
def longest_common_prefix(strs):
    if not strs:
        return ""
    shortest_str = min(strs, key=len)
    for i in range(len(shortest_str)):
        for other_str in strs:
            if other_str[i] != shortest_str[i]:
                return shortest_str[:i]
    return shortest_str

strs = ["flower", "flow", "flight"]
longest_prefix = longest_common_prefix(strs)
print(longest_prefix)
```

IDLE Shell 3.12.2

File Edit Shell Debug Options Window Help

Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]  
n32

Type "help", "copyright", "credits" or "license()" for more information.

>>>

= RESTART: C:/Users/jayan/OneDrive/Documents/DAA/Longest common prefix.py

fl

>>>

|

**15. 3 Sum** Given an integer array `nums`, return all the triplets `[nums[i], nums[j], nums[k]]` such that  $i \neq j$ ,  $i \neq k$ , and  $j \neq k$ , and  $nums[i] + nums[j] + nums[k] == 0$ . Notice that the solution set must not contain duplicate triplets.



Example 1: Input: `nums = [-1,0,1,2,-1,-4]` Output: `[[-1,-1,2],[-1,0,1]]` Explanation: `nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0`. `nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0`. `nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0`. The distinct triplets are `[-1,0,1]` and `[-1,-1,2]`. Notice that the order of the output and the order of the triplets does not matter

```
Three sum.py - C:/Users/jayan/OneDrive/Documents/DAA/Three sum.py (3.12.2)
File Edit Format Run Options Window Help

def three_sum(nums):
    nums.sort()
    triplets = []
    for i in range(len(nums) - 2):
        if i > 0 and nums[i] == nums[i - 1]:
            continue
        left, right = i + 1, len(nums) - 1
        while left < right:
            sum_of_three = nums[i] + nums[left] + nums[right]
            if sum_of_three == 0:
                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
            elif sum_of_three < 0:
                left += 1
            else:
                right -= 1
    return triplets

nums = [-1, 0, 1, 2, -1, -4]
triplet_sums = three_sum(nums)
print(triplet_sums)
```

```
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help

Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
n32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/jayan/OneDrive/Documents/DAA/Three sum.py
[[-1, -1, 2], [-1, 0, 1]]
>>>
```

**16. 3 Sum Closest** Given an integer array `nums` of length `n` and an integer `target`, find three integers in `nums` such that the sum is closest to `target`. Return the sum of the three integers. You may assume that each input would have exactly one solution.

Example 1: Input: `nums = [-1,2,1,-4]`, `target = 1` Output: 2 Explanation: The sum that is closest to the target is 2. ( $-1 + 2 + 1 = 2$ ).

3 Sum closest.py - C:/Users/jayan/OneDrive/Documents/DAA/3 Sum closest.py (3.12.2)

File Edit Format Run Options Window Help

```
def three_sum_closest(nums, target):
    nums.sort()
    closest_sum = float('inf')
    for i in range(len(nums) - 2):
        if i > 0 and nums[i] == nums[i - 1]:
            continue

        left, right = i + 1, len(nums) - 1
        while left < right:
            current_sum = nums[i] + nums[left] + nums[right]
            if abs(current_sum - target) < abs(closest_sum - target):
                closest_sum = current_sum
            if current_sum < target:
                left += 1
            else:
                right -= 1
        return closest_sum

nums = [-1, 2, 1, -4]
target = 1
closest_value = three_sum_closest(nums, target)
print(closest_value)
```

```
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
n32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/jayan/OneDrive/Documents/DAA/3 Sum closest.py
2
>>>|
```

**17. Letter Combinations of a Phone Number** Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in any order. A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.



Example 1: Input: digits = "23" Output: ["ad","ae","af","bd","be","bf","cd","ce","cf"]



```
Letter combinations.py - C:/Users/jayan/OneDrive/Documents/DAA/Letter combinations.py (3.12.2)
File Edit Format Run Options Window Help

def letter_combinations(digits):
    digit_to_letters = {
        "2": "abc",
        "3": "def",
        "4": "ghi",
        "5": "jkl",
        "6": "mno",
        "7": "pqrs",
        "8": "tuv",
        "9": "wxyz"
    }
    if not digits:
        return []
    combinations = []
    def backtrack(current_combination, index):
        if index == len(digits):
            combinations.append(current_combination)
            return
        letters = digit_to_letters[digits[index]]
        for letter in letters:
            backtrack(current_combination + letter, index + 1)
    backtrack("", 0)
    return combinations

digits = "23"
combinations_list = letter_combinations(digits)
print(combinations_list)
```

```
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help

Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
n32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/jayan/OneDrive/Documents/DAA/Letter combinations.py
['ad', 'ae', 'af', 'bd', 'be', 'bf', 'cd', 'ce', 'cf']
>>>
```

**18. 4 Sum** Given an array nums of n integers, return an array of all the unique quadruplets [nums[a], nums[b], nums[c], nums[d]] such that: •  $0 \leq a, b, c, d < n$  • a, b, c, and d are distinct. •  $\text{nums}[a] + \text{nums}[b] + \text{nums}[c] + \text{nums}[d] == \text{target}$  You may return the answer in any order.

Example 1: Input: nums = [1,0,-1,0,-2,2], target = 0 Output: [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]

```
Four sum.py - C:/Users/jayan/OneDrive/Documents/DAA/Four sum.py (3.12.2)
File Edit Format Run Options Window Help

def four_sum(nums, target):
    nums.sort()
    quadruplets = []
    for i in range(len(nums) - 3):
        if i > 0 and nums[i] == nums[i - 1]:
            continue
        for j in range(i + 1, len(nums) - 2):
            if j > i + 1 and nums[j] == nums[j - 1]:
                continue
            left, right = j + 1, len(nums) - 1
            while left < right:
                current_sum = nums[i] + nums[j] + nums[left] + nums[right]
                if current_sum == target:
                    quadruplets.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 current_sum < target:
                    left += 1
                else:
                    right -= 1
            return quadruplets
nums = [1, 0, -1, 0, -2, 2]
target = 0
quad_list = four_sum(nums, target)
print(quad_list)
```

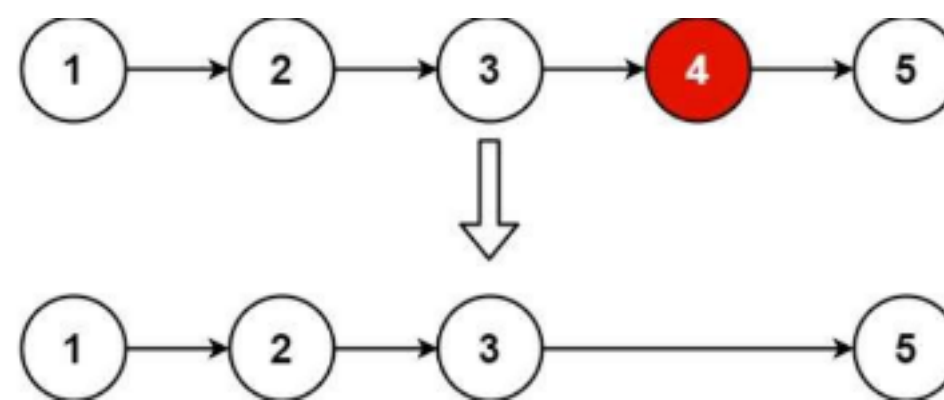
```
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help

Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
2
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/jayan/OneDrive/Documents/DAA/Four sum.py
[[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]
>>>
```

**19. Remove Nth Node From End of List** Given the head of a linked list, remove the nth node from the end of the list and return its head.

Example 1: Input: head = [1,2,3,4,5], n = 2 Output: [1,2,3,5]





Remove nth node from end.py - C:/Users/jayan/OneDrive/Documents/DAA/Remove nth node from end.py (3.12.2)

File Edit Format Run Options Window Help

```

class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

class Solution:
    def removeNthFromEnd(self, head: ListNode, n: int) -> ListNode:
        dummy = ListNode(0)
        dummy.next = head
        fast, slow = dummy, dummy
        for _ in range(n):
            if fast.next is None:
                return head
            fast = fast.next
        while fast.next:
            fast = fast.next
            slow = slow.next
        slow.next = slow.next.next
        return dummy.next
  
```

```

head = ListNode(1, ListNode(2, ListNode(3, ListNode(4, ListNode(5)))))
n = 2
linked_list = Solution()
new_head = linked_list.removeNthFromEnd(head, n)
while new_head:
    print(new_head.val, end=" -> ")
    new_head = new_head.next
  
```

IDLE Shell 3.12.2

File Edit Shell Debug Options Window Help

Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]

2

Type "help", "copyright", "credits" or "license()" for more information.

>>>

= RESTART: C:/Users/jayan/OneDrive/Documents/DAA/Remove nth node from end.py

1 -> 2 -> 3 -> 5 ->

>>>

**20. Valid Parentheses** Given a string *s* containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid. An input string is valid if: 1. Open brackets must be closed by the same type of brackets. 2. Open brackets must be closed in the

correct order. 3. Every close bracket has a corresponding open bracket of the same type.

Example 1: Input: s = "()" Output: true

```
Valid punctuation.py - C:/Users/jayan/OneDrive/Documents/DAA/Valid punctuation.py (3.12.2)
File Edit Format Run Options Window Help

def is_valid(s):
    bracket_map = {
        '(': ')',
        '{': '}',
        '[': ']'
    }
    stack = []
    for char in s:
        if char in bracket_map:
            stack.append(char)
        else:
            if not stack or bracket_map[stack.pop()] != char:
                return False
    return not stack

s1 = "()"
s2 = "()[{}]"
s3 = "["
s4 = "[()]"

print(is_valid(s1))
print(is_valid(s2))
print(is_valid(s3))
print(is_valid(s4))
```

```
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
2
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/jayan/OneDrive/Documents/DAA/Valid punctuation.py
True
True
False
False
>>>
```