INF1 : Algorithmique et Programmation

Cours 5 :
Listes et Tableaux

Domitile Lourdeaux

Université de technologie de Compiègne

Printemps 2024

Style de codage en Python

Défini dans PEP8 :

- https://www.python.org/dev/peps/pep-0008

Principales recommandations

- Utiliser des indentations de 4 espaces et pas de tabulation
- Utiliser des lignes vides pour séparer les fonctions ou pour scinder de gros blocs de code à l'intérieur de fonctions
- Lorsque c'est possible, placez les commentaires avant Utiliser des espaces autour des opérateurs et après les virgules, mais pas juste à l'intérieur des parenthèses
  - Exemple : a = f(1, 2) + g(3, 4)
- N'utiliser que des caractères ASCII pour vos noms de variables
- Utiliser des minuscules avec_trait_bas pour les noms de variables et camelCase pour les noms de fonction
- Faire en sorte que les lignes ne dépassent pas 79 caractères, au besoin en insérant des retours à la ligne

Retours à la ligne

Pour couper une ligne de code utiliser le caractère

```
>>> x = "on peut couper une chaine de \
        caractères si elle est trop  \
        longue"

>>> x
"on peut couper une chaine de caractères si elle est trop longue"
```

Il est possible aussi de couper n'importe quelle ligne de code avec ce moyen

```
if number % 3 == 0 \
        and number % 5 == 0:
    print("FizzBuzz")
```

Motivations

Pourquoi a-t-on besoin de tableaux / listes

- Par exemple, pour stocker les n notes de math d'un étudiant

Problème : comment représenter/stocker ces notes en mémoire ?

- Solution 1 : n variables (pas très pratique !)
- Solution 2 : tableau de nombres
  - Un seul nom pour tout le tableau
  - Accès aux nombres à l'aide d'un indice

Définitions

Tableau

- Un tableau est une collection ordonnée d'éléments ayant tous le même type
- Exemple : $etus = ["Alice", "Anna", "Bob", "Toto"]$

Liste

- Une liste est une collection ordonnée d'éléments pouvant avoir des types différents
- Exemple : $etu\_1 = ["Anna", 20, 16.5]$

Accès aux éléments

- On accède à chacun de ces éléments individuellement à l'aide d'un indice
- Exemples : $etus[1]$ correspond à "Anna", $etu\_1[1]$ correspond à 20

Notions (1)

Exemple :

T	12	132	35	5	63
	0	1	2	3	4

- T est la variable de type tableau
- Les indices 0, 1, 2, 3, 4 correspondent aux valeurs colonne
- Dimension = 1
- Longueur = 5
- t[1] = 132

Notions (2)

Exemple :

T	12	132	35	5	63
	-5	-4	-3	-2	-1

- Les indices -1, -2, -3, -4, -5 correspondent aux valeurs colonne en partant de la fib
- t[-1] = 63
- t[-2] = 5

Représentation

En python, un tableau peut être représenté par une liste :

- Une liste se note entre crochet
- Les éléments sont séparés par des virgules

```
>>> t = [12, 132, 35, 5, 63]
>>> t[0]
12
>>> len(t)
5
>>> type(t)
<class 'list'>
>>> t = []
>>> t
[]
>>> len(t)
0
```

Accès aux éléments

Accès à l'aide d'un index / indice

```
>>> ut = ["Belfort", "Compiègne", "Troyes"]
>>> ut[1]
'Compiègne'
```

Contrairement aux chaînes de caractères, les listes sont "mutables"

```
>>> ut[0] = "Belfort-Montbelliard"
>>> ut
['Belfort-Montbelliard', 'Compiègne', 'Troyes']
```

Initialisation (1)

Voir en ligne

Initialisation directe

```
>>> t = [1, 2, 3, 4, 5]
```

Initialisation à 0

```
>>> t = [0]*5
>>> t
[0, 0, 0, 0, 0]
```

Initialisation (2)

Réécriture

```
>>> for i in range(5) :
...     t[i] = int(input("Saisir un entier : "))
...
Saisir un entier : >? 5
Saisir un entier : >? 4
Saisir un entier : >? 3
Saisir un entier : >? 2
Saisir un entier : >? 1
```

Pas possible d'initialiser :

```
>>> for i in range(5) :
...     u[i] = int(input("Saisir un entier : "))
...
Saisir un entier : >? 5
Traceback (most recent call last):
  File "<input>", line 2, in <module>
NameError: name 'u' is not defined
```

```
>>> for i in range(6) :
...     t[i] = int(input("Saisir un entier : "))
...
Saisir un entier : >? 6
Saisir un entier : >? 5
Saisir un entier : >? 4
Saisir un entier : >? 3
Saisir un entier : >? 2
Saisir un entier : >? 1
Traceback (most recent call last):
  File "<input>", line 2, in <module>
IndexError: list assignment index out of range
```

Initialisation (3)

Méthode append() : ajout d'éléments

- Méthode à privilégier

```
N = 5
t = []
for i in range(N):
    t.append(i)
```

Méthode initalisation + nouvelle affectation

```
N = 5
t = [0]*N
for i in range(N):
    t[i] = i
```

Parcours d'un tableau

Affichage du tableau entier

```
>>> t = [1, 2, 3, 4, 5]
>>> print(t)
[1, 2, 3, 4, 5]
```

Parcours des éléments par itération sur les indices

```
>>> for i in range(len(t)) :
...     print(t[i], end = "")
...
12345
```

Parcours des éléments par itération sur les éléments

```
>>> for element in t:
...     print(element, end=",")
...
1,2,3,4,5,
```

Exemple (1)

Ecrire un programme permettant de calculer la somme de 2 vecteurs de R_N

- Exemple : R_3
  - $(3.5, 12, -6) + (-1, 6.3, 0) = (2.5, 18.3, -6)$

Algorithme ?

Exemple (2)

Algorithm 1 Somme de deux vecteurs

Afficher ("Taille des vecteurs : ")
Lire (n)
Initialiser u, v et s

```
pour i allant de 1 à n (inclus) faire
 |  Lire (u[i])
fin
pour i allant de 1 à n (inclus) faire
 |  Lire (v[i])
fin
pour i allant de 1 à n (inclus) faire
 |  s[i] = u[i] + v[i]
fin
```

Exemple (3)

```python
# Somme de deux vecteurs Rn

# Taille des vecteurs
n = int(input("Taille des deux vecteurs : "))
# Initialisation des vecteurs
u = []
v = []
s = []
# Saisie vecteur u
for i in range (n) :
    x = float(input(f"Saisie u[{i}] : "))
    u.append(x)
# Saisie vecteur v
for i in range (n) :
    x = float(input(f"Saisie v[{i}] : "))
    v.append(x)
# Somme des deux vecteurs
for i in range (n) :
    somme = u[i] + v[i]
    s.append(somme)
# Affichage du vecteur somme
print(s)
```

```
Taille des deux vecteurs : 3
Saisie u[0] : 3.5
Saisie u[1] : 12
Saisie u[2] : -6
Saisie v[0] : -1
Saisie v[1] : 6.3
Saisie v[2] : 0
[2.5, 18.3, -6.0]
```

Concaténation

Opérateur +

```
>>> t1 = ["Alice", "Anna"]
>>> t2 = ["Bob", "Toto"]
>>> t = t1 + t2
>>> t
['Alice', 'Anna', 'Bob', 'Toto']
```

Extraction

On peut extraire des parties de liste (comme pour les chaines) : *Voir tableau*

```
>>> t
['Alice', 'Anna', 'Bob', 'Toto']
>>> t[1:2]
['Anna']
>>> t
['Alice', 'Anna', 'Bob', 'Toto']
>>> t[1:3]
['Anna', 'Bob']
>>> t[:3]
['Alice', 'Anna', 'Bob']
>>> t[2:]
['Bob', 'Toto']
```

```
>>> t[-3:-1]
['Anna', 'Bob']
>>> t[-1:-3:-1]
['Toto', 'Bob']
>>> t[::-1]
['Toto', 'Bob', 'Anna', 'Alice']
>>> t[::2]
['Alice', 'Bob']
>>> t[::-2]
['Toto', 'Anna']
```

Copie d'un tableau (1)

Copie de t dans $t2$

- $t2 = t[:]$
- Copie effective : on peut ensuite modifier $t2$ sans que t ne soit modifié

Attention

On serait tenté d'écrire tout simplement :

- $t3 = t$
- Ici $t3$ et t référencent la même adresse
- Si un élément est modifié dans $t3$, il le sera aussi dans t

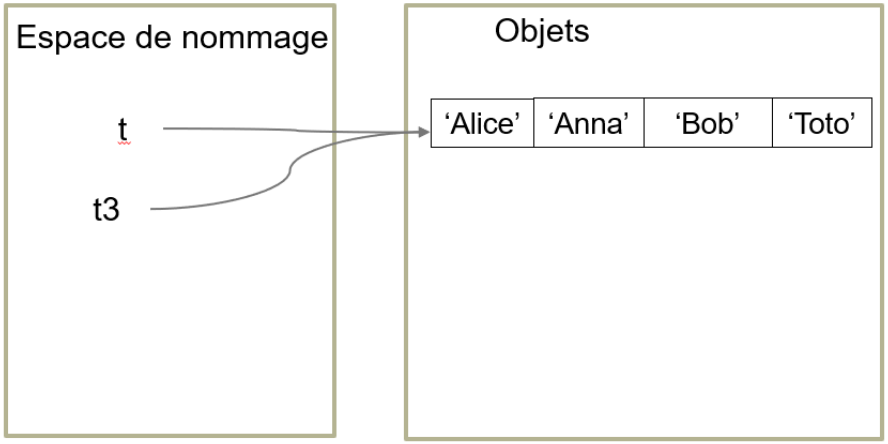
```
>>> t
['Alice', 'Anna', 'Bob', 'Toto']
>>> t2 = t[:]
>>> t2[2] = "Robert"
>>> t2
['Alice', 'Anna', 'Robert', 'Toto']
>>> t
['Alice', 'Anna', 'Bob', 'Toto']
>>> t3 = t
>>> t3[2] = "Marcel"
>>> t3
['Alice', 'Anna', 'Marcel', 'Toto']
>>> t
['Alice', 'Anna', 'Marcel', 'Toto']
```

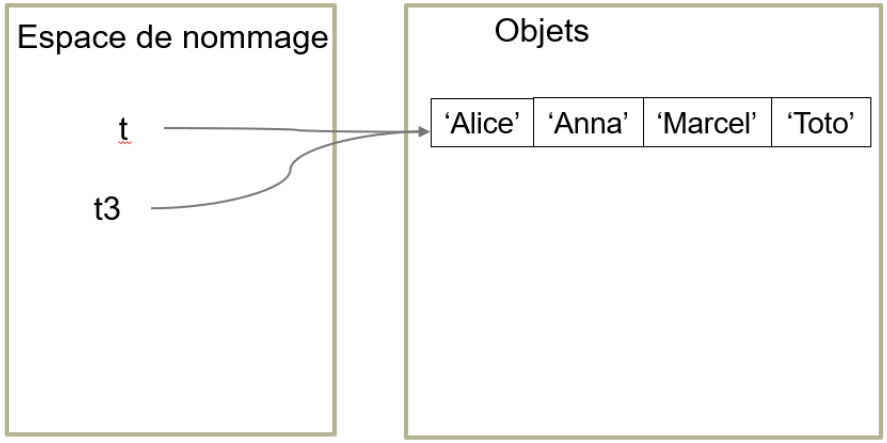
Copie d'un tableau (2)

Représentation en mémoire

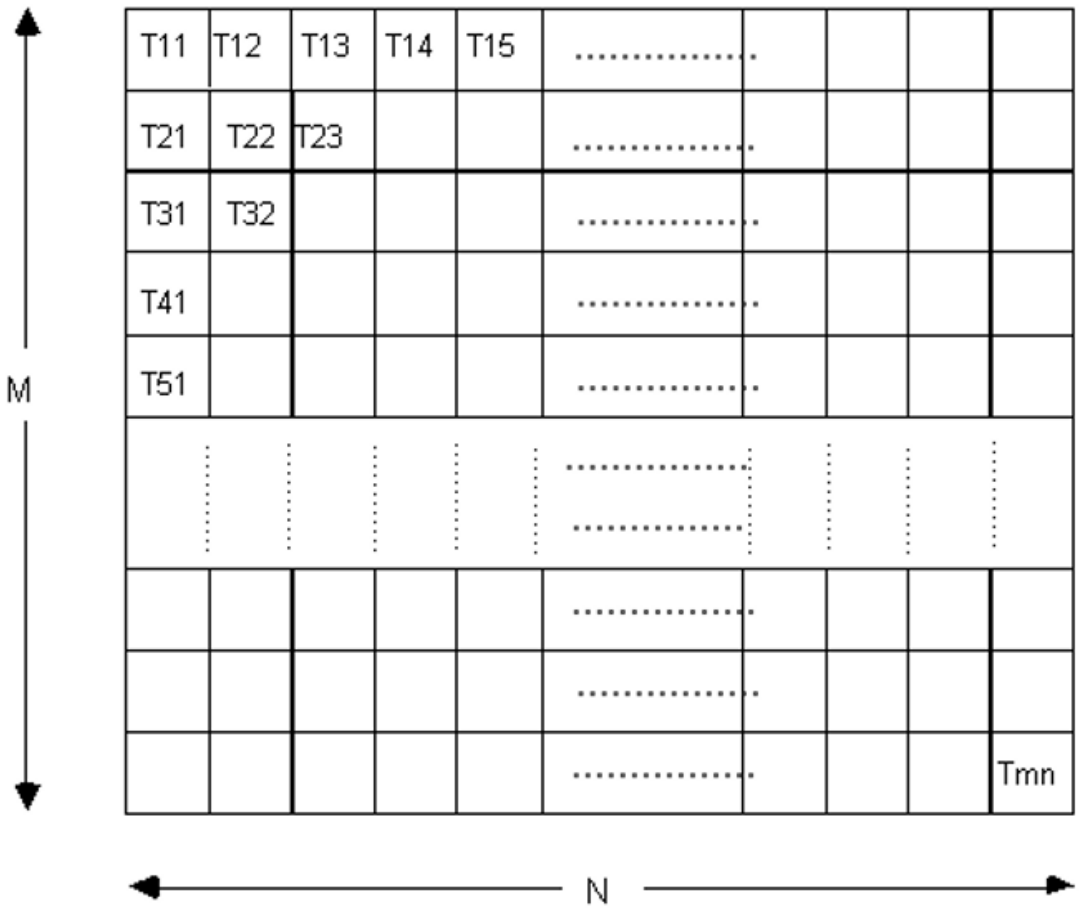
```
>>> t
['Alice', 'Anna', 'Bob', 'Toto']
>>> t3 = t
```

```
>>> t3[2] = "Marcel"
>>> t3
['Alice', 'Anna', 'Marcel', 'Toto']
>>> t
['Alice', 'Anna', 'Marcel', 'Toto']
```





Représentation en matrice



Représentation en Python

- Tableau à une dimension de lignes (ou de colonnes)
- Liste de listes

Exemple :

2	6	7
-1	21	12
10	-23	0

matrice = [[2, 6, 7], [-1, 21, 12], [10, -23, 0]]

Accès aux éléments

Accès à l'élément i, j

- $t[i][j]$

Exemple :

2	6	7
-1	21	12
10	-23	0

matrice = [[2, 6, 7], [-1, 21, 12], [10, -23, 0]]

Initialisation d'une matrice

Il faut initialiser avec une liste vide la matrice et chacune des lignes qui la composent

```
N = 3
M = 2
mat = []

# Saisie d'une matrice
for i in range (N) :
    mat.append([])
    for j in range (M) :
        mat[i].append(int(input(f"Element mat[{i}][{j}]:")))

# Affichage d'une matrice
for ligne in mat :
    print(ligne)
```

Affichage d'une matrice

```
mat = [[1,2,3],[4,5,6],[7,8,9],[10,11,12]]
# Affichage : par indice version 1
for i in range(4):
    for j in range(3):
        print(f"{mat[i][j]:2}", end=" ")
    print()
# Affichage : par indice version 2
for i in range(len(mat)):
    for j in range(len(mat[i])):
        print(f"{mat[i][j]:2}", end=" ")
    print()
# Affichage : par element
for ligne in mat:
    for colonne in ligne:
        print(f"{colonne:2}", end=" ")
    print()
```

Exercice (1)

Ecrire un programme permettant d'initialiser une matrice unité

Exemple : Matrice unité de dimension 5

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Exercice (1)

Algorithme permettant d'initialiser une matrice unité

Algorithm 2 Somme de deux vecteurs

```
N = 5
Initialiser mat
pour chaque ligne faire
    Initialiser la ligne
    pour chaque colonne faire
        si i == j alors
            Ajouter 1 à mat
        sinon
            Ajouter 0 à mat
        fin
    fin
fin
```

Exercice (3)

Programme Python permettant d'initialiser une matrice unité

```
N = 5
mat = []
for i in range (N) :
    mat.append([])
    for j in range (N) :
        if i == j :
            mat[i].append(1)
        else :
            mat[i].append(0)
for ligne in mat :
    for colonne in ligne :
        print(colonne,end=" ")
    print()
```

Affectations

Programme Python permettant d'initialiser une matrice unité

- Avec range

```
>>> t = [x * x for x in range(10)]
>>> t
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

- En parcourant un tableau

```
>>> t2 = [x * x for x in t]
>>> t2
[0, 1, 16, 81, 256, 625, 1296, 2401, 4096, 6561]
```

- Avec une condition

```
>>> t3 = [x * x for x in t if x * x % 3 == 0]
>>> t3
[0, 81, 1296, 6561]
```

Méthodes avancées

```
>>> t = ["Alice", "Anna", "Bob", "Toto"]
>>> # Remplacer des valeurs
>>> t[2] = "Bobby"
>>> t
['Alice', 'Anna', 'Bobby', 'Toto']
>>> t[2:4] = ["Robert", "Zoe"]
>>> t
['Alice', 'Anna', 'Robert', 'Zoe']
>>> # Ajouter un élément à la fin
>>> t.append("Zorro")
>>> t
['Alice', 'Anna', 'Robert', 'Zoe', 'Zorro']
>>> # Ajouter plusieurs éléments à la fin
>>> t.extend(["Zoulou", "Zulu"])
>>> t
['Alice', 'Anna', 'Robert', 'Zoe', 'Zorro', 'Zoulou', 'Zulu']
```

```
>>> # Enlever un élément
>>> del t[2]
>>> t
['Alice', 'Anna', 'Zoe', 'Zorro', 'Zoulou', 'Zulu']
>>> t.pop(2)
'Zoe'
>>> t
['Alice', 'Anna', 'Zorro', 'Zoulou', 'Zulu']
>>> t.remove('Zorro')
>>> t
['Alice', 'Anna', 'Zoulou', 'Zulu']
>>> # Supprimer tous les éléments de la liste
>>> del t[:] # del la liste complète
>>> t.clear() # ou avec méthode clear
```

Questions...