Séance 6 : Exceptions et fichiers textes

L1 – Université Côte d'Azur

Exercice 1 — Watson (⋆)

On considère le programme suivant

```
f_out = open('watson.txt', 'w', encoding='utf-8')
f_out.write('Élémentaire, mon cher!')
f_out.close()
```

1. Que fait ce programme ?
2. Que se passe-t-il si le fichier `watson.txt` existe déjà avant d'exécuter le programme ?
3. Que se passe-t-il si on oublie la ligne `f_out.close()` ?
4. Que se passe-t-il si on remplace le paramètre optionnel par `encoding='latin-1'` ? Et si on oublie ce paramètre ?
5. Comment faire pour ajouter le texte ci-dessus à la fin du fichier si le fichier contient déjà du texte ?
6. Que se passe-t-il si le fichier `watson.txt` est protégé en écriture ?

Exercice 2 — Le temps qui passe (⋆)

Définissez une fonction `jour(s)` qui crée un fichier `s` dans lequel elle écrit ligne après ligne les différentes heures de la journée, à intervalle de 5 minutes. Le fichier ressemblera à :

```
00:00
00:05
00:10
...
23:50
23:55
```

Indication : si `x=42`, la chaîne `f'{x:*>10}'` vaut `'********42'`, c'est-à-dire une chaîne de 10 caractères représentant le nombre `x` aligné à droite en utilisant le caractère `*` comme caractère de remplissage.

Exercice 3 — Afficher un fichier (⋆)

1. Écrivez une fonction `affiche_fichier(s)` qui lit le fichier nommé `s` et l'affiche. *Rappel* : On peut lire un fichier d'au moins deux manières distinctes :
   - d'un seul coup, en chargeant tout le contenu sous la forme d'une longue chaîne avec la méthode `read`.

     ```
     def afficher_fichier(s):
         f_in = open(s, 'r', encoding = 'utf-8')
         texte = f_in.read()
         f_in.close()
         print(texte, end ='')
     ```

   - avec une boucle `for` en itérant ligne par ligne sur le descripteur du fichier.

```python
def afficher_fichier4(s):
    f_in = open(s, 'r', encoding = 'utf-8')
    for ligne in f_in:
        print(ligne, end = '') # ligne contient le retour à la ligne
    print()
    f_in.close() # Il faut fermer le fichier après la boucle
```

2. Que se passe-t-il si l'on demande d'afficher un fichier qui n'existe pas ?
   On obtient un message d'erreur : `FileNotFoundError`
3. Modifiez votre fonction pour faire afficher le message « `fichier` *`nom_du_fichier`* `non trouvé` ».

```python
def afficher_fichier(s):
    try:
        f_in = open(s, 'r', encoding = 'utf-8')
        for ligne in f_in:
            print(ligne, end = '') # la chaîne ligne contient déjà le '\n'
        f_in.close()
    except FileNotFoundError:
        print('fichier', s, 'non trouvé.')
```

Exercice 4 — Nombre de lignes (⋆)

Définissez la fonction `nombre_lignes(s)` qui renvoie le nombre de lignes du fichier `s`, ou -1 si le fichier `s` n'existe pas ou ne peut pas être lu.

```python
def nombre_lignes(s):
    try:
        fichier = open(s, 'r', encoding = 'utf-8')
        n = 0 # n=nombre de lignes lues
        for ligne in fichier:
            n = n+1
        fichier.close()
        return n
    except FileNotFoundError:
        return -1
    except PermissionError:
        return -1
```

Exercice 5 — Mise en majuscule (⋆⋆)

On suppose que l'on a créé un fichier texte contenant plusieurs paragraphes en français. Programmez une procédure `majuscules(f)`, sans résultat, fonctionnant de la manière suivante. Elle prend en entrée une chaîne de caractères `f` représentant un nom de fichier, se terminant par `.txt` comme `'texte.txt'`.

1. La fonction va commencer à fabriquer un nouveau nom de fichier `f1` identique au précédent, mais en ajoutant `-maj` au nom du fichier (donc ici `f1` sera : `'texte-maj.txt'`). Définissez la variable `f1` en fonction de la variable `f`. On rappelle que l'on peut facilement obtenir une sous-chaîne avec la notation `chaine[début:fin:pas]`.
2. Elle va ensuite ouvrir `f` en lecture et `f1` en écriture, pour envoyer tout le texte de `f` dans `f1`, en le tranformant au passage en `MAJUSCULES`. Elle ferme ensuite les deux fichiers. On pourra utiliser la méthode `chaîne.upper()`

```python
def majuscules(f) :
    f1 = f[0:len(f)-4] + '-maj' + '.txt'
    ancien  = open(f,  'r', encoding = 'utf-8')
    nouveau = open(f1, 'w', encoding = 'utf-8')
    for ligne in ancien :
        nouveau.write(ligne.upper())
    ancien.close()
    nouveau.close()
```

Exercice 6 — Statistiques (⋆⋆-⋆ ⋆ ⋆)

Définissez une fonction `statistiques(L)` qui prend en argument une liste `L` de notes entre 0 et 20 et qui affiche des statistiques sur ces notes au format suivant. L'alignement sera géré avec l'indication de l'exercice 2.

```
Note la plus basse :     3/20
Note la plus haute :    19/20
Moyenne :             9.83/20
Taux d'admission :      40.2%
```

On commence par définir les fonctions qui font les calculs (minimum, maximum, etc.)

```python
# Existe déjà c'est la fonction min
def minimum(L):
    assert len(L)!=0 # On lève une exception si la liste est vide
    m=L[0] # L[0] est forcèment définie si le assert n'a pas levé d'exceptions
    for e in L:
        if e<m:
            m=e
    return m

# Existe déjà c'est la fonction max
def maximum(L):
    assert len(L)!=0 # On lève une exception si la liste est vide
    m=L[0] # L[0] est forcèment définie si le assert n'a pas levé d'exceptions
    for e in L:
        if e>m:
            m=e
    return m

# En une ligne : return  sum(L)/len(L) (seulement si L!=[])
def moyenne(L):
    assert len(L)!=0 # On lève une exception si la liste est vide
    s=0 # on calcul la somme des termes dans s
    for e in L:
        s=s+e
    return s/len(L) # len(L) est forcément différent de 0

# En une ligne : return sum([1 for e in L if e>10] * 100/len(L))
def pourcentage_admis(L):
    assert len(L)!=0  # On lève une exception si la liste est vide
    n=0 # on calcul le nombre d'admis dans n
    for e in L:
        if e>=10:
            n=n+1
    return n/len(L)*100 # len(L) est forcément différent de 0
```

Puis on s'occupe de l'affichage.

```python
def affichage(L):
    # L = liste de triplets : texte, nombre, fin (% ou /20)
    # On calcule la plus longue description pour pouvoir aligner les notes.
    liste_description=[]
    for triplet in L:
        liste_description.append(triplet[0])
    longueur_max = maximum(liste_decription)
    # On aurait pu simplement écrire : longueur_max =  max([len(e[0]) for e in L])

    for triplet in L:
        (description,nombre,unité)=triplet
        print('{: <{lg}}  '.format(description, lg = longueur_max),end='')
        if type(nombre) == int:
            print(' {: >5}{}'.format(nombre,unité))
        else:
            print(' {:5.2f}{}'.format(nombre,unité))
            # La partie entière du nombre fait 2 caractères maximum (entre 0 et 20)
            # On choisit deux chiffres après la virgule
            # Si on ajoute le . de la virgule, cela fait 5 caractères
            # on prend 5 caractères pour gérer le taux 100%,

def statistiques(L):
    # calculs statistiques
    m = minimum(L)
    M = maximum(L)
    moy = moyenne(L)
    taux_admission = pourcentage_admis(L)
    affichage([('Note la plus basse :',m, '/20'),
               ('Note la plus haute :',M, '/20'),
               ('Moyenne :', moy, '/20'),
               ("Taux d'admission :", taux_admission, '%')])
```