Programme du 24 novembre

- Structure de donnée linéaire: la liste
- Boucle `for`

Les listes en python

- La structure de données la plus répandue: une variable de type liste contient une collection numérotée de valeurs de types quelconques.

```
toto = [3, 7, 9, 1, 0, 8, 8, 8]
titi = [9, False, 8.34, True, 22, 1e-3, 'a', -1]
tutu = [[0,1], ['a', 'b', 'c'], 78.5]
vecteur = [0]*100
lv = []
```

- Les données peuvent être de tous les types existants (y compris des listes)
- Une liste peut contenir plusieurs occurrences d'une même valeur

Accès aux éléments d'une liste

Dans une liste, les éléments sont numérotés à partir de 0.
Affichage du $3^{\text{ième}}$ élément de la liste `titi` :

```
print(titi[2])
```

Modification du $3^{\text{ième}}$ élément de la liste `titi` :

```
titi[2] = 8
```

Accès aux éléments d'une liste

Dans une liste, les éléments sont numérotés à partir de 0.
Affichage du $3^{\text{ième}}$ élément de la liste `titi` :

```
print(titi[2])
```

Modification du $3^{\text{ième}}$ élément de la liste `titi` :

```
titi[2] = 8
```

Ce n'est pas la seule méthode pour modifier une liste, et ce n'est pas la plus recommandée. Il faut impérativement que la liste contienne déjà un élément à cette place.

Sorbonne Nouvelle

Slices

On peut aussi “extraire” une sous-liste d’une liste donnée:

par exemple `titi[2:5]` vaut `[8.34, True, 22]`

Le premier nombre correspond au premier élément extrait (on numérote à partir de 0) ; le second est le numéro **du premier élément exclu**

```
[  9   False   8.34   True   22   1e-3   'a'   -1  ]
   0     1      [2      3     4    [5      6     7
```

Extrait du mémento

pour les listes, tuples, chaînes de caractères, bytes…

Indexation conteneurs séquences

index négatif	−5	−4	−3	−2	−1
index positif	0	1	2	3	4
`lst=[10, 20, 30, 40, 50]`					
tranche positive	0	1	2	3	4	5
tranche négative	−5	−4	−3	−2	−1	

Nombre d'éléments

`len(lst)→5`

index à partir de 0
(de 0 à 4 ici)

Accès individuel aux **éléments** par `lst[`*index*`]`

`lst[0]→10` ⇒ *le premier* `lst[1]→20`

`lst[-1]→50` ⇒ *le dernier* `lst[-2]→40`

Sur les séquences modifiables (`list`), suppression avec `del lst[3]` et modification par affectation `lst[4]=25`

Accès à des **sous-séquences** par `lst[`*tranche début*`:`*tranche fin*`:`*pas*`]`

`lst[:-1]→[10,20,30,40]` `lst[::-1]→[50,40,30,20,10]` `lst[1:3]→[20,30]` `lst[:3]→[10,20,30]`

`lst[1:-1]→[20,30,40]` `lst[::-2]→[50,30,10]` `lst[-3:-1]→[30,40]` `lst[3:]→[40,50]`

`lst[::2]→[10,30,50]` `lst[:]→[10,20,30,40,50]` *copie superficielle de la séquence*

Indication de tranche manquante → à partir du début / jusqu'à la fin.

Sur les séquences modifiables (`list`), suppression avec `del lst[3:5]` et modification par affectation `lst[1:4]=[15,25]`

Source: `https://perso.limsi.fr/pointal/python:memento`

Sorbonne Nouvelle

Une nouvelle boucle

Si `lst` est une liste quelconque:
On dispose d'un nouveau type de boucle pour « parcourir » la liste:

```
for x in lst:
    <bloc d'instructions>
```

Une nouvelle boucle

Si `lst` est une liste quelconque:
On dispose d'un nouveau type de boucle pour « parcourir » la liste:

```
for x in lst:
    <bloc d'instructions>
```

```
c = 0
while c < len(lst):
    x = lst[c]
    <bloc d'instructions>
    c += 1
```

Sorbonne Nouvelle

Une nouvelle boucle

Si `lst` est une liste quelconque:
On dispose d'un nouveau type de boucle pour « parcourir » la liste:

```
for x in lst:
    <bloc d'instructions>
```

```
c = 0
while c < len(lst):
    x = lst[c]
    <bloc d'instructions>
    c += 1
```

Principe: la variable `x` va successivement contenir (à chaque tour) la valeur suivante de la liste.
On fera autant de tours de boucle qu'il y a d'éléments dans la liste.