**STRINGS 2**

**1) Given a string, return a string where for every char in the original, there are two chars.**

**double\_char('The') → 'TThhee'**

**double\_char('AAbb') → 'AAAAbbbb'**

**double\_char('Hi-There') → 'HHii--TThheerree'**

**def double\_char(str):**

**i=0**

**res=''**

**while i<len(str):**

**res=res+str[i]+str[i]**

**i+=1**

**return res**

**2) Return the number of times that the string "code" appears anywhere in the given string, except we'll accept any letter for the 'd', so "cope" and "cooe" count.**

**count\_code('aaacodebbb') → 1**

**count\_code('codexxcode') → 2**

**count\_code('cozexxcope') → 2**

**def count\_code(str):**

**i=0**

**count=0**

**while i<len(str)-3:**

**if str[i]=='c' and str[i+1]=='o' and str[i+3]=='e':**

**count=count+1**

**i+=1**

**return count**

**3) Return the number of times that the string "hi" appears anywhere in the given string.**

**count\_hi('abc hi ho') → 1**

**count\_hi('ABChi hi') → 2**

**count\_hi('hihi') → 2**

**def count\_hi(str):**

**return str.count('hi')**

**4) Given two strings, return True if either of the strings appears at the very end of the other string, ignoring upper/lower case differences (in other words, the computation should not be "case sensitive"). Note: s.lower() returns the lowercase version of a string.**

**end\_other('Hiabc', 'abc') → True**

**end\_other('AbC', 'HiaBc') → True**

**end\_other('abc', 'abXabc') → True**

**def end\_other(a, b):**

**f=a.lower()**

**s=b.lower()**

**if f.endswith(s) or s.endswith(f):**

**return True**

**else:**

**return False**

**5) Return True if the string "cat" and "dog" appear the same number of times in the given string.**

**cat\_dog('catdog') → True**

**cat\_dog('catcat') → False**

**cat\_dog('1cat1cadodog') → True**

**def cat\_dog(str):**

**return (str.count('cat') == str.count('dog'))**

**6)**

|  |
| --- |
| **Return True if the given string contains an appearance of "xyz" where the xyz is not directly preceeded by a period (.). So "xxyz" counts but "x.xyz" does not.**  **xyz\_there('abcxyz') → True**  **xyz\_there('abc.xyz') → False**  **xyz\_there('xyz.abc') → True** |

**def xyz\_there(str):**

**l = len(str)**

**i = 0**

**while i<l-2:**

**if str[i] == '.' and str[i+1] == 'x':**

**i+=2**

**continue**

**elif str[i] == 'x' and str[i+1] == 'y' and str[i+2] == 'z':**

**return True**

**i+=1**

**return False**

**LIST2**

1. **Return the number of even ints in the given array. Note: the % "mod" operator computes the remainder, e.g. 5 % 2 is 1.**

**count\_evens([2, 1, 2, 3, 4]) → 3**

**count\_evens([2, 2, 0]) → 3**

**count\_evens([1, 3, 5]) → 0**

**def count\_evens(nums):**

**i=0**

**count=0**

**while i<len(nums):**

**if nums[i]%2==0:**

**count=count+1**

**i+=1**

**return count**

**2) Return the sum of the numbers in the array, returning 0 for an empty array. Except the number 13 is very unlucky, so it does not count and numbers that come immediately after a 13 also do not count.**

**sum13([1, 2, 2, 1]) → 6**

**sum13([1, 1]) → 2**

**sum13([1, 2, 2, 1, 13]) → 6**

**def sum13(nums):**

**l=len(nums)**

**i=0**

**sum=0**

**if l==0:**

**return 0**

**temp = 0**

**while i<len(nums):**

**if nums[i]==13:**

**i+=2**

**continue**

**else:**

**sum=sum+nums[i]**

**i+=1**

**return sum**

**3)**

|  |
| --- |
| **Given an array length 1 or more of ints, return the difference between the largest and smallest values in the array. Note: the built-in min(v1, v2) and max(v1, v2) functions return the smaller or larger of two values.**  **big\_diff([10, 3, 5, 6]) → 7**  **big\_diff([7, 2, 10, 9]) → 8**  **big\_diff([2, 10, 7, 2]) → 8** |

**def big\_diff(nums):**

**l=len(nums)**

**min\_value=min(nums)**

**max\_value=max(nums)**

**return max\_value-min\_value**

**4)**

|  |
| --- |
| **Return the sum of the numbers in the array, except ignore sections of numbers starting with a 6 and extending to the next 7 (every 6 will be followed by at least one 7). Return 0 for no numbers.**  **sum67([1, 2, 2]) → 5**  **sum67([1, 2, 2, 6, 99, 99, 7]) → 5**  **sum67([1, 1, 6, 7, 2]) → 4** |

**def sum67(nums):**

**l = len(nums)**

**if l == 0:**

**return 0**

**ignore = False**

**sum = 0**

**for i in nums:**

**if i == 6 and not ignore:**

**ignore = True**

**continue**

**elif i == 7 and ignore:**

**ignore = False**

**continue**

**elif ignore:**

**continue**

**else:**

**sum+=i**

**return sum**

**5)**

|  |
| --- |
| **Return the "centered" average of an array of ints, which we'll say is the mean average of the values, except ignoring the largest and smallest values in the array. If there are multiple copies of the smallest value, ignore just one copy, and likewise for the largest value. Use int division to produce the final average. You may assume that the array is length 3 or more.**  **centered\_average([1, 2, 3, 4, 100]) → 3**  **centered\_average([1, 1, 5, 5, 10, 8, 7]) → 5**  **centered\_average([-10, -4, -2, -4, -2, 0]) → -3** |

**def centered\_average(nums):**

**largestValue=max(nums)**

**smallestValue=min(nums)**

**i=0**

**avg=0**

**sum=0**

**met\_smallest = False**

**met\_largest = False**

**for l in nums:**

**if l==smallestValue and not met\_smallest:**

**met\_smallest = True**

**continue**

**elif l == largestValue and not met\_largest:**

**met\_largest = True**

**continue**

**else:**

**sum+=l**

**avg=(sum//(len(nums)-2))**

**return avg**

**6) Given an array of ints, return True if the array contains a 2 next to a 2 somewhere.**

**has22([1, 2, 2]) → True**

**has22([1, 2, 1, 2]) → False**

**has22([2, 1, 2]) → False**

**def has22(nums):**

**l=len(nums)**

**i=0**

**while i<l-1:**

**if nums[i]==2 and nums[i+1]==2:**

**return True**

**i+=1**

**return False**