Slovak University of Technology in Bratislava

Faculty of Informatics and Information Technologies

Ilkovičova 2, 842 16 Bratislava 4

Introduction to Cryptography

HomeWork in Python

Academic year: 2021/2022, Winter Semester

Supervisor:

prof. Ing. Volodymyr Khylenko, PhD.

Students:

Ahmed Lotfi Alqnatri

1A.

1a.

```
01]: def encrypt_1a(surname):

        encrypted_surname = ""

        for letter in surname:

            if letter.isupper():
                new_unicode = ord("Z") - ((ord(letter) + 25) - ord("Z"))
                new_character = chr(new_unicode)
                encrypted_surname += new_character

            elif letter.islower():
                new_unicode = ord("z") - ((ord(letter) + 25) - ord("z"))
                new_character = chr(new_unicode)
                encrypted_surname += new_character

            else:
                encrypted_surname += letter

        return encrypted_surname
```

To encrypt the message we iterate over the input and then check for each letter and perform a couple of checks, first check is checking if the letter is an Upper letter and then we do a logic to get the encryption by getting the ASCII value for the letter using the function ord() and then we add the value 25, then we subtract the ASCII value of the last letter in the alphabet “Z”, then we get a value of how much we should subtract from the ASCII value of “Z” after we get the ASCII value of the final encrypted letter we transfer to Char using the function char() and then we add it to our string.
Example: Letter “A”, ord(“A”) = 65, 65 + 25 = 90, 90 - ord(“Z”) = 90 - 90 = 0, ord(“Z”) - 0 = 90
Encrypted letter = char(90) = ‘Z’

If the letter is a Lower letter we do very similar operation as for the Upper case but instead of using the ASCII value of ord(“Z") we use the ASCII value of ord(“z”) lower case
Example: Letter “z”, ord(“z”) = 122, 122 + 25 = 147, 147- ord(“z”) = 147 - 122 = 25, ord(“z”) - 25 = 122 - 25 = 97
Encrypted letter = char(97) = ‘a’

If the letter is not Upper orLower we ignore it and just append it to the string

```python
def decrypt_1a(encrypted_surname):

    decrypted_surname = ""

    for letter in encrypted_surname:

        if letter.isupper():
            new_unicode = ord("Z") - ((ord(letter) + 25) - ord("Z"))
            new_character = chr(new_unicode)
            decrypted_surname += new_character

        elif letter.islower():
            new_unicode = ord("z") - ((ord(letter) + 25) - ord("z"))
            new_character = chr(new_unicode)
            decrypted_surname += new_character

        else:
            decrypted_surname += letter

    return decrypted_surname
```

For decryption we basically do an encrypt for the encrypted string to decrypted, so it has the same logic as above

For the time of running the algorithm we got very similar results between encryption and decryption
seconds for encrypting --- 4.2100000428035855e-05
seconds for decrypting --- 3.609999839682132e-05

```
[114]: string = input("Enter your message: ")
       start = timeit.default_timer()

       encrypted_surname = encrypt_1a(string)

       end = timeit.default_timer()

       print("Encrypted message: ", encrypted_surname)
       print("---  seconds for encrypting ---" , end-start)


       start = timeit.default_timer()

       decrypted_surname = decrypt_1a(encrypted_surname)

       end = timeit.default_timer()

       print("Decrypted message: ",decrypted_surname)
       print("---  seconds for decrypting ---" , end-start)

       Enter your message:  Alqnatri
       Encrypted message:  Zojmzgir
       ---  seconds for encrypting --- 4.2100000428035855e-05
       Decrypted message:  Alqnatri
       ---  seconds for decrypting --- 3.609999839682132e-05
```

1B.

```
def encrypt_1b(surname):

    encrypted_surname = ""

    for letter in surname:

        if letter.isupper():
            if ord(letter) <= ord("M"):

                new_unicode = ord("M") - ((ord(letter) + 12) - ord("M"))
                new_character = chr(new_unicode)
                encrypted_surname += new_character

            else:

                new_unicode = ord("Z") - ((ord(letter) + 12) - ord("Z"))
                new_character = chr(new_unicode)
                encrypted_surname += new_character

        elif letter.islower():
            if ord(letter) <= ord("m"):

                new_unicode = ord("m") - ((ord(letter) + 12) - ord("m"))
                new_character = chr(new_unicode)
                encrypted_surname += new_character

            else:

                new_unicode = ord("z") - ((ord(letter) + 12) - ord("z"))
                new_character = chr(new_unicode)
                encrypted_surname += new_character

        else:
            encrypted_surname += letter

    return encrypted_surname
```

To encrypt the message we iterate over the input and then check for each letter and perform a couple of checks, first check is checking if the letter is an Upper letter and then we do another check to see if the letter is between the first half of alphabet ord(letter) <= ord("M") and then we do a logic to get the encryption by getting the ASCII value for the letter using the function ord() and then we add the value 12, then we subtract the ASCII value of the last letter in the alphabet “M”, then we get a value of how much we should subtract from the ASCII value of “M” after we get the ASCII value of the final encrypted letter we transfer to Char using the function char() and then we add it to our string.

Second check is checking if the letter is an Upper letter and then we do another check to see if the letter is between the first half of alphabet ord(letter) > ord("M") and then we do a logic to get the encryption by getting the ASCII value for the letter using the function ord() and then we add the value 12, then we subtract the ASCII value of the last letter in the alphabet “Z”, then we get a value of how much we should subtract from the ASCII value of “Z” after we get the ASCII value of the final encrypted letter we transfer to Char using the function char() and then we add it to our string.
For the lower letter we do the exact same logic but we use lower letter in the ord(“A”) function

```
def decrypt_1b(encrypted_surname):

    decrypted_surname = ""

    for letter in encrypted_surname:

        if letter.isupper():
            if ord(letter) <= ord("M"):

                new_unicode = ord("M") - ((ord(letter) + 12) - ord("M"))
                new_character = chr(new_unicode)
                decrypted_surname += new_character

            else:

                new_unicode = ord("Z") - ((ord(letter) + 12) - ord("Z"))
                new_character = chr(new_unicode)
                decrypted_surname += new_character

        elif letter.islower():
            if ord(letter) <= ord("m"):

                new_unicode = ord("m") - ((ord(letter) + 12) - ord("m"))
                new_character = chr(new_unicode)
                decrypted_surname += new_character

            else:

                new_unicode = ord("z") - ((ord(letter) + 12) - ord("z"))
                new_character = chr(new_unicode)
                decrypted_surname += new_character

        else:
            decrypted_surname += letter

    return decrypted_surname
```

For decryption we basically do an encrypt for the encrypted string to decrypted, so it has the same logic as above

For the time of running the algorithm we got very similar results between encryption and decryption

--- seconds for encrypting --- 4.1799998143687844e-05

--- seconds for decrypting --- 3.560000186553225e-05

```
string = input("Enter your message: ")
start = timeit.default_timer()

encrypted_surname = encrypt_1b(string)

end = timeit.default_timer()

print("Encrypted message: ", encrypted_surname)
print("---  seconds for encrypting ---" , end-start)


start = timeit.default_timer()

decrypted_surname = decrypt_1b(encrypted_surname)

end = timeit.default_timer()

print("Decrypted message: ",decrypted_surname)
print("---  seconds for decrypting ---" , end-start)
```

```
Enter your message:  Alqnatri
Encrypted message:  Mbwzmtve
---  seconds for encrypting --- 4.1799998143687844e-05
Decrypted message:  Alqnatri
---  seconds for decrypting --- 3.560000186553225e-05
```

1C.

1c.

```
def encrypt_1c (surname):

    if surname[0].isupper():
        shift = ord(surname[0]) - ord("A") + 1
    elif surname[0].islower():
        shift = ord(surname[0]) - ord("a") + 1
    else:
        return

    encrypted_surname = ""

    for letter in surname:

        if letter.isupper():

            new_index = ((ord(letter) - ord("A")) + shift) % 26
            new_unicode = new_index + ord("A")
            new_character = chr(new_unicode)
            encrypted_surname += new_character

        elif letter.islower():

            new_index = ((ord(letter) - ord("a")) + shift) % 26
            new_unicode = new_index + ord("a")
            new_character = chr(new_unicode)
            encrypted_surname += new_character

        else:
            encrypted_surname += letter

    return encrypted_surname
```

For this algorithm we first check if the value of K of how much we need to shift the letters for both letter and upper cases, after that we iterate over the input and check which case we have the letter and act accordingly if the letter is an upper case we get the ASCII value of the letter and subtract from the ASCII value of the letter “A” using the function ord() and then we add the shift value that we obtained before and use the mod operator of value 26 to check if the sum is less than 26 or bigger and get the its value, after we get the index of the letter we add the ASCII value of the letter “A” using the function ord() and extract the Char value of the decrypted letter using the function char() and then append it to the decrypted string we do the same logic incase of lower case but we use the ord(“a”) instead of ord(“A”).

```python
def decrypt_1c(encrypted_surname, shift):

    decrypted_surname = ""

    for letter in encrypted_surname:

        if letter.isupper():

            new_index = ((ord(letter) - ord("A")) - shift) % 26
            new_unicode = new_index + ord("A")
            new_character = chr(new_unicode)
            decrypted_surname += new_character

        elif letter.islower():

            new_index = ((ord(letter) - ord("a")) - shift) % 26
            new_unicode = new_index + ord("a")
            new_character = chr(new_unicode)
            decrypted_surname += new_character

        else:
            decrypted_surname += letter

    return decrypted_surname
```

For decryption we pass the encrypted message and the value we used to shift the letters with it and we do the same logic as for encryption but instead of adding the shift value
new_index = ((ord(letter) - ord("A")) + shift) % 26, we subtract the shift value
new_index = ((ord(letter) - ord("A")) - shift) % 26,
To decrypt this algorithm we need to use a technique called brute force by trying all the shift values of the alphabet which are 26.

```python
string = input("Enter your message: ")
start = timeit.default_timer()

encrypted_surname = encrypt_1c(string)
end = timeit.default_timer()
print("Encrypted message: ", encrypted_surname)
print("---  seconds for encrypting ---" , end-start)

if encrypted_surname != None:
    list_decrypted_surname = []
    start = timeit.default_timer()

    for shift in range (1, 27):
        result = decrypt_1c(encrypted_surname, shift)
        list_decrypted_surname.append(result)

    end = timeit.default_timer()
    print("List of all possible decryptions...")
    for surname in list_decrypted_surname:
        print(surname)

    print("---  seconds for decrypting ---" , end-start)
```

```
Enter your message:  Alqnatri
Encrypted message:  Bmrobusj
---  seconds for encrypting --- 4.289999924367294e-05
List of all possible decryptions...
Alqnatri
Zkpmzsqh
Yjolyrpg
Xinkxqof
Whmjwpne
Vglivomd
Ufkhunlc
Tejgtmkb
Sdifslja
Rcherkiz
Qbgdqjhy
Pafcpigx
Ozebohfw
Nydangev
Mxczmfdu
Lwbylect
Kvaxkdbs
Juzwjcar
Ityvibzq
Hsxuhayp
Grwtgzxo
Fqvsfywn
Epurexvm
Dotqdwul
Cnspcvtk
Bmrobusj
---  seconds for decrypting --- 7.440000263159163e-05
```

As for the execution time we see that the decryption took almost double the time of the encryption or other algorithms because we did not have the value of shift and we had to use brute force to get all possible values.
seconds for encrypting --- 4.380000245873816e-05
seconds for decrypting --- 7.809999806340784e-05

Link for the code on Github.