**PYTHON TEST 24DEC (MANDATORY)**

* **TEST TIME => 2 HOUR 30 MINUTES.**
* **SOLVE ALL THESE QUESTIONS AND CREATE A REPOSITORY ON GITHUB BY NAME YOUR\_NAME\_PYTHON\_TEST\_24 AND PUSH IT ON GITHUB.**
* **FILL THE GOOGLE FORM WITH YOUR DETAILS - https://forms.gle/CSMkwHTHNw3pm6Av7**
* **TOP PERFORMERS NAMES WILL BE SENT TO THE CIPHER TEAM FOR RECOGNITION**
* **FORM LINK WILL  BE CLOSED BY 3pm**

1. """You can use this class to represent how classy someone

or something is.

"Classy" is interchangable with "fancy".

If you add fancy-looking items, you will increase

your "classiness".

Create a function in "Classy" that takes a string as

input and adds it to the "items" list.

Another method should calculate the "classiness"

value based on the items.

The following items have classiness points associated

with them:

"tophat" = 2

"bowtie" = 4

"monocle" = 5

Everything else has 0 points.

Use the test cases below to guide you!"""

class Classy(object):

    def \_\_init\_\_(self):

        self.items = []

# Test cases

me = Classy()

# Should be 0

print me.getClassiness()

me.addItem("tophat")

# Should be 2

print me.getClassiness()

me.addItem("bowtie")

me.addItem("jacket")

me.addItem("monocle")

# Should be 11

print me.getClassiness()

me.addItem("bowtie")

# Should be 15

print me.getClassiness()

class Classy(object):  
 def \_\_init\_\_(self):  
 self.items = []  
 self.classiness = 0  
  
 def getClassiness(self):  
 try:  
 for i in self.items:  
 if i == "tophat":  
 self.classiness += 2  
 elif i == "bowtie":  
 self.classiness += 4  
 elif i == "monocle":  
 self.classiness += 5  
 self.items.pop(0)  
 self.items.pop(0)  
 except IndexError as e:  
 pass  
 return self.classiness  
  
 def addItem(self, item):  
 self.items.append(item)  
  
me = Classy()  
  
  
print (me.getClassiness())  
  
me.addItem("tophat")  
  
print (me.getClassiness())  
  
me.addItem("bowtie")  
me.addItem("jacket")  
me.addItem("monocle")  
  
print (me.getClassiness())  
  
me.addItem("bowtie")  
  
print (me.getClassiness())

2.# Write a function called "show\_excitement" where the string

# "I am super excited for this course!" is returned exactly

# 5 times, where each sentence is separated by a single space.

# Return the string with "return".

# You can only have the string once in your code.

# Don't just copy/paste it 5 times into a single variable!

def show\_excitement():

    # Your code goes here!

    return ""

print show\_excitement()

def show\_excitement():  
 for i in range(1, 6):  
 print ("I am super excited for this course! ")  
 return " "  
  
print (show\_excitement())

3.Create a **Bus** child class that inherits from the Vehicle class. The default fare charge of any vehicle is **seating capacity \* 100**. If Vehicle is **Bus** instance, we need to add an extra 10% on full fare as a maintenance charge. So total fare for bus instance will become the **final amount = total fare + 10% of the total fare.**

Note: The bus seating capacity is **50**. so the final fare amount should be **5500.** You need to override the fare() method of a Vehicle class in Bus class.

Use the following code for your parent Vehicle class. We need to access the parent class from inside a method of a child class.

class Vehicle:

    def \_\_init\_\_(self, name, mileage, capacity):

        self.name = name

        self.mileage = mileage

        self.capacity = capacity

    def fare(self):

        return self.capacity \* 100

class Bus(Vehicle):

    pass

School\_bus = Bus("School Volvo", 12, 50)

print("Total Bus fare is:", School\_bus.fare())

**Expected Output**:

Total Bus fare is: 5500.0

class Vehicle:  
 def \_\_init\_\_(self, name, mileage, capacity):  
 self.name = name  
 self.mileage = mileage  
 self.capacity = capacity  
  
 def fare(self):  
 return self.capacity \* 100  
  
  
class Bus(Vehicle):  
  
 def fare(self):  
 fare\_car = self.capacity \* 100  
 total\_fare = fare\_car + (0.1 \* fare\_car)  
 return total\_fare  
  
  
School\_bus = Bus("School Volvo", 12, 50)  
print("Total Bus fare is:", School\_bus.fare())

**4.: Rename key of a dictionary**

Write a program to rename a key city to a location in the following dictionary.

**Given**:

sample\_dict = {

  "name": "Kelly",

  "age":25,

  "salary": 8000,

  "city": "New york"

}

**Expected output:**

{'name': 'Kelly', 'age': 25, 'salary': 8000, 'location': 'New york'}

sample\_dict = {  
 "name": "Kelly",  
 "age": 25,  
 "salary": 8000,  
 "city": "New york"  
}  
  
sample\_dict['location'] = sample\_dict.pop('city')  
print(sample\_dict)

5. Given an array **arr[]** of integers. Find a peak element i.e. an element that is **not smaller** than its neighbors.

**Note:** For corner elements, we need to consider only one neighbor

***Example:***

***Input:*** *array[]= {5, 10, 20, 15}*

***Output:*** *20*

***Explanation:*** *The element 20 has neighbors 10 and 15, both of them are less than 20.*

***Input:*** *array[] = {10, 20, 15, 2, 23, 90, 67}*

***Output:*** *20 or 90*

***Explanation:*** *The element 20 has neighbors 10 and 15, both of them are less than 20, similarly 90 has neighbors 23 and 67.*

The following corner cases give a better idea about the problem.

1. If the input array is sorted in a strictly increasing order, the last element is always a peak element. For example, 50 is peak element in {10, 20, 30, 40, 50}.
2. If the input array is sorted in a strictly decreasing order, the first element is always a peak element. 100 is the peak element in {100, 80, 60, 50, 20}.
3. If all elements of the input array are the same, every element is a peak element.

It is clear from the above examples that there is always a peak element in the input array.

def findPeak(arr, n):  
 if (n == 1):  
 return 0  
 if (arr[0] >= arr[1]):  
 return 0  
 if (arr[n - 1] >= arr[n - 2]):  
 return n - 1  
  
 for i in range(1, n - 1):  
  
 if (arr[i] >= arr[i - 1] and arr[i] >= arr[i + 1]):  
 return i  
  
  
arr = [1, 3, 20, 4, 1, 0]  
n = len(arr)  
print("Index of a peak point is", findPeak(arr, n))

6.Given an array and a number **K** where **K** is smaller than the size of the array. Find the K’th smallest element in the given array. Given that all array elements are distinct.

**Examples:**

***Input****: arr[] = {7, 10, 4, 3, 20, 15}, K = 3*

***Output****: 7*

***Input****: arr[] = {7, 10, 4, 3, 20, 15}, K = 4*

***Output****: 10*

def kthSmallest(arr, n, k):  
   
 arr.sort()  
  
   
 return arr[k - 1]  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 arr = [7, 10, 4, 3, 20, 15]  
 n = len(arr)  
 k = 3  
   
  
 print("K'th smallest element is",  
 kthSmallest(arr, n, k))

7.Given an array of **N** integers, and a number **sum,** the task is tofind the **number of pairs** of integers in the array whose sum is equal to sum.

**Examples:**

***Input:****arr[] = {1, 5, 7, -1}, sum = 6*

***Output:****2*

***Explanation:*** *Pairs with sum 6 are (1, 5) and (7, -1).*

***Input:****arr[] = {1, 5, 7, -1, 5}, sum = 6*

***Output:****3*

***Explanation:*** *Pairs with sum 6 are (1, 5), (7, -1) & (1, 5).*

***Input:****arr[] = {1, 1, 1, 1}, sum = 2*

***Output:****6*

***Explanation:*** *Pairs with sum 2 are (1, 1), (1, 1), (1, 1), (1, 1), (1, 1).*

***Input:****arr[] = {10, 12, 10, 15, -1, 7, 6, 5, 4, 2, 1, 1, 1}, sum = 11*

***Output:****9*

***Explanation:*** *Pairs with sum 11 are (10, 1), (10, 1), (10, 1), (12, -1), (10, 1), (10, 1), (10, 1), (7, 4), (6, 5).*

def getPairsCount(arr, n, sum):  
 count = 0  
 for i in range(0, n):  
 for j in range(i + 1, n):  
 if arr[i] + arr[j] == sum:  
 count += 1  
  
 return count  
  
  
  
arr = [1, 5, 7, -1, 5]  
n = len(arr)  
sum = 6  
print("Count of pairs is",  
 getPairsCount(arr, n, sum))

*8.An array contains both positive and negative numbers in random order. Rearrange the array elements so that all negative numbers appear before all positive numbers.*

***Examples :***

***Input:*** *-12, 11, -13, -5, 6, -7, 5, -3, -6*

***Output:*** *-12 -13 -5 -7 -3 -6 11 6 5*

def rearrange(arr, n):  
 j = 0  
 for i in range(0, n):  
 if (arr[i] < 0):  
 temp = arr[i]  
 arr[i] = arr[j]  
 arr[j] = temp  
 j = j + 1  
 print(arr)  
  
  
arr = [-12, 11, -13, -5, 6, -7, 5, -3, -6]  
n = len(arr)  
rearrange(arr, n)

*9.Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.*

*You may assume that each input would have exactly one solution, and you may not use the same element twice.*

*You can return the answer in any order.*

*Example 1:*

*Input: nums = [2,7,11,15], target = 9*

*Output: [0,1]*

*Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].*

*Example 2:*

*Input: nums = [3,2,4], target = 6*

*Output: [1,2]*

*Example 3:*

*Input: nums = [3,3], target = 6*

*Output: [0,1]*

*Constraints:*

* *2 <= nums.length <= 104*
* *-109 <= nums[i] <= 109*
* *-109 <= target <= 109*
* *Only one valid answer exists.*

*10.The product difference between two pairs (a, b) and (c, d) is defined as (a \* b) - (c \* d).*

* *For example, the product difference between (5, 6) and (2, 7) is (5 \* 6) - (2 \* 7) = 16.*

*Given an integer array nums, choose four distinct indices w, x, y, and z such that the product difference between pairs (nums[w], nums[x]) and (nums[y], nums[z]) is maximized.*

*Return the maximum such product difference.*

*Example 1:*

*Input: nums = [5,6,2,7,4]*

*Output: 34*

*Explanation: We can choose indices 1 and 3 for the first pair (6, 7) and indices 2 and 4 for the second pair (2, 4).*

*The product difference is (6 \* 7) - (2 \* 4) = 34.*

*Example 2:*

*Input: nums = [4,2,5,9,7,4,8]*

*Output: 64*

*Explanation: We can choose indices 3 and 6 for the first pair (9, 8) and indices 1 and 5 for the second pair (2, 4).*

*The product difference is (9 \* 8) - (2 \* 4) = 64.*

*Constraints:*

* *4 <= nums.length <= 104*
* *1 <= nums[i] <= 104*

*11.A sentence is a list of words that are separated by a single space with no leading or trailing spaces.*

*You are given an array of strings sentences, where each sentences[i] represents a single sentence.*

*Return the maximum number of words that appear in a single sentence.*

*Example 1:*

*Input: sentences = ["alice and bob love leetcode", "i think so too", "this is great thanks very much"]*

*Output: 6*

*Explanation:*

*- The first sentence, "alice and bob love leetcode", has 5 words in total.*

*- The second sentence, "i think so too", has 4 words in total.*

*- The third sentence, "this is great thanks very much", has 6 words in total.*

*Thus, the maximum number of words in a single sentence comes from the third sentence, which has 6 words.*

*Example 2:*

*Input: sentences = ["please wait", "continue to fight", "continue to win"]*

*Output: 3*

*Explanation: It is possible that multiple sentences contain the same number of words.*

*In this example, the second and third sentences (underlined) have the same number of words.*

*Constraints:*

* *1 <= sentences.length <= 100*
* *1 <= sentences[i].length <= 100*
* *sentences[i] consists only of lowercase English letters and ' ' only.*
* *sentences[i] does not have leading or trailing spaces.*
* *All the words in sentences[i] are separated by a single space.*

12. Balanced strings are those that have an equal quantity of 'L' and 'R' characters.

Given a balanced string s, split it into some number of substrings such that:

* Each substring is balanced.

Return *the maximum number of balanced strings you can obtain.*

Example 1:

Input: s = "RLRRLLRLRL"

Output: 4

Explanation: s can be split into "RL", "RRLL", "RL", "RL", each substring contains same number of 'L' and 'R'.

Example 2:

Input: s = "RLRRRLLRLL"

Output: 2

Explanation: s can be split into "RL", "RRRLLRLL", each substring contains same number of 'L' and 'R'.

Note that s cannot be split into "RL", "RR", "RL", "LR", "LL", because the 2nd and 5th substrings are not balanced.

Example 3:

Input: s = "LLLLRRRR"

Output: 1

Explanation: s can be split into "LLLLRRRR".

Constraints:

* 2 <= s.length <= 1000
* s[i] is either 'L' or 'R'.
* s is a balanced string.

def BalancedPartition(str1, n):  
 if (n == 0):  
 return 0  
  
 r = 0  
 l = 0  
  
 ans = 0  
  
 for i in range(n):  
  
 if (str1[i] == 'R'):  
 r += 1  
  
 elif (str1[i] == 'L'):  
 l += 1  
  
 if (r == l):  
 ans += 1  
  
 return ans  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 str1 = "RLRRLLRLRL"  
 n = len(str1)  
  
 print(BalancedPartition(str1, n))

13. Given an integer n, return *a string array* answer *(1-indexed) where*:

* answer[i] == "FizzBuzz" if i is divisible by 3 and 5.
* answer[i] == "Fizz" if i is divisible by 3.
* answer[i] == "Buzz" if i is divisible by 5.
* answer[i] == i (as a string) if none of the above conditions are true.

Example 1:

Input: n = 3

Output: ["1","2","Fizz"]

Example 2:

Input: n = 5

Output: ["1","2","Fizz","4","Buzz"]

Example 3:

Input: n = 15

Output: ["1","2","Fizz","4","Buzz","Fizz","7","8","Fizz","Buzz","11","Fizz","13","14","FizzBuzz"]

Constraints:

* 1 <= n <= 104

14.Given a list of numbers of list, write a Python program to create a list of tuples having first element as the number and second element as the cube of the number. **Example:**

Input: list = [1, 2, 3]

Output: [(1, 1), (2, 8), (3, 27)]

Input: list = [9, 5, 6]

Output: [(9, 729), (5, 125), (6, 216)]

list1 = [9, 5, 6]  
  
res = [(val, val \*\* 3) for val in list1]  
  
print(res)

15.With a given integral number n, write a program to generate a dictionary that contains (i, i\*i) such that is an integral number between 1 and n (both included). and then the program should print the dictionary.

|  |
| --- |
| Suppose the following input is supplied to the program: |
| 8 |
| Then, the output should be: |
| {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64} |

number = int(input("Type a number: "))  
  
numberDict = {}  
for i in range(1, number+1):  
 numberDict[i] = i\*i  
  
print(numberDict)