Tranchage/Slicing

Essayez et comprenez les codes suivants :

```
L=[5,8,20,8,9,-2,-3]
M=L[2:6]
print(M)
print(L[0:len(L)])
print(L[0:len(L)-1])
print(L[0:len(L)-2])
s="Je m'appelle encore Dupont"
print(s[2:6])
print(s[0:len(s)])
print(s[3:len(s)-1])
print(s[0:len(s)-2])
```

Conversion

Les chaînes de caractères ne sont pas des nombres et inversement, mais on peut transformer un nombre en une chaîne de caractère. Et si une chaîne de caractère représente un nombre, on peut la convertir en nombre

```
a=3#Nombre
b=str(a)#Conversion en chaîne de caractère
print(a,b)
print(b==a)#b==a est un booléen qui vaudra True ssi s=a
s="3"#Chaîne de caractère
n=int(s)#Conversion en un nombre entier, int pour integer
x=float(s)#Conversion en un nombre à virgule,
#float pour nombre flottant (sera vu plus tard)
print(s,n,x)
```

La commande `type` vous donne le type de variable que vous avez, essayez :

```
print(type(a))
print(type(b))
print(type(s))
print(type(n))
print(type(x))
```

Dans ce TP, nous allons voir l'utilisation de bibliothèques Python pour afficher des graphiques ainsi que l'utilisation de fichiers que ce soit en lecture ou en écriture.

Attention les commandes ne sont pas à apprendre

Conformément au programme, les commandes utilisant des bibliothèques Python et l'utilisation des fichiers ne sont pas à connaître. La documentation contenant les commandes concernant les bibliothèques ainsi que l'utilisation de fichiers devra être fournie aux concours.

Affichage de graphe

Pour afficher un graphe, on peut utiliser une bibliothèque `pyplot`, il faut appeler cette bibliothèque grâce à la commande suivante :

```
import matplotlib.pyplot as plt# plt sera donc le raccourci
# pour appeler les commandes de cette bibliothèque
#cette commande est à mettre en début de fichier
```

Si on veut afficher une liste de points, il suffit de donner la liste des abscisses et celles des ordonnées :

```
A=[1,4,4,1,1]
O=[2,2,3,3,2]
plt.plot(A,O)
plt.show()#Le dessin est-il cohérent avec les listes A et O?
```

Par défaut, Python relie les points entre eux (il trace des segments), mais cela peut changer :

```
plt.plot(A,O,'ro')# r pour red et o pour des o, essayez
plt.show()#k, b ou g à la place de r et x à la place de o
```

Exercice 1.

1. Définir une fonction Python `f` qui à `x` renvoie x^2.
2. Créer deux listes, la première `A` telle que `A[i]=i/100` pour tout `i` entre 0 et 100 et la seconde `O` telle que `O[i]=f(A[i])`.

3. Afficher tous les points d'abscisse `A[i]` et d'ordonnée `O[i]` pour tout `i`. Par défaut, Python risque d'afficher ces points sur la figure précédemment créée. Pour éviter cela, utilisez la commande `plt.figure()` avant la commande `plt.plot(...)`.
4. Rajouter des légendes sur les axes ainsi qu'un titre grâce aux commandes `plt.xlabel("légende axe des abscisses")`, `plt.ylabel` et `plt.title`

Écriture dans un fichier

Si on veut écrire dans un fichier, placé dans le même répertoire que votre fichier python actuel, on utilise les commandes suivantes :

```
fichier=open("nom_de_mon_fichier.txt", "w")
#"w" pour write: mode écriture
fichier.write("Vive le vent\n")#\n sert à dire
#qu'on veut aller à la ligne, \n est un caractère spécial.
fichier.write("Vive le vent\n")
fichier.write("Vive le vent d'hiver\n")
fichier.close()
```

Ouvrir le fichier avec bloc-note pour vérifier que c'est bien conforme à ce qu'on a fait. Attention, Pyzo a une spécificité un peu embêtante, par défaut, il ne va pas forcément mettre ce fichier là où vous voulez, car il ne se place pas forcément dans le dossier où est placé votre fichier Python actuel[1]. Pour y remédier, exécuter votre code avec `Ctrl+Shift+E` (ou `run file as a script`). Il n'y a pas ce problème sous Spyder.

Lecture d'un fichier

Pour lire le contenu d'un fichier, utiliser les commandes suivantes :

```
fichier=open("nom_de_mon_fichier.txt" ,"r" )#'r' pour read
M=fichier.read()
print(M)
fichier.close()
```

```
fichier=open("nom_de_mon_fichier.txt" ,"r" )
N=fichier.readlines()#permet de créer une liste,
print(N)# où N[i] contient la i-ème ligne du fichier
fichier.close()
```

Exercice 2 (⋆).
1. Téléchargez les deux fichiers textes[2] `TP6fichier1.txt` et `TP6fichier2.txt` qui sont sur cahier de prépa dans le dossier TP et mettez les dans **le même dossier** que votre fichier Python actuel.
2. En utilisant les commandes de la partie «lecture d'un fichier», ouvrez ces fichiers en mode lecture.
3. Extrayez-les avec `readlines`. Vous obtenez alors les données sous forme de listes, appelez `A` la première liste et `O` la deuxième.
4. Pour chacun des éléments de cette liste, grâce au slicing, enlevez les `\n` qu'il y a et convertissez la chaîne de caractères en nombres. *Petite subtilité : `\n` est considéré comme **un** caractère et non deux : tester `len("\n")`.*
5. Ensuite, dans une boucle `for`, afficher le segment des points `C` et `D` où `C` est le point d'abscisse `A[2i]` et d'ordonnée `O[2i]` et `D` est le point d'abscisse `A[2i+1]` et d'ordonnée `O[2i+1]` pour tous les `i` possibles.
6. Adaptez la couleur.

Exercice 3 (⋆⋆).
1. Téléchargez le fichier `TP6ListeMotsFrancais.txt` qui se trouve sur cahier de prépa.
2. De même qu'à l'exercice précédent, retirer le `\n` à chaque élément de la liste des mots en français.
3. En s'appuyant sur les fonctions déjà écrites lors du TP5, trouver la liste des palindromes de la langue française.
4. Calculer le nombre moyen de lettres des mots du fichier, le nombre médian ainsi que l'écart-type. Représenter sur un graphique le nombre de mots à n lettres en fonction de n.

Si on a fini ?

Finir les TP précédents.

1. Rappelons que vous devez écrire le contenu dans un fichier dont le nom est `TP6` placé dans un dossier `TPInfo`.

2. Ces deux fichiers ont été crées à partir du travail de Lilian Besson et Arnaud Basson, merci à eux !