21. Cifrado por transposición

Wednesday, October 28, 2020 8:20 PM

Cifrado

```
def main():
    myMessage = input('Ingresa el mensaje a cifrar: ')
    espacios = 1
    while espacios > 0:
        clave = input('Ingresa tu palabra clave para cifrar: ')
        espacios = clave.count(' ')
        if clave.isalpha() == False:
            espacios += 1
    myKey = len(clave)

    ciphertext = encryptMessage(myKey, myMessage)

    # Print the encrypted string in ciphertext to the screen, with
    # a | ("pipe" character) after it in case there are spaces at
    # the end of the encrypted message.
    print(ciphertext + '|')
```

```
def encryptMessage(key, message):
    # Each string in ciphertext represents a column in the grid.
    ciphertext = [''] * key
    print(ciphertext)
    input()

    # Loop through each column in ciphertext.
    for column in range(key):
        currentIndex = column

        # Keep looping until currentIndex goes past the message le
        while currentIndex < len(message):
            # Place the character at currentIndex in message at th
            # end of the current column in the ciphertext list.
            ciphertext[column] += message[currentIndex]

            # move currentIndex over
            currentIndex += key

    # Convert the ciphertext list into a single string value and r
    return ''.join(ciphertext)
```

cifradoTr...

Descifrado

```
def decryptMessage(key, message):
    # The transposition decrypt function will simulate the "column
    # "rows" of the grid that the plaintext is written on by using
```

```
    # of strings. First, we need to calculate a few values.

    # The number of "columns" in our transposition grid:
    numOfColumns = int(math.ceil(len(message) / float(key)))
    # The number of "rows" in our grid will need:
    numOfRows = key
    # The number of "shaded boxes" in the last "column" of the gri
    numOfShadedBoxes = (numOfColumns * numOfRows) - len(message)

    # Each string in plaintext represents a column in the grid.
    plaintext = [''] * numOfColumns

    # The column and row variables point to where in the grid the
    # character in the encrypted message will go.
    column = 0
    row = 0

    for symbol in message:
        plaintext[column] += symbol
        column += 1 # Point to next column.

        # If there are no more columns OR we're at a shaded box, g
        # the first column and the next row:
        if (column == numOfColumns) or (column == numOfColumns - 1
            column = 0
            row += 1

    return ''.join(plaintext)
```

descifrad...