Exercice 1

Écrire une fonction qui prend en entrée un nom de fichier texte, ouvre le fichier, et **affiche** la longueur moyenne des mots du texte. On prendra comme définition d'un mot celle qui correspond à l'emploi de la fonction `split()`.
On demande le code complet de la fonction, abondamment commenté (en français), soit sous la forme de commentaires dans le corps de la fonction, soit sous la forme d'une description de quelques lignes du fonctionnement du programme. Attention : les commentaires du genre "j reçoit la valeur 3" au niveau de l'instruction "`j=3`" sont à proscrire et seront pénalisés.

Corrigé

```
def long_moy(ficname):
    with open(ficname, "r") as file:
        somme = 0
        nb = 0
        for mot in file.read().split():
            somme += len(mot)
            nb += 1
    print("La longueur moyenne des %d mots est %f." % (nb, somme/nb))
```

Exercice 2

Écrire une fonction qui prend en entrée un nom de fichier texte, ouvre le fichier, et **affiche** le nombre de mots qui commencent et finissent par la même lettre. On prendra comme définition d'un mot celle qui correspond à l'emploi de la fonction `split()`.

Corrigé

```
def nb_mots_ml(ficname):
    with open(ficname, "r") as file:
        compteur = 0
        for mot in file.read().split():
            if mot[0] == mot[-1]:
                compteur += 1
    print("Il y a %d mots qui commencent et finissent par la même lettre." % compteur)
```

Exercice 3

Écrire une fonction qui prend en entrée un nom de fichier texte, ouvre le fichier, et **affiche** le mot le plus long du texte. On prendra comme définition d'un mot celle qui correspond à l'emploi de la fonction `split()`.

Corrigé

```
def mot_long(ficname):
    with open(ficname, "r") as file:
        champion = ""
        for mot in file.read().split():
            if len(mot) > len(champion):
                champion = mot
    print("Le mot le plus long est %s." % champion)
```

Exercice 4

Écrire une fonction qui prend en entrée un nom de fichier texte, ouvre le fichier, et **affiche** le nombre de mots dont la longueur est comprise entre 7 et 9 (bornes incluses). On prendra comme définition d'un mot celle qui correspond à l'emploi de la fonction `split()`.

Corrigé

```
def nb_motsl(ficname):
    with open(ficname, "r") as file:
        compteur = 0
        for mot in file.read().split():
            if 7 <= len(mot) <= 9:
                compteur += 1
    print("Il y a %d mots de longueur entre 7 et 9." % compteur)
```

Exercice 5

Écrire une fonction qui prend en entrée une liste d'entiers, et renvoie le nombre de paires de valeurs voisines qui ne sont pas dans le bon ordre (le bon ordre étant un ordre croissant). Il s'agit donc du nombre d'échanges qui seraient faits à la première passe dans un tri à bulle.
Exemple : avec l'entrée `[1,9,8,3,5,4]` la fonction répondra 3.

Corrigé

```
def compte_paires(l):
    c = 0
    for i in range(len(l)-1):
        if l[i] > l[i+1]:
            c += 1
    return c
```

Exercice 6

Écrire une fonction qui prend en entrée une liste d'entiers, et renvoie le nombre de paires de valeurs voisines qui sont dans le bon ordre (le bon ordre étant un ordre croissant).
Exemple : avec l'entrée `[1,9,8,3,5,4]` la fonction répondra 2.

Corrigé

```
def compte_paires_triees(l):
    c = 0
    for i in range(len(l)-1):
        if l[i] <= l[i+1]:
            c += 1
    return c
```

Exercice 7

Écrire une fonction qui prend en entrée une liste d'entiers, calcule la moyenne de la liste pour en faire un pivot, et renvoie le nombre d'échanges que coûte une réorganisation de la liste par rapport à ce pivot, pour faire en sorte que tous les éléments inférieurs au pivot précèdent tous les éléments supérieurs au pivot (sans que ces éléments soient nécessairement triés entre eux). Il s'agit donc du nombre d'échanges qui seraient faits à la première passe d'un quicksort.
Exemple : avec l'entrée `[1,9,8,3,5,4]` la fonction répondra `2`.

Corrigé

```
def compte_echange_qsort(l):
    pivot = int(sum(l)/len(l))
    g, d = 0, len(l)-1
    c = 0
    while g < d:
        if l[g] > pivot:
            l[g], l[d] = l[d], l[g]
            c += 1
            d -= 1
        else:
            g +=1
    return c, pivot
```