Fiche mémo Python
INFO201 - TD

1 Fonctions natives en Python

Opération sur un tableau/liste `tab`:

```
min(tab), max(tab)      # Minimum et maximum de tab
sum(tab)                # Somme des elements de tab
mean(tab)               # Moyenne des elements de tab
list.append(obj)        # Ajoute un element a list, equivalent a list += [obj]
list.insert(i, obj)     # Insertion d'un obj (int, caractere) a l'indice i
list.count(obj)         # Compte le nombre d'occurences de obj dans list
list.sort()             # Trie dans l'ordre croissant
```

Quelques bibliothèques utiles :

```
numpy, math, scipy  # Calcul scientifique, statistiques, maths
matplotlib          # Affichage graphique)

import numpy as np  # Importation de la bibliotheque numpy

# Quelques fonctions de numpy
np.cos(x), np.tanh(x), np.pi, np.array(obj), np.zeros(shape)
```

2 Boucles for

Il existe deux grandes manières de parcourir un tableau en utilisant une boucle for :

```
for i in range(0, len(tab)): # parcours des indices allant de 0 a len(tab)-1
    tab[i] = ...

for e in tab:                # parcours des elements de tab
    e = ...
```

Le *slicing* permet d'accèder à certaines parties d'un tableau `tab`. On utilise la syntaxe `tab[start:end:step]` pour découper un tableau à partir de l'indice `start` jusqu'à l'indice `end` (l'élément à cet indice n'étant pas inclu) avec un pas `step`.
Quelques exemples :

```
tab = [2, 5, 20, 4, 9, 11]

tab[-2]                         # 9
tab[0:3]                        # [2, 5, 20]
tab[1:len(tab):2]               # [5, 4, 11]
tab[::-1]                       # [11, 9, 4 ,20, 5, 2]

# parcours des elements ayant un indice pair
for i in range(0, len(tab), 2): # acces aux indices
for e in tab[0:len(tab):2]:     # acces aux elements

# parcours des elements ayant un indice impair
for i in range(1, len(tab), 2):
for e in tab[1:len(tab):2]:
```

3 Boucle for ou while ?

Pour savoir laquelle choisir, on peut se poser les questions suivantes : (1) Ai-je besoin de parcourir tous les éléments de mon tableau dans ce problème ? (2) Est-il nécessaire d'imposer une condition d'arrêt ?

- Quelques exemples avec une/des boucles `for` : recherche d'un min/max, addition/multiplication de deux matrices, recherche de **toutes** les occurences dans un tableau, tri rapide, etc.
- Avec une/des boucles `while` : recherche d'**une** occurence dans un tableau, *test* de monotonie/tri, *test* de matrice diagonale, recherche dichotomique/ternaire, calcul de trajectoire balistique, etc.

...faites les tests par vous-même !

4 Tableaux 2-D

Opérations sur tableaux à deux dimensions `mat`:

```
len(mat)          # Nombre de lignes de mat
len(mat[0])       # Nombre de colonnes de mat
mat[0][0]         # Acces a l'element a la 1ere ligne et 1ere colonne

# Acces a tous les elements de mat
for i in range(len(mat)):
    for j in range(len(mat[0])):
        mat[i][j] = ...
```

5 Dictionnaires

Un dictionnaire est un type de représentation de données. Au lieu d'utiliser des indices comme dans les tableaux, on utilise des clefs. D un dictionnaire.

```
>>> D = {}                    # Initialisation d'un dictionnaire
>>> D["nom"] = Dupond
>>> D["prenom"] = Isabelle
>>> D
{"nom": Dupond, "prenom": Isabelle}
```

Opération sur les dictionnaires :

```
>>> D.get("nom", v)           # Acces a la valeur correspondante a la clef "nom"
"Dupond"

>>> del D["nom"]              # Suppression d'une entree de D
>>> D
{"prenom": Isabelle}

>>> D = {"nom": Dupond, "prenom": Isabelle}
>>> for clefs in D.keys():    # Parcours des clefs de D et affichage
...     print clefs
>>> for vals in D.values(): # Parcours des valeurs de D et affichage
...     print vals
```

Cette section est directement inspirée de :
`https://python-django.dev/page-apprendre-dictionnaire-python`

6 Chaînes de caractères

Une chaîne de caractère est un type de données en Python. Elle est définie par des guillemets et on y accède comme un tableau, donc par indiçage. Les chaînes de caractères sont concaténables mais leurs éléments sont non mutables (impossible de faire `chaine[0]='e'` par exemple).
Le symbole \ est spécial : il permet de transformer le caractère qui le suit en une autre information :

- \n est un saut de ligne
- \t est une tabulation
- \' est un ' mais il ne ferme pas la chaîne
- \" est un " mais il ne ferme pas la chaîne
- \\ est un \

Quelques fonctions (à ne pas utiliser si cela est explicitement demandé) : `s.upper()`, `s.lower()`, `s.strip()`, `s.count(letter)`...
On peut formater une chaîne de caractère en utilisant les accolades {:} et la fonction `format(arg)` comme suit :

```
>>> for i in range(3):
>>>     print("{:02}".format(i))
00
01
02
```

D'autres exemples pour mieux comprendre la fonction `format()` : `https://pyformat.info/`

7 Fichiers

Fonctions utiles :

```
f = open(nom_fichier, mode) # Ouverture du fichier nom_fichier
f.readline()                # Lecture de la premiere ligne du fichier
f.readlines()               # Lectures de toutes les lignes du fichier
f.write(str)                # Ecriture de str dans le fichier
f.close()                   # Fermeture de l'acces au fichier
```

8 Quelques algorithmes utiles

8.1 Le tableau est-il trié ?

Principe : on compare les valeurs du tableau deux à deux. Tant que la valeur est plus grande (ou petite suivant que l'on veuille vérifier une croissance ou une décroissance) que sa valeur précédente, on continue, sinon on s'arrête.

Algorithm 1 Vérification si un tableau est croissant

Input: T un tableau de nombre
Output: **True** si le tableau est trié dans l'ordre croissant, **False** sinon

```
1: i = 1
2: inc = True
3: while inc and i < len(T) do
4:     if T[i-1] > T[i] then
5:         inc = False
6:     end if
7:     i += 1
8: end while
9: return inc
```

8.2 Recherche dichotomique d'un nombre p

Principe : on calcule la moyenne entre a et b, puis on compare cette moyenne au nombre p. Si la moyenne est inférieur à p, on remplace la borne inférieure de l'intervalle de recherche par la moyenne + 1, sinon on remplace la borne supérieur par la moyenne. Lorsque la moyenne n'est ni inférieure ni supérieure à p, alors elle est égale à p, on a donc trouvé p.

Algorithm 2 Recherche dichotomique d'un nombre $p \in [a, b]$

```
1: found = False
2: while not found do
3:     m = (a+b)/2
4:     if m < p then
5:         a = m + 1
6:     else if m > p then
7:         b = m
8:     else
9:         found = True
10:    end if
11: end while
12: return found
```

Liens utiles:

- Doc `numpy` : `https://docs.scipy.org/doc/numpy/index.html`
- Fonctions mathématiques de la bibliothèque `numpy` : `https://docs.scipy.org/doc/numpy/reference/routines.math.html`
- Fonctions utiles pour la manipulation de listes : `https://www.tutorialspoint.com/python/python_lists.htm`