UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

Info. TP2: Vers les tableaux à n dimensions

- https://learn.heeere.com/python
- https://learn.heeere.com/2020-infospichi-0b73/
- https://learn.heeere.com/2020-info-0s42 (cours de L1)

Rendu : sous forme d'archive

À la fin, vos réponses seront rendues dans une archive zip : cette archive contiendra un fichier `compte-rendu.txt` (avec votre **nom**, **groupe**, et les réponses aux questions précédées d'une apostrophe) et vos **fichiers pythons**.

Pour faire une **archive zip** de votre TP pour le rendu, exécutez, dans le terminal :

```
cd ~/info-spichi/
zip -r tp2.zip tp2/
```

Important : mise en place (5 minutes)

- travaillez dans un dossier dédié au TP, lui même dans `~/info-spichi`,
- créez un fichier `compte-rendu.txt` pour écrire **la date, vos noms** et les réponses aux questions précédées d'une apostrophe,
- pour chaque exercice, créez un dossier et placez y les fichiers python que vous écrivez.

|↦ **Aide** |↦ Il faut lancer un terminal et y taper les commandes (les `# ...` sont des commentaires) :

```
cd info-spichi              # Se déplacer dans le dossier du cours
mkdir tp2                   # Créer le dossier du TP
cd tp2                      # Se déplacer dedans
touch compte-rendu.txt      # Créer un fichier vide
emacs compte-rendu.txt &    # L'ouvrir avec l'éditeur emacs
```

Téléchargement de la bibliothèque qtido

Si besoin, la bibliothèque qtido peut être téléchargée à l'aide de la commande « wget » avec la ligne de commande :

```
wget https://learn.heeere.com/2020-infospichi-0b73/outils-python/qtido.py
```

Exercice 1 – Liste de listes

Rappel : pour cet exercice, travaillez dans un dossier `ex1` lui même dans le dossier `tp2`.

|↦ **Aide** |↦ En Python, une liste dispose d'une fonction `append()` qui permet d'ajouter un élément à la fin de cette liste, comme illustré dans ce bout de code :

```
1 fruits = ["banane", "kiwi", "pomme"]
2 print(len(fruits))       # 3
3 fruits.append("poire")
4 print(len(fruits))       # 4
5 print(fruits)            # ['banane', 'kiwi', 'pomme', 'poire']
```

Voici un exemple d'utilisation de `append` dans une fonction qui reçoit un paramètre n et qui créée une liste avec n éléments, qui sont tous égaux à la valeur 0 :

```python
def liste_zeros(n):
    resultat = []           # liste vide
    for i in range(n):      # répéter n fois
        resultat.append(0)  # ajouter un 0
    return resultat         # retourner/renvoyer la liste calculée
```

Q1) Écrire un programme `liste.py` qui demande à l'utilisateur de rentrer un nombre entier n, qui utilise la fonction `liste_zeros` pour créer une liste avec n 0 et qui affiche cette liste (directement avec un `print`).

Rappel : vos réponses aux questions précédées d'une apostrophe doivent être écrite à l'aide d'emacs dans le fichier `compte-rendu.txt`. Vérifiez bien qu'il est au bon endroit et pensez bien à le sauvegarder.

'Q2) À quel endroit avez-vous fait la conversion de l'entrée utilisateur à l'aide de `int` ? Où pourrait aussi être faite cette conversions ? Quels sont les avantages des différentes solutions ?

Q3) Écrire un programme `listeliste.py` qui demande à l'utilisateur deux nombres entiers m et n (avec 1 ou 2 `input`, au choix) et qui crée une liste de m éléments, chacun étant une nouvelle liste de n 0 (que l'on peut obtenir à l'aide de la fonction `liste_zeros`). Votre programme doit ensuite afficher la liste pour que vous puissiez vérifier son contenu.

Note : bien qu'ici toutes nos m lignes ont le même nombres de valeurs (cela forme une grille) et les valeurs sont toutes du même type (des entiers), Python n'imposent pas ces contraintes.

Q4) Écrire un programme `func.py` qui reprends exactement le comportement de `listeliste.py` mais dans lequel vous définissez une fonction `liste_zeros2d(m, n)` qui crée et renvoie la liste de liste voulue.

Exercice 2 – Fonction sur liste

Rappel : pour cet exercice, travaillez dans un dossier `ex2` lui même dans le dossier `tp2`.

$\mapsto$ **Aide** $\mapsto$ Il est possible de modifier les éléments d'une liste, par exemple avec `ma_liste[mon_indice] = ma_nou` les indices commençant à 0 (le premier élément est celui d'indice 0). Ainsi, par ex. dans le TP1, si on a une liste de chaînes de caractères, on peut prendre chaque élément, le convertir en entier et le remplacer dans la liste avec :

```python
lll = ["20", "200", "4"]     # ou par exemple issu de input + split
for i in range(len(lll)):
    lll[i] = int(lll[i])
```

Assurez-vous de bien comprendre ce que fait ce bout de code (le choix des noms de variables est arbitraire). Important : la liste d'origine est modifiée (on remplace ses éléments).

Q5) Écrivez un programme `nouv.py` qui fait la même chose que l'exemple mais en créant une nouvelle liste (en s'inspirant de l'exercice précédent qui crée une liste vide puis utilise la fonction `append`).

'Q6) Comment vérifier que votre programme fonctionne, c'est à dire qu'il produit bien une liste avec des entiers et aussi qu'il laisse la liste d'origine intacte ?

Q7) Écrivez un programme `nouvfunc.py` qui défini une fonction `vers_entiers(l)` qui reçoit en paramètre une liste (supposée de chaîne de caractères) et renvoie une nouvelle liste avec les entiers correspondants. On veut par exemple pouvoir écrire :

```python
tailles = vers_entier(input("Entrez les tailles : ").split(" "))
...
```

'Q8) Jusque là, avez-vous utilisé la « technique de la fonction principale » ? Pensez à l'utiliser (cf TP1), c'est une bonne habitude.

Exercice 3 – « Liste en compréhension »

Rappel : pour cet exercice, travaillez dans un dossier `ex3` lui même dans le dossier `tp2`.

Si l'on voulait calculer une liste des logarithmes des carrés des nombres des multiples de 5 allant de 0 à 100, on pourrait écrire en Python :

```
les_log(les_carrés(range(0, 100, 5)))
```

mais pour cela il faudrait définir les fonctions `les_log` et `les_carrés`, en suivant exactement le même modèle que `vers_entiers`.

|↦ **Aide** |↦ Ce besoin est tellement courant que Python propose une syntaxe (notation) appelée « liste en compréhension ». Cette syntaxe est une sorte de mélange entre la notation pour créer une nouvelle liste `[ ... ]` et celle du `for` (il est possible d'ajouter en plus un `if`). Voici un exemple qui permet de définir `les_carrés` en utilisant cette syntaxe :

```
def les_carrés(lll):
    resultat = [ e**2 for e in lll ]
    return resultat
```

ou sans la variable `resultat`, maintenant inutile :

```
def les_carrés(lll):
    return [ e**2 for e in lll ]
```

Q9) Écrivez un programme `logcarre.py` qui crée trois fonctions `vers_floats`, `les_log` et `les_carrés`, demande à l'utilisateur d'entrer une liste de nombres réels, les convertis, les passe au carré, les passe au log, et finalement les affiche.

|↦ **Aide** |↦ la fonction `log` existe dans la bibliothèque (module) `math`, on peut donc l'importer avec `from math import log`. Elle correspond au logarithme naturel (ln en math).

Q10) Écrivez un programme `logcarredirect.py` qui fait exactement la même choses mais utilise uniquement `vers_float` et utilise une seule « liste en compréhension » pour calculer directement la liste des log des carrés, à partir de la valeur renvoyée par `vers_float`.

Q11) Écrivez un programme `cubelogunplus.py` qui demande aussi une liste de réels et calcule la liste des $(log(1+x))^3$.

'Q12) Combien de lignes sont différentes entre les 2 programmes `logcarredirect.py` et `cubelogunplus.py` ?

Exercices 4 – Accès dans une grille

Voici une fonction qui crée une grille, avec m lignes et n colonnes, remplie de 0 :

```
def grille(m, n):
    return [ [ 0 for j in range(n) ] for i in range(m) ]
```

Q13) Assurez-vous de comprendre ce qui est écrit, et d'écrire un programme `testgrille.py` qui utilise cette fonction pour bien comprendre ce qu'elle fait.

'Q14) À l'issue de votre étude de cette fonction, comment expliqueriez vous ce que l'on appelle ici une grille ?

Q15) Écrivez un programme `grillehasard.py` qui défini une fonction `grille_random(m, n, a, b)` qui crée une grille de taille m, n où chaque élément ne vaut pas 0 mais un nombre tiré au hasard entre a et b. Le programme utilisera cette fonction pour créer et afficher (au moins) une grille de 4 lignes et 7 colonnes, avec des nombres entre 0.1 et 0.75.

$\mapsto$ **Aide** $\mapsto$ la fonction `uniform(a, b)` du module `random` permet de tirer un nombre aléatoirement entre a et b.

(répondez dans votre compte-rendu et vérifiez dans votre programme)

'Q16) Quelle ligne Python permet d'afficher la valeur de la 1ère case de la 1ère ligne ?

'Q17) Quelle ligne Python permet d'afficher la valeur de la 1ère case de la 2ème ligne ?

'Q18) Comment mettre la valeur de la 3ème case de la 2ème ligne à 1 ?

'Q19) Comment mettre la valeur de la 1ère case de la dernière ligne à 1 ?

'Q20) Comment mettre la valeur de la dernière case de la dernière ligne 1 ?

(vous pouvez copier le programme précédent pour commencer celui ci)

Q21) Écrivez un programme `affgrille.py` qui crée une grille au hasard, puis va créer une fenêtre qtido pour l'afficher et utiliser `attendre_fermeture` pour attendre que l'utilisateur ferme la fenêtre. Les questions suivantes sont des étapes pour arriver à afficher toute la grille.

Q22) Pour l'instant essayez d'afficher uniquement un carré pour la 1ère valeur de la 1ère ligne. Faites que le carré (rectangle) aie une taille de 50 par 50 (en pixels). Faites aussi que sa couleur dépende de la valeur de la case, par exemple avec `couleur(f, v, 0, 0)` pour que le carré soit rouge mais d'une intensité qui dépend de v.

Q23) Écrivez le code pour afficher les 4 premiers carrés de la première ligne (sans boucle).

Q24) Essayez maintenant de faire que la première ligne soit affichée complètement, chaque carré ayant une couleur qui dépend de sa valeur.

Q25) Essayez maintenant de faire que l'ensemble de la grille soit affichée.

Q26) Challenge : copier votre programme en tant que `challenge.py` et faite que l'affichage se fasse en boucle (comme une simulation) et qu'à chaque 25 millisecondes, la grille soit remplacée par une nouvelle grille au hasard.

Q27) Challenge : au lieu de remplacer par une nouvelle grille au hasard, faite que la nouvelle grille contienne une moyenne pondére entre la grille précédente et une grille tirée au hasard, par exemple en utilisant 0.9/0.1 comme poids ($0.9 \times valgrille1 + 0.1 \times valgrille2$).