Российский университет дружбы народов Научный факультет

Математические основы защиты информации и

информационной безопасности

шифрование маршрута

шифр Виженера

сетевое шифрование

Подготовлено студентом:

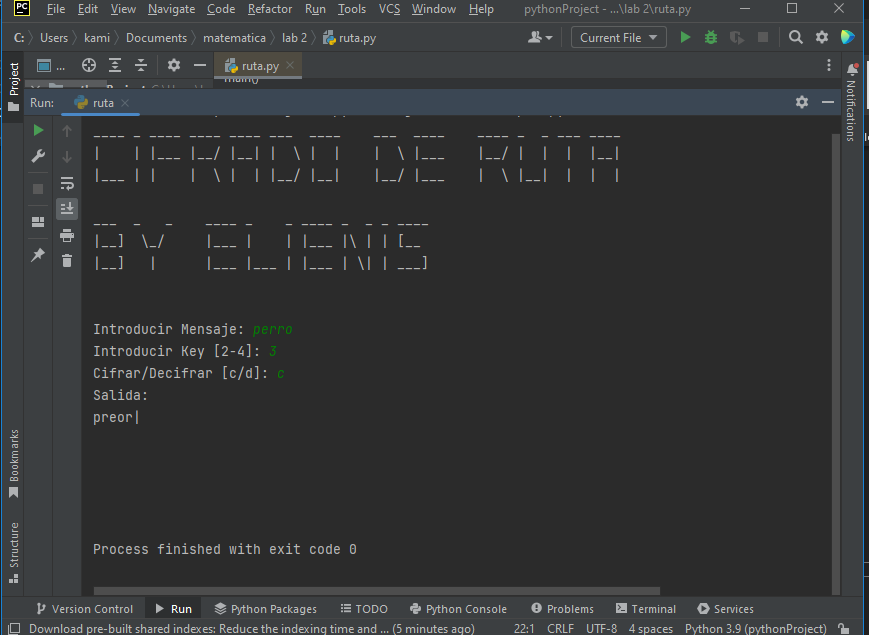
Елиенис Санчес Родригес.

Преподаватель: Дмитрий Сергеевич

Cifrado de Ruta o transposición

Codigo

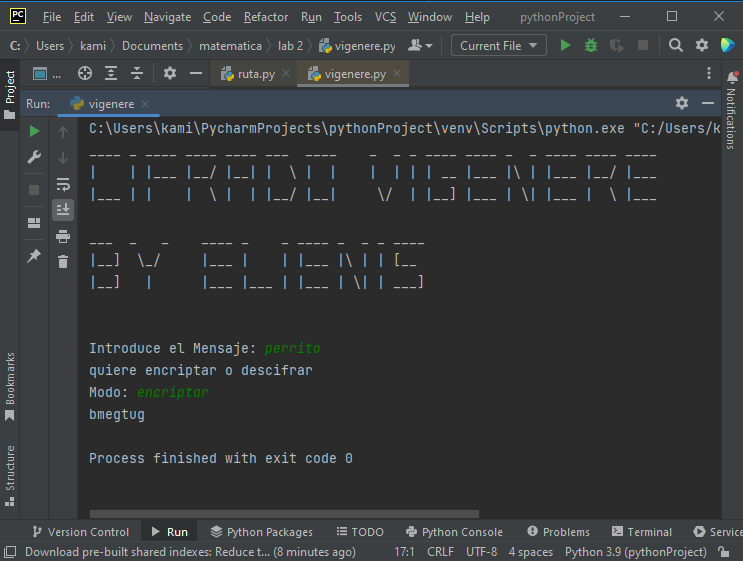
from pyfiglet import figlet\_format  
print(figlet\_format( "Cifrado de Ruta by Elienis", font = "cybermedium"))  
import math  
# Función principal  
def main():  
 message = input('Introducir Mensaje: ')  
 key = int(input('Introducir Key [2-%s]: ' % (len(message) - 1)))  
 mode = input('Cifrar/Decifrar [c/d]: ')  
  
 if mode.lower().startswith('c'): # Si mode es igual a "c" se llamara a "encryptMessage"  
 text = cifrarMensaje(key, message)  
 elif mode.lower().startswith('d'): # De lo contrario se llamara a "decryptMessage"  
 text = descifrarMensaje(key, message)  
  
 # Imprime la cedena cifrada y usando '|' para indicar el fin  
 #del mensaje cifrado  
 print('Salida:\n%s' %(text + '|'))  
 print("\n")  
 input("")  
  
def cifrarMensaje(key, message):  
#Cada cadena de texto en el mensaje cifrado representa una columna en la matriz  
 cipherText = [''] \* key  
 # recorremos las colimnas  
 for col in range(key):  
 pointer = col  
  
 #Lee cada elemento de la columna  
 while pointer < len(message):  
 cipherText[col] += message[pointer]  
  
 #mueve el puntero al siguiente elemento de la columna  
 pointer += key  
 return ''.join(cipherText)  
  
def descifrarMensaje(key, message):  
  
 numCols = math.ceil(len(message) / key)  
 numRows = key  
 numShadedBoxes = (numCols \* numRows) - len(message)  
 plainText = [""] \* numCols  
 col = 0; row = 0;  
  
 for symbol in message:  
 plainText[col] += symbol  
 col += 1  
  
 if (col == numCols) or (col == numCols - 1) and (row >= numRows - numShadedBoxes):  
 col = 0  
 row += 1  
 # Convertimos la lista en una cadena de texto y lo retornamos  
 return "".join(plainText)  
# si el nombre de la funcion es main la manda a ejecutar  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

****

**El cifrado de Vigenère**

Codigo

from colorama import init, Fore  
from pyfiglet import figlet\_format  
print(figlet\_format( "Cifrado Vigenere by Elienis", font = "cybermedium"))  
  
#!/usr/bin/env python  
# -\*- coding: utf-8 -\*-  
LETRAS = ("ABCDEFGHIJKLMNÑOPQRSTUVWXYZ")  
  
def main():  
 mensaje=input("Introduce el Mensaje: ")  
 myKey="MINOMBREESANTONIOALFONSO"  
 print ("quiere encriptar o descifrar")  
 accion=input("Modo: ")  
  
 if accion=='encriptar':  
 traducido=cifrar\_mensaje(myKey,mensaje)  
 elif accion=='descifrar':  
 traducido=descifrar\_mensaje(myKey,mensaje)  
 print(traducido)  
  
def cifrar\_mensaje(clave,mensa):  
 return traductor\_mensaje(clave,mensa,'encriptar')  
  
def descifrar\_mensaje(clave,mensa):  
 return traductor\_mensaje(clave,mensa,'descifrar')  
  
def traductor\_mensaje(clave,mensa,accion):  
 traducido=[]  
 indice\_clave=0  
 clave=clave.upper()  
  
 for symbol in mensa:  
 num=LETRAS.find(symbol.upper())  
 if num!=-1:  
 if accion=='encriptar':  
 num+=LETRAS.find(clave[indice\_clave])  
 elif accion=='descifrar':  
 num-=LETRAS.find(clave[indice\_clave])  
 num%=len(LETRAS)  
 if symbol.isupper():  
 traducido.append(LETRAS[num])  
 elif symbol.islower():  
 traducido.append(LETRAS[num].lower())  
 indice\_clave+=1  
 if indice\_clave==len(clave):  
 indice\_clave=0  
  
 else:  
 traducido.append(symbol)  
 return ('').join(traducido)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()



**Rejillas de Fleissner**

Codigo

print("\033[9;35m"+"Codificacion Rejilla by Elienis")  
  
def grille\_encrypt(plaintext: str, grille: List[str]) -> Optional[str]:  
 # Hay un trozo de papel de cuadrícula cuadrada con orificios (4\*4) en el papel blanco, rejilla: posición del orificio  
 # Después de girar la posición del orificio durante una semana, se puede llenar el cuadrado de 4\*4. Si no puede, no será válido y volverá directamente a None.  
 # Escriba las letras de texto sin formato en los agujeros de izquierda a derecha y de arriba a abajo. Después de girar el papel de orificio 90 grados en el sentido de las agujas del reloj, el orificio se mueve a una posición en blanco y las letras de texto sin formato continúan escribiéndose.  
 # Si el mensaje no termina después de girar 3 veces, continúe en la siguiente hoja de papel  
 # La letra final se completará, lea el texto cifrado de salida por línea  
  
 # Regrese a la posición cubierta por el orificio después de que la rejilla gire 3 veces  
 def check\_grille(grille):  
 # Ubicación actual del hoyo  
 K = [(i, j) for i, l in enumerate(grille) for j, v in enumerate(l) if v == 'X']  
 # Convertir lista de rejillas a matriz  
 # Rotación de matriz, obtenga la posición del orificio después de cada rotación, y después de 3 rotaciones, obtenga la posición del orificio.  
 for i in range(3):  
 grille = [[j for j in g ] for g in grille ]  
 A = np.mat(grille) # Lista matriz de par  
 B = np.rot90(A,k=-1) # Rotación de la matriz, k es un número positivo en sentido antihorario, k es un número negativo en sentido horario.  
 grille = B.tolist() # Matriz para listar  
 k = [(i, j) for i, l in enumerate(grille) for j, v in enumerate(l) if v == 'X']  
 K.extend(k)  
 return K  
 K = check\_grille(grille) # La posición en la que los orificios se cubren a su vez es también la posición en la que las letras de texto sin formato se rellenan a su vez  
 # imprimir('=============================',conjunto (K), len(conjunto (K)))  
  
 if len(set(K)) != 16 or len(K) > 16: # Después de girar 3 veces, el orificio no cubre 4\*4=16 cuadrados, o los cuadrados se cubren repetidamente, la rejilla no es válida  
 result = None  
 else:  
 ciphertext = []  
 for i in range(0,len(plaintext),16):  
 sub\_t = plaintext[i:i+16]  
 # Narr = np.array([['.', '.', '.', '.'], ['.', '.', '.', '.'], ['.', '.', '.', '.'], ['.', '.', '.', '.']]) #创建一个4\*4矩阵  
 Narr = np.ones((4, 4)).astype(np.str\_) # # Crear una matriz de caracteres de 4\*4  
 # Complete las letras de texto sin formato en la posición de K a su vez  
 for j in range(16):  
 x = K[j][0]  
 y = K[j][1]  
 Narr[x][y] = sub\_t[j] if j < len(sub\_t) else '' # Evita que la longitud del texto sin formato sea inferior a 16  
 # Después de completar, convierta la matriz en una cadena por fila  
 sub\_c = Narr.tolist() # Añadir a la lista  
 sub\_c = ''.join([''.join(i) for i in sub\_c])  
 ciphertext.append(sub\_c)  
 result = ''.join(ciphertext)  
 # print(result)  
 return result  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print("\n")  
 numero1 = input("\033[0;51m"+"Por favor ingrese la palabra a encriptar:")  
 print("\n")  
 print(grille\_encrypt(numero1, [".X..", ".X..", "...X", "X..."]))  
 print("\n")  
 # Estas "afirmaciones" se utilizan para la autocomprobación y no para una prueba automática  
 assert (  
 grille\_encrypt("cardangrilletest", [".X..", ".X..", "...X", "X..."])  
 == "actilangeslrdret"  
 )  
 assert (  
 grille\_encrypt(  
 "quickbrownfoxjumpsoverthelazydog", ["X...", "...X", "..X.", ".X.."]  
 )  
 == "qxwkbnjufriumcoopyeerldsatoogvhz"  
 )  
 assert (  
 grille\_encrypt(  
 "quickbrownfoxjumpsoverthelazydog", [".XX.", ".XX.", "..X.", "X..."]  
 )  
 == None  
 )  
 assert grille\_encrypt("cardangrilletest", ["...X", "....", "....", "...."]) == None  
  
  
 print("\033[1;33m" + "Codigo encriptado satisfactoriamente \n" + '\033[0;m')  
 print("\033[2;35m" + "Ahora largate hijo , que me estorbas")

