Mohamed Ali Damergi

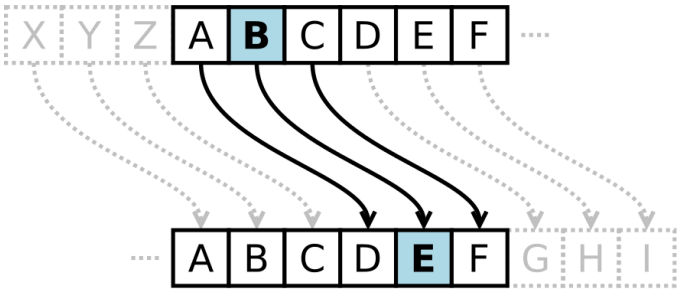
Tp Math a l'usage de l'info

Remarques: - Le code ne s'exécute que avec la version 3.10 de python.
- Tout le code est disponible en pièce jointe sous le nom: Tp1.py.
- Si l'OS est different de MACos il faut ajouter un (#) Avant la ligne 4.

Lors de ce projet on se propose de coder un programme en python qui crypte et décrypte des message en utilisant le code de césar (affine):

Definition :

le code de césar aussi nommé chiffrement par décalage fut utilisé pour la première fois par Jules césar afin d'envoyer des messages secrets que seul lui et le destinataire pouvaient déchiffrer.



Cette méthode de chiffrement consiste a décaler a l'aide d'une clé l'ordre des lettres de l'alphabet suivant la fonction suivante :

f(L) = (x+L) [26] => avec la clé (x) et l'ordre de la lettre (L)
Pour déchiffrer le message il suffit de soustraire: F(x) = (x-n) [26]

Le code de César (Affine) repose sur le meme principe que le code de césar mais a la place d'utiliser qu'une seule clé la on va en utiliser deux a l'aide d'une fonction affine:

f(L) = ((x*L) +y) [26]

Pour ce qui est de l'algorithme du code de césar affine nous avons plusieurs façons de procéder, pour ma part j'ai choisi de permettre a l'utilisateur de saisir les clés et le code a chiffrer mais aussi en modifiant un peut la fonction de base de césar affine j'ai fait en sorte de permettre a mon programme de fonctionner avec n'importe quel alphabet et pas seulement les 26 lettres:

F(L) = ((x*L)+y) [la longueur de mon alphabet]

J'ai aussi rajouter une condition a ma fonction de cryptage afin d'éviter les erreur
=> Si le pgcd entre la premiere clé et la longueur de mon alphabet est égal a 1
le programme va retourner une erreur.

La **fonction pgcd** fonctionne de la sorte:

```
def pgcd(a,b):
    r = 1
    while r != 0:
        mini = min(a,b)
        maxi = max(a,b)
         q = maxi // mini
        r = maxi % mini
        a = mini
        b = r
    return mini
```

La **fonction inverse_Modulaire**:

```
def Inv_Mod(x,y):
    for i in range(y):
        if (x*i)%y==1:
            return i
```

La **fonction Cryptage**:

```
def cryptage():
    Alphabet = entrée utilisateur
    char = ""
    code = ""
    message = entrée utilisateur

    if pgcd (int(key1),len(Alphabet)) = 1:
        for char in message:
            num = (Alphabet.find(char))
            num = (num *int(Key1)+int(Key2)) % len(Alphabet)
            code += Alphabet[num]
            if char == ' ':
                code += ' '

    else:
        code = "ERROR"

    print(code)
```

La **fonction Décryptage**:

Pour décrypter il faut d'abord trouver x' qui est l'inverse modulaire de a par la longueur de notre alphabet, puis il faut chercher y' = -x' * y [longueur de l'alphabet]

=> une autre méthode sans avoir besoin de y' : L' = x' * (L-y) [longueur de l'alphabet]

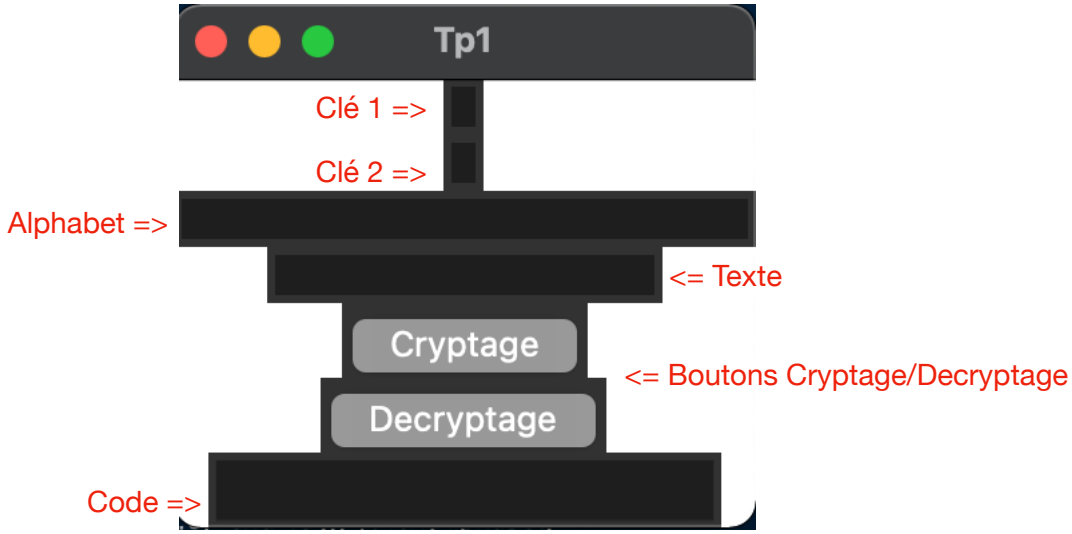
```
def Decryptage():
    Alphabet = entrée utilisateur
    char = ""
    code = ""
    message = entrée utilisateur

    if x is not None :
        for char in message:
            num = (Alphabet.find(char))
            x = Inv_Mod(int(Key1)),len(Alphabet))
            y = (-(int(x) * int(Key2) % len(Alphabet))
            print(x)
            print(y)
            num = (num*x+y) % len(Alphabet)
            code += Alphabet[num]
    else:
        code = "ERROR MOD_DEC/=1"
```

Observations personnelles :

On aurait pu ajouter une méthode de décryptage par brute force et enlever la condition du PGCD ce qui nous aurait permis de décrypter beaucoup plus de codes mais ce qui aurait rendu certain codes contenants des doublons presque impossible a déchiffrer sans une intervention humaine ou sans l'aide de dictionnaires.

Execution :

Pour plus de confort lors de l'utilisation de mon programme, au lieu de faire un programme qui fonctionne avec le terminal, j'ai préféré opter pour une interface graphique a l'aide de la bibliothèque TKinter meme si pour tester mon programme j'ai mis un retour des résultats sur le terminal.



Test :

Resources :

https://www.dcode.fr/chiffre-affine => *pour vérifier mes résultats*

shorturl.at/fyLM6 => *chaine YouTube francophone pour apprendre le python*

shorturl.at/sABFU => *chaine YouTube francophone pour apprendre a utiliser la bibliothèque TKinter avec python*

https://atom.io => *l'éditeur de texte que j'utilise pour coder*

https://www.python.org/downloads/release/python-3100/ => *la version de python que j'utilise (3.10)*

```python
#Mohamed Ali Damergi
##################################################
##########
from tkinter import *
from tkmacosx import Button # ====> a desactiver
sous windows #

window = Tk()
window.title("Cesar[affine]")
window.geometry("")
window.config(background = '#004466')

##################################################
##########

Key1_entry = Text(window,height = 1,width = 3)
Key1_entry.pack()
Key2_entry = Text(window,height = 1,width = 3)
Key2_entry.pack()

Ab_entry = Text(window,height = 1,width = 30)
Ab_entry.pack()

message_entry = Text(window,height = 1,width = 20)
message_entry.pack()

##################################################
##########

print("")
print("    ___ ___  ___  __ _ _ __  ")
print("   / __/ _ \/ __|/ _` | '__| ")
print("  | (_|  __/\__ \ (_| | |    ")
print("   \___\___||___/\__,_|_|         [affine]")
print("")

##################################################
##########

def Pgcd(a,b):
    r = 1
```

```
37      while r != 0:
38          mini = min(a,b)
39          maxi = max(a,b)
40          q = maxi // mini
41          r = maxi % mini
42          a = mini
43          b = r
44      return mini
45
46  ##########################################################
    ##########
47
48  def Inv_Mod(x,y):
49      for i in range(y):
50          if (x*i)%y==1:
51              return i
52
53  ##########################################################
    ##########
54
55  def Cryptage():
56      Alphabet = Ab_entry.get(1.0, "end-1c")
57
58      char = ""
59      code = ""
60      message = message_entry.get(1.0, "end-1c")
61
62      if Pgcd (int(Key1_entry.get(1.0, "end-1c")),len
    (Alphabet)) == 1:
63          for char in message:
64              num = (Alphabet.find(char))
65              num = (num *int(Key1_entry.get(1.0, "
    end-1c"))+ int(Key2_entry.get(1.0, "end-1c"))) %
    len(Alphabet)
66              code += Alphabet[num]
67              if char == ' ':
68                  code += ' '
69      else:
70          code = "ERROR 1"
71
72      code_aff.delete(0,END)
```

```
 73     code_aff.insert(0, code)
 74
 75
 76     print(" =>Fonction Cryptage")
 77     print("la premiere cle est: " + (Key1_entry.
    get(1.0, "end-1c")))
 78     print("la deuxieme cle est: " +  (Key2_entry.
    get(1.0, "end-1c")))
 79     print("l'Alphabet: " + Alphabet)
 80     print("la longueur de l'alphabet est: "+ str(
    len(Alphabet)))
 81     print("le code chiffre: "+ code)
 82     print("
    ------------------------------------------------")
 83 ##################################################
    ###########
 84
 85 def Decryptage():
 86     Alphabet = Ab_entry.get(1.0, "end-1c")
 87     char = ""
 88     code = ""
 89     message = message_entry.get(1.0, "end-1c")
 90     x = Inv_Mod(int(Key1_entry.get(1.0, "end-1c"
    )),len(Alphabet))
 91     if x is not None:
 92         for char in message:
 93             num = (Alphabet.find(char))
 94             y = (-(int(x) * int(Key2_entry.get(1.0
    , "end-1c"))) % len(Alphabet))
 95             num = (num*x+y) % len(Alphabet)
 96             code += Alphabet[num]
 97             if char == ' ':
 98                 code += ' '
 99     else:
100         code = "ERROR 2"
101     code_aff.delete(0,END)
102     code_aff.insert(0, code)
103
104
105     print(" =>Fonction Decryptage")
106     print("la premiere cle est: " + (Key1_entry.
```

```
106 get(1.0, "end-1c")))
107     print("la deuxieme cle est: " +  (Key2_entry.
    get(1.0, "end-1c")))
108     print("l'Alphabet: " + Alphabet)
109     print("la longueur de l'alphabet est: "+ str(
    len(Alphabet)))
110     print("le code dechiffre: "+ code)
111     print("
    ------------------------------------------")
112     return 0
113
114 ##################################################
    ###########
115
116 Generate_button = Button(window,text = "Cryptage",
    bg='white', fg='black', command =Cryptage)
117 Generate_button.pack()
118
119
120
121 Generate_button = Button(window,text = "Decryptage
    ",bg='white', fg='black', command =Decryptage)
122 Generate_button.pack()
123
124 code_aff = Entry(window)
125 code_aff.pack()
126
127
128 window.mainloop()
129
```