# 11.Container with Most Water

```
def maxArea(A, Len) :
        area = 0
        for i in range(Len) :
                for j in range(i + 1, Len) :
                        area = max(area, min(A[j], A[i]) * (j - i))
        return area
a = [ 1, 5, 4, 3 ]
b = [ 3, 1, 2, 4, 5 ]
len1 = len(a)
print(maxArea(a, len1))
len2 = len(b)
print(maxArea(b, len2))
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/DAA/Container With Most Water.py
6
12
>>>
```

# 12.Integer to Roman

```
def int_to_roman(num):
    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 =  roman_num+syms[i]
            num = num-val[i]
        i =i+1
    return roman_num
print(int_to_roman(58))
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/DAA/Integer to Roman.py
LVIII
>>>
```

# 13.Roman to Integer

```
def roman_to_integer(roman):
    roman_values = {
        'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000
    }
    total = 0
    prev_value = 0
    for char in reversed(roman):
        value = roman_values[char]
        if value >= prev_value:
            total += value
        else:
            total -= value
        prev_value = value
    return total
roman_numeral = "III"
integer_value = roman_to_integer(roman_numeral)
print(f"The integer value of {roman_numeral} is {integer_value}")
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/DAA/Roman to Integer.py
The integer value of III is 3
>>>
```

# 14.Longest Common Prefix

```
def longest_common_prefix(strs):
    if not strs:
        return ""
    prefix = strs[0]
    for s in strs[1:]:
        min_length = min(len(prefix), len(s))
        for i in range(min_length):
            if prefix[i] != s[i]:
                prefix = prefix[:i]
                break
        else:
            prefix = prefix[:min_length]
        if not prefix:
            return ""
    return prefix
strings = ["flower", "flow", "flight"]
common_prefix = longest_common_prefix(strings)
print(f"The longest common prefix is: '{common_prefix}'")
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/DAA/Longest Common Prefix.py
The longest common prefix is: 'fl'
>>>
```

# 15.3Sum

```
def three_sum(nums):
    nums.sort()
    result = []
    n = len(nums)
    for i in range(n):
        if i > 0 and nums[i] == nums[i-1]:
            continue
        left, right = i + 1, n - 1
        while left < right:
            current_sum = nums[i] + nums[left] + nums[right]
            if current_sum == 0:
                result.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 current_sum < 0:
                left += 1
            else:
                right -= 1
    return result
nums = [0,1,1]
print(three_sum(nums))
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/DAA/3Sum.py
[]
>>>
```

# 16.3Sum Closest

```
def three_sum_closest(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(current_sum - target) < abs(closest_sum - target):
                closest_sum = current_sum
            if current_sum == target:
                return current_sum
            elif current_sum < target:
                left += 1
            else:
                right -= 1
    return closest_sum
nums = [0,0,0]
target = 1
print(three_sum_closest(nums, target))
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/DAA/3Sum Closest.py
0
>>>
```

# 17.Letter Combinations of a phone Number

```
from collections import deque
def letterCombinationsUtil(number, n, table):
        list = []
        q = deque()
        q.append("")
        while len(q) != 0:
                s = q.pop()
                if len(s) == n:
                        list.append(s)
                else:
                        for letter in table[number[len(s)]]:
                                q.append(s + letter)
        return list
def letterCombinations(number, n):
        table = ["0", "1", "abc", "def", "ghi", "jkl",
                 "mno", "pqrs", "tuv", "wxyz"]
        list = letterCombinationsUtil(number, n, table)
        s = ""
        for word in list:
                s += word + " "
        print(s)
        return
number = [2]
n = len(number)
letterCombinations(number, n)
```

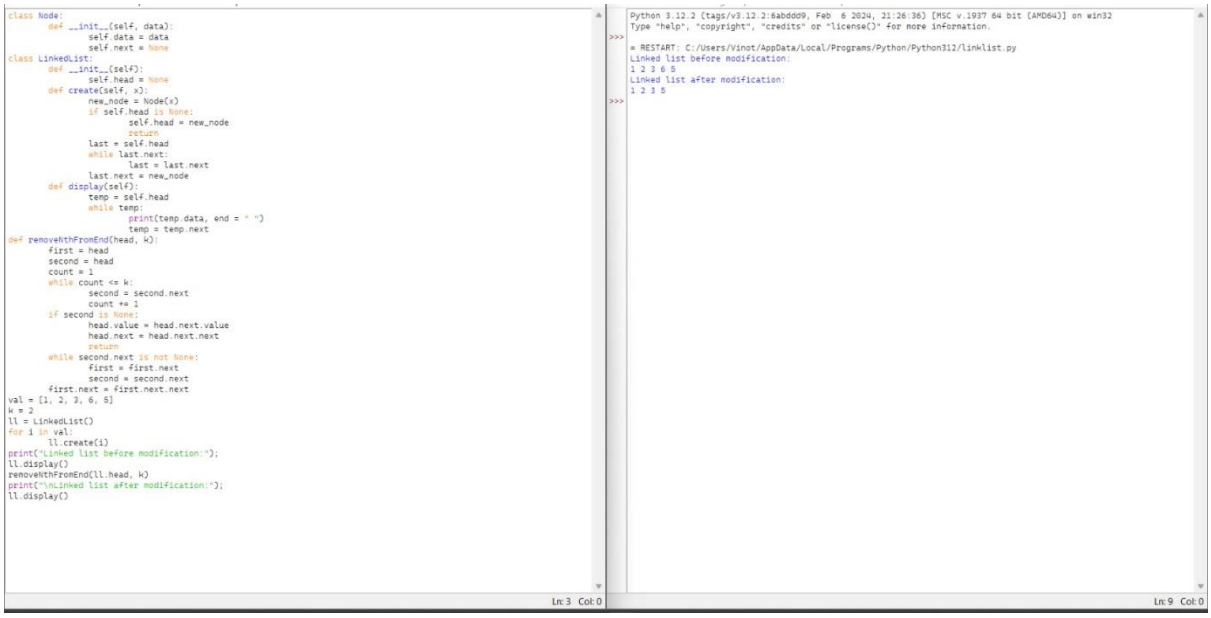
```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/DAA/Letter Combinations of a Phone Number.py
c b a
>>>
```

# 18.4Sum

```
def four_sum(nums, target):
    nums.sort()
    results = []
    length = len(nums)
    for i in range(length - 3):
        if i > 0 and nums[i] == nums[i - 1]:
            continue
        for j in range(i + 1, length - 2):
            if j > i + 1 and nums[j] == nums[j - 1]:
                continue
            left, right = j + 1, length - 1
            while left < right:
                total = nums[i] + nums[j] + nums[left] + nums[right]
                if total == target:
                    results.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 results
nums = [2,2,2,2,2]
target = 8
print(four_sum(nums, target))
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/DAA/4Sum.py
[[2, 2, 2, 2]]
>>>
```

# 19.Remove Nth Node from End of List



```
class Node:
        def __init__(self, data):
                self.data = data
                self.next = None
class LinkedList:
        def __init__(self):
                self.head = None
        def create(self, x):
                new_node = Node(x)
                if self.head is None:
                        self.head = new_node
                        return
                last = self.head
                while last.next:
                        last = last.next
                last.next = new_node
        def display(self):
                temp = self.head
                while temp:
                        print(temp.data, end = " ")
                        temp = temp.next
def removeNthFromEnd(head, k):
        first = head
        second = head
        count = 1
        while count <= k:
                second = second.next
                count += 1
        if second is None:
                head.value = head.next.value
                head.next = head.next.next
                return
        while second.next is not None:
                first = first.next
                second = second.next
        first.next = first.next.next
val = [1, 2, 3, 6, 5]
k = 2
ll = LinkedList()
for i in val:
        ll.create(i)
print("Linked list before modification:");
ll.display()
removeNthFromEnd(ll.head, k)
print("\nLinked list after modification:");
ll.display()
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Vinot/AppData/Local/Programs/Python/Python312/linklist.py
Linked list before modification:
1 2 3 6 5
Linked list after modification:
1 2 3 5
>>>
```

# 20.Valid Parentheses

```
def areBracketsBalanced(expr):
        stack = []
        for char in expr:
                if char in ["(", "{", "[":
                        stack.append(char)
                else:
                        if not stack:
                                return False
                        current_char = stack.pop()
                        if current_char == '(':
                                if char != ")":
                                        return False
                        if current_char == '{':
                                if char != "}":
                                        return False
                        if current_char == '[':
                                if char != "]":
                                        return False
        if stack:
                return False
        return True
expr = "[()]{"
if areBracketsBalanced(expr):
    print("true")
else:
    print("false")
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/DAA/Valid Parentheses.py
false
>>>
```