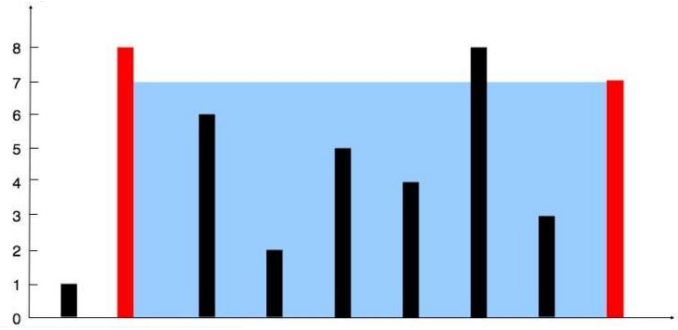
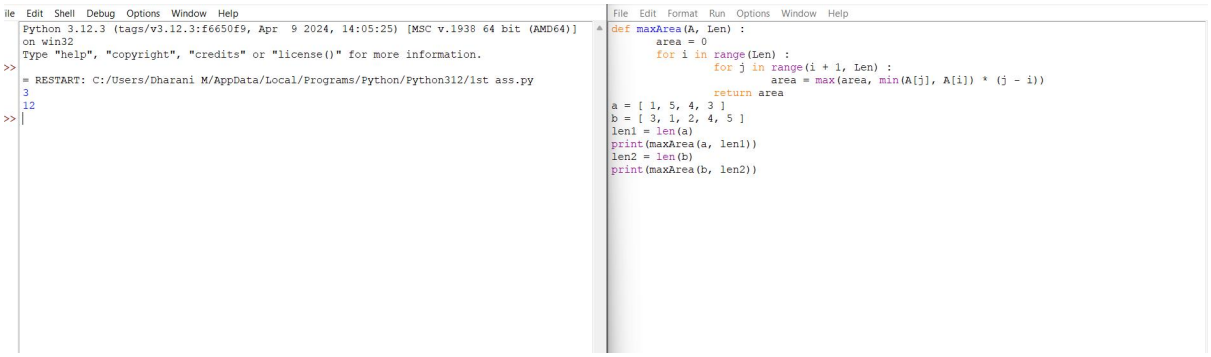
11 .Container With Most Water You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]). Find two lines that together with the x-axis form a container, such that the container contains the most water.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



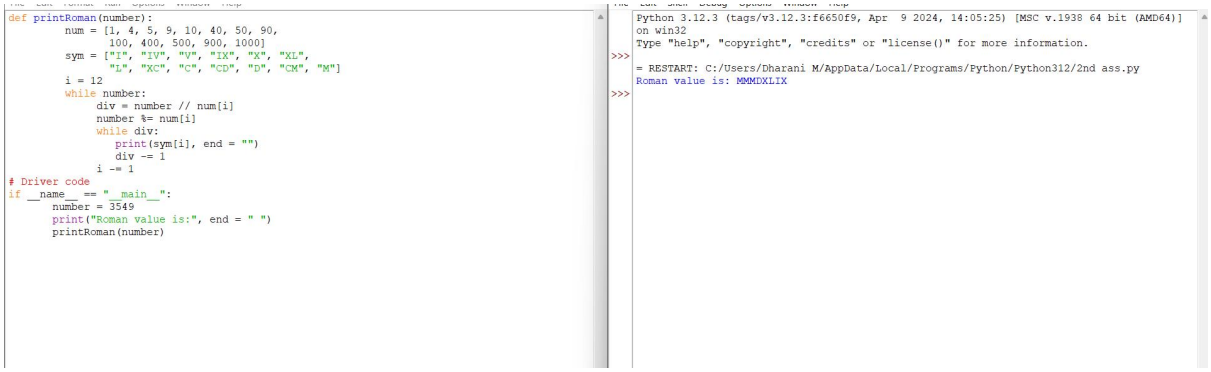
Example 2:

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.



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.

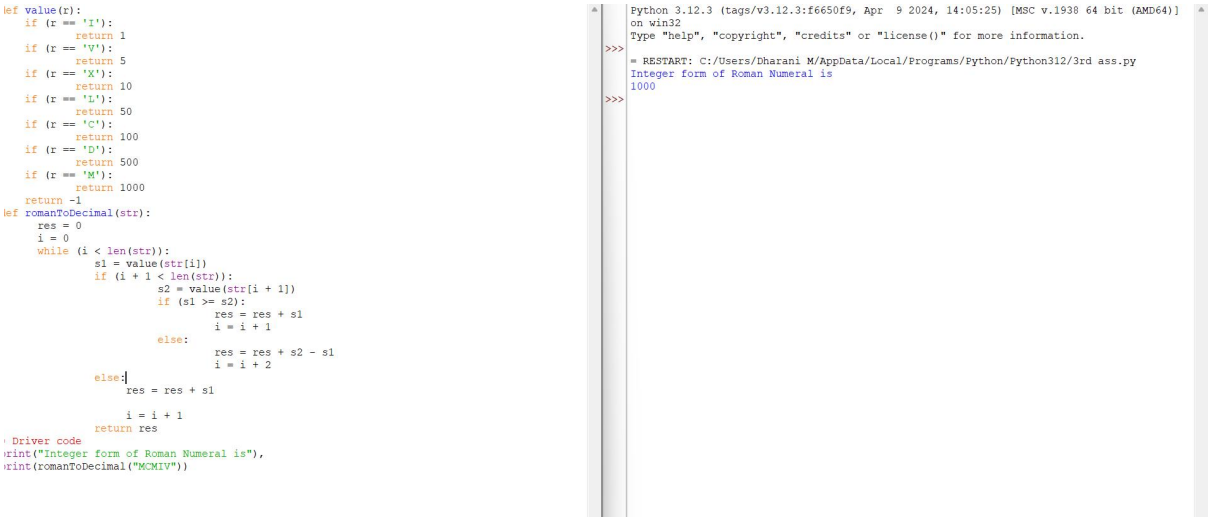


```
def printRoman(number):
        num = [1, 4, 5, 9, 10, 40, 50, 90,
               100, 400, 500, 900, 1000]
        sym = ["I", "IV", "V", "IX", "X", "XL",
               "L", "XC", "C", "CD", "D", "CM", "M"]
        i = 12
        while number:
            div = number // num[i]
            number %= num[i]
            while div:
                print(sym[i], end = "")
                div -= 1
            i -= 1
# Driver code
if __name__ == "__main__":
    number = 3549
    print("Roman value is:", end = " ")
    printRoman(number)
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/2nd ass.py
Roman value is: 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.

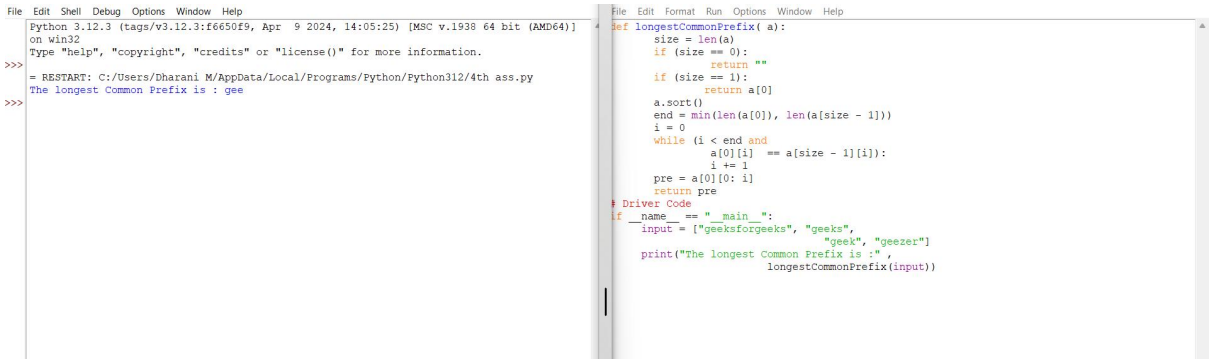


```
def value(r):
    if (r == 'I'):
            return 1
    if (r == 'V'):
            return 5
    if (r == 'X'):
            return 10
    if (r == 'L'):
            return 50
    if (r == 'C'):
            return 100
    if (r == 'D'):
            return 500
    if (r == 'M'):
            return 1000
    return -1
def romanToDecimal(str):
    res = 0
    i = 0
    while (i < len(str)):
            s1 = value(str[i])
            if (i + 1 < len(str)):
                    s2 = value(str[i + 1])
                    if (s1 >= s2):
                            res = res + s1
                            i = i + 1
                    else:
                            res = res + s2 - s1
                            i = i + 2
            else:
                res = res + s1
                i = i + 1
            return res
# Driver code
print("Integer form of Roman Numeral is"),
print(romanToDecimal("MCMIV"))
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/3rd ass.py
Integer form of Roman Numeral is
1000
>>>
```

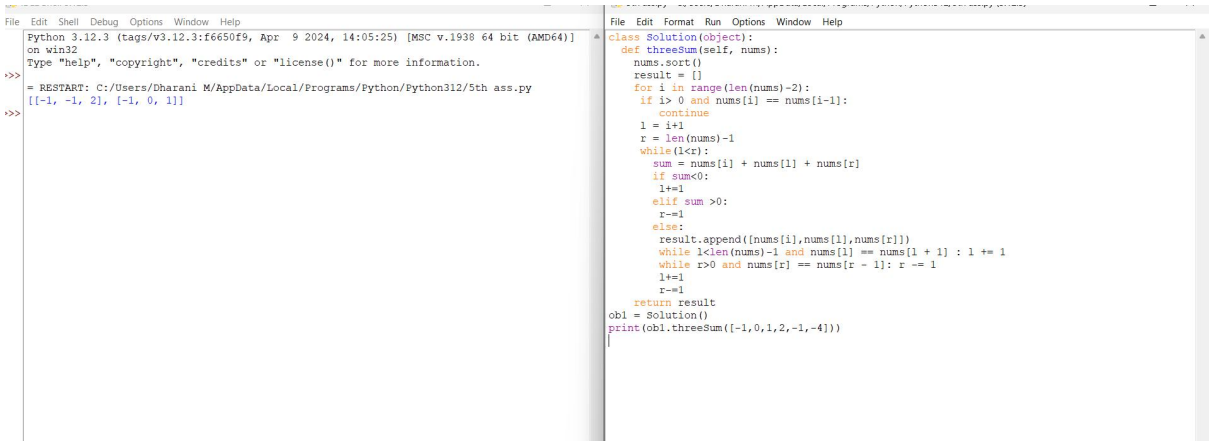
14. Longest Common Prefix Write a function to find the longest common prefix string amongst an array of strings. If there is no common prefix, return an empty string "".

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



15. 3 Sum Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that i != j, i != k, and j != 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



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).

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/5th ass.py
[[-1, -1, 2], [-1, 0, 1]]
>>>
==== RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/5th ass.py ====
2
>>>
```

```
File  Edit  Format  Run  Options  Window  Help
import sys
def solution(arr, x):
        closestSum = sys.maxsize
        for i in range (len(arr)) :
                for j in range(i + 1, len(arr)):
                        for k in range(j + 1, len( arr)):
                                if(abs(x - closestSum) >
                                abs(x - (arr[i] +
                                arr[j] + arr[k]))):
                                        closestSum = (arr[i] +arr[j] + arr[k])
        return closestSum
# Driver code
if __name__ == "__main__":
        arr = [ -1, 2, 1, -4 ]
        x = 1
        print(solution(arr, x))
```

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"]

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/5th ass.py
cf ce cd bf be bd af ae ad
>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/5th ass.py
cf ce cd bf be bd af ae ad
>>
```

```
File  Edit  Format  Run  Options  Window  Help
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


# Driver code
number = [2, 3]
n = len(number)

# Function call
letterCombinations(number, n)
```

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 <= a, b, c, d < n ● a, b, c, and d are distinct. ● nums[a] + nums[b] + nums[c] + nums[d] == target You may return the answer in any order.

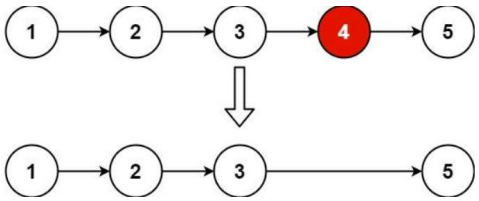
Example 1: Input: nums = [1,0,-1,0,-2,2], target = 0 Output: [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]

```
lass Pair:
        def __init__(self, x, y):
                self.index1 = x
                self.index2 = y

ef GetQuadruplets(nums, target):
        map = {}

        for i in range(len(nums) - 1):
                for j in range(i + 1, len(nums)):
                        sum = nums[i] + nums[j]

                        if sum not in map:
                                map[sum] = [Pair(i, j)]
                        else:
                                map[sum].append(Pair(i, j))

        ans = set()

        for i in range(len(nums) - 1):
                for j in range(i + 1, len(nums)):
                        lookUp = target - (nums[i] + nums[j])

                        if lookUp in map:
                                temp = map[lookUp]

                                        if pair.index1 != i and pair.index1 != j and pair.inc
                                                ll = [nums[pair.index1], nums[pair.index2], r

                                                ll.sort()
                                                ans.add(tuple(ll))
        print(*reversed(list(ans)), sep = '\n')

rr = [1, 0, -1, 0, -2, 2]
 = 0
etQuadruplets(arr, K)
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/8thh.py =
(-2, 0, 0, 2)
(-1, 0, 0, 1)
(-2, -1, 1, 2)
>>>
```

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]



```
File  Edit  Format  Run  Options  Window  Help
class Node:
    def __init__(self, value):
        self.data = value
        self.next = None

def length(head):
    temp = head
    count = 0
    while(temp != None):
        count += 1
        temp = temp.next
    return count

def printList(head):
    ptr = head
    while(ptr != None):
        print (ptr.data, end =" ")
        ptr = ptr.next
    print()

def deleteNthNodeFromEnd(head, n):
    Length = length(head)
    nodeFromBeginning = Length - n + 1
    prev = None
    temp = head
    for i in range(1, nodeFromBeginning):
        prev = temp
        temp = temp.next
    if(prev == None):
        head = head.next
        return head
    else:
        prev.next = prev.next.next
        return head

if __name__ == '__main__':
    head = Node(1)
    head.next = Node(2)
    head.next.next = Node(3)
    head.next.next.next = Node(4)
    head.next.next.next.next = Node(5)
    print("Linked List before Deletion:")
    printList(head)
```

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/8thh.py =
(-2, 0, 0, 2)
(-1, 0, 0, 1)
(-2, -1, 1, 2)
>>>
====== RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/8thh.py =====
Linked List before Deletion:
1 2 3 4 5
Linked List after Deletion:
1 3 4 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

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/8thh.py
{[]{()}} - Balanced
[{}{}](] - Unbalanced
((() - Unbalanced
>>>
```

```
open_list = ["[","{","("]
close_list = ["]","}",")"]

# Function to check parentheses
def check(myStr):
        stack = []
        for i in myStr:
                if i in open_list:
                        stack.append(i)
                elif i in close_list:
                        pos = close_list.index(i)
                        if ((len(stack) > 0) and
                                (open_list[pos] == stack[len(stack)-1])):
                                stack.pop()
                        else:
                                return "Unbalanced"
        if len(stack) == 0:
                return "Balanced"
        else:
                return "Unbalanced"


# Driver code
string = "{[]{()}}"
print(string,"-", check(string))

string = "[{}{}](]"
print(string,"-", check(string))

string = "((()"
print(string,"-",check(string))
```