Les Listes en Python

Semaine 6

Ali Assi, PhD

Une liste est une suite ordonnée d'éléments. Voici un exemple de la façon de les créer:

In [1]:
```
# Exemple 1
paires = [2, 4, 6, 8]
print(paires)
```

```
[2, 4, 6, 8]
```

In [26]:
```
# Exemple 2
couleurs = ['red', 'blue', 'yellow']
print(couleurs)
```

```
['red', 'blue', 'yellow']
```

Une liste peut contenir un mélange de différents types de variables :

In [27]:
```
# Exemple 3
mes_choix = [14, 'Ice cream', 'soccer']
print(mes_choix)
```

```
[14, 'Ice cream', 'soccer']
```

Nous pouvons même faire une liste de listes :

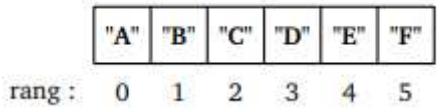
In [28]:
```
# Exemple 4
mains = [['J', 'Q', 'K'], ['2', '2', '2'], ['6', 'A', 'K']]
print(mains)
```

```
[['J', 'Q', 'K'], ['2', '2', '2'], ['6', 'A', 'K']]
```

Indexing

Vous pouvez accéder à des éléments de liste individuels avec des crochets `[]` . On commence à compter à partir du rang 0 !

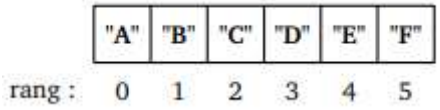
In [6]:
```
# Exemple 5
liste = ["A", "B", "C", "D", "E", "F"]
```



In [7]:
```
# Exemple 6
print(liste[0])
print(liste[1])
print(liste[2])
print(liste[3])
print(liste[4])
print(liste[5])
```

```
A
B
C
D
E
F
```

Les éléments en fin de liste sont accessibles par des nombres négatifs, à partir de `-1` :

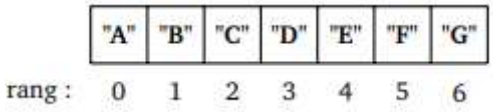


In [4]:
```
# Exemple 7
print(liste[-1])
print(liste[-2])
```

```
F
E
```

Trancher des listes - Slicing

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`.



In [8]:
```
# Exemple 8
liste = ["A", "B", "C", "D", "E", "F", "G"]
print(liste[1:4])
print(liste[0:2])
```

```
['B', 'C', 'D']
['A', 'B']
```

> Les indices de début et de fin sont tous deux **facultatifs**. Si je n'indique pas l'indice de début, il est supposé être égal à `0` . Je pourrais donc réécrire l'expression ci-dessus comme suit :

In [8]:
```
# Exemple 9
print(liste[ :2])
print(liste[1:])
```

```
['A', 'B']
['B', 'C', 'D', 'E', 'F', 'G']
```

L'expression ci-dessus signifie **"donnez-moi toutes les valeurs à partir de l'indice 1"**.

Modification des listes

Ajouter un élément

Pour ajouter un élément à la fin de la liste, il suffit d'utiliser la commande `maliste.append(element)`

In [9]:
```
# Exemple 10
liste.append("H")
liste
```

Out[9]:
```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
```

ou bien

In [32]:
```
# Exemple 11
liste = liste + ["I"]
liste
```

Out[32]:
```
['A', 'B', 'C', 'D', 'E', 'F', 'H', 'I']
```

Ajouter un élément au début. Avec :

In [33]:
```
# Exemple 12
liste = ["0"] + liste
liste
```

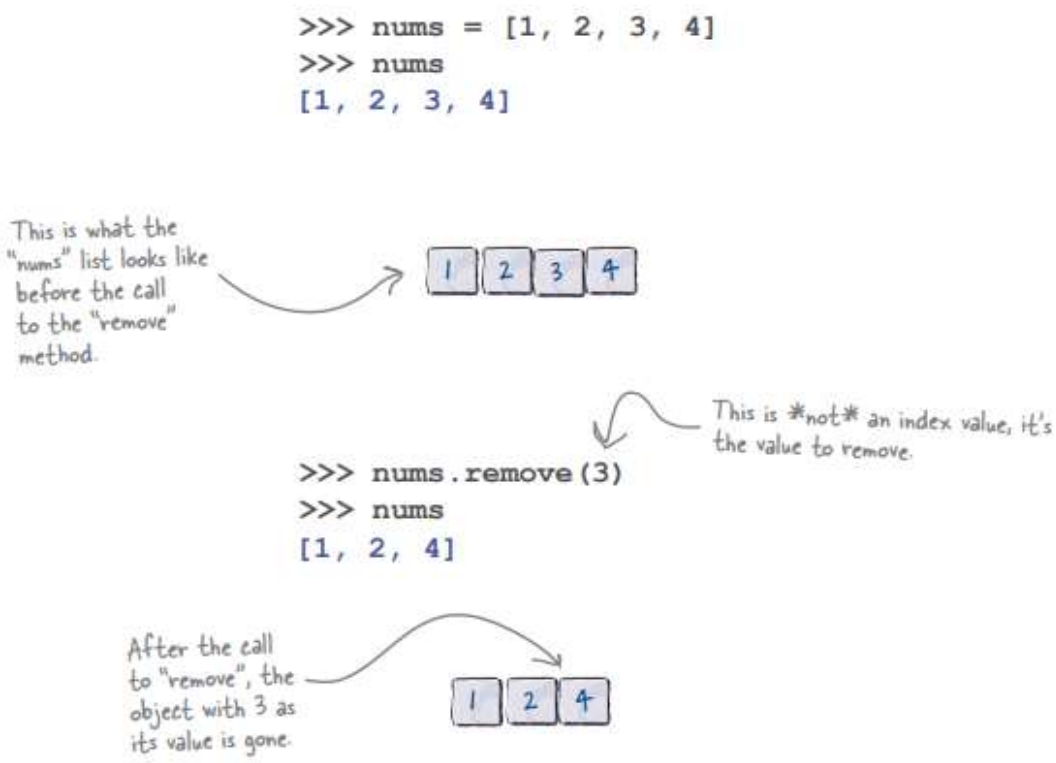
Out[33]:
```
['0', 'A', 'B', 'C', 'D', 'E', 'F', 'H', 'I']
```

Concaténer deux listes. Si on a deux listes, on peut les fusionner par l'opérateur « + ». Par exemple

In [16]:
```
# Exemple 13
liste1 = [4,5,6]
liste2 = [7,8,9, 9]
liste = liste1 + liste2
liste.remove(9)
liste
```

Out[16]:
```
[4, 5, 6, 7, 8, 9]
```

Supprimer un élément. La commande `liste.remove(element)` supprime la première occurrence trouvée (la liste est modifiée).



Les fonctions prédéfinies

Python possède plusieurs fonctions utiles pour travailler avec des listes.

len ()

`len` retourne la longueur d'une liste. La longueur d'une liste est le nombre d'éléments qu'elle contient:

In [17]:
```
# Exemple 13
# Combien de choix y a-t-il ?
liste = ["A", "B", "C", "D", "E", "F", "G"]
longueur = len(liste)
print(longueur)
```

```
7
```

sort

`sort` trie la liste en place en échangeant les valeurs à ses index. Par exemple:

In [19]:
```
# Exemple 14
# Les couleurs triées par ordre alphabétique
liste = ["E", "F", "A",  "C", "D", "B", "G"]
print(liste)
liste.sort()
print(liste)
liste.append()
```

```
['E', 'F', 'A', 'C', 'D', 'B', 'G']
['A', 'B', 'C', 'D', 'E', 'F', 'G']
```

Lorsque nous appelons la fonction `sort` sur une liste, une nouvelle liste n'est pas renvoyée;

count

`count` permet de compter le nombre d'occurrences d'un élément dans une liste. Par exemple:

In [22]:
```
# Exemple 15
# Les couleurs triées par ordre alphabétique
liste = ["E", "F", "A",  "a", "D", "B", "G"]
occurrences = liste.count('A')
print(occurrences)
```

```
1
```

Veuillez noter que cette méthode renvoie `0` si elle reçoit un paramètre non valide ou inexistant. Par exemple:

In [38]:
```
# Exemple 16
# Les couleurs triées par ordre alphabétique
liste = ["E", "F", "A",  "C", "D", "B", "G"]
occurrences = liste.count('H')
print(occurrences)
```

```
0
```

sum

`sum` fait ce à quoi on peut penser:

In [24]:
```
# Exemple 17
paires = [2, -2, 4, 6, 8]
somme = sum(paires)
print(somme)
```

```
18
```

min et max

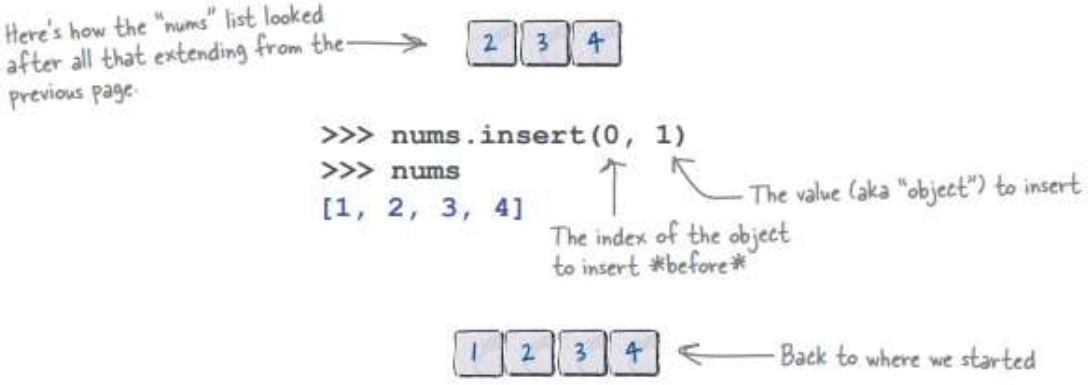
les fonctions min et max pour obtenir le minimum ou le maximum de plusieurs arguments. Mais nous pouvons également passer un seul argument de liste.

In [39]:
```
# Exemple 18
paires = [2, -2, 4, 6, 8]
max_value = max(paires)
min_value = min(paires)
print(max_value)
print(min_value)
```

```
8
-2
```

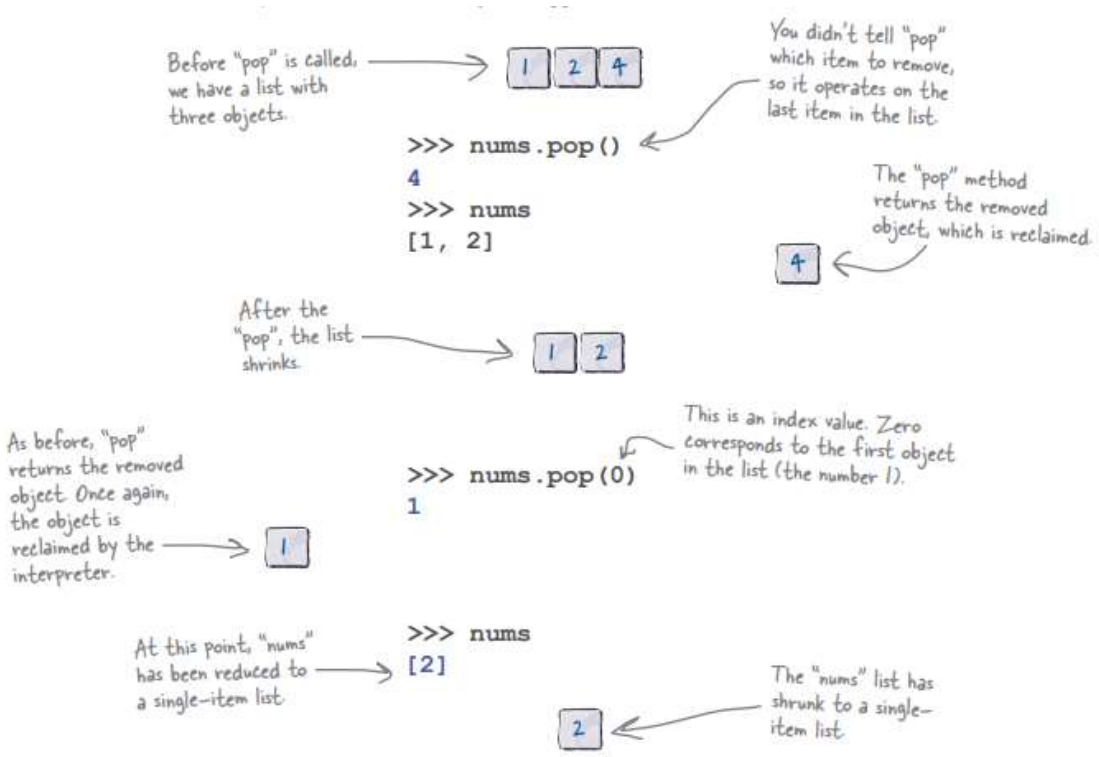
insert

`insert()` permet d'ajouter un élément à l'index donné d'une liste existante. Par exemple:



pop

`list.pop` supprime et renvoie le dernier élément d'une liste :



reverse

`reverse` inverse les éléments d'une liste. Par exemple:

In [40]:
```
# Exemple 19
couleurs = ['red', 'blue', 'yellow', 'Aqua']
couleurs.reverse()
print(couleurs)
```

```
['Aqua', 'yellow', 'blue', 'red']
```

Copier une liste

Utiliser la methode `copy()` pour copier les contenues d'une liste dans une nouvelle liste:

index

Nous pouvons obtenir l'indice pour recherche un élément dans une liste:

In [25]:
```
# Exemple 20
couleurs = ['red', 'blue', 'yellow',  'blue']
idx = couleurs.index('blue')
idx
```

Out[25]:
```
1
```

In [42]:
```
# Exemple 21
couleurs = ['red', 'blue', 'yellow']
idx = couleurs.index('Aqua')
idx
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_7648/1019990370.py in <module>
      1 # Exemple 21
      2 couleurs = ['red', 'blue', 'yellow']
----> 3 idx = couleurs.index('Aqua')
      4 idx

ValueError: 'Aqua' is not in list
```

In [26]:
```
couleurs = ['red', 'blue', 'yellow']
if 'Aqua' in couleurs:
    idx = couleurs.index('Aqua')
    print(idx)
else:
    print('Aqua is not in couleurs')
```

```
Aqua is not in couleurs
```

Pour éviter ce genre de surprises désagréables, nous pouvons utiliser l'opérateur `in` pour déterminer si une liste contient une valeur particulière :

In [35]:
```
# Exemple 21
# Est-ce que Aqua est une couleur ?
idx = 'Aqua' in couleurs
idx
```

Out[35]:
```
False
```

In [43]:
```
# Exemple 22
# Est-ce que red est une couleur ?
idx = 'red' in couleurs
idx
```

Out[43]:
```
True
```

Les boucles - Parcourir une liste

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

In [11]:
```
# Exemple 23
liste = ["A", "B", "C", "D", "E", "F"]
for element in liste:
    print(element)
```

```
A
B
C
D
E
F
```

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).

In [44]:
```
# Exemple 24
n = len(liste)
for i in range(n):
    print(i, liste[i])
```

```
0 E
1 F
2 A
3 C
4 D
5 B
6 G
```

In [34]:
```
liste = ["A", "B", "C", "D", "E", "F"]
print(list(enumerate(liste))) # list of tuples
for idx, value in enumerate(liste, start = 8):
    print(idx, value)
```

```
[(0, 'A'), (1, 'B'), (2, 'C'), (3, 'D'), (4, 'E'), (5, 'F')]
8 A
9 B
10 C
11 D
12 E
13 F
```

In [ ]: