Correction - DS 4

Devoir surveillé du Mardi 12 Novembre

Exercice 1

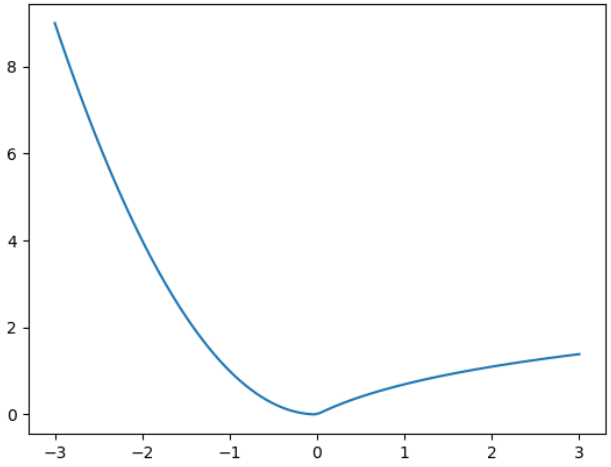
1. Pour définir f :

```
def f(x):
    if x<0:
        return(x**2)
    else:
        return(np.log(x+1))
```

2. Pour tracer f :

```
x = np.linspace(-3,3,100)
y = np.zeros(100)
for k in range(100):
    y[k] = f(x[k])
plt.plot(x,y)
plt.show()
```

On obtient le tracé suivant :



Exercice 2

1. Pour la fonction `indicemin` :

```
def indicemin(L):
    imin = 0
    for k in range(len(L)):
        if L[imin] > L[k] :
            imin = k
    return imin
```

2. `del L[i]` permet de supprimer de la liste `L` l'élément d'indice `i`.

3. Avec les deux questions précédentes, on propose une fonction qui :

- cherche l'indice du minimum de la liste ;
- supprime l'élément correspondant de la liste ;
- cherche l'indice du minimum dans la liste des éléments restants ;
- retourne la valeur associée.

On obtient ainsi la valeur du deuxième minimum de la liste de départ.

```
def min2(L):
    imin = indicemin(L)
    del L[imin]
    imin2 = indicemin(L)
    return L[imin2]
```

4. Voici la fonction complétée :

```
def tri(L):
    M = []
    while len(L) > 0 :
        i = indicemin(L)
        M.append(L[i])
        del L[i]
    return M
```

On crée la liste vide `M` qui contiendra à la fin du programme la liste triée dans l'ordre croissant des éléments de `L`. Puis, tant que la liste `L` n'est pas vide, c'est-à-dire tant que sa longueur `len(L)` est strictement positive,

- on cherche l'indice `i` du minimum de `L` avec la commande `i = indicemin(L)`,
- on ajoute le minimum à la fin de `M` avec la commande `M.append(L[i])`,
- on le supprime de `L` avec la commande `del L[i]`.

On obtient ainsi le résultat demandé.

Exercice 3

1. Voici la fonction `facto` demandée :

```
def facto(n):
    p = 1
    for k in range(1,n+1):
        p = p*k
    return(p)
```

2. Voici la fonction `coefbin` demandée :

```
def coefbin(k,n):
    return(facto(n)/(facto(k)*facto(n-k))
```

3. Voici la fonction `somme` demandée :

```
def somme(n):
    S = 0
```

```
3       for k in range(n+1):
4           S = S + coefbin(k,n)
5       return(S)
```

4. La variable `M` est une vecteur de longueur 10 dont le coefficient k contient $\sum_{i=0}^{k} \binom{k}{i}$.

   On remarque que les résultats obtenus satisfont la formule $\sum_{k=0}^{n} \binom{n}{k} = 2^n$. On peut démontrer ce résultat avec la formule du binôme de Newton :

$$\sum_{k=0}^{n} \binom{n}{k} = \sum_{k=0}^{n} \binom{n}{k} 1^k 1^{n-k} = (1+1)^n = 2^n.$$

Exercice 4

1. On entre les commandes :

```
>>> P = rd.normal(72, np.sqrt(6), 1000)
>>> T = rd.normal(1.78, np.sqrt(0.04), 1000)
```

2. Il faut entrer la commande `I = P/T**2`.

3. Voici les instructions demandées :

   - Pour la médiane : `np.median(I)`
   - Pour la moyenne : `np.mean(I)`
   - Pour la variance : `np.var(I)`
   - Pour l'écart type : `np.std(I)`

4. Pour déterminer les personnes en sur-poids, on utilise la commande `np.sum(I>=25)/1000`.

5. Comme il y a un très grand nombre de modalités (à priori, `I` contient 1000 valeurs différentes), on trie la série statistique par classes.

6. On commence par déterminer la plus petite et la plus grand modalité de la série statistique. On sépare ensuite l'intervalle $[m, M]$ en 4 de manière uniforme et on définit ainsi 4 classes (contenues dans la variable `c`).

   Le graphique obtenu est un histogramme qui représente la fréquence de chaque classe. On remarque que la classe modale est la classe 2 (celle qui correspond au plus grand bâton).

Exercice 5

1. (a) `df.shape` permet d'obtenir le nombre de lignes et de colonnes de la table `df`.
   (b) `df['Age'].std()` permet d'obtenir l'écart type de la colonne `Age` de la table `df`.
   (c) `df[df['Ville']=='Paris']` permet d'obtenir les lignes de la table `df` où la ville est `Paris`.

2. (a) Pour obtenir la médiane de la colonne `'Dispo'` : `df['Dispo'].median()`
   (b) Pour sélectionner les personnes qui ont au moins un enfant : `df[df['Enfants']>0]`
   (c) Pour sélectionner les personnes qui ont le permis et pas d'enfant :
   `df[(df['Permis']=='oui') & (df['Enfants']==0)]`
