| **SORU** | **SORU İÇERİĞİ & ÇÖZÜMÜ** | **METOTLAR & ANAHTAR KELİMELER** |
| --- | --- | --- |
| **Ex - 101** | Create a function that retrieves every number that is strictly larger than every number that follows it.  Examples  [3, 13, 11, 2, 1, 9, 5] ➞ [13, 11, 9, 5]   * 13 is larger than all numbers to its right, etc.   [5, 5, 5, 5, 5, 5] ➞ [5]   * Must be strictly larger. * Always include the last number.   [5, 9, 8, 7] ➞ [9, 8, 7]  Notes  The last number in an array is trivially strictly larger than all numbers that follow it (no numbers follow it).  n, sub\_list = [3, 13, 11, 2, 1, 9, 5], []  if len(set(n)) == 1 :    print(n[-1])  else :    for i in n :      if i >= max(n[n.index(i)::]) :        sub\_list.append(i)    print(sub\_list)  >>> [13, 11, 9, 5] | **?len()**  **?set()**  **?max()**  **?index()**  **?append()** |
| **Ex - 102** | Given a list of words in the singular form, return a set of those words in the plural form if they appear more than once in the list.  Examples  ["cow", "pig", "cow", "cow"] ➞ { "cows", "pig" }  ["table", "table", "table"] ➞ { "tables" }  ["chair", "pencil", "arm"] ➞ { "chair", "pencil", "arm" }  Notes This is an oversimplification of the English language so no edge cases will appear. Only focus on whether or not to add an s to the ends of words.  liste = ["cow", "pig", "cow", "cow"]  listem = []  for i in liste :    if liste.count(i) == 1 :      listem.append(i)    elif liste.count(i) > 1:      i += "s"      listem.append(i)  print(set(listem))  >>> {'cows', 'pig'}  arry = ["cow", "pig", "cow", "cow"]  set([i + "s" if arry.count(i) > 1 else i for i in arry])  >>> {'cows', 'pig'} | **?comprehension**  **?count()**  **?set()**  **?append()** |
| **Ex - 103** | Write a Python program to calculate the sum of three given numbers, if the values are equal then return three times of their sum.  numbers = list(map(int, input("Please enter 3 numbers: ").split()))  print(sum(numbers) \* 3 if numbers[0] == numbers[1] == numbers[2] else sum(numbers))  >>> Please enter 3 numbers: 5 5 5  >>> 45 | **?list()**  **?input()**  **?map()**  **?split()**  **?sum()**  **?comprehension** |
| **Ex - 104** | Given a string s, reverse only all the vowels in the string and return it. The vowels are 'a', 'e', 'i', 'o', and 'u', and they can appear in both cases.  Example 1:  Input: s = "hello"  Output: "holle"  Example 2:  Input: s = "leetcode"  Output: "leotcede"  text = "leetcode"  vowels = "aeiou"  text\_vowels = [x for x in text if x in vowels]  print("".join([x if x not in vowels else text\_vowels.pop(-1) for x in text]))  >>> loetcede  word, vowels = input(""), "aeiou"  word\_vowels = []  for i in word :    if i in vowels :      word\_vowels += [i]  output= ""  for j in word :    if j not in vowels :      output += j    else :      output += word\_vowels.pop(-1)  print(output)  >>> hello  >>> holle  s = "hello"  indeks = [i for i in range(len(s)) if s[i] in "AEIOUaeiou"]  consonants = [j for j in s if j not in "AEIOUaeiou" ]  vowels = [k for k in s if k in "AEIOUaeiou"][::-1]  for i in range(len(indeks)) :    consonants.insert(indeks[i], vowels[i])  print("".join(consonants))  >>> holle | **?join()**  **?input()**  **?pop()**  **?for loop**  **?range()**  **?len()**  **?comprehension** |
| **Ex - 105** | Create a string of up to 1000 numbers, then create a pyhton code that finds how many 3 digits occur in this string of numbers.  print(len([i for i in range(1, 1000) if "3" in str(i)]))  >>> 271  liste = []  for i in range(1, 1000) :    if "3" in str(i) :      liste.append(i)  print(len(liste))  >>> 271 | **?len()**  **?range()**  **?append()**  **?comprehension** |
| **Ex - 106** | Write a Python program to count the number of characters (character frequency) in a string.  Sample String : 'google.com'  Expected Result : {'g': 2, 'o': 3, 'l': 1, 'e': 1, '.': 1, 'c': 1, 'm': 1}  x, my\_dict = input("Please enter your input: "), {}  for i in x:    if i not in my\_dict :      my\_dict[i] = 1    else :      my\_dict[i] += 1  print(my\_dict)  >>> Please enter your input: clarusway  >>> {'c': 1, 'l': 1, 'a': 2, 'r': 1, 'u': 1, 's': 1, 'w': 1, 'y': 1} | **?for loop**  **?input()** |
| **Ex - 107** | Robot Return to Origin  There is a robot starting at the position  (0, 0)  , the origin, on a 2D plane. Given a sequence of its moves, judge if this robot **ends up** at (0, 0) after it completes its moves.  You are given a string moves that represents the move sequence of the robot where moves[i] represents its ith move. Valid moves are 'R' (right), 'L' (left), 'U' (up), and 'D' (down).  Return True if the robot returns to the origin after it finishes all of its moves, or False otherwise.  **Note:** The way that the robot is "facing" is irrelevant. 'R' will always make the robot move to the right once, 'L' will always make it move left, etc. Also, assume that the magnitude of the robot's movement is the same for each move.  **Example 1:**  Input: moves = "UD"  Output: True  Explanation: The robot moves up once, and then down once. All moves have the same magnitude, so it ended up at the origin where it started. Therefore, we return true.  **Example 2:**  Input: moves = "LL"  Output: False  Explanation: The robot moves left twice. It ends up two "moves" to the left of the origin. We return false because it is not at the origin at the end of its moves.  **Example 3:**  Input: moves = "RRDD"  Output: False  **Example 4:**  Input: moves = "LDRRLRUULR  Output: False  moves = "LLLUUD"  x,y = 0 , 0  for i in moves :    if i == "L" : x += 1    elif i == "R" : x -= 1    elif i == "D" : y -= 1    elif i == "U" : y += 1  print(True if x == 0 and y == 0 else False)  >>> False  moves = "LDRRLRUULR"  print((moves.count('L') == moves.count('R')) & (moves.count('D') == moves.count('U')))  >>> False | **?for loop**  **?count()** |
| **Ex - 108** | Fair Candy Swap  Alice and Bob have a different total number of candies. You are given two integer arrays AliceSizes and bobSizes where aliceSizes[i] is the number of candies of the ith box of candy that Alice has and bobSizes[j] is the number of candies of the jth box of candy that Bob has.  Since they are friends, they would like to exchange one candy box each so that after the exchange, they both have the same total amount of candy. The total amount of candy a person has is the sum of the number of candies in each box they have.  Return an integer array answer where answer[0] is the number of candies in the box that Alice must exchange, and answer[1] is the number of candies in the box that Bob must exchange. If there are multiple answers, you may return any one of them. It is guaranteed that at least one answer exists.  Example 1:  Input: aliceSizes = [1,1], bobSizes = [2,2]  Output: [1,2]  Example 2:  Input: aliceSizes = [1,2], bobSizes = [2,3]  Output: [1,2]  Example 3:  Input: aliceSizes = [2], bobSizes = [1,3]  Output: [2,3]  Example 4:  Input: aliceSizes = [1,2,5], bobSizes = [2,4]  Output: [5,4]  Constraints:  Alice and Bob have a different total number of candies.  There will be at least one valid answer for the given input.  aliceSizes = [1,2,5]  bobSizes = [2,4]  need\_alice = (sum(aliceSizes) + sum(bobSizes)) // 2 - sum(aliceSizes)  for i in aliceSizes :    temp = i + need\_alice    if temp in set(bobSizes) :      print([i,temp])      break  >>> [5, 4] | **?sum()**  **?set()**  **?for loop** |
| **Ex - 109** | Given two arrays of strings a1 and a2 return a sorted array r in lexicographical order of the strings of a1 which are substrings of strings of a2.  Example 1:  a1 = ["arp", "live", "strong"]  a2 = ["lively", "alive", "harp", "sharp", "armstrong"]  returns ["arp", "live", "strong"]  Example 2:  a1 = ["tarp", "mice", "bull"]  a2 = ["lively", "alive", "harp", "sharp", "armstrong"]  returns []  Notes:  Arrays are written in "general" notation. See "Your Test Cases" for examples in your language.  In Shell bash a1 and a2 are strings. The return is a string where words are separated by commas.  Beware: r must be without duplicates.  a1 = ["arp", "live", "strong", "arp"]  a2 = ["lively", "alive", "harp", "sharp", "armstrong"]  liste = []  for i in a1 :    for j in a2 :      if i in j :        if i in liste :          continue        else :          liste.append(i)  print(liste)  >>> ['arp', 'live', 'strong']  a1 = ["arp", "live", "strong"]  a2 = ["lively", "alive", "harp", "sharp", "armstrong"]  print(sorted(list({i for i in a1 for j in a2 if set(i).issubset(set(j))})))  >>> ['arp', 'live', 'strong'] | **?nested for loop ?comprehension** |
|  | a1 = ["arp", "live", "strong"]  a2 = ["lively", "alive", "harp", "sharp", "armstrong"]  print(sorted(set(list(i for i in a1 for j in a2 if set(i).issubset(set(j))))))  >>> ['arp', 'live', 'strong']  mylist = []  a1 = ["arp", "live", "strong"]  a2 = ["lively", "alive", "harp", "sharp", "armstrong"]  for i in a1 :    for k in a2 :      if i in k :        mylist.append(i)  print(sorted(list(set(mylist))))  >>> ['arp', 'live', 'strong'] |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |