In [1]:
```
"""You are given two strings word1 and word2. Merge the strings by adding letters in a
order, starting with word1. If a string is longer than the other, append the additional
end of the merged string."""

def mergeAlternately(word1, word2):
    """
    :type word1: str
    :type word2: str
    :rtype: str
    """
    merged = []
    i, j = 0,0

    while i < len(word1) and j < len(word2):
        merged.append(word1[i])
        merged.append(word2[j])
        i += 1
        j += 1

    merged.extend(word1[i:])
    merged.extend(word2[j:])

    return ''.join(merged)
```

In [2]:
```
mergeAlternately('abc','xyz')
```

Out[2]: 'axbycz'

In [3]:
```
"""String t is generated by random shuffling string s and then add one more letter at a

from collections import Counter

def findTheDifference(s, t):
        """
        :type s: str
        :type t: str
        :rtype: str
        """
        count_s = Counter(s)
        count_t = Counter(t)

        for char in count_t:
            if count_t[char] != count_s.get(char, 0):
                return char
```

In [4]:
```
findTheDifference('abc','abcd')
```

Out[4]: 'd'

In [5]:
```
"""String t is generated by random shuffling string s and then add one more letter at a


def findTheDifference(s, t):
```

```
    """
    :type s: str
    :type t: str
    :rtype: str
    """
    record = {}

    for c in s:
        if c in record:
            record[c] += 1
        else:
            record[c] = 1

    for c in t:
        if record.get(c, 0) == 0:
            return c
        else:
            record[c] -= 1
```

In [6]:
```
findTheDifference('abc','abcd')
```

Out[6]: 'd'

In [7]:
```
"""Given two strings needle and haystack, return the index of the first occurrence of n
haystack, or -1 if needle is not part of haystack."""


def strStr(haystack, needle):
    """
    :type haystack: str
    :type needle: str
    :rtype: int
    """
    if needle in haystack:
        for i in range (0,len(haystack)):
            if haystack[i:(i+len(needle))]==needle:
                print(i)
                break
        return(i)
    else:
        return(-1)
```

In [8]:
```
strStr('European','Euro')
```

```
0
```

Out[8]: 0

In [9]:
```
"""Given two strings s and t, return true if t is an
anagram of s, and false otherwise"""

def isAnagram(s, t):
    """
    :type s: str
    :type t: str
```

```
    :rtype: bool
    """
    return Counter(s) == Counter(t)
```

In [10]:
```
isAnagram("anagram", "nagaram")
```

Out[10]: True

In [11]:
```
def isAnagram(s, t):
    """
    :type s: str
    :type t: str
    :rtype: bool
    """
    record = {}

    for c in s:
        if c in record:
            record[c] += 1
        else:
            record[c] = 1

    for c in t:
        if record.get(c, 0) == 0:
            return False
        else:
            record[c] -= 1

    return sum(record.values()) == 0
```

In [12]:
```
isAnagram("anagram", "nagaram")
```

Out[12]: True

In [13]:
```
"""Given a string s, check if it can be constructed by taking a substring of it and appe
copies of the substring together."""

def repeatedSubstringPattern(s):
    """
    :type s: str
    :rtype: bool
    """
    return s in (s + s)[1:-1]
```

In [14]:
```
repeatedSubstringPattern('abcdeabcdeabcde')
```

Out[14]: True

In [15]:
```
"""Given an integer array nums, move all 0's to the end of it while maintaining the rela
non-zero elements."""
def moveZeroes(nums):
    """
```

```
    :type nums: List[int]
    :rtype: None Do not return anything, modify nums in-place instead.
    """
    j = 0  # Pointer to place the next non-zero element

    # Move non-zero elements to the front
    for i in range(len(nums)):
        if nums[i] != 0:
            nums[j] = nums[i]
            j += 1

    # Fill the rest of the array with zeros
    for i in range(j, len(nums)):
        nums[i] = 0
```

In [16]:
```
nums = [0,0,15,5,8,7,2]
moveZeroes(nums)
```

In [17]:
```
"""You are given a large integer represented as an integer array digits, where each dig
the ith digit of the integer. The digits are ordered from most significant to least sig
to-right order. The large integer does not contain any leading 0's."""

def plusOne(digits):
    """
    :type digits: List[int]
    :rtype: List[int]
    """
    num = ""
    for item in digits:
        num += str(item)
    num = int(num) + 1
    answer = [int(digit) for digit in str(num)]
    return answer
```

In [18]:
```
plusOne([1,2,3])
```

Out[18]:
```
[1, 2, 4]
```

In [19]:
```
"""Implement a function signFunc(x) that returns:

1 if x is positive.
-1 if x is negative.
0 if x is equal to 0."""

def arraySign(nums):
    """
    :type nums: List[int]
    :rtype: int
    """
    sum = 1
    for i in nums:
        if i == 0:
            return 0
        sum *= i
```

```
    return 1 if sum > 0 else -1
```

In [20]:
```
arraySign([1,2,5,-5])
```

Out[20]: -1

In [21]:
```
"""A sequence of numbers is called an arithmetic progression if the difference between a
consecutive elements is the same."""

def canMakeArithmeticProgression(arr):
    """
    :type arr: List[int]
    :rtype: bool
    """
    if len(arr) < 2:
        return True

    arr.sort()

    diff = arr[1] - arr[0]

    for i in range(2, len(arr)):
        if arr[i] - arr[i - 1] != diff:
            return False

    return True
```

In [22]:
```
canMakeArithmeticProgression([3,5,1])
```

Out[22]: True

In [23]:
```
"""An array is monotonic if it is either monotone increasing or monotone decreasing."""

def isMonotonic(nums):
    """
    :type nums: List[int]
    :rtype: bool
    """
    increasing = True
    decreasing = True

    for i in range(1, len(nums)):
        if nums[i] > nums[i - 1]:
            decreasing = False
        elif nums[i] < nums[i - 1]:
            increasing = False

    return increasing or decreasing
```

In [24]:
```
isMonotonic([1,5,9])
```

Out[24]: True

In [ ]: