Debugging the Caesar Cipher Program

Lab overview

Recall that a debugger is a computer program that is used to test and find bugs (debug) other programs. In this lab, you will use the Python Debugger (pdb) to find and fix bugs in a Python program.

In this lab, you will:

- Use the Python Debugger
- Debug the different versions of the Caesar cipher program that you created in a previous lab

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**
6. Delete the sample code from the template file.
7. Choose **File > Save As...**, and provide a suitable name for the exercise file (for example, *debug-caesar-1.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 you created in the previous section.

Exercise 1: Working with the buggy Caesar cipher program - Part 1

In the Functions lab, you created a Caesar cipher program to encrypt and decrypt a message. In this lab, you will use the Python Debugger (pdb) to find and fix errors in buggy versions of the program.

11. From the navigation pane of the IDE, choose the **.py** file that you created in the previous *Creating your Python exercise file* section. Copy the following code and paste it in the file:

```
# Module Lab: Caesar Cipher Program Bug #1
#
# In a previous lab, you created a Caesar cipher program. This version of
# the program is buggy. Use a debugger to find the bug and fix it.

# Double the given alphabet
def getDoubleAlphabet(alphabet):
    doubleAlphabet = alphabet + alphabet
    return doubleAlphabet

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

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

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

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

# Main program logic
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'Decrypted Message: {myDecryptedMessage}')

# Main logic
runCaesarCipherProgram()
```

12. Save the file.
13. Try running the first buggy Caesar cipher program. You should receive an error similar to the one in the following example.

```
Alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Alphabet2: ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ
Please enter a message to encrypt: AWS Restart rocks
AWS Restart rocks
Please enter a key (whole number from 1-25): 2
2
Traceback (most recent call last):
  File "/home/ec2-user/environment/caesar_cipher_program_bug_1.py", line 56, in <module>
    runCaesarCipherProgram()
  File "/home/ec2-user/environment/caesar_cipher_program_bug_1.py", line 50, in runCaesarCiph
erProgram
    myEncryptedMessage = encryptMessage(myMessage, myCipherKey, myAlphabet2)
  File "/home/ec2-user/environment/caesar_cipher_program_bug_1.py", line 28, in encryptMessag
e
    newPosition = position + cipherKey
TypeError: unsupported operand type(s) for +: 'int' and 'str'

Process exited with code: 0
```

    The program ends in a *traceback*. A traceback is a stack trace that starts from the point of an exception handler. It then goes down the call chain to the point where the exception was raised. In other words, an error occurred.

14. Use the debugger to find and fix the bug in the first lab file for the buggy Caesar cipher.

Exercise 2: Working with the buggy Caesar cipher program - Part 2

Errors that result in a traceback are usually easier to fix because the traceback provides helpful clues, like line numbers.

15. From the menu bar, choose **File > New From Template > Python File**
16. Delete the sample code from the template file.
17. Choose **File > Save As...**, and provide a suitable name for the exercise file (for example, *debug-caesar-2.py*) and save it under the **/home/ec2-user/environment** directory.
18. Copy the following code and paste it into the newly created Python file:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

17

```
# Module Lab: Caesar Cipher Program Bug #2
#
# In a previous lab, you created a Caesar cipher program. This version of
# the program is buggy. Use a debugger to find the bug and fix it.

# Double the given alphabet
def getDoubleAlphabet(alphabet):
    doubleAlphabet = alphabet + alphabet
    return doubleAlphabet

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

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

# Encrypt message
def encryptMessage(message, cipherKey, alphabet):
    encryptedMessage = ""
    uppercaseMessage = ""
    uppercaseMessage = message
    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

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

# Main program logic
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'Decrypted Message: {myDecryptedMessage}')

# Main logic
runCaesarCipherProgram()
```

19. Save the file.
20. Run the second buggy Caesar cipher program. The program seems to end correctly, but double-check the output. The message is only partially encrypted, as shown in the example.

```
Alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Alphabet2: ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ
Please enter a message to encrypt: AWS Restart rocks!
AWS Restart rocks!
Please enter a key (whole number from 1-25): 2
2
Encrypted Message: CYU Testart rocks!
Decrypted Messgae: AWS Restart rocks!

Process exited with code: 0
```

21. Step through the program by using the debugger, and try to find the bug.
22. To see if you can get clues about the bug, run the program several times with different inputs. What do you notice?
23. When you find the bug, fix it, and validate your fix by running the program and entering different inputs.

Exercise 3: Working with the buggy Caesar cipher program - Part 3

In this exercise, you will debug a third buggy version of the Caesar cipher program.

24. From the menu bar, choose **File > New From Template > Python File**
25. Delete the sample code from the template file.
26. Choose **File > Save As...**, and provide a suitable name for the exercise file (such as *caesar_debug-3.py*) and save it under the **/home/ec2-user/environment** directory.
27. Copy the following code and paste it into the newly created Python file:

```
# Module Lab: Caesar Cipher Program Bug #3
#
# In a previous lab, you created a Caesar cipher program. This version of
# the program is buggy. Use a debugger to find the bug and fix it.

# Double the given alphabet
def getDoubleAlphabet(alphabet):
    doubleAlphabet = alphabet + alphabet
    return doubleAlphabet

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

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

# Encrypt message
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

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

# Main program logic
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'Decrypted Message: {myDecryptedMessage}')

# Main logic
runCaesarCipherProgram()
```

28. Save the file.
29. Run the third buggy Caesar cipher program. The output looks almost correct. However, the decryption of the message *AWS Restart* message is incorrect, as shown in the following example:

```
Alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Alphabet2: ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ
Please enter a message to encrypt: AWS Restart rocks!
AWS Restart rocks!
Please enter a key (whole number from 1-25): 2
2
Encrypted Message: CYU TGUVCTV TQEMU!
Decrypted Message: EAW VIWXEVX VSGOW!

Process exited with code: 0
```

30. It’s time to start the debugger again! Find and fix the bug.

Exercise 4: Working with the buggy Caesar cipher program - Part 4

In this exercise, you will debug the fourth (and final) buggy version of the Caesar cipher program.

31. From the menu, choose **File > New From Template > Python File**
32. Delete the sample code from the template file.
33. Choose **File > Save As...**, provide a suitable name for the exercise file (such as *debug-caesar-4.py*), and save it under the **/home/ec2-user/environment** directory.
34. Copy the following into this file:

```
# Module Lab: Caesar Cipher Program Bug #4
#
# In a previous lab, you created a Caesar cipher program. This version of
# the program is buggy. Use a debugger to find the bug and fix it.

# Double the given alphabet
def getDoubleAlphabet(alphabet):
    doubleAlphabet = alphabet + alphabet
    return doubleAlphabet

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

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

# Encrypt message
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

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

# Main program logic
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'Decrypted Message: {myEncryptedMessage}')

# Main logic
runCaesarCipherProgram()
```

35. Save the file.
36. Run the fourth buggy Caesar cipher program. The output should be similar to the following example:

```
Alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Alphabet2: ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ
Please enter a message to encrypt: AWS Restart rocks!
AWS Restart rocks!
Please enter a key (whole number from 1-25): 2
2
Encrypted Message: CYU TGUVCTV TQEMU!
Decrypted Message: CYU TGUVCTV TQEMU!

Process exited with code: 0
```

37. The output seems buggy. Find and fix the final bug.

Congratulations! You have debugged four programs, and you have completed all the labs for this course.

End Lab

Congratulations! You have completed the lab.

38. 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.*

39. 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/ (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 (https://support.aws.amazon.com/#/contacts/aws-training).

- ○ Yes
- ○ No

Previous