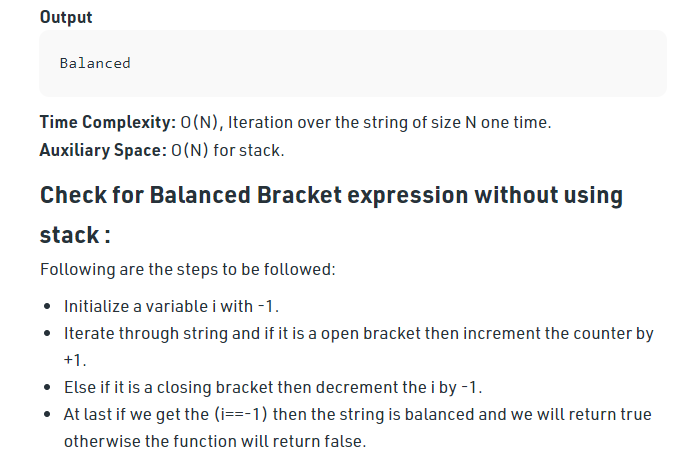
1.

|  |
| --- |
| # Python3 program to check for  # balanced brackets.    # function to check if  # brackets are balanced      **def** areBracketsBalanced(expr):      stack **=** []        # Traversing the Expression  **for** char **in** expr:  **if** char **in** ["(", "{", "["]:                # Push the element in the stack              stack.append(char)  **else**:                # IF current character is not opening              # bracket, then it must be closing.              # So stack cannot be empty at this point.  **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        # Check Empty Stack  **if** stack:  **return** False  **return** True      # Driver Code  **if** \_\_name\_\_ **==** "\_\_main\_\_":      expr **=** "{()}[]"        # Function call  **if** areBracketsBalanced(expr):  **print**("Balanced")  **else**:  **print**("Not Balanced")    # This code is contributed by AnkitRai01 and improved  # by Raju Pitta |



[Check for Balanced Brackets in an expression (well-formedness) using Stack - GeeksforGeeks](https://www.geeksforgeeks.org/check-for-balanced-parentheses-in-an-expression/)

2. [Loading... (leetcode.com)](https://leetcode.com/discuss/interview-question/444027/linkedin-oa-2019-hashed-ports-threshold-alerts)

def numberOfAlerts(preceedingMinutes, alertThreshold, numCalls):

for i in range(preceedingMinutes):

total + = (double)numCalls[i]

if(total/preceedingMinutes > alertThreshold)

count\_alerts+=1

for i in range(numCalls)

total + = (double)numCalls[i]

total -= (double)numCalls[i – preceedingMinutes]

if(total/preceedingMinutes > alertThreshold)

count\_alerts+=1

return count\_alerts

3. # Python3 implementation of above idea

# A function to check if n is palindrome

**def** isPalindrome(n: int) **-**> bool:

    # Find reverse of n

    rev **=** 0

    i **=** n

**while** i > 0:

        rev **=** rev **\*** 10 **+** i **%** 10

        i **//=** 10

    # If n and rev are same,

    # then n is palindrome

**return** (n **==** rev)

# prints palindrome between min and max

**def** countPal(minn: int, maxx: int) **-**> None:

**for** i **in** range(minn, maxx **+** 1):

**if** isPalindrome(i):

            print(i, end **=** " ")

# Driver Code

**if** \_\_name\_\_ **==** "\_\_main\_\_":

    countPal(100, 2000)

# This code is contributed by

# sanjeev2552

[Program to print all palindromes in a given range - GeeksforGeeks](https://www.geeksforgeeks.org/program-print-palindromes-given-range/?ref=lbp)

[Rearrange characters to form palindrome if possible - GeeksforGeeks](https://www.geeksforgeeks.org/rearrange-characters-form-palindrome-possible/?ref=lbp)

4. [Given a sequence of words, print all anagrams together | Set 2 - GeeksforGeeks](https://www.geeksforgeeks.org/given-a-sequence-of-words-print-all-anagrams-together-set-2/)

5. [Binary Search - GeeksforGeeks](https://www.geeksforgeeks.org/binary-search/)

6. [Print all subsequences of a string - GeeksforGeeks](https://www.geeksforgeeks.org/print-subsequences-string/)

7. [Loading... (leetcode.com)](https://leetcode.com/discuss/interview-question/1369452/Goldman-Sachs-Coderpad-Interview-Question-Magic-Potion-Help!!)

8. [Find the smallest and second smallest elements in an array - GeeksforGeeks](https://www.geeksforgeeks.org/to-find-smallest-and-second-smallest-element-in-an-array/)

9. [Find first non-repeating character of given String - GeeksforGeeks](https://www.geeksforgeeks.org/given-a-string-find-its-first-non-repeating-character/)

10. [Alternatively Merge two Strings - GeeksforGeeks](https://www.geeksforgeeks.org/alternatively-merge-two-strings-in-java/)

11. [Maximum difference between two elements such that larger element appears after the smaller number - GeeksforGeeks](https://www.geeksforgeeks.org/maximum-difference-between-two-elements/)

12. [Maximum difference between two elements such that larger element appears after the smaller number - GeeksforGeeks](https://www.geeksforgeeks.org/maximum-difference-between-two-elements/)

13. [Maximum and minimum of an array using minimum number of comparisons - GeeksforGeeks](https://www.geeksforgeeks.org/maximum-and-minimum-in-an-array/)

14. [Prime Numbers - GeeksforGeeks](https://www.geeksforgeeks.org/prime-numbers/)

[Recursive program for prime number - GeeksforGeeks](https://www.geeksforgeeks.org/recursive-program-prime-number/)

15. [Delete a node in a Doubly Linked List - GeeksforGeeks](https://www.geeksforgeeks.org/delete-a-node-in-a-doubly-linked-list/)

16. [Josephus Problem - GeeksforGeeks](https://www.geeksforgeeks.org/josephus-problem/)

17. [Find the Rotation Count in Rotated Sorted array - GeeksforGeeks](https://www.geeksforgeeks.org/find-rotation-count-rotated-sorted-array/)

[Search an element in a sorted and rotated Array - GeeksforGeeks](https://www.geeksforgeeks.org/search-an-element-in-a-sorted-and-pivoted-array/)

18. [Minimum number of jumps to reach end - GeeksforGeeks](https://www.geeksforgeeks.org/minimum-number-of-jumps-to-reach-end-of-a-given-array/)

19. [Reverse string in Python (6 different ways) - GeeksforGeeks](https://www.geeksforgeeks.org/reverse-string-python-5-different-ways/)

[Reverse string without using any temporary variable - GeeksforGeeks](https://www.geeksforgeeks.org/reverse-string-without-using-any-temporary-variable/)

20. [Minimum number of jumps to reach end - GeeksforGeeks](https://www.geeksforgeeks.org/minimum-number-of-jumps-to-reach-end-of-a-given-array/)

21. [Lowest Common Ancestor in a Binary Tree - GeeksforGeeks](https://www.geeksforgeeks.org/lowest-common-ancestor-binary-tree-set-1/)

22. [Maximum size subset with given sum - GeeksforGeeks](https://www.geeksforgeeks.org/maximum-size-subset-given-sum/)

[Dynamic Programming - Subset Sum Problem (geeksforgeeks.org)](https://www.geeksforgeeks.org/subset-sum-problem-dp-25/)

23. [Find the element before which all the elements are smaller than it, and after which all are greater - GeeksforGeeks](https://www.geeksforgeeks.org/find-the-element-before-which-all-the-elements-are-smaller-than-it-and-after-which-all-are-greater-than-it/)

[Find a peak element which is not smaller than its neighbours - GeeksforGeeks](https://www.geeksforgeeks.org/find-a-peak-in-a-given-array/)

24. [Design a stack that supports getMin() in O(1) time and O(1) extra space - GeeksforGeeks](https://www.geeksforgeeks.org/design-a-stack-that-supports-getmin-in-o1-time-and-o1-extra-space/)

[Design and Implement Special Stack Data Structure | Added Space Optimized Version - GeeksforGeeks](https://www.geeksforgeeks.org/design-and-implement-special-stack-data-structure/)

25. [Nuts & Bolts Problem (Lock & Key problem) using Quick Sort - GeeksforGeeks](https://www.geeksforgeeks.org/nuts-bolts-problem-lock-key-problem-using-quick-sort/)

26. [Largest subarray with equal number of 0s and 1s - GeeksforGeeks](https://www.geeksforgeeks.org/largest-subarray-with-equal-number-of-0s-and-1s/)

27.

28.

29. [Position of robot after given movements - GeeksforGeeks](https://www.geeksforgeeks.org/position-of-robot-after-given-movements/)

|  |
| --- |
| # Python3 implementation to find final position  # of robot after the complete movement    # function to find final position of  # robot after the complete movement      **def** finalPosition(move):        l **=** len(move)      countUp, countDown **=** 0, 0      countLeft, countRight **=** 0, 0        # traverse the instruction string      # 'move'  **for** i **in** range(l):            # for each movement increment          # its respective counter  **if** (move[i] **==** 'U'):              countUp **+=** 1    **elif**(move[i] **==** 'D'):              countDown **+=** 1    **elif**(move[i] **==** 'L'):              countLeft **+=** 1    **elif**(move[i] **==** 'R'):              countRight **+=** 1        # required final position of robot  **print**("Final Position: (", (countRight **-** countLeft),            ", ", (countUp **-** countDown), ")")      # Driver code  **if** \_\_name\_\_ **==** '\_\_main\_\_':      move **=** "UDDLLRUUUDUURUDDUULLDRRRR"      finalPosition(move)    # This code is contributed by 29AjayKumar |

**Output**

Final Position: (2, 3)

30.

31.

32. [Check loop in array according to given constraints - GeeksforGeeks](https://www.geeksforgeeks.org/check-loop-array-according-given-constraints/)

|  |
| --- |
| # Python3 program to check if a  # given array is cyclic or not    # A simple Graph DFS based recursive  # function to check if there is cycle  # in graph with vertex v as root of DFS.  # Refer below article for details.  # https:#www.geeksforgeeks.org/detect-cycle-in-a-graph/  **def** isCycleRec(v, adj, visited, recur):      visited[v] **=** True      recur[v] **=** True  **for** i **in** range(len(adj[v])):  **if** (visited[adj[v][i]] **==** False):  **if** (isCycleRec(adj[v][i], adj,                                 visited, recur)):  **return** True            # There is a cycle if an adjacent is visited          # and present in recursion call stack recur[]  **else** **if** (visited[adj[v][i]] **==** True **and**                  recur[adj[v][i]] **==** True):  **return** True        recur[v] **=** False  **return** False    # Returns true if arr[] has cycle  **def** isCycle(arr, n):        # Create a graph using given      # moves in arr[]      adj **=** [[] **for** i **in** range(n)]  **for** i **in** range(n):  **if** (i !**=** (i **+** arr[i] **+** n) **%** n):              adj[i].append((i **+** arr[i] **+** n) **%** n)        # Do DFS traversal of graph      # to detect cycle      visited **=** [False] **\*** n      recur **=** [False] **\*** n  **for** i **in** range(n):  **if** (visited[i] **==** False):  **if** (isCycleRec(i, adj,                             visited, recur)):  **return** True  **return** True    # Driver code  **if** \_\_name\_\_ **==** '\_\_main\_\_':        arr **=** [2, **-**1, 1, 2, 2]      n **=** len(arr)  **if** (isCycle(arr, n)):          print("Yes")  **else**:  **print**("No")    # This code is contributed by PranchalK |

**Output**

Yes

32.

33.

34. [Goldman Sachs Interview Experience | 1 year experienced - GeeksforGeeks](https://www.geeksforgeeks.org/goldman-sachs-interview-experience-1-year-experienced/)

35. [Program to generate all possible valid IP addresses from given string - GeeksforGeeks](https://www.geeksforgeeks.org/program-generate-possible-valid-ip-addresses-given-string/)

36. [Loading... (leetcode.com)](https://leetcode.com/discuss/interview-question/949160/find-the-most-frequent-ip-address-from-the-logs)

37. import re

from collections import Counter

class Solution():

def mostFrequentIP(self, lines):

cnt = Counter()

# regex can be changed as per the ip constraints

ipre = re.compile(r'[0-9]{2}\.[0-9]{1}\.[0-9]{1}\.[0-9]{1}')

for line in lines:

m = ipre.match(line)

if m is not None:

# considering 1 ip in each line

ip = m.group()

cnt[ip] += 1

return dict(cnt)

s = Solution()

lines = ["10.0.0.1 - GET 2020-08-24", "10.0.0.1 - GET 2020-08-24", "10.0.0.2 - GET 2020-08-20", "10.0.0.2 - GET 2020-08-20"]

print(s.mostFrequentIP(lines))

37. **Create a function that checks if the number is a power of 10.**

import math

def ispowerofTen(x):

    p = math.log10(x)

    if x == 10 \*\* math.floor(p):

        return True

    else:

        return False

1. Given a list, find the all valid timestamps (hrs:mins) in 24hrs digital clock.

Example :

Input : L = [1,2,3,4,5,6]

Output : 12:13, 12:14, 12:15, 12:16, 12:21, 12:23, 12:24, …

Constraints :

1)Number should not be repeated in combination of hrs:min. Example : As per Input 12:12,13:13,.. are not allowed.

2)Same Digit is also not allowed in hrs or mins part in timestamp calculation. Example : As per Input 11, 22, 33, … are not allowed. So in timestamp 11:xx, 22:xx, xx:11, xx:22, xx:33, xx:44, xx:55 are not allowed.

***hr=[]***

***mn=[]***

***for part in l:***

***if(part>2):***

***continue***

***for part1 in l:***

***if(part==part1):***

***continue***

***elif(part ==2 and part1>4):***

***continue***

***else:***

***hr.append(str(part)+str(part1))***

***print(hr)***

***for part in l:***

***if(part>6):***

***continue***

***for part1 in l:***

***if(part==part1):***

***continue***

***elif(part ==6 and part1>0):***

***continue***

***else:***

***mn.append(str(part)+str(part1))***

***print(mn)***

***for part in hr:***

***for part1 in mn:***

***if(part==part1):***

***continue***

***print(part+":"+part1)***

def ip\_add (1):

max = 1

max\_ip-[]

d-dict ()

for i in 1:

ip = i[:i.index (" ")]

if ip in d.keys():

d[ip]= d[ip]+1

if d[ip]>max:

max = d[ip]

else:

d[ip] - 1

for key, values in d.items():

if values == max:

max\_ip.append (key)

return max ip