🔑

Lab 124: Cifrado Cesar

🔗Link: https://awsrestart.instructure.com/courses/1632/modules/items/886839

☁️ Repositorio en la nube: https://github.com/francopig/aws-python/tree/main/13. Cifrado Cesar

Uso de funciones para implementar un cifrado César

Información general sobre el laboratorio

En programación, una función es una sección con nombre dentro de un programa que realiza una tarea específica. Python tiene funciones integradas como `print()` proporcionadas por el mismo lenguaje. Además, puede utilizar funciones proporcionadas por otros desarrolladores a través de la instrucción `import`. Por ejemplo, puede utilizar `import math` si desea utilizar la función `math.floor()`. En Python, puede crear sus propias funciones, denominadas *funciones definidas por el usuario*.

Para continuar el debate sobre las funciones definidas por el usuario, escribirá un programa para implementar un cifrado César, que es un método sencillo de cifrado. El cifrado César toma las letras de un mensaje y desplaza cada letra a lo largo del alfabeto un número determinado de posiciones.

En este laboratorio, deberá realizar lo siguiente:

- crear funciones definidas por el usuario
- utilizar varias funciones para implementar un programa de cifrado César

---

Ejercicio 1: Creación de una función definida por el usuario

Para comenzar el proceso de implementación de un cifrado César en Python, creará una función simple definida por el usuario.

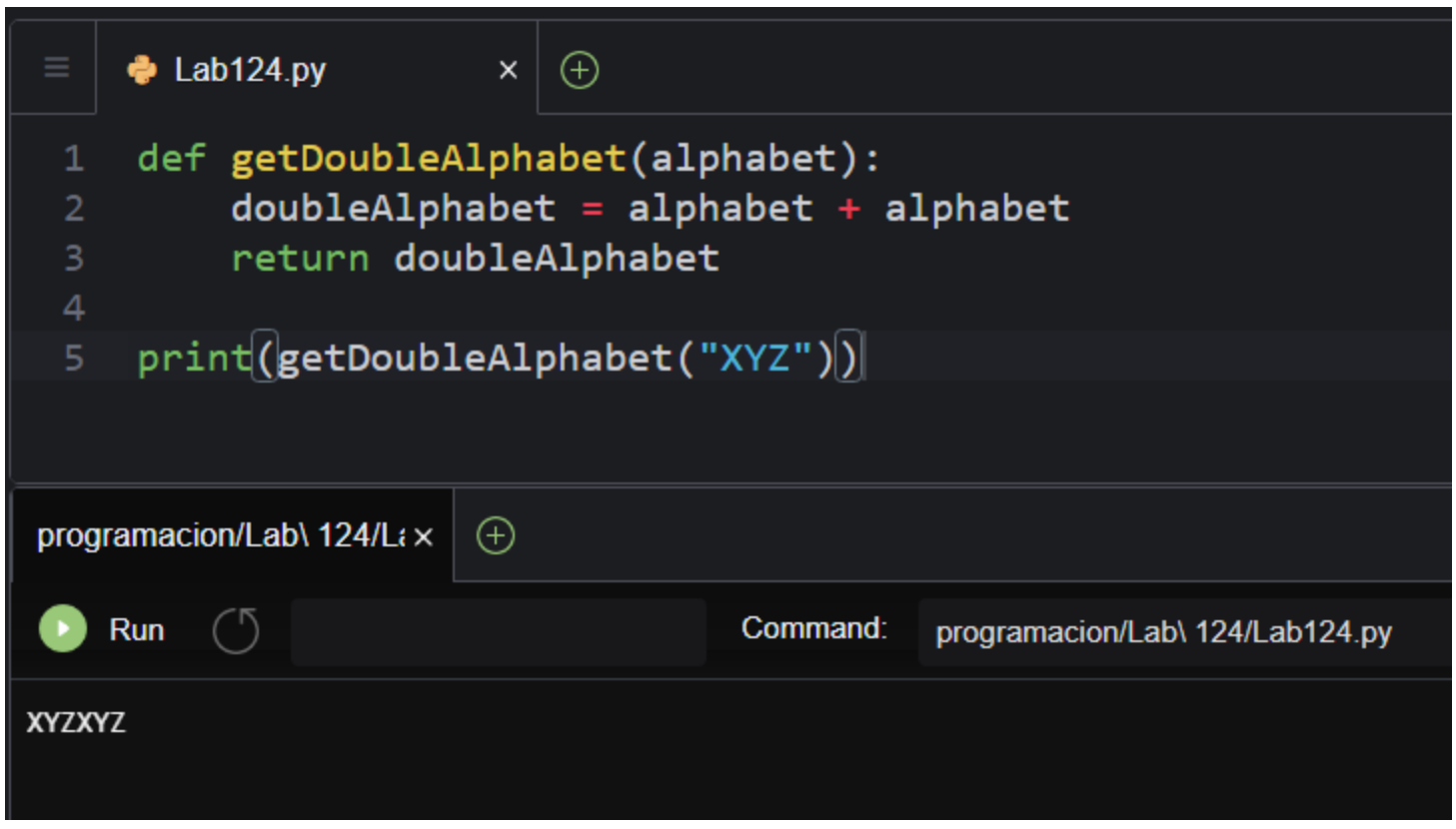
1. En el panel de navegación del IDE, elija el archivo que creó en la sección *Creación del archivo de ejercicios de Python* anterior.
2. Defina una función denominada `getDoubleAlphabet` que tome un argumento de tipo cadena y concatene o combine dicha cadena consigo misma de la siguiente manera:

```
def getDoubleAlphabet(alphabet):
    doubleAlphabet = alphabet + alphabet
    return doubleAlphabet
```

> Nota: Las partes obligatorias de la instrucción de una función son la palabra clave def, un nombre y los dos puntos (:). Además, en Python, no es necesario declarar las variables, y sus tipos de datos se deducen a partir de la instrucción de asignación.

3. Guarde el archivo.
4. Para comprender lo que hace la función, tome una entrada de muestra, como `alphabet="ABC"`. La cadena devuelta para esta entrada sería `"ABC" + "ABC" = "ABCABC"`. El signo más (+) concatena los textos en uno solo.

En los siguientes ejercicios, definirá más funciones que realizan una tarea simple. Luego, combinará estas funciones para crear un programa de cifrado César.



Ejercicio 2: Cifrado de un mensaje

La siguiente función que definirá solicitará al usuario un mensaje para cifrar. Utilizará la función integrada denominada `input()`.

1. En el editor de texto, escriba el siguiente código y guarde el archivo:

```
def getMessage():
    stringToEncrypt = input("Please enter a message to encrypt: ")
    return stringToEncrypt
```

> Nota: Las funciones deben realizar tareas específicas. Por lo general, debido a que las funciones realizan una tarea específica, es probable que sus funciones también sean cortas. Aunque esta función devuelve una cadena, no necesita un argumento como la función getDoubleAlphabet().

Ejercicio 3: Obtención de una clave de cifrado

La *clave de cifrado* es la distancia con la que desplazará las letras. Mediante el uso de dos alfabetos, puede tener una clave de cifrado que sea cualquier número entero entre 1 y 25. No considere la clave en el índice 26 porque esa clave lo dirigirá de nuevo al mensaje original.

1. Defina una función para solicitar una clave de cifrado al usuario y escriba el siguiente código:

```
def getCipherKey():
    shiftAmount = input( "Please enter a key (whole number from 1-25): ")
```

```
    return shiftAmount
```

2. Guarde el archivo.

```
 9  def getCypherKey():
10      shiftAmount = input("Ingrese la clave de cifrado (1-25):")
11      return shiftAmount
```

Ejercicio 4: Cifrado de un mensaje

Hasta ahora, las funciones han sido cortas y sencillas. Este suele ser el caso cuando mantiene una tarea específica dentro de una función. La función `encryptMessage` será un poco más larga.

1. Antes de escribir el código, debe diseñar el algoritmo de cifrado de la siguiente manera:

> Tome tres argumentos: el mensaje, la clave de cifrado y el alfabeto. Inicie las variables. Utilice un bucle for para recorrer cada letra del mensaje. Para una letra específica, busque su posición. Para una letra específica, determine la nueva posición de acuerdo con la clave de cifrado.Si la letra actual está en el alfabeto, agregue la nueva letra al mensaje cifrado.Si la letra actual no está en el alfabeto, añada la letra actual. Devuelva el mensaje cifrado tras agotar todas las letras del mensaje.

2. En el archivo de ejercicios, escriba el siguiente código y siga la lógica a partir de la revisión de los pasos del algoritmo anterior:

```
def encryptMessage(message, cipherKey, alphabet):
    encryptedMessage = ""
    uppercaseMessage = ""
    uppercaseMessage = message.upper()
    for currentCharacter in uppercaseMessage:
        position = alphabet.find(currentCharacter)
        newPosition = position + int(cipherKey)
        if currentCharacter in alphabet:
            encryptedMessage = encryptedMessage + alphabet[newPosition]
        else:
            encryptedMessage = encryptedMessage + currentCharacter
    return encryptedMessage
```

☁️ La función `find()` se utiliza para buscar la posición de un carácter dentro de una cadena. A continuación, te explicaré cómo funciona y cómo se aplica en este contexto.

La función `find()` se utiliza en Python para encontrar la primera aparición de un subtexto (carácter o cadena) dentro de una cadena más grande. Toma el subtexto como argumento y devuelve el índice de la primera ocurrencia de ese subtexto dentro de la cadena. Si el subtexto no se encuentra en la cadena, se devuelve -1.

En el código que has proporcionado, `find()` se utiliza para o**btener la posición de cada carácter** en `uppercaseMessage` dentro del `alphabet`

☁️ La parte del if: Si el `currentCharacter` está presente en el `alphabet`, significa que es un carácter válido para el cifrado y se debe reemplazar por el carácter correspondiente en la nueva posición desplazada. En ese caso, se agrega el carácter en la posición `newPosition` del `alphabet` a la `encryptedMessage` utilizando la concatenación `encryptedMessage + alphabet[newPosition]`.

Si el `currentCharacter` no está presente en el `alphabet`, esto indica que es un carácter que no debe modificarse durante el cifrado (por ejemplo, un espacio, un símbolo o un dígito). En este caso, se agrega el `currentCharacter` original sin modificar a la `encryptedMessage` utilizando la concatenación `encryptedMessage + currentCharacter`.

3. Guarde el archivo.

```python
def encryptMessage(message, cipherKey, alphabet):

    encryptedMessage = ""
    uppercaseMessage = ""
    uppercaseMessage = message.upper()

    for currentCharacter in uppercaseMessage:
        position = alphabet.find(currentCharacter)
        newPosition = position + int(cipherKey)
        if currentCharacter in alphabet:
            encryptedMessage = encryptedMessage + alphabet[newPosition]
        else:
            encryptedMessage = encryptedMessage + currentCharacter

    return encryptedMessage
```

Ejercicio 5: Descifrado de un mensaje

Las funciones son útiles, ya que pueden reutilizarse. Escribirá una función `decryptMessage()` al reutilizar la función `encryptMessage()`. Para este cifrado sencillo, puede deshacer el cifrado si mueve cada letra hacia atrás. De este modo, en lugar de agregar la clave de cifrado, la eliminará. Para evitar reescribir la mayor parte de la lógica, pasará una clave de cifrado negativa.

1. Luego, escriba el siguiente código y guarde el archivo:

```
def decryptMessage(message, cipherKey, alphabet):
    decryptKey = -1 * int(cipherKey)
    return encryptMessage(message, decryptKey, alphabet)
```

```python
def decryptMessage(message, cipherKey, alphabet):
    decryptKey = -1 * int(cipherKey)
    return encryptMessage(message, decryptKey, alphabet)
```

Realiza la operación inversa del cifrado para descifrar un mensaje encriptado. Toma tres parámetros: `message` (mensaje a descifrar), `cipherKey` (clave de cifrado) y `alphabet` (alfabeto).

La función realiza los siguientes pasos:

1. Se calcula `decryptKey` multiplicando la `cipherKey` por -1 y convirtiéndolo a entero. Esto es necesario para obtener el valor negativo de la clave de cifrado, que se utilizará para descifrar el mensaje.
2. Se llama a la función `encryptMessage` pasando el `message` a descifrar, el `decryptKey` negativo y el `alphabet` original. La función `encryptMessage` es la misma función que se usó para cifrar el mensaje en el código previo. Aquí se utiliza para realizar el descifrado, ya que el descifrado es el proceso inverso del cifrado.
3. Se devuelve el resultado de la llamada a `encryptMessage`, que corresponde al mensaje descifrado.

Ejercicio 6: Creación de una función principal

Ha creado una colección de funciones definidas por el usuario que lo ayudarán a escribir un programa de cifrado César. Desde luego, la lógica principal del programa también estará incluida en una función.

1. Antes de examinar el código, diseñe la lógica:

> Defina una variable de cadena para que contenga el alfabeto inglés. Para poder desplazar las letras, duplique la cadena del alfabeto. Obtenga del usuario un mensaje para cifrar. Obtenga del usuario una clave de cifrado. Cifre el mensaje. Descifre el mensaje.

2. En el archivo de ejercicios, escriba el siguiente código y siga la lógica a partir de la revisión de los pasos del algoritmo anterior:

```
def runCaesarCipherProgram():
    myAlphabet="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    print(f'Alphabet: {myAlphabet}')
    myAlphabet2 = getDoubleAlphabet(myAlphabet)
    print(f'Alphabet2: {myAlphabet2}')
    myMessage = getMessage()
    print(myMessage)
    myCipherKey = getCipherKey()
    print(myCipherKey)
    myEncryptedMessage = encryptMessage(myMessage, myCipherKey, myAlphabet2)
    print(f'Encrypted Message: {myEncryptedMessage}')
    myDecryptedMessage = decryptMessage(myEncryptedMessage, myCipherKey, myAlphabet2)
    print(f'Decypted Message: {myDecryptedMessage}')
```

Para ayudarlo con la depuración y la comprensión del programa, se agregaron las instrucciones `print()`, pero no son estrictamente necesarias para que el programa opere de forma correcta.

3. Guarde y ejecute el archivo y luego, vea los resultados.

No ocurre nada. ¿Por qué? Recuerde que una función es una sección de un programa que realiza una tarea específica. Todavía no llamado la función.

4. Para darle un nombre a la función, agregue la siguiente línea al archivo **.py** y guarde el archivo:

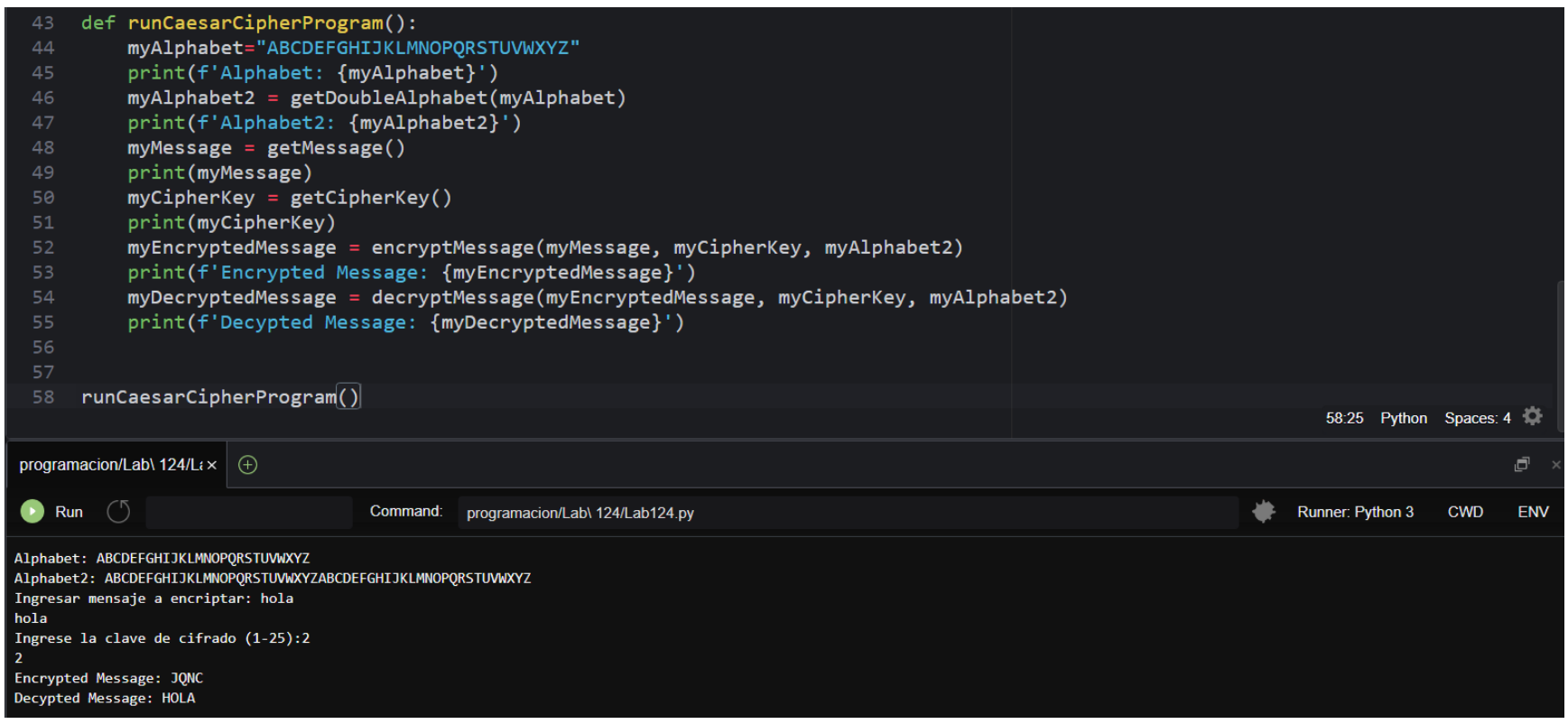
```
runCaesarCipherProgram()
```

5. Ejecute el programa de nuevo. El resultado debería ser similar al siguiente:

```
Alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Alphabet2: ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ
Please enter a message to encrypt: new message
new message
Please enter a key (whole number from 1-25): 4
4
Encrypted Message: RIA QIWWEKI
Decypted Message: NEW MESSAGE
```

6. Vuelva a ejecutar el programa con diferentes entradas.

¡Felicitaciones! ¡Ha trabajado con funciones definidas por el usuario e implementado un programa de cifrado!



Sobre Alphabet2:

n la función `runCaesarCipherProgram`, se llama a `getDoubleAlphabet` para obtener `alphabet2`. Esto se hace antes de encriptar o descifrar el mensaje. El propósito de `alphabet2` es permitir que el cifrado de César realice desplazamientos más allá del final del `alphabet` original.

Al tener dos copias consecutivas del `alphabet`, cuando se busca la posición de un carácter en `alphabet2`, el desplazamiento se realiza correctamente incluso si el desplazamiento lleva el índice más allá del límite del `alphabet` original. Por ejemplo, si se tiene un desplazamiento de 3 y el carácter actual es 'X' (posición 23 en el `alphabet` original), se buscará la posición de 'X' en `alphabet2` y se obtendrá 23 + 3 = 26. Esto permitirá realizar el desplazamiento circular y obtener la letra correspondiente ('A' en este caso).