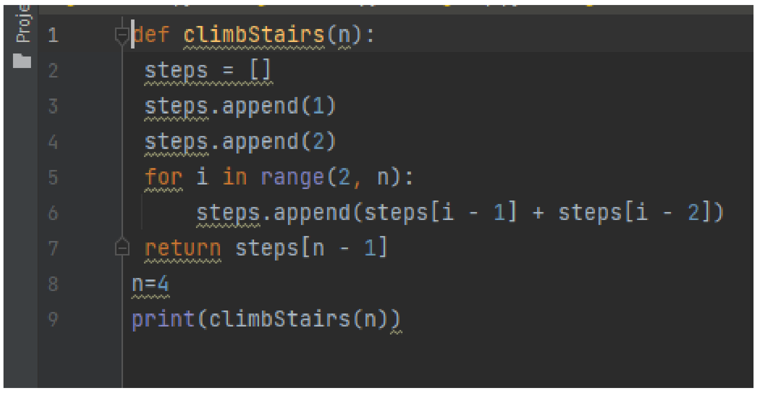
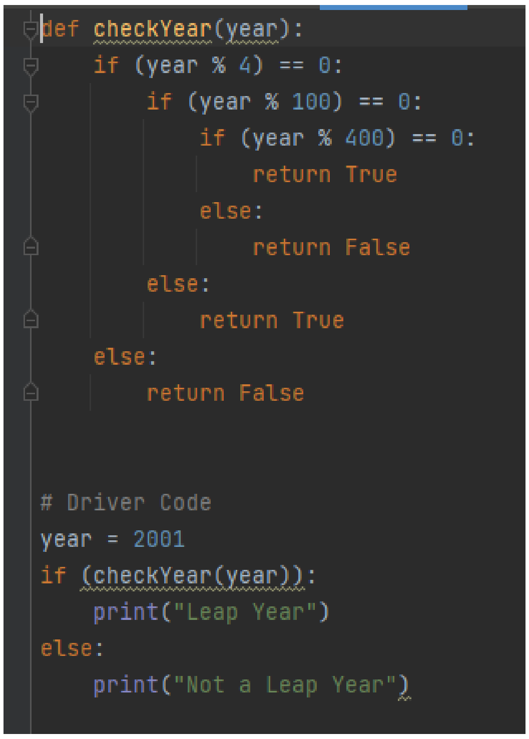
DAY 2 PROGRAMS

1.



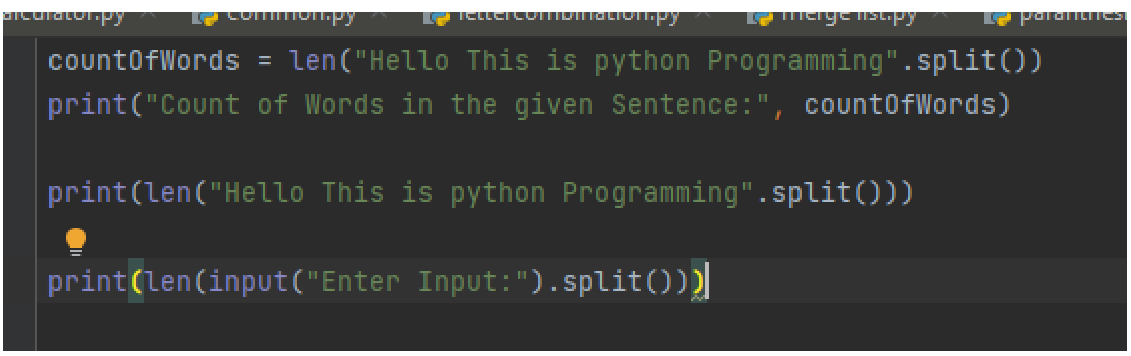
```python
def climbStairs(n):
    steps = []
    steps.append(1)
    steps.append(2)
    for i in range(2, n):
        steps.append(steps[i - 1] + steps[i - 2])
    return steps[n - 1]
n=4
print(climbStairs(n))
```

2.



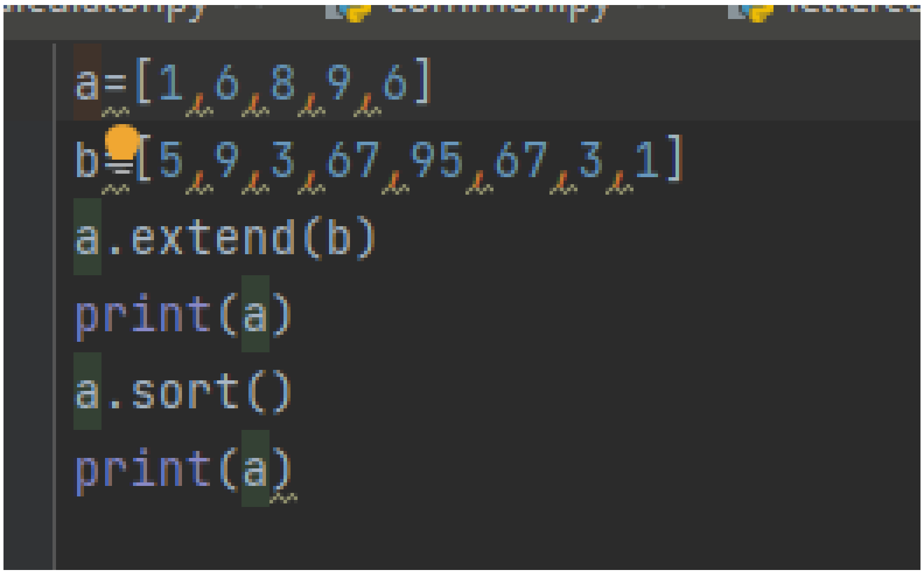
```python
def checkYear(year):
    if (year % 4) == 0:
        if (year % 100) == 0:
            if (year % 400) == 0:
                return True
            else:
                return False
        else:
            return True
    else:
        return False


# Driver Code
year = 2001
if (checkYear(year)):
    print("Leap Year")
else:
    print("Not a Leap Year")
```

3.



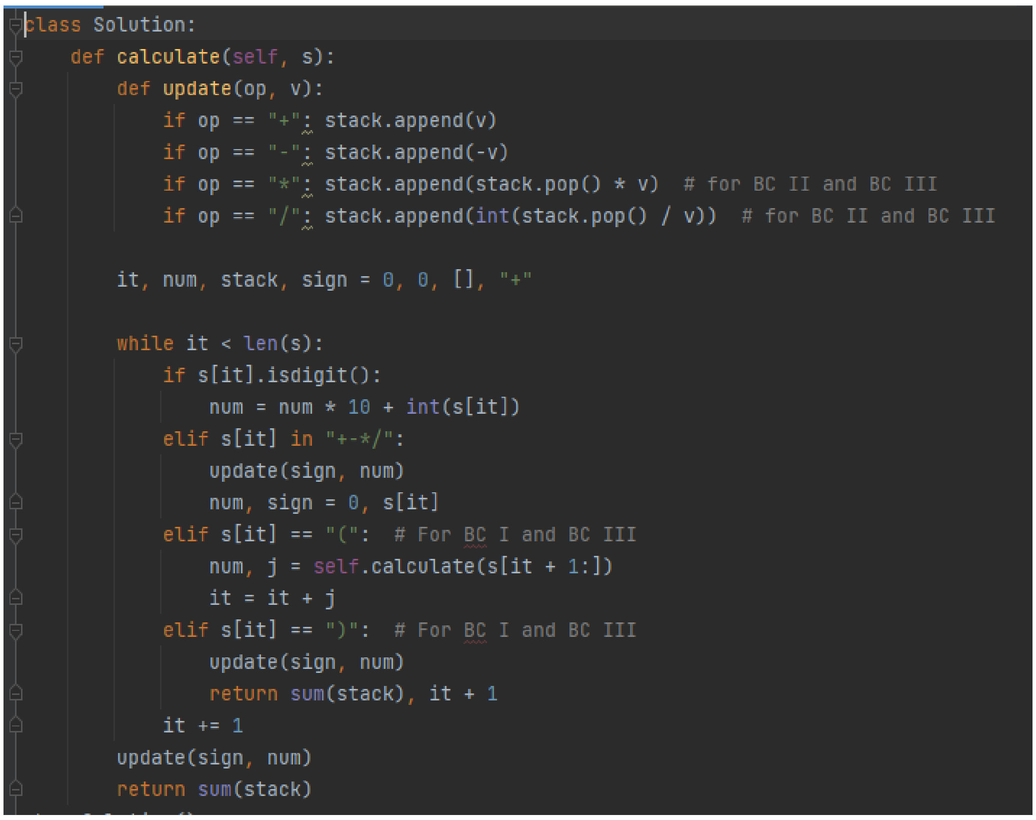
```python
countOfWords = len("Hello This is python Programming".split())
print("Count of Words in the given Sentence:", countOfWords)

print(len("Hello This is python Programming".split()))

print(len(input("Enter Input:").split()))
```

4.



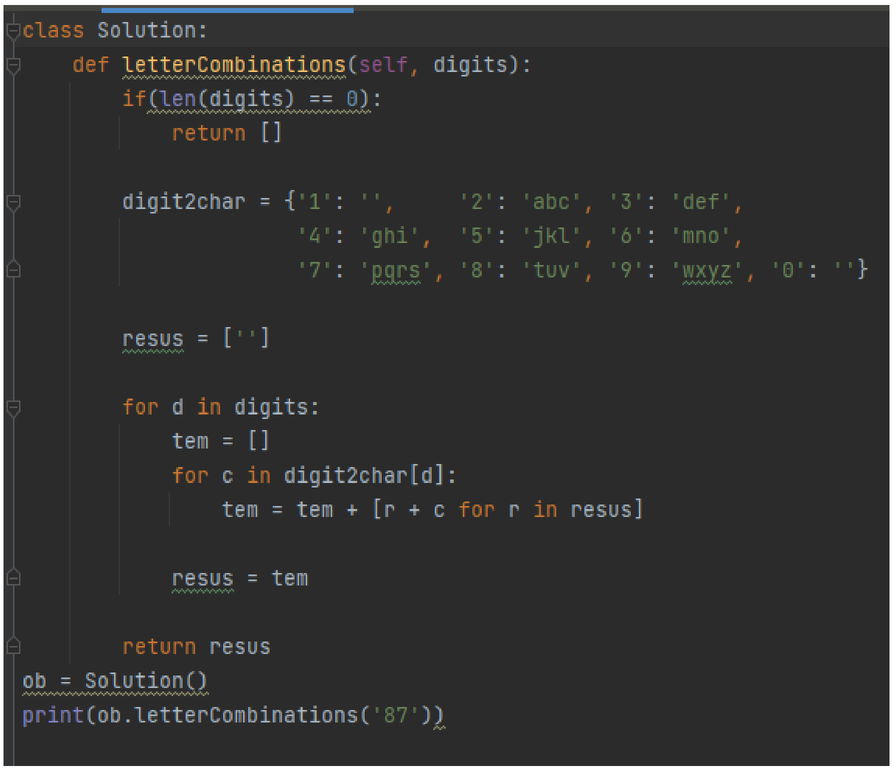
```python
a=[1,6,8,9,6]
b=[5,9,3,67,95,67,3,1]
a.extend(b)
print(a)
a.sort()
print(a)
```

5.



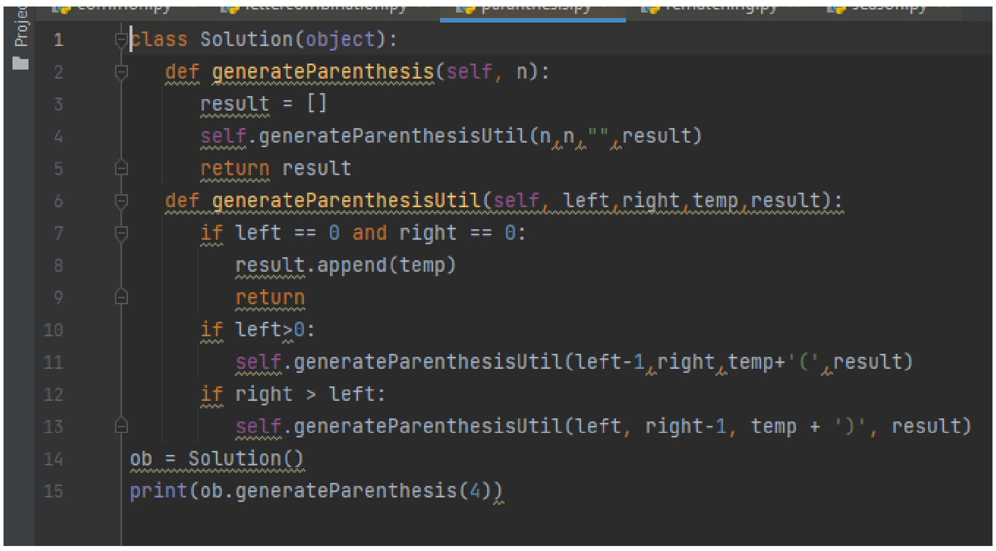
```python
class Solution:
    def calculate(self, s):
        def update(op, v):
            if op == "+": stack.append(v)
            if op == "-": stack.append(-v)
            if op == "*": stack.append(stack.pop() * v)  # for BC II and BC III
            if op == "/": stack.append(int(stack.pop() / v))  # for BC II and BC III

        it, num, stack, sign = 0, 0, [], "+"

        while it < len(s):
            if s[it].isdigit():
                num = num * 10 + int(s[it])
            elif s[it] in "+-*/":
                update(sign, num)
                num, sign = 0, s[it]
            elif s[it] == "(":  # For BC I and BC III
                num, j = self.calculate(s[it + 1:])
                it = it + j
            elif s[it] == ")":  # For BC I and BC III
                update(sign, num)
                return sum(stack), it + 1
            it += 1
        update(sign, num)
        return sum(stack)
```

6.



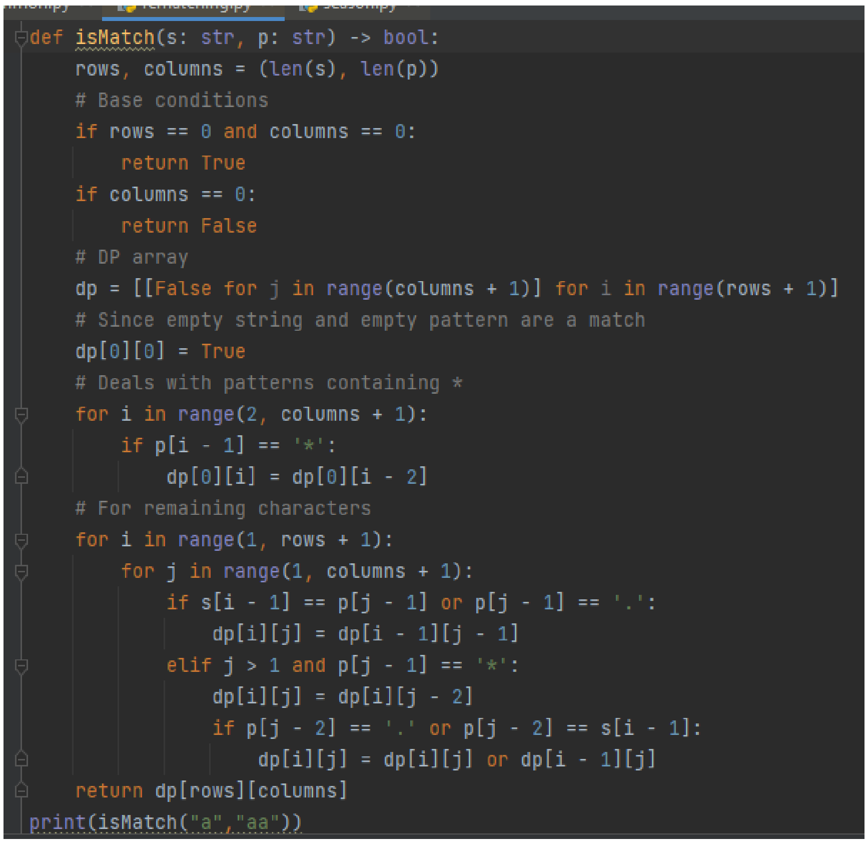
```python
class Solution:
    def letterCombinations(self, digits):
        if(len(digits) == 0):
            return []

        digit2char = {'1': '',     '2': 'abc', '3': 'def',
                      '4': 'ghi',  '5': 'jkl', '6': 'mno',
                      '7': 'pqrs', '8': 'tuv', '9': 'wxyz', '0': ''}

        resus = ['']

        for d in digits:
            tem = []
            for c in digit2char[d]:
                tem = tem + [r + c for r in resus]

            resus = tem

        return resus
ob = Solution()
print(ob.letterCombinations('87'))
```

7.



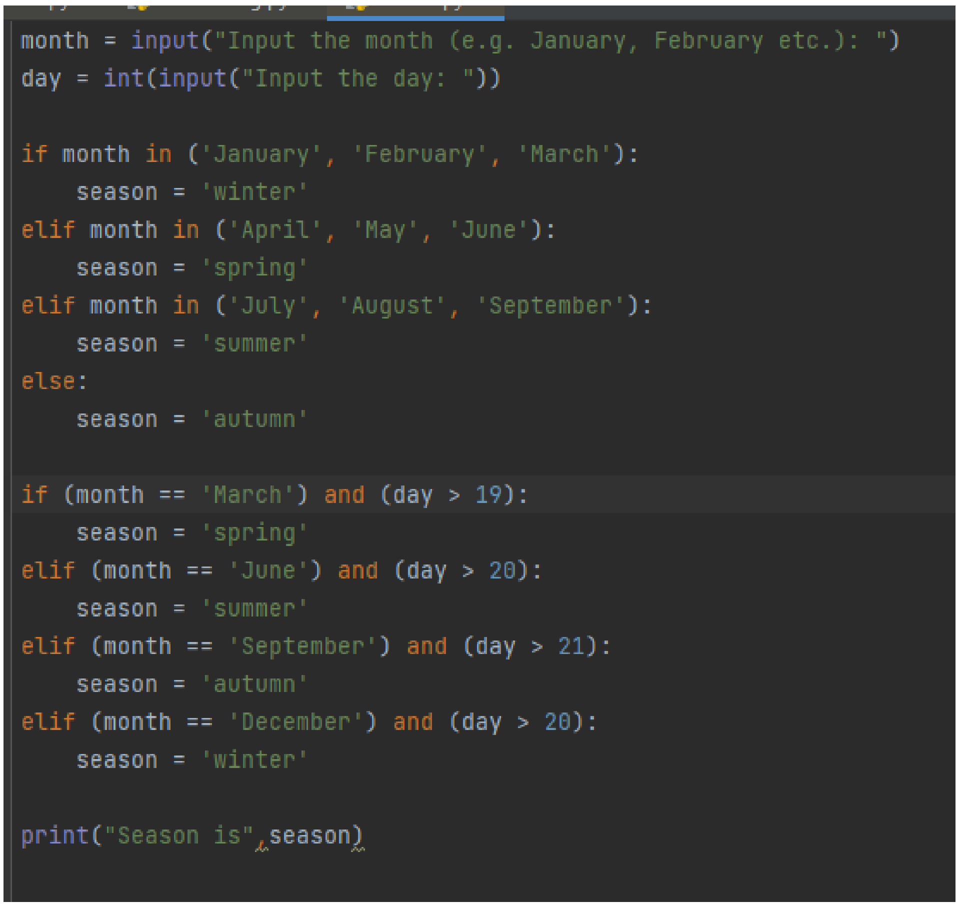
```python
class Solution(object):
   def generateParenthesis(self, n):
      result = []
      self.generateParenthesisUtil(n,n,"",result)
      return result
   def generateParenthesisUtil(self, left,right,temp,result):
      if left == 0 and right == 0:
         result.append(temp)
         return
      if left>0:
         self.generateParenthesisUtil(left-1,right,temp+'(',result)
      if right > left:
         self.generateParenthesisUtil(left, right-1, temp + ')', result)
ob = Solution()
print(ob.generateParenthesis(4))
```

8.



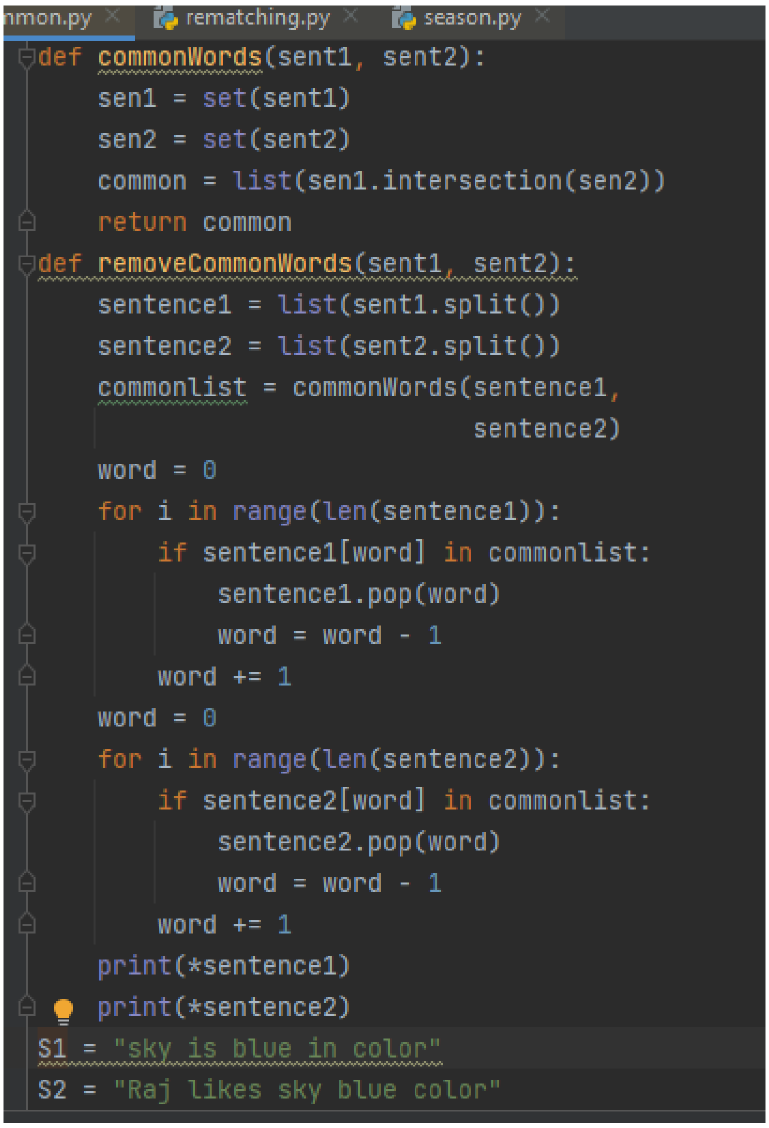
```
def isMatch(s: str, p: str) -> bool:
    rows, columns = (len(s), len(p))
    # Base conditions
    if rows == 0 and columns == 0:
        return True
    if columns == 0:
        return False
    # DP array
    dp = [[False for j in range(columns + 1)] for i in range(rows + 1)]
    # Since empty string and empty pattern are a match
    dp[0][0] = True
    # Deals with patterns containing *
    for i in range(2, columns + 1):
        if p[i - 1] == '*':
            dp[0][i] = dp[0][i - 2]
    # For remaining characters
    for i in range(1, rows + 1):
        for j in range(1, columns + 1):
            if s[i - 1] == p[j - 1] or p[j - 1] == '.':
                dp[i][j] = dp[i - 1][j - 1]
            elif j > 1 and p[j - 1] == '*':
                dp[i][j] = dp[i][j - 2]
                if p[j - 2] == '.' or p[j - 2] == s[i - 1]:
                    dp[i][j] = dp[i][j] or dp[i - 1][j]
    return dp[rows][columns]
print(isMatch("a","aa"))
```

9.



```
month = input("Input the month (e.g. January, February etc.): ")
day = int(input("Input the day: "))

if month in ('January', 'February', 'March'):
    season = 'winter'
elif month in ('April', 'May', 'June'):
    season = 'spring'
elif month in ('July', 'August', 'September'):
    season = 'summer'
else:
    season = 'autumn'

if (month == 'March') and (day > 19):
    season = 'spring'
elif (month == 'June') and (day > 20):
    season = 'summer'
elif (month == 'September') and (day > 21):
    season = 'autumn'
elif (month == 'December') and (day > 20):
    season = 'winter'

print("Season is",season)
```

10.



```
def commonWords(sent1, sent2):
    sen1 = set(sent1)
    sen2 = set(sent2)
    common = list(sen1.intersection(sen2))
    return common
def removeCommonWords(sent1, sent2):
    sentence1 = list(sent1.split())
    sentence2 = list(sent2.split())
    commonlist = commonWords(sentence1,
                             sentence2)
    word = 0
    for i in range(len(sentence1)):
        if sentence1[word] in commonlist:
            sentence1.pop(word)
            word = word - 1
        word += 1
    word = 0
    for i in range(len(sentence2)):
        if sentence2[word] in commonlist:
            sentence2.pop(word)
            word = word - 1
        word += 1
    print(*sentence1)
    print(*sentence2)
S1 = "sky is blue in color"
S2 = "Raj likes sky blue color"
```