124- [PF] - Lab - Uso de funciones para implementar un cifrado César

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

Tiempo de finalización estimado

60 minutos

Acceso al IDE de AWS Cloud9

1. Para activar el entorno de su laboratorio, desplácese hasta la parte superior de estas instrucciones y seleccione **Start Lab** (Iniciar laboratorio).

   Se abrirá el panel **Start Lab** (Iniciar laboratorio), donde se muestra el estado del laboratorio.

2. Espere hasta que aparezca el mensaje *Lab status: ready* (Estado del laboratorio: listo) y, a continuación, para cerrar el panel **Start Lab** (Iniciar laboratorio), haga clic en la **X**.

3. En la parte superior de estas instrucciones, elija **AWS**.

   AWS Management Console se abrirá en una pestaña nueva del navegador. El sistema iniciará la sesión de forma automática.

   > Nota: Si no se abre una pestaña nueva del navegador, generalmente aparece un anuncio o un icono en la parte superior de este, el cual indica que el navegador no permite que se abran ventanas emergentes en el sitio. Haga clic en el anuncio o en el icono, y elija Allow pop ups (Permitir ventanas emergentes).

4. En AWS Management Console, elija **Services** (Servicios) > **Cloud9**. En el panel **Your environments** (Sus entornos), busque la tarjeta **reStart-python-cloud9** y elija **Open IDE** (Abrir IDE).

   Se abre el entorno de AWS Cloud9.

> Nota: Si se abre una ventana emergente con el mensaje .c9/project.settings have been changed on disk (Se ha modificado la configuración de .c9/project. en el disco), elija Discard (Descartar) para ignorarlo. Del mismo modo, si una ventana de diálogo le pide que Show third-party content (Muestre contenido de terceros), elija No para rechazar la indicación.

Creación del archivo de ejercicio de Python

1. En la barra de menú, elija **File > New From Template > Python File** (Archivo > Nuevo a partir de plantilla > Archivo en Python).

   Esta acción crea un archivo sin título.

2. Elimine el código de muestra del archivo de plantilla.
3. Elija **File > Save As…** (Archivo > Guardar como…), proporcione un nombre adecuado para el archivo de ejercicios (por ejemplo, *caesar-cipher.py*) y guárdelo en el directorio **/home/ec2-user/environment**.

Acceso a la sesión del terminal

1. En su IDE de AWS Cloud9, elija el icono **+** y seleccione **New Terminal** (Nuevo terminal).

   Se abre una sesión de terminal.

2. Para ver el directorio en el que está trabajando actualmente, escriba `pwd`. Este comando lleva a **/home/ec2-user/environment**.
3. En este directorio, localice el archivo que creó en la sección anterior.

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.

1. Guarde el archivo.
2. 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
```

1. Guarde el archivo.

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.

1. 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
```

1. Guarde el archivo.

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)
```

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.

1. 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}')
```

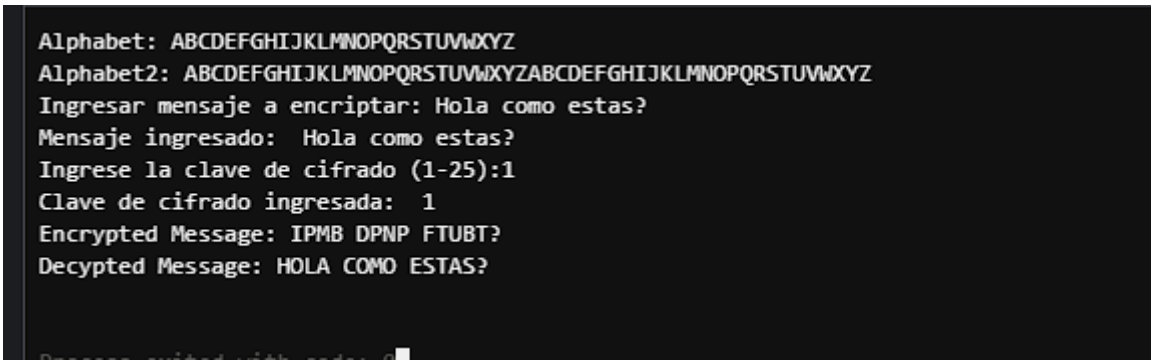
Para ayudarlo con la depuración y la comprensión del programa, se agregaron las instrucciones `print()`, pero no son estrictamente necesarias para que el programa opere de forma correcta.

1. 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.

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

```
runCaesarCipherProgram()
```



1. 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
```

1. Vuelva a ejecutar el programa con diferentes entradas.

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

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

```
def getMessage():
stringToEncrypt = input("Ingresar mensaje a encriptar: ")
return stringToEncrypt
```

```
def getCipherKey():
shiftAmount = input("Ingrese la clave de cifrado (1-25):")
return shiftAmount
```

```
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
```

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

```
def runCaesarCipherProgram():
myAlphabet="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
print(f'Alphabet: {myAlphabet}')
```

```
myAlphabet2 = getDoubleAlphabet(myAlphabet)
print(f'Alphabet2: {myAlphabet2}')

myMessage = getMessage()
print("Mensaje ingresado: ", myMessage)

myCipherKey = getCipherKey()
print("Clave de cifrado ingresada: ", myCipherKey)

myEncryptedMessage = encryptMessage(myMessage, myCipherKey, myAlphabet2)
print(f'Encrypted Message: {myEncryptedMessage}')

myDecryptedMessage = decryptMessage(myEncryptedMessage, myCipherKey, myAlphabet2)
print(f'Decypted Message: {myDecryptedMessage}')
```

```
runCaesarCipherProgram()
```