

```

#!/usr/bin/env python3
from simple_term_menu import TerminalMenu
import string, math, os

mainMenuOptions = ["a) Cifrado de César", "b) RSA", "c) Información del Programa", "d) Imprimir código fuente", "e) Salir"]
mainMenu = TerminalMenu(mainMenuOptions)

characters = list(string.printable)
characters.remove('\t')
characters.remove('\n')
characters.remove('\r')
characters.remove('\x0b')
characters.remove('\x0c')

def isPrime(number):
    for i in range(2, number):
        if (number % i) == 0:
            return False
    return True

class CesarCipher(object):
    characters = characters

    def __init__(self, k):
        self.k = k
        pass

    def encrypt(self, text):
        new_characters = []
        for x in text:
            new_character_index = (self.characters.index(x) + self.k) % len(self.characters)
            new_character = self.characters[new_character_index]
            new_characters.append(new_character)
        return "".join(new_characters)

    def decrypt(self, text):
        new_characters = []
        for x in text:
            new_character_index = (self.characters.index(x) - self.k) % len(self.characters)
            new_character = self.characters[new_character_index]
            new_characters.append(new_character)
        return "".join(new_characters)

class RSACipher(object):
    characters = characters
    characters.insert(0, "®")

    def __init__(self, p, q):
        self.p, self.q = p, q #number.getPrime(b), number.getPrime(b)
        self.n = self.p * self.q
        self.phi = (self.p - 1) * (self.q - 1)

        self.e = 2
        while(self.e < self.phi):

```

```

        if (math.gcd(self.e, self.phi) == 1):
            break
        else:
            self.e += 1

x = self.xgcd(self.e, self.phi)

if (x < 0):
    self.d = x + self.phi
else:
    self.d = x

self.pad_length = len(str(self.n))

print(f'==> Llave pública: {self.e, self.n}')
print(f'==> Llave privada: {self.d, self.n}')

def xgcd(self, a, b):
    x, old_x = 0, 1
    y, old_y = 1, 0

    while (b != 0):
        quot = a // b
        a, b = b, a - quot * b
        old_x, x = x, old_x - quot * x
        old_y, y = y, old_y - quot * y

    return old_x

def encrypt(self, text):
    return "".join([str(pow(self.characters.index(x), self.e, mod=self.n)).zfill(self.pad_length) for x in text])

def decrypt(self, text):
    encrypted_numbers = [int(x) for x in map("".join, zip(*[iter(text)]*self.pad_length))]
    return "".join([self.characters[pow(number, self.d, mod=self.n)] for number in encrypted_numbers])

class MenuOptions(object):
    cipherMenuOptions = ["a) Encriptar", "b) Desencriptar", "d) Regresar"]
    cipherMenu = TerminalMenu(cipherMenuOptions)

    def __init__(self):
        pass

    def Cesar(self):
        print("+++ Menu Cesar +++")
        condition = True
        while condition:
            try:
                k = int(input("Introduce un valor de desplazamiento, k = "))
                self.cesarCipher = CesarCipher(k)
                condition = False
            except ValueError:
                print("!!! Error: Introduzca un número.\n")
                condition = True
            except TypeError:

```

```

        print("!!! Error: Introduzca un número.\n")
        condition = True
    except EOFError:
        print("!!! Error: Introduzca un valor.\n")
        condition = True
while True:
    selectedOption = self.cipherMenu.show()
    match selectedOption:
        case 0:
            text = input("Introduce el texto a encriptar: ")
            print(f'#=> El texto cifrado es: <<{self.cesarCipher.encrypt(text)}>>\n')
        case 1:
            text = input("Introduce el texto a desencriptar: ")
            print(f'#=> El texto plano es: <<{self.cesarCipher.decrypt(text)}>>\n')
        case 2:
            break

def RSA(self):
    print("+++ Menu RSA +++")
    condition = True
    while condition:
        try:
            p = int(input("Introduce un número primo, p = "))

            if not isPrime(p):
                print("!!! CUIDADO: Introduce un número primo.")
                continue

            q = int(input("Introduce otro número primo, q = "))

            while (not isPrime(q)):
                print("!!! CUIDADO: Introduce un número primo.")
                q = int(input("Introduce otro número primo, q = "))

            self.rsaCipher = RSACipher(p, q)
            condition = False
        except ValueError:
            print("!!! Error: Introduzca un número.\n")
            condition = True
        except TypeError:
            print("!!! Error: Introduzca un número.\n")
            condition = True
        except EOFError:
            print("!!! Error: Introduzca un valor.\n")
            condition = True
    while True:
        selectedOption = self.cipherMenu.show()
        match selectedOption:
            case 0:
                text = input("Introduce el texto a encriptar: ")
                try:
                    print(f'#=> El texto cifrado es: <<{self.rsaCipher.encrypt(text)}>>\n')
                except ValueError as error:
                    print(error, "\n")
            case 1:

```

```
text = input("Introduce el texto a desencriptar: ")
print(f'#=> El texto plano es: <<{self.rsaCipher.decrypt(text)}>>\n')
```

```
case 2:
    break
```

```
def Info(self):
    print("+++ Información del Programa +++\n")
    print("### AUTOR ###")
    print("## Nombre: Roberto Treviño Cervantes")
    print("## Matricula: 1915003")
    print("## Grupo: 32")
    print("## Carrera: Ciencias Computacionales")
    print("## Semestre: Séptimo\n")
    print("### CIFRADO DE CÉSAR ###")
    print("Es uno de los metodos más simples y antiguos, fue empleado por Julio César para comunica
ciones privadas. Funciona mediante la sustitución de los caracteres en el texto por el caracter correspon
diente recorrido un cierto numero de lugares en un alfabeto preestablecido.\n")
    print("### RSA ###")
    print("Es un algoritmo de encriptación llamado con las iniciales de sus creadores muy ampliamente
utilizado hoy en dia. Es un algoritmo de encriptación asimetria, es decir, que genera una llave privada, u
sada para encriptar, y una publica, para desencriptar. Funciona bajo la premisa de que es facil generar u
n numero semiprimo al multiplicar dos numeros primos, pero factorizar ese numero de vuelta a los primos
que lo originaron es bastante más complejo computacionalmente.\n")
```

```
def Print(self):
    print("+++ Imprimir código fuente +++\n")
    os.system("./gtk-print main.pdf")
```

```
def main():
    print("*** PIA Criptografia ***\n")
    print(f'#=> El conjunto de caracteres empleado es el conjunto ASCII con {len(characters)} elementos.')
    print(characters, "\n")
    options = MenuOptions()
    while True:
        selectedOption = mainMenu.show()
        match selectedOption:
            case 0:
                options.Cesar()
            case 1:
                options.RSA()
            case 2:
                options.Info()
            case 3:
                options.Print()
            case 4:
                exit()
```

```
if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        exit()
```