Using Functions to Implement a Caesar Cipher

Lab overview

In programming, a function is a named section of a program that performs a specific task. Python has built-in functions like `print()` that are provided by the language. Additionally, you can use functions provided by other developers through the `import` statement. For example, you can use `import math` if you want to use the `math.floor()` function. In Python, you can make your own functions, which are called *user-defined functions*.

To drive the discussion of user-defined functions, you will write a program that implements a Caesar cipher, which is a simple method of encryption. A Caesar cipher takes the letters of a message and shifts each letter along the alphabet by a certain number of places.

In this lab, you will:

- Create user-defined functions
- Use several functions to implement a Caesar cipher encryption program

Estimated completion time

60 minutes

Accessing the AWS Cloud9 IDE

1. Start your lab environment by going to the top of these instructions and choosing **Start Lab**.

   A **Start Lab** panel opens, displaying the lab status.

2. Wait until you see the message *Lab status: ready*, and then close the **Start Lab** panel by choosing the **X**.

3. At the top of these instructions, choose **AWS**.

   The AWS Management Console opens in a new browser tab. The system automatically logs you in.

   > **Note:** If a new browser tab does not open, a banner or icon at the top of your browser typically indicates that your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and choose **Allow pop ups**.

4. In the AWS Management Console, choose **Services** > **Cloud9**. In the **Your environments** panel, locate the **reStart-python-cloud9** card, and choose **Open IDE**.

   The AWS Cloud9 environment opens.

   > **Note:** If a pop-up window opens with the message *.c9/project.settings have been changed on disk*, choose **Discard** to ignore it. Likewise, if a dialog window prompts you to *Show third-party content*, choose **No** to decline.

Creating your Python exercise file

5. From the menu bar, choose **File > New From Template > Python File**.

   This action creates an untitled file.

6. Delete the sample code from the template file.

7. Choose **File > Save As...**, provide a suitable name for the exercise file (for example, *caesar-cipher.py*), and save it under the **/home/ec2-user/environment** directory.

Accessing the terminal session

8. In your AWS Cloud9 IDE, choose the **+** icon and select **New Terminal**.

   A terminal session opens.

9. To display the present working directory, enter `pwd`. This command points to **/home/ec2-user/environment**.

10. In this directory, locate the file that you created in the previous section.

Exercise 1: Creating a user-defined function

To start the process of implementing a Caesar cipher in Python, you will create a simple user-defined function.

11. From the navigation pane of the IDE, choose the file that you created in the previous *Creating your Python exercise file* section.

12. Define a function called `getDoubleAlphabet` that takes a string argument and concatenates, or combines, the given string with itself as follows:

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

> **Note:** The required parts of the function statement are the keyword `def`, a name, and the colon (:). Additionally, in Python, variables don't need to be declared, and their data types are inferred from the assignment statement.

13. Save the file.

14. To understand what the function does, take a sample input of `alphabet="ABC"`. The return string for this input would be `"ABC" + "ABC" = "ABCABC"`. The plus sign (+) concatenates the strings into one string.

Across the following exercises, you will define more functions that perform a simple task. You will then combine these functions to make a Caesar cipher program.

Exercise 2: Encrypting a message

The next function you define will request a message to encrypt from the user. You will use the built-in function called `input()`.

15. In the text editor, enter the following code, and save the file:

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

> **Note:** Functions should perform a specific task. Usually, because functions perform a specific task, your functions will also probably be short. Though this function returns a string, it doesn't take an argument like the `getDoubleAlphabet()` function.

Exercise 3: Getting a cipher key

The *cipher key* is how far you will shift the letters. By using two alphabets, you can have a cipher key that is any integer from 1 to 25. Don't count the key at index 26 because that key would shift you back to the original message.

16. Define a function to request a cipher key from the user by entering the following code:

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

17. Save the file.

Exercise 4: Encrypting a message

So far, the functions have been short and simple. That is usually the case when you keep to a specific task inside a function. The `encryptMessage` function will be a little longer.

18. Before writing the code, you should plan out the algorithm for encryption as follows:

> 1. Take three arguments: the message, the cipherKey, and the alphabet.
> 2. Initialize variables.
> 3. Use a `for` loop to traverse each letter in the message.
> 4. For a specific letter, find the position.
> 5. For a specific letter, determine the new position given the cipher key.
> 6. If current letter is in the alphabet, append the new letter to the encrypted message.
> 7. If current letter is not in the alphabet, append the current letter.
> 8. Return the encrypted message after exhausting all the letters in the message.

19. In the exercise file, enter the following code, and follow the logic by reviewing the steps of the previous algorithm:

```
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
```

20. Save the file.

Exercise 5: Decrypting a message

Functions are useful because you can reuse them. You will write a `decryptMessage()` function by reusing the `encryptMessage()` function. For this simple encryption, you can undo the encryption by shifting each letter back. Thus, instead of adding the cipher key, you will subtract the cipher key. To avoid rewriting most of the logic, you will pass in a negative cipher key.

21. Next, enter the following code, and save the file:

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

Exercise 6: Creating a main function

You have built a collection of user-defined functions that will help you write a Caesar cipher program. The main logic of the program will, of course, also be contained in a function.

22. Before you look at the code, plan out your logic:

> 1. Define a string variable to contain the English alphabet.
> 2. To be able to shift letters, double your alphabet string.
> 3. Get a message to encrypt from the user.
> 4. Get a cipher key from the user.
> 5. Encrypt the message.
> 6. Decrypt the message.

23. In the exercise file, enter the following code, and follow the logic by reviewing the steps of the previous algorithm:

```
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}')
```

To help with debugging and understanding the program, `print()` statements were added, but they are not strictly necessary for the program to operate correctly.

24. Save and run the file, and then view the results.

    Nothing happens. Why? Recall that a function is a named section of a program that performs a specific task. You have not called your function.

25. To call the function, add the following line to your **.py** file and save the file:

```
runCaesarCipherProgram()
```

26. Run the program again. The output should be similar to the following:

```
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
```

27. Re-run the program with different inputs.

Congratulations! You have worked with user-defined functions and implemented an encryption program!

End Lab

Congratulations! You have completed the lab.

28. Choose **End Lab** at the top of this page, and then select Yes to confirm that you want to end the lab.

    A panel indicates that *DELETE has been initiated... You may close this message box now.*

29. A message *Ended AWS Lab Successfully* is briefly displayed, indicating that the lab has ended.

Additional Resources

For more information about AWS Training and Certification, see https://aws.amazon.com/training/.

Your feedback is welcome and appreciated. If you would like to share any suggestions or corrections, please provide the details in our AWS Training and Certification Contact Form.

© 2022 Amazon Web Services, Inc. and its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

< Rubric: 13 - Caesar Cipher | Points: 0 >