European University Cyprus
MSc Cybersecurity

Individual Assignment 1

Second Coursework

CYS 625X – Cryptography

Dr Philippos Isaia

Panayiotis Stamatis
ps205054@students.euc.ac.cy
ID:20205054

8 December 2020

Exercise 1.

Software Description

Caesar Cipher is a command line program implemented in Python 3.8 with the help of PyCharm 2020.3 editor. It takes as input a file with plaintext or ciphertext and based on the selected mode, encrypts or decrypts the corresponded file using the provided key, and the result exported to a file with the desired filename. The default dataset is the English alphabet. If a mandatory option is omitted, program through an exception.

Source: https://github.com/stamatispanos/cys625/caesar.py

The file is called "caesar.py" and can be executed with the following command,

>> caesar.py [options]

The available [options] are listed below,

-h or - -help	| show help	-> optional
-i or - -input	| (text) input filename	-> mandatory
-o or - -output	| (text) output filename	-> mandatory
-k or - -key	| (int) encryption/decryption key to use	-> mandatory
-m or - -mode	| use 'enc' for encryption, 'dec' for decryption	-> mandatory

Options description

- *Input file* -> should be a text file (plaintext or ciphertext), with letters from A to Z, in upper or lower case. Other characters processed without change.
- *output file* -> will be a) a plaintext in lower case if 'dec' mode is selected,
  b) a ciphertext in lower case if 'enc' mode is selected.
- *Key* -> can be any integer number.
- *mode* -> 'enc' encrypt the input file and 'dec' decrypt it.

Cipher.py usage examples

- encryption mode >> Caesar.py -i plaintext.txt -o ciphertext.txt -k 15 -m enc
- decryption mode >> Caesar.py --input ciphertext.txt --output plaintext.txt --key 15 --mode dec

Key points of the source code

Program using the **getopt()** module to parse the parameters from the command line and pass them to corresponding variables. Next, it loads the content of the input file and pass it to the main function with the key and the desired mode to execute. The function **main()**, starts with the conversion of the input data to lower case letters. Next, a for loop starts the translation of every character based on the selected mode and key, by the end it returns a string. In more detail, with **find()** we get the position of the char found in our alphabet, if result is -1, it means that char is not listed and not processed. If result is not -1 and based on the mode, we add or subtract the key from the initial char's index position. Because key value could be any integer number, shift value should be calculated using mod26. Now we know the shift, we can get the new char from the alphabet and append it to a string. Throughout the program execution, any exception might occur will handled with the try, except statements.

Source Code of caesar.py

```
1 import sys
2 import getopt
3
4
5 # exercise 1
6 def main(data_in, key_, mode_):
```

```
7       global new_index
8       # english alphabet
9       alphabet = 'abcdefghijklmnopqrstuvwxyz'
10      # initialize variable
11      data_out = ''
12      # turn input data to lowercase
13      data_in = data_in.lower()
14
15      for c in data_in:
16
17          # find character position
18          index = alphabet.find(c)
19
20          if index == -1:
21              # if Character not listed in the alphabet, return char
22              data_out += c
23          else:
24              # find the shift based on mode and key
25              if mode_ == 'enc':
26                  new_index = index + key_
27              elif mode_ == 'dec':
28                  new_index = index - key_
29
30              # find the shifted char position
31              new_index %= len(alphabet)
32              # append new char to string
33              data_out += alphabet[new_index:new_index + 1]
34      # Return the encrypted/decrypted string
35      return data_out
36
37
38  if __name__ == '__main__':
39      syntaxShort = "hi:o:k:m:"
40      syntaxLong = ["help", "input=", "output=", "key=", "mode="]
41      # Variables initialization
42      in_file = []
43      out_file = []
44      key = []
45      mode = []
46
47      try:
48          opts, args = getopt.getopt(sys.argv[1:], syntaxShort, syntaxLong)
49          for option, a in opts:
50
51              if option in ("-h", "--help"):
52                  print("usage: caesar.py [options]")
53                  print("short  long        function")
54                  print(" -h   --help       show this help")
55                  print(" -i   --input      input filename (text, for example
input.txt)")
56                  print(" -o   --output     output filename (text, for example
output.txt)")
57                  print(" -k   --key        encryption key to use (integer)")
58                  print(" -m   --mode       function mode. Type 'enc' for encryption
mode")
59                  print("                                  Type 'dec' for decryption
mode")
60                  sys.exit()
61              elif option in ("-i", "--input"):
62                  in_file = a
63              elif option in ("-o", "--output"):
64                  out_file = a
65              elif option in ("-k", "--key"):
66                  key = int(a)
67              elif option in ("-m", "--mode"):
68                  mode = str(a)
69
```

```
70          # Open file, read it and load contents to data
71          with open(in_file, 'rt') as filein:
72              data = filein.read()
73          # Call the encryption/decryption function
74          output_data = main(data, key, mode)
75          # print results to output file
76          with open(out_file, 'wt') as fileout:
77              fileout.write(output_data)
78
79      except getopt.GetoptError as err:
80          print('Error parsing args:', err)
81          print('type -h or --help for options')
82          sys.exit(1)
83      except Exception as e:
84          print('Error', e)
85          print('type -h or --help for options')
86          sys.exit(2)
87
```

Exercise 2.

Software Description

Caesar Cipher with custom Alphabet is a command line program implemented in Python 3.8 with the help of PyCharm 2020.3 editor. It takes as input, a file with plaintext or ciphertext and a file with a custom alphabet, and based on the selected mode, encrypts or decrypts the corresponded file, using the provided alphabet and key. The result exported to a file with the desired filename. If a mandatory option is omitted, program through an exception.

Source: https://github.com/stamatispanos/cys625/caesarAlpha.py

The file is called “caesarAlpha.py” and can be executed with the following command,

>> caesarAlpha.py [options]

The available [options] are listed below,

-h or - -help	| show help	-> optional
-i or - -input	| (text) input filename	-> mandatory
-o or - -output	| (text) output filename	-> mandatory
-k or - -key	| (int) encryption/decryption key to use	-> mandatory
-a or - -alphabet	| (text) alphabet filename	-> mandatory
-m or - -mode	| use ‘enc’ for encryption, ‘dec’ for decryption	-> mandatory

Options description

- *Input file* -> should be a text file (plaintext or ciphertext), with letters from A to Z, in upper or lower case. Other characters processed without changed.
- *output file* -> will be a) a plaintext in lower case if ‘dec’ mode is selected,
  b) a ciphertext in lower case if ‘enc’ mode is selected.
- *Key* -> can be any integer number.
- *alphabet* -> should be a csv file. Value separator (delimiter) can be character “,” or “;”.
- *mode* -> ‘enc’ encrypt the input file and ‘dec’ decrypt it.

Cipher.py usage examples

- encryption mode >> CaesarAlpha.py -i plaintext.txt -o ciphertext.txt -k 15 -a myAlphabet.csv -m enc
- decryption mode >> CaesarAlpha.py -i ciphertext.txt -o plaintext.txt -k 15 -a myAlphabet.csv -m dec

Key points of the source code

Program using the **getopt()** module to parse the parameters from the command line and pass them to corresponding variables. Next, it loads the contents of the input files (text file and alphabet file) and pass them to the main function with the key and the desired mode to execute. The function **main()**, starts with the conversion of the input data to lower case letters. Next, a for loop starts the translation of every character based on the selected mode and key, by the end it returns a string. In more detail, with **find()** we get the position of the char found in our custom alphabet, if result is -1, it means that char is not listed and not processed. If result is not -1 and based on the mode, we add or subtract the key from the initial char's index position. Because key value could be any integer number, shift value should be calculated using **mod** with the length of our alphabet. Now we know the shift, we can get the new char from the custom alphabet and append it to a string. Throughout the program execution, any exception might occur will handled with the try, except statements.

Source Code of caesarAlpha.py

```
import sys
import getopt


# exercise 2
def main(data_in, key_, mode_, custom_alphabet):
    global new_index
    # load custom alphabet
    alphabet = custom_alphabet
    # initialize variable
    data_out = ''
    # turn input data to lowercase
    data_in = data_in.lower()

    for c in data_in:

        # find character position
        index = alphabet.find(c)

        if index == -1:
            # if Character is not listed in the alphabet, return char
            data_out += c
        else:
            # find the shift based on mode and key
            if mode_ == 'enc':
                new_index = index + key_
            elif mode_ == 'dec':
                new_index = index - key_

            # find the shifted char position
            new_index %= len(alphabet)
            # append new char to string
            data_out += alphabet[new_index:new_index + 1]
    # Return the encrypted/decrypted text
    return data_out


if __name__ == '__main__':
    syntaxShort = "hi:o:k:a:m:"
    syntaxLong = ["help", "input=", "output=", "key=", "alphabet=", "mode="]
    # Variables initialization
    in_file = []
    out_file = []
    key = []
    alphabetFile = []
    mode = []

    try:
```

```
49          opts, args = getopt.getopt(sys.argv[1:], syntaxShort, syntaxLong)
50          for option, a in opts:
51
52              if option in ("-h", "--help"):
53                  print("usage: caesar.py [options]")
54                  print("short  long         function")
55                  print(" -h   --help       show this help")
56                  print(" -i   --input      input filename (text, for example
input.txt)")
57                  print(" -o   --output     output filename (text, for example
output.txt)")
58                  print(" -k   --key        encryption key to use (integer)")
59                  print(" -a   --alphabet   custom alphabet filename (text, for
example alphabet.csv)")
60                  print(" -m   --mode       function mode. Type 'enc' for encryption
mode")
61                  print("                                   Type 'dec' for decryption
mode")
62                  sys.exit()
63              elif option in ("-i", "--input"):
64                  in_file = a
65              elif option in ("-o", "--output"):
66                  out_file = a
67              elif option in ("-k", "--key"):
68                  key = int(a)
69              elif option in ("-a", "--alphabet"):
70                  alphabetFile = a
71              elif option in ("-m", "--mode"):
72                  mode = str(a)
73
74          # Open file, read it and load contents to data
75          with open(in_file, 'rt') as filein:
76              data = filein.read()
77          # Read alphabet file
78          with open(alphabetFile, 'rt') as csvfile:
79              raw_data = csvfile.read()
80              # variable init
81              alpha = []
82              # remove most common csv delimiter chars like (;) and (,)
83              if raw_data.find(";") != -1:
84                  alpha = raw_data.replace(";", "")
85              elif raw_data.find(",") != -1:
86                  alpha = raw_data.replace(",", "")
87
88          # Call the encryption/decryption function
89          new_data = main(data, key, mode, alpha)
90          # print results to output file
91          with open(out_file, 'wt') as fileout:
92              fileout.write(new_data)
93
94      except getopt.GetoptError as err:
95          print('Error parsing args:', err)
96          print('type -h or --help for options')
97          sys.exit(1)
98      except Exception as e:
99          print('Error', e)
100         print('type -h or --help for options')
101         sys.exit(2)
102
```

Exercise 3.

Software Description

Frequency counter is a command line program implemented in Python 3.8 with the help of PyCharm 2020.3 editor. It takes as input, a file with ciphertext and counts the occurrences of every letter found into the English alphabet, no matter if it is lower or upper case. The results represented as a list of characters with the corresponded occurrences, comma separated and ordered alphabetically from A to Z. Output format example "`"'A': 1,'B': 1,'C': 2,……`". Finally, results exported to a file with the desired filename. If a mandatory option is omitted, program through an exception.

Source: https://github.com/stamatispanos/cys625/frequency.py

The file is called "frequency.py" and can be executed with the following command,

>> frequency.py [options]

The available [options] are listed below,

-h or - -help	| show help	-> optional
-i or - -input	| (text) input filename	-> mandatory
-o or - -output	| (text) output filename	-> mandatory

Options description

- *Input file* -> should be a text file (ciphertext), with letters from A to Z, in upper or lower case. Other characters processed without counted.
- *output file* -> will be a text file with an alphabetically listed chars with their occurrences. The format is "`"'A': 1,'B': 1,'C': 2,……`". Characters not listed in the alphabet will not represented.

Cipher.py usage examples

- >> frequency.py -i ciphertext.txt -o letterFrequency.txt

Key points of the source code

Program using the **getopt()** module to parse the parameters from the command line and pass them to corresponding variables. Next, it loads the content of the input file and pass it to the counter function. The function **counter()**, starts with the conversion of the input data to upper case letters. Next, using python's counter container (**collections.counter()**), it count the letter occurrences in the provided text file and the results stored to a counter{} type variable (letters_count). With a for loop we sort the output by keys. In more detail, with **find()** we want to check if the character exists in the alphabet, if result is -1, it means that char is not listed and not processed. If result is not -1 the character append to a string with the corresponded frequency number and the appropriate format. Throughout the program execution, any exception might occur will handled with the try, except statements.

Source Code of frequency.py

```
1  import sys
2  import getopt
3  import collections
4
5
6  # exercise 3
7  def counter(data_in):
8      # english alphabet
9      alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
10     # turn input data to lowercase
```

```
    raw_text = data_in.upper()
    # initialize variable
    data_out = '"'
    # find letter frequency
    letters_count = collections.Counter(raw_text)

    for c in alphabet:
        # check if letter is valid
        index = alphabet.find(c)

        # if Character is valid (listed in the alphabet)
        if index != -1:
            # get letter appearances from counter list
            single_letter_count = letters_count[c]
            # append key and value to string
            data_out = data_out + "'" + str(c) + "': " + str(single_letter_count) +
","

    # Return frequencies string
    data_out += '"'
    return data_out


if __name__ == '__main__':
    syntaxShort = "hi:o:"
    syntaxLong = ["help", "input=", "output="]
    # Variables initialization
    in_file = []
    out_file = []

    try:
        opts, args = getopt.getopt(sys.argv[1:], syntaxShort, syntaxLong)
        for option, a in opts:

            if option in ("-h", "--help"):
                print("usage: caesar.py [options]")
                print("short  long        function")
                print(" -h   --help       show this help")
                print(" -i   --input      input filename (text, for example
input.txt)")
                print(" -o   --output     output filename (text, for example
output.txt)")
                sys.exit()
            elif option in ("-i", "--input"):
                in_file = a
            elif option in ("-o", "--output"):
                out_file = a

        # Open file, read it and load contents to data
        with open(in_file, 'rt') as filein:
            data = filein.read()
        # Call the counter function
        output_data = counter(data)
        # print results to output file
        with open(out_file, 'wt') as fileout:
            fileout.write(output_data)

    except getopt.GetoptError as err:
        print('Error parsing args:', err)
        print('type -h or --help for options')
        sys.exit(1)
    except Exception as e:
        print('Error', e)
        print('type -h or --help for options')
        sys.exit(2)

```