Martin Creuze

Camille Bauvais

Exercice 1 :

```python
def check_palindrome(mot):
    #condition d'arret
    if len(mot)==0 or len(mot)==1:
        return True

    #appel récursif
    elif mot[0]==mot[len(mot)-1]:
        return check_palindrome(mot[1:(len(mot)-1)])

    else:
        return False


if __name__ == '__main__':
    mot=str(input("Entrer le mot à tester  :  "))
    print(check_palindrome(mot))
```

Essais :

Essai sur un palindrome :

```
Entrer le mot à tester  :  kayak
True

Process finished with exit code 0
```

Essai sur un mit qui n'est pas un palindrome :

```
Entrer le mot à tester  :  bonjour
False
```

Essai sur un mot vide :

```
Entrer le mot à tester  :
True
```

Exercice 2 :

```
def crypter(ch):
    lettres='abcdefghijklmnopqrstuvwxyz'
    chaine_cryptee=''
    for j in range (len(ch)):
        if ch[j] in lettres:
            if ch[j]=='z':
                chaine_cryptee+='a'
            else:
                i=0
                #parcours de lettres pour trouver l'indice de la lettre
                while ch[j]!=lettres[i]:
                    i+=1

                chaine_cryptee+=lettres[i+1] #ajout de la lettre située après lettres[i] dans chaine_cryptee
    return (chaine_cryptee)
```

```
import string

def crypter(ch):
    chaine_cryptee=''
    for lettre in ch:
        if lettre in string.ascii_uppercase: #vérifie si c'est une lettre majuscule
            #traitement du cas particulier Z
            if lettre=='Z':
                chaine_cryptee+='A'

            #traitement du cas général
            else:
                chaine_cryptee+=chr(ord(lettre)+1) #ajout de la lettre située après lettre dans
chaine_cryptee
    return (chaine_cryptee)


if __name__ == '__main__':
    mot = str(input("Entrer le mot à crypter  :  "))
    print(crypter(mot))
```

Essai :

```
Entrer le mot à modifier  :  zab
abc

Process finished with exit code 0
```

```
Entrer le mot à crypter  :  ESSAI
FTTBJ

Process finished with exit code 0
```

Exercice 3

```
#fonction qui retourne le nombre de caractères minuscules
def nb_min(password):
    nb=0          #compteur initialisé à 0
    for i in range (len(password)):
        if password[i] in [chr(k) for k in range(ord('a'), ord('z')+1)]: #test : lettre est elle minuscule?
            nb+=1
    return nb

#fonction qui retourne le nombre de caractères majuscules
def nb_maj(password):
    nb=0          #compteur initialisé à 0
    for i in range (len(password)):
        if password[i] in [chr(k) for k in range(ord('A'), ord('Z')+1)]: #test : lettre est elle majuscule?
            nb+=1
    return nb

#focntion qui retourne le nombre de caractères non alphabétiques
def nb_alpha(password):
    nb=0          #compteur initialisé à 0
    for i in range (len(password)):
        if (password[i] not in [chr(k) for k in range(ord('A'),        ord('Z')+1)]) and (password[i] not in
[chr(k) for k in range(ord('a'), ord('z')+1)]):
 #test : lettre est elle ni minuscule ni majuscule?
            nb+=1
    return nb

#fonction qu retourne la longueur de la plus longue séquence de lettres minuscules
def long_min(password):
    #initialisation des variables à 0
    max=0
    taille=0
    #parcours de la chaine lettre par lettre
    for i in range(len(password)):
        #si c'est une lettre minuscule
        if password[i] in [chr(k) for k in range(ord('a'), ord('z')+1)]: #test : lettre est elle minuscule?
            taille+=1
            #on remplace le maximum par cette nouvelle séquence si elle est plus longue
            if taille > max:
                max = taille
        #sinon on remet le compteur à 0
        else:
            taille=0
    return max

#fonction qui retourne la longueur de la plus longue séquence de lettres majuscules
def long_maj(password):
    # initialisation des variables à 0
    max=0
    taille=0
    # parcours de la chaine lettre par lettre
    for i in range(len(password)):
        # si c'est une lettre majuscule
        if password[i] in [chr(k) for k in range(ord('A'), ord('Z')+1)]: #test : lettre est elle majuscule?
            taille+=1
            # on remplace le maximum par cette nouvelle sequence si elle est plus longue
```

```
            if taille > max:
                max = taille
        # sinon on remet le compteur à 0
        else:
            taille=0
    return max

#fonction qui affiche la force du mot de passe
def score(password) ->(str, int):
    #calcul des bonus
    bonus= len(password)*4 + (len(password) - nb_maj(password)) * 2 +
(len(password)-nb_min(password))*3 + nb_alpha(password)*5
    #calcul des malus
    malus= long_min(password)*2 + long_maj(password)*3
    #différence :
    force=bonus-malus
    #Conditions
    if force<20:
        valeur= "Tres faible"
    elif force <40:
        valeur= "Faible"
    elif force <80:
        valeur= "Fort"
    else :
        valeur= "Tres fort"
    return(valeur, force)



if __name__=='__main__':
    mdp=str(input("Entrez un mot de passe : "))
    valeur, score=score(mdp)
    print(valeur)
    print(score)
```

Essais :

Essai pour un mot de passe Très Fort

```
Entrez un mot de passe : HopTyij45_uypoRT
Tres fort
110
```

Essai pour un mot de passe Fort

```
Entrez un mot de passe : fpp47
Fort
40
```

Essai pour un mot de passe Faible

```
Entrez un mot de passe : hU7
Faible
22
```

Essai pour un mot de passe Très Faible

```
Entrez un mot de passe : yuo
Tres faible
12
```