TP sur les tableaux

Thème types construits

Première NSI, Lycée du Parc

On désigne par tableaux, les objets de type `list` du langage Python. Dans les exercices, comme dans les QCM d'E3C, on rencontrera parfois l'appellation Python liste pour désigner la structure de données tableau.

Exercice 1

Créer un fichier *TP-Tableaux.py* pour rassembler les scripts rédigés pendant le TP.

Enregistrer ce fichier dans un dossier pertinent de son espace personnel sur le réseau pédagogique.

Exercice 2

QCM type E3C

1. On exécute le code suivant :

```
t = [1,2,3,4,5,6,7,8,9]

v = [c for c in t if c%3 == 0]
```

Quelle est la valeur de la variable `v` à la fin de cette exécution ?

- **Réponse A :** 18 **Réponse B :** `[1,4,7]`
- **Réponse C :** `[3,6,9]` **Réponse D :** `[1,2,3,4,5,6,7,8,9]`

2. On définit : `resultat = [ i*2 for i in range(10)]`.

Quelle est la valeur de `resultat` ?

- **Réponse A :** `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
- **Réponse B :** `[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]`
- **Réponse C :** `[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]`
- **Réponse D :** `[2, 4, 6, 8, 10, 12, 14, 16, 18]`

3. On considère la fonction suivante :

```
def somme(tab):
```

```
s = 0
for i in range(len(tab)):
    .....
return s
```

Par quelle instruction faut-il remplacer les points de suspension pour que l'appel `somme([10,11,12,13,14])` renvoie 60 ?

- **Réponse A :** `s = tab[i]` **Réponse B :** `s = s + tab[i]`
- **Réponse C :** `tab[i] = tab[i] + s` **Réponse D :** `s = s + i`

Exercice 3

1. La fonction suivante doit calculer la moyenne d'un tableau de nombres, passé en paramètre. Avec quelles expressions faut-il remplacer les points de suspension pour que la fonction soit correcte ? *Auteur Sylvie Genre*

```
def moyenne(tableau):
    total = ...
    for valeur in tableau:
        total = total + valeur
    return total / ...
```

2. Donner la valeur des expressions Python suivantes :

```
>>> [1, 2, 3] + [4, 5, 6]
>>> 2 * [1, 2, 3]
```

Méthode

À partir de cet exercice, on précisera pour chaque fonction, dans une chaîne de documentation (ou *docstring*) multiligne placée juste après l'en-tête de la fonction entre triples quotes ou guillemets, sa **spécification**.

Celle-ci est constituée :

- du ou des **type(s)** du ou des paramètre(s), en précisant si la fonction ne prend pas de paramètre
- de la ou des **précondition(s)** que doivent vérifier ces paramètres
- du ou des **type(s)** de la (ou les) valeur(s) renvoyée(s), en précisant si la fonction ne renvoie rien (`None` par défaut en Python)
- de la ou des **postcondition(s)** que doivent vérifier la ou les valeur(s) renvoyée(s)

Pour vérifier les préconditions on utilise l'instruction `assert` qui permet de vérifier l'assertion (à valeur booléenne) qu'elle préfixe : si l'assertion est vraie, l'exécution continue sinon une exception `AssertionError` est levée et l'exécution s'interrompt. On peut passer un message d'erreur

personnalisée en le séparant de l'assertion par une virgule :

```
>>> assert (1 + 1) % 2 == 0, 'attention on calcule modulo 2'
>>> assert (1 + 1) % 2 == 2, 'attention on calcule modulo 2'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AssertionError: attention on calcule modulo 2
```

Voici un exemple de rédaction pour la deuxième fonction de l'exercice 4 :

```
def vsom(tab1, tab2):
    """
    Paramètres :
        tab1 et tab2 des tableaux de type list contenant des nombres de
             type int ou float
    Préconditions :
        tab1 non vide et len(tab1) == len(tab2)
    Valeur renvoyée :
        un tableau de type list contenant des nombres de type int ou
             float
    Postcondition :
        le tableau renvoyé est constitué des sommes
        terme à terme des éléments de tab1 et tab2
    """
    assert len(tab1) > 0 and len(tab1) == len(tab2); "tab1 et tab2
         doivent être de même longueur"
    return ................. #à  compléter
```

Exercice 4

1. Écrire une fonction Python `smul` à deux paramètres, un nombre et une liste de nombres, qui multiplie chaque élément de la liste par le nombre et renvoie une nouvelle liste :

```
>>> smul(2, [1, 2, 3])
[2, 4, 6]
```

2. Écrire une fonction Python `vsom` qui prend en paramètre deux listes de nombres de même longueur et qui renvoie une nouvelle liste constituée de la somme terme à terme de ces deux listes :

```
>>> vsom([1, 2, 3], [4, 5, 6])
[5, 7, 9]
```

3. Écrire une fonction Python `vdif` qui prend en paramètre deux listes de nombres de même longueur et qui renvoie une nouvelle liste constituée de la différence terme à terme de ces deux listes (la deuxième moins la première).

```
>>> vdif([1, 2, 3], [4, 5, 6])
[3, 3, 3]
```

4. Écrire une fonction Python `vprod` qui prend en paramètre deux listes de nombres de même longueur et qui retourne une nouvelle liste constituée des produits terme à terme de ces deux listes.

```
>>> vprod([1, 2, 3], [4, 5, 6])
[4, 10, 18]
```

Exercice 5

1. Écrire une fonction `produit(tab)` qui retourne le produit des éléments d'un tableau de nombres `tab`.
2. Écrire une fonction `all_positive(tab)` qui retourne un booléen indiquant si tous les éléments du tableau de nombres `tab` sont strictement positifs.
3. Écrire une fonction `any_positive(tab)` qui retourne un booléen indiquant si au moins un élément du tableau de nombres `tab` est strictement positif.

Exercice 6

1. Écrire une fonction `tableau_aleatoire(n, a, b)` qui renvoie un tableau de `n` entiers tirés aléatoirement entre les entiers `a` et `b` inclus. On utilisera la fonction `randint` du module `random`.

```
Help on method randint in module random:

randint(a, b) method of random.Random instance
    Return random integer in range [a, b], including both end points.
```

2. Écrire une fonction `histo_echantillon(nbexp)` qui renvoie un tableau de taille 6 comptant le nombre d'occurences de chaque face numérotée de 1 à 6 sur un échantillon de `nbexp` lancers d'un dé cubique équilibré.

3. On donne ci-dessous une fonction qui prend en paramètre un tableau et renvoie un élément extrait au hasard du tableau. Elle permet par exemple de simuler un tirage sans remise d'une boule dans une urne.

```
from random import randint
def tirage_sans_remise(urne):
    """
    Paramètres :
        urne un tableau homogène type list
    Précondition :
```

```
        urne non vide
    Valeur renvoyée :
        un élément du même type que ceux dans urne
    Postcondition :
        l'élement renvoyée a été extrait aléatoirement de urne
        urne a été modifiée
    """
    return urne.pop(randint(0, .......))
```

- Compléter les pointillés pour respecter la spécification de la fonction.
- Écrire une fonction `echantillon_tirage_sans_remise` respectant la spécification ci-dessous. Les annotations de type sont des métadonnées optionnelles décrivant les types des paramètres et des valeurs renvoyées.

```
def echantillon_tirage_sans_remise(urne:list, nbtirage:int) -> list:
    """
    Paramètres :
        urne un tableau homogène type list
        nbtirage un entier de type int
    Préconditions :
        nbtirage >= 0
        urne non vide
    Valeur renvoyée :
        un tableau d'entiers de type list
    Postcondition :
        le tableau renvoyé est un échantillon extrait
        aléatoirement de l'urne sans remise
    """
```

Exercice 7

- La fonction `ord` prend en paramètre un caractère de type `string` et renvoie son point de code dans le jeu de caractères Unicode.
- La fonction `chr` prend en paramètre un point de code Unicode et renvoie le caractère correspondant.

```
>>> ord('a')
97
>>> chr(97)
'a'
```

1. Construire un tableau `alphabet` qui contient toutes les lettres minuscules de l'alphabet romain.
2. Construire un tableau `consonne` qui contient toutes les consonnes dans `alphabet`.
3. Écrire une fonction `occurences(chaine)` qui prend en paramètre une chaîne de caractère et renvoie un tableau de taille 26 avec le nombre d'occurences dans `chaine` des 26 lettres de

- l'alphabet romain.

Exercice 8

Exercice du manuel de NSI de Thibault Balabonski chez Ellipses

En mathématiques, la suite de Fibonacci est une séquence d'entiers définie ainsi : on part des entiers 0 et 1 puis on construit à chaque fois l'entier suivant comme la somme des deux précédents :

$$0, 1, 1, 2, 3, 5....$$

Compléter la fonction `fibonacci(n)` ci-dessous pour qu'elle renvoie un tableau contenant les `n` premiers termes de la suite de Fibonacci avec `n` entier supposé supérieur ou égal à 2. Les deux assertions proposées en tests unitaires, doivent être vérifiées.

```
def fibonacci(n):
    #à  compléter

# Tests unitaires (vérification de postconditions)
f6 = fibonacci(6)
assert f6 == [0,1, 1, 2,3,5]
f30 = fibonacci(30)
assert f30[29] == 514229
```

Méthode

Pour stocker de façon persistante (sur le disque et pas seulement en mémoire vive) des informations lisibles par un humain, on utilise un **fichier texte** qui est une simple séquence de caractères. Le contenu du fichier doit respecter un **format** qui fixe des contraintes pour que les informations soient interprétables lors de la lecture. Par exemple le code source d'un programme Python ou d'une page Web en HTML, un fichier de mesures au format CSV, sont des fichiers textes.

Tester les séquences d'instructions suivantes dans une console Python pour découvrir des manipulations de base de fichiers textes.

- Création d'un fichier `test.txt` dans le répertoire de travail

```
>>> f = open('test.txt', 'w') # ouverture du fichier en écriture
>>> f.write('10\n') #  ligne 1 : '\n' est un caractère de saut de ligne
3
>>> f.write('11 12\n') #ligne 2 avec 6 caractères
6
>>> f.write('13 14 15\n') #ligne 3 avec 9 caractères
9
>>> f.close()  #fermeture du fichier
```

- Lecture et traitement ligne par ligne du fichier `test.txt` :

```
>>> g = open('test.txt', 'r') #ouverture du fichier
>>> ligne1 = g.readline() #lecture de ligne 1
>>> ligne1
'10\n'
>>> ligne1.rstrip() #nettoyage du caractère de fin de ligne
'10'
>>> int(ligne1.rstrip()) #conversion de chaine vers entier
10
>>> ligne2 = g.readline() #lecture de ligne 2
>>> ligne2
'11 12\n'
>>> ligne2.rstrip() #nettoyage du caractère de fin de ligne
'11 12'
>>> ligne2.rstrip().split(' ') #découpage de ligne 2 selon le caractère
    par défaut de séparation ' ', on peut écrire juste split()
['11', '12']
>>> [int(c) for c in ligne2.rstrip().split()] #conversion en entier
[11, 12]
>>> g.close()  #fermeture du fichier
```

Exercice 9

- **Énoncé :**

Comme le disait le grand-père de Joseph Marchand, « Le plus beau voyage est celui qu'on n'a pas encore fait ». New York était dans la tête de Joseph depuis plusieurs années maintenant, et il a décidé aujourd'hui d'acheter son billet d'avion. Quelques secondes avant de cliquer sur le bouton « Acheter ! », son amie Haruhi lui envoie une liste de prix qu'elle a trouvés sur Internet. Joseph est curieux de voir si ces derniers sont moins chers que le billet qu'il s'apprêtait à acheter.

- **Entrée :**

Sur la première ligne le prix initial du billet de Joseph. Sur la deuxième ligne un entier N , correspondant au nombre de billets envoyés par Haruhi. La ligne suivante contient les N prix trouvés par Haruhi.

- **Sortie :**

Si Haruhi a trouvé au moins 3 prix strictement moins chers que celui de Joseph, affichez « ARNAQUE ! » pour l'avertir. Sinon « Ok bon voyage, bisous, n'oublie pas de m'envoyer des photos ! ».

- Exemple 1 :
  - Entrée :

```
570
4
```

```
495 1200 540 450
```

  - Sortie :

```
ARNAQUE !
```

- Exemple 2:
  - Entrée :

```
820
5
580 2000 970 1050 820
```

  - Sortie :

```
Ok bon voyage, bisous, n'oublie pas de m'envoyer des photos !
```

1. Récupérer les fichiers textes `arnaque-exemple1.txt` et `arnaque-exemple2.txt` avec les entrées des deux exemples.
2. En s'inspirant de la méthode de lecture de fichier texte présentée précédemment, compléter la fonction ci-dessous pour qu'elle résolve le problème. Les deux assertions doivent être vérifiées.
3. Se créer un compte sur le site Prologin et soumettre sa solution au juge en ligne sur https://prologin.org/train/2019/qualification/arnaque_aerienne.

   Avant de soumettre, remplacer dans la fonction l'instruction `f = open(inputfile)` par `f = inputfile` et le code du programme principal par :

```
import sys
arnaque(sys.stdin)
```

   `sys.stdin` est l'entrée standard de l'interpréteur Python.
4. Sur la même plateforme, résoudre cet autre problème du même type : https://prologin.org/train/2018/qualification/faites_place

```
def arnaque(inputfile):
    """Résolution du problème
    https://prologin.org/train/2019/qualification/arnaque_aerienne"""
    f = open(inputfile) #ouverture du fichier d'entrée
    # TO DO à compléter
    f.close()
    c = 0
    # TO DO à compléter
    if c >= 3:
        print("ARNAQUE !")
        return 1
    else:
```

```
        print("Ok bon voyage, bisous, n'oublie pas de m'envoyer des
             photos !")
        return 0

# Tests unitaires
assert arnaque('arnaque-exemple1.txt') == 1
assert arnaque('arnaque-exemple2.txt') == 0
print("Tests unitaires réussis !")
```

Exercice 10

Écrire une fonction `maximum_intervalle(t, n)` qui prend en paramètres un tableau d'entiers `t` et un entier `n` supposé inférieur ou égal à la longueur de `t` et qui retourne la somme maximale sur tous les sous-tableaux de longueur `n` inclus dans `t` .

```
>>> maximum_intervalle([4,1,10,2,8,5], 3)
20
```