Listes I

Construction d'une liste

Une *liste* est une suite d'éléments.

- `[5,-7,12,99]`
- `["Mars","Avril","Mai"]`
- `[3.14,"pi",10e-3,"x",True]`

- Une liste se définit par des éléments entre crochets :
  — `liste1 = [5,4,3,2,1]`
  — `liste2 = ["Vendredi","Samedi","Dimanche"]`
  — `liste3 = []` la liste vide (très utile pour la compléter plus tard !).

Accéder à un élément

Pour obtenir un élément de la liste, il suffit d'écrire

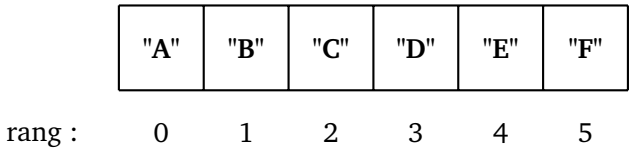
```
liste[i]
```

où i est le rang de l'élément souhaité.

Attention. On commence à compter à partir du rang 0 !

Par exemple après l'instruction `liste = ["A","B","C","D","E","F"]` alors

- `liste[0]` renvoie `"A"`
- `liste[1]` renvoie `"B"`
- `liste[2]` renvoie `"C"`
- `liste[3]` renvoie `"D"`
- `liste[4]` renvoie `"E"`
- `liste[5]` renvoie `"F"`



Ajouter un élément

Pour ajouter un élément à la fin de la liste, il suffit d'utiliser la commande

`maliste.append(element)`

- `premiers = [2,3,5,7]`,
- puis `premiers.append(11)` rajoute 11 à la liste,
- puis `premiers.append(13)` alors maintenant la liste `premiers` vaut `[2,3,5,7,11,13]`.

Une construction

Voici comment construire la liste qui contient les premiers carrés.

```
liste_carres = []           # On part d'un liste vide
for i in range(10):
    liste_carres.append(i**2)   # On ajoute un carré
```

À la fin `liste_carres` vaut :

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Parcourir une liste

- **Longueur d'une liste.** `len(liste)`
  — La liste `[5,4,3,2,1]` est de longueur 5,
  — la liste `["Vendredi","Samedi","Dimanche"]` de longueur 3,
  — la liste vide `[]` de longueur 0.

- **Parcourir une liste.** Voici la façon la plus simple de parcourir une liste (et ici d'afficher chaque élément) :

```
for element in liste:
    print(element)
```

- **Parcourir une liste (bis).** Parfois on a besoin de connaître le rang des éléments. Voici une autre façon de faire (qui affiche ici le rang et l'élément).

```
n = len(liste)
for i in range(n):
    print(i,liste[i])
```

- Pour obtenir une liste à partir de `range()` il faut écrire :

```
list(range(n))
```

Concaténer

- **Concaténer deux listes.** Si on a deux listes, on peut les fusionner par l'opérateur « + ».
  Exemple. `liste1 = [4,5,6]` et `liste2 = [7,8,9]`
  `liste1 + liste2` vaut `[4,5,6,7,8,9]`.

- **Ajouter un élément à la fin.**
  `liste = liste + [element]`
  Exemple. `[1,2,3,4] + [5]` vaut `[1,2,3,4,5]`.
  C'est une méthode alternative à `liste.append(element)`.

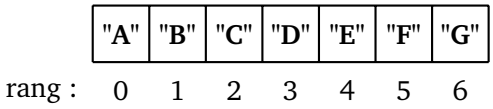
- **Ajouter un élément au début.**
  `liste = [element] + liste`
  Exemple `[5] + [1,2,3,4]` vaut `[5,1,2,3,4]`.

Trancher des listes

On peut extraire d'un seul coup toute une partie de la liste :

`liste[a:b]`

renvoie la sous-liste des éléments de rang a à $b-1$.



Par exemple si `liste = ["A","B","C","D","E","F","G"]` alors

- `liste[1:4]` renvoie `["B","C","D"]`
- `liste[0:2]` renvoie `["A","B"]`
- `liste[4:7]` renvoie `["E","F","G"]`

Il faut encore une fois faire attention à ce que le rang d'une liste commence à 0 et que le tranchage `liste[a:b]` s'arrête au rang $b-1$.

Manipulation des listes

- **Inverser une liste.** Voici trois méthodes :
  — `maliste.reverse()` modifie la liste sur place (c'est-à-dire que `maliste` est maintenant renversée, la commande ne renvoie rien) ;
  — `list(reversed(maliste))` renvoie une nouvelle liste ;
  — `maliste[::-1]` renvoie une nouvelle liste.

- **Supprimer un élément.** La commande `liste.remove(element)` supprime la première occurrence trouvée (la liste est modifiée). Par exemple avec `liste = [2,5,3,8,5]` la commande `liste.remove(5)` modifie la liste qui maintenant vaut `[2,3,8,5]` (le premier 5 a disparu).

- **Supprimer un élément (bis).** La commande `del liste[i]` supprime l'élément de rang i (la liste est modifiée).

Tri

python : `sorted()`

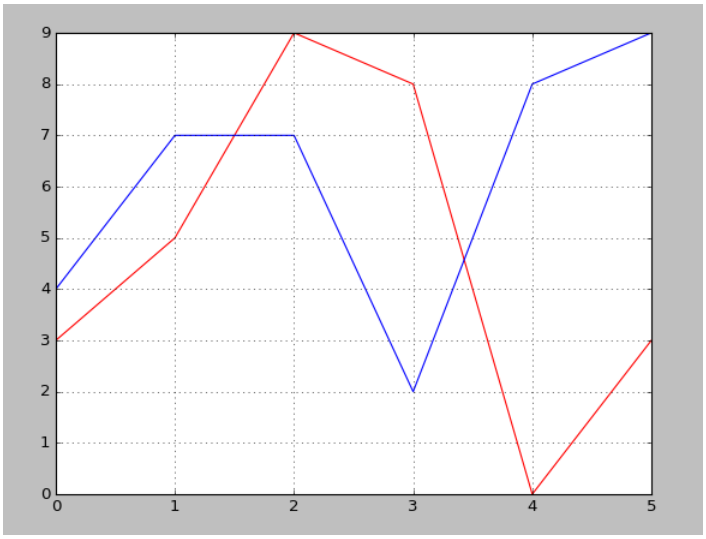
Usage : `sorted(liste)`
Entrée : une liste
Sortie : la liste ordonnée des éléments

Exemple : `sorted([13,11,7,4,6,8,12,6])` renvoie la liste `[4,6,6,7,8,11,12,13]`.

Attention ! Il existe aussi une méthode `liste.sort()` qui modifie la liste sur place.

Visualiser une liste

Avec le module `matplotlib` il est très facile de visualiser les éléments d'une liste de nombres.



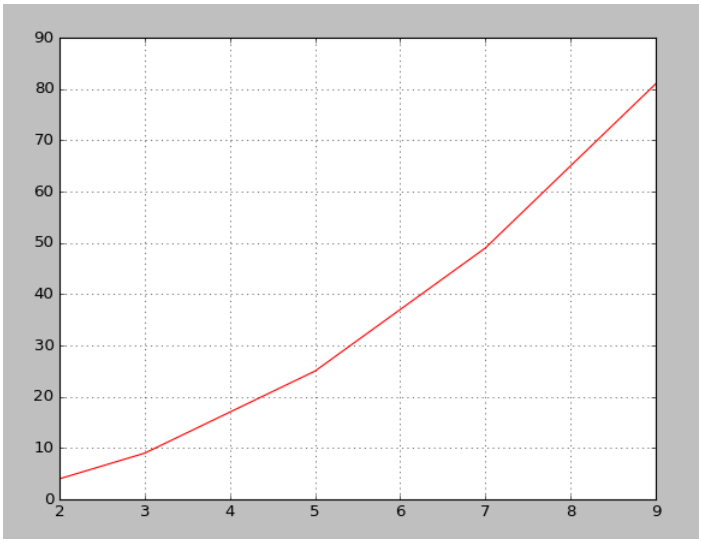

```
import matplotlib.pyplot as plt

liste1 = [3,5,9,8,0,3]
liste2 = [4,7,7,2,8,9]

plt.plot(liste1,color="red")
plt.plot(liste2,color="blue")
plt.grid()
plt.show()
```

Visualiser des points

Pour afficher des points (x_i, y_i) il faut fournir la listes des abscisses puis la listes des ordonnées :

`plt.plot(liste_x,liste_y,color="red")`

Voici un exemple de graphe obtenu en affichant des points de coordonnées du type (x, y) avec $y = x^2$.




```
import matplotlib.pyplot as plt

liste_x = [2, 3, 5, 7, 9]
liste_y = [4, 9,25,49,81]
plt.plot(liste_x,liste_y,color="red")
plt.grid()
plt.show()
```