TD Info

Corrigé : Cryptographie - chiffrement de César

1. Chiffrement de César

2. Caractère et chaîne de caractères sont de type `string`.

3. La fonction `ord()` renvoie le code ASCII du caractère rentré en argument alors que la fonction `chr()` fait l'opération inverse. Avec une boucle `for` on obtient les lettres minuscules de l'alphabet dans l'ordre alphabétique : a b c d e f g h i j k l m n o p q r s t u v w x y z.

4.

Python
```
def ordre(ch):
    """ prend un caractère (string) en argument
    renvoie son code simplifié (en entier enntre 0 et 25)"""
    return ord(ch)-97
```

5.

Python
```
def lettre (nb):
    """ prend un code simlpifié en argument (en entier enntre 0 et 25)
    renvoie le caractère correspondant (string)"""
    return chr(nb+97)
```

6.

Python
```
def code_cesar(texte,key):
    """prend en argument un texte (string) sans accent ni caractères spéciaux
    renvoie le texte codé (string) avec la clé key
    Remarque : on laisse les espaces à leur place"""
    code = '' # texte codé
    for k in texte:
        if k == ' ': # on laisse les espaces à leur place
            code = code + ' '
        else:
            code = code + str( lettre ((ordre(k)+key)%26))
    return code
```

7.

Python
```
code_cesar('la physique c est fantastique ',5)
'qf umdxnvzj h jxy kfsyfxynvzj '
```

8.

Python
```
def decode_cesar(texte,key):
    """prend un texte codé (string) sans accent ni caractères spéciaux
    et une clé key (integer)
    renvoie le texte décodé (string)"""
    texte_decode =code_cesar(texte,-key)
    return texte_decode

decode_cesar('qf umdxnvzj h jxy kfsyfxynvzj ',5)
'la physique c est fantastique '
```

9.

Python
```
def formatage(chaine, espace=False):
    """prend un texte (string) en argument
    renvoie le texte formaté sans espace (par défaut)
    si on ajoute l'argument True, on garde les espaces"""
    chaine = chaine.lower()
    for car in chaine:
        if ord(car)>122 or ord(car)<97:
            if car in 'éèëê':
                chaine = chaine.replace(car,'e')
            elif car in 'àäâ':
                chaine = chaine.replace(car,'a')
            elif car in 'öô':
                chaine = chaine.replace(car,'o')
            elif car in 'ùûü':
                chaine = chaine.replace(car,'u')
            elif car in 'ïî':
                chaine = chaine.replace(car,'i')
            else :                          #Ponctuation ou autres lettres ou symboles particuliers
                chaine = chaine.replace(car,' ')
        if not espace and car ==' ':
            chaine = chaine.replace(car,'')
    return chaine
```

10.

```python
def frequence(texte):
    """prend un texte (string) en argument
    renvoie une liste de float avec la fréquence de chaque lettre """
    texte = formatage(texte) # on ne garde que les lettres
    lgth = len(texte) # nombre de caractères du texte
    ls = [0 for k in range(26)] # liste des fréquences pour les 26 lettres
    for car in texte:
        ls[ordre(car)] +=1
    for k in range(len(ls)): # on divise par le nombre de lettres
        ls[k] = float(ls[k])/lgth
    return ls
```

11.

```python
def distance(frequence_ref,frequence_text):
    """prend en argument deux listes de fréquences (de référence et celle du texte à décoder)
    renvoie la liste des distances """
    ls = [0 for i in range(26)] #liste des distances
    for i in range(26):
        for j in range(26):
            ls[i] = ls[i] + abs(frequence_ref[j] - frequence_text[(i+j)%26])
    return ls
```

12.

```python
def craquer_cesar(distance):
    """prend la liste donnée par distance
    rend la clé (integer)"""
    mini = min(distance)
    return distance.index(mini)
```

13.

```python
def decodage(texte,frequence_ref):
    """prend un texte codé (string) en argument et une liste de fréquences d'une langue donnée
    rend le texte décodé (string) sans connaitre la clé"""
    frequence_text = frequence(texte)
    distance_texte = distance(frequence_ref,frequence_text)
    key = craquer_cesar(distance_texte)
    return decode_cesar(texte,key)

file = open('texte_secret.txt','r')
contenu = file.read()
file.close()
print(decodage(contenu))
```

2. Chiffrement de Vigénère

14.

Python
```
def codage_vigenere(chaine, cle):
    """prend en arguments une chaîne de caractères et la clé (chaine de caractères)
    renvoie le message codé (chaîne de caractères)"""
    message_code =""
    L = len(chaine)
    C = len(cle)
    for i in range(L) :
        if chaine[i] !=" ":
            message_code =message_code +lettre((ordre(chaine[i]) + ordre(cle[i%C]))%26)
    return message_code
```

Python
```
def decodage_vigenere(chaine, cle):
    """prend en arguments une chaîne de caractères et la clé (chaine de caractères)
    renvoie le message décodé (chaîne de caractères)"""
    message_decode =""
    L = len(chaine)
    C = len(cle)
    for i in range(L) :
        if chaine[i] ==" ":
            message_decode +=" "
        else:
            message_decode =message_decode +lettre((ordre(chaine[i]) - ordre(cle[i%C]))%26)
    return message_decode
```

Python
```
def apparition(chaine):
    """prend en argument une chaine de caractères
    renvoie une liste des fréquences d'apparition de chaque lettre de l'alphabet.
    Cette fonction n'est pas normalisée.
    Liste d'entiers """
    N = len(chaine)
    freq = []
    for i in range(26):
        freq = freq + [round(chaine.count(chr(97 + i)),4)]
    return freq
```

Python
```
def indice_coincidence(chaine) :
    """prend en argument une chaîne de caractères (message codé)
    applique l'algo de Friedman
    renvoie un float """
    app = apparition(chaine)
    s = sum(n*(n-1) for n in app)
    somme =sum(app)
    return s / (somme*(somme-1))
```

Python
```
def craquer_longueur(chaine_codee):
    """prend en argument une chaine codée par Vigenere
    renvoie la longueur de la clé (int)"""
    L_max =15 # longueur maximale de la clé
    L=0     # Initialisation : longueur clé
    indice =0 # Initialisation indice de coïncidence
    seuil = 0.07

    for k in range(1,L_max):
        moyenne=0
        for j in range(k):
            moyenne+=indice_coincidence(chaine_codee[j::k])
        moyenne =moyenne/k
        if indice <moyenne:     # on stocke la valeur max,
            indice = moyenne   #si jamais aucune valeur ne dépasse le seuil
            L=k
        if moyenne > seuil :     # dès qu'on dépasse le seuil, on valide
            return k             # cela évite les pb de multiple de k
    return L
```

Python
```
def craquer_vigenere(chaine_codee,chaine_ref):
    """prend en argument une chaine de caractères codée
    renvoie la clé (chaine de caractères)"""
    L=craquer_longueur(chaine_codee)
    freq_ref = frequence(chaine_ref)
    cle=''
    for k in range(L):
        cle+=chr(craquer_cesar(freq_ref, frequence(chaine_codee[k::L]))+97)
    return(cle)
```