Math93.com

TD - Algorithmique
Débuter en Python - Partie 2

Table des matières

I Les Listes 2

1 Définition d'une liste 2

2 Moyenne d'une liste de valeurs 3

3 Un premier bilan sur les listes 4

4 TD sur les listes 4

II Les Dictionnaires (NSI only) 5

1 Les dictionnaires 5
- 1.1 Première définition . . . 5
- 1.2 Exemples de dictionnaires . . . 5

2 Les méthodes liés aux dictionnaires 5
- 2.1 .get() : Comment récupérer une valeur dans un dictionnaire python ? . . . 5
- 2.2 .key() : Comment récupérer les clés d'un dictionnaire python par une boucle ? . . . 6
- 2.3 .value() : Comment récupérer les valeurs d'un dictionnaire python par une boucle ? . . . 6
- 2.4 .item() : Comment récupérer les clés et les valeurs d'un dictionnaire python par une boucle ? . . . 6

3 Un TD sur les dictionnaires 6

III Complément : l'affichage sous python 7

Première partie

Les Listes

La notion de liste est explicitement au programme de mathématiques et de NSI en classe de première (et terminale).

I.1. Définition d'une liste

Une liste : L

Une liste est une suite d'éléments numérotés dont le premier indice est 0. En Python, une liste s'écrit entre crochets [... , ..., ..., ...] avec les éléments séparés par des virgules.

- Le premier élément de la liste est L[0], le 2e est L[1], ...
- Une liste peut être écrite de manière explicite : L = ["Lundi", "Mardi","Mercredi"]
- Sa longueur est donnée par $len(L)$.
- Si les éléments de la liste sont comparables, le max. est donné par *max(L)*, le min. par *min(L)*
- L=[] permet de définir une liste vide.
- Si L est une liste, l'instruction *L.append(x)* va ajouter l'élémentt [x] à la liste L.

```
# Par exemple
>>> maliste=[31,'salut',78,'bonjour',2022,'NSI']
>>> maliste[0]
31
>>> maliste[1]
'salut'
>>> maliste[5]
'NSI'
>>> len(maliste)
6
>>> maliste.append('toto')
>>> maliste
[31,'salut',78,'bonjour',2022,'NSI','toto']
```

Exercice 1

1. Écrire une liste nommée **semaine** contenant les jours de la semaine.
2. Comment afficher le premier jour de la semaine?
3. Comment afficher le deuxième jour de la semaine?
4. Comment afficher le dernier jour de la semaine?
5. Comment afficher la taille de la liste?
6. Ajouter à votre liste le 8e jour de la semaine nommé 'crésudi' avec l'instruction **.append**

I.2. Moyenne d'une liste de valeurs

Exercice 2

1. Définissez deux listes de nombres L1 et L2 ainsi :

```
# Dans l'éditeur PYTHON
L1 =  [-1,3,5,7,10]  # Liste 1 pour les exemples
L2 =  [5,12,15,7,10,19]  # Liste 2
```

2. Calculer à la main la moyenne de ces deux séries de valeurs.

```
# Dans l'éditeur PYTHON
# moyenne L1 = ...
# moyenne L2 = ...
```

3. On veut écrire une fonction qui renvoie la moyenne des termes d'une liste. Complétez-la et testez-la sur les listes L1 et L2 .

```
# Dans l'éditeur PYTHON
def moyenne(L):
    '''IN : L une liste de nombres flottants (décimaux)
      OUT : la moyenne des nombres de la liste'''
    s = 0 # on initialise s à 0
    ...
```

4. Modifiez-la pour que la fonction renvoie aussi la longueur de L, la valeur maximale et la minimale de la liste.

max(L) et min(L) :

max(L) et min(L) renvoie les valeurs maximale et minimale des éléments de la liste L.

Question Bonus

(a) Écrire une fonction **maxi(liste)** qui donne le maximum de la liste, sans utiliser la fonction max.

(b) Écrire une fonction **mini(liste)** qui donne le minimum de la liste, sans utiliser la fonction min.

I.3. Un premier bilan sur les listes

Créer une liste vide	$L = []$ ou $L = list()$	
Créér une valeur	Pour récupérer une valeur dans une liste python : nom_liste[index] *Attention le 1er élément de la liste L est L[0], le 2e est L[1] ...*	L[0]=25 *le 1er item est 25*
Remplacer item	On peut ajouter ou remplacer un élément dans une liste : $liste[index] = valeur$	$L[1] = 76$ $L[2] = 20$ Donc $L = [25, 76, 20]$
Ajouter item : — .append()	On peut ajouter un élément dans une liste, à la fin : $liste.append(valeur)$	L.append(15) *on ajoute 15 à la fin de la liste L* $L = [25, 76, 20, 15]$
Supprimer item : — del — .remove() — .pop()	On peut supprimer un valeur d'une liste — Avec l'index : **del ou pop** — La fonction **del liste[i]** va supprimer l'item à l'index i spécifié (pas besoin de connaître l'item). — La méthode **liste.pop(i)** va supprimer l'item à l'index i spécifié et renvoyer sa valeur. — Avec sa valeur : la méthode **.remove()** liste.remove(x) : supprime de la liste le premier élément dont la valeur est égale à x	$L = [25, 76, 20, 15]$ > del L[0] $L = [76, 20, 15]$ > L.pop(1) $L = [76, 15]$ > L.remove(15) $L = [76]$
Parcourir	On peut parcourir une liste : — par les index : **for index in range(len(liste))** — directement : **for élément in liste**	
index()	Trouver l'index d'une valeur $v1$ La méthode : liste.index($v1$)	> L2=['a' , 'c' , 'f'] > L2.index('f') 2
sort() : trier	liste.sort() va trier la liste par ordre croissant	L.sort()
count()	liste.count(x) : renvoie le nombre d'apparitions de x dans la liste	
liste[i : j]	Liste[i : j] : renvoie une sous liste composée des éléments Liste[i] à Liste[$j-1$]	$L[2:4] = \left[L[2], L[3]\right]$
insert()	Insère un élément à la position indiquée. Le premier argument est la position de l'élément courant avant lequel l'insertion doit s'effectuer. Donc **liste.insert(0, x)** insère l'élément x en tête de la liste et **liste.insert(len(a), x)** est équivalent à **liste.append(x)**.	
Copier une liste $L2 = list(L1)$	Attention, l'instruction $L1 = L2$ ne peut pas être utilisée car dans ce cas les deux listes seront modifiée simultanément.	L2 = list(L1) ou L2 = L1[:]

I.4. TD sur les listes

Faire le TD : - Recherche des occurrences sur des valeurs de type quelconque.

Deuxième partie

Les Dictionnaires (NSI only)

Cette section peut être omise ne première lecture. Les dictionnaires ne sont au programme que de la fameuse spécialité NSI de première.

II.1. Les dictionnaires

II.1.1 Première définition

$\{clé_1 : val_1, clé_2 : val_2, ..., clé_n : val_n\}$:

$\{clé_1 : val_1, clé_2 : val_2, ..., clé_n : val_n\}$ est une liste dont la clé n'est pas l'indice de position mais qui une valeur qui peut être de n'importe quel type.
- Un dictionnaire en python est donc une sorte de liste mais au lieu d'utiliser des index , on utilise des clés alphanumériques.
- Dans un dictionnaire, les informations ne sont pas stockées dans un ordre précis. Pour accéder aux valeurs, on utilise les clés.

II.1.2 Exemples de dictionnaires

```
# Dans la console PYTHON
# Pour initialiser un dictionnaire
>>>dico = {} # ou dico=dict{}

# Pour ajouter des valeurs à un dictionnaire il faut indiquer
# une clé ainsi qu'une valeur :
>>> dico["nom"] = "Turing" # clé1 = "nom" et valeur associée "Turing"
>>> dico["prenom"] = "Alan"
>>> dico["annee_naissance"] = 1912

>>> dico
{'nom': 'Turing', 'prenom': 'Alan', 'annee_naissance': 1912}

# On peut vérifier que la variable dico est bien de type dictionnaire
>>>type(dico)
<class 'dict'>
```

II.2. Les méthodes liés aux dictionnaires

II.2.1 .get() : Comment récupérer une valeur dans un dictionnaire python ?

La méthode get vous permet de récupérer une valeur dans un dictionnaire et si la clé est introuvable, vous pouvez donner une valeur à retourner par défaut :

```
>>> dico
{'nom': 'Turing', 'prenom': 'Alan', 'annee_naissance': 1912}

>>> dico
{'nom': 'Turing', 'prenom': 'Alan', 'annee_naissance': 1912}
>>>dico.get("nom")
'Turing'
>>>dico.get("annee_naissance")
1912
>>>dico.get("poids") # si la clé n'est pas trouvée, rien ne s'affiche
```

II.2.2 .key() : Comment récupérer les clés d'un dictionnaire python par une boucle ?

Pour récupérer les clés on utilise la méthode keys .

```
fiche = {"nom":"Wayne", "prenom":"Bruce"}
for cle in fiche.keys():
    print cle
```

II.2.3 .value() : Comment récupérer les valeurs d'un dictionnaire python par une boucle ?

Pour cela on utilise la méthode values .

```
fiche = {"nom":"Wayne", "prenom":"Bruce"}
for valeur in fiche.values():
    print valeur
```

II.2.4 .item() : Comment récupérer les clés et les valeurs d'un dictionnaire python par une boucle ?

Pour récupérer les clés et les valeurs en même temps, on utilise la méthode items qui retourne un tuple .

```
fiche = {"nom":"Wayne","prenom":"Bruce"}
for cle,valeur in fiche.items():
    print cle, valeur
```

II.3. Un TD sur les dictionnaires

Vous pouvez maintenant traiter le TD sur les dictionnaires disponible sur : https://www.math93.com/lycee/nsi-1ere/nsi-1ere.html.

Troisième partie

Complément : l'affichage sous python

Les instructions *input* et *print* ne sont plus à mettre en avant. Il est cependant très utile pour comprendre un programme de faire des affichages de valeurs, parfois même dans une boucle afin de comprendre comment les variables sont modifiées.

1. Affichage de texte et de variables : deux méthodes

Deux méthodes pour afficher du texte et la valeur des variables.

```
# Méthode 1
# On place le texte entre "" et les variables séparées par des virgules

var1 = 2
var2 = -9
print("Méthode 1 : var1 =", var1," et var2 =", var2)

# Méthode 2 :  f (format) devant une chaîne de caractères

print(f"Méthode 2 : var1 = {var1} et var2 =  {var2}")
```

On remarque vite l'intérêt de la deuxième écriture.

```
# Dans la console PYTHON
Methode 1 : var1 = 2  et var2 = -9
Methode 2 : var1 = 2  et var2 =  -9
```

2. Affichage de texte et de variables : une 3e méthode par formatage

```
# Méthode 3 :  Affichage par formatage
# %d pour les entiers int(),
# %f pour les flottants float()
# %s pour les chaînes de caractères str()

var1=2       # var1 est un entier int()
var2=2.6     # var2 est un float()
var3="BEBE" # var3 est un str()

print("Méthode 3: var1 = %d, var2 =%f et var3 =%s" % (var1,var2,var3))
```

La méthode par formatage permet de varier la forme d'écritures surtout pour les flottants. Par exemple

```
var3=775776878.68767

# Méthode 3 :  Affichage par formatage
# %e pour forme a10^n
# %0.nf à 10^-n près

print("Méthode 3: var3 = %e, var3 = %0.2f " % (var3,var3))
```

```
Methode 3: var3 = 7.757769e+08, var3 = 775776878.69
```

3. Quelques compléments

Print("mon Texte",var)

- *print()* affiche la valeur numérique ou le texte qui suit.
- *print(a, b)* affiche à la suite sans passer à la ligne les éléments a et b.
- *print("mon Texte",var)* affiche le texte suivi de la valeur de la variable *var*.
- *print(a,end="")* affiche l'élément a et ne passe pas à la ligne : le prochain affichage continuera sur cette ligne.
- \n : permet de passer à la ligne dans l'affichage d'une même instruction print().
- \ : si une ligne de script est trop longue pour notre écran, on peut la casser avec un \ sans que Python interprète cela comme un saut de ligne.