

Listes

Saisie d'une liste

Syntaxe : [objet1, objet2, ...]

Exemple

```
In : joursOuvres = ['lundi', 'mzrdi',  
                    'mercredi', 'jeudy', 'vendredy']
```

```
In : type(joursOuvres)
```

```
Out: list
```

Les listes peuvent être hétérogènes

Les éléments d'une liste peuvent être de types différents.

`int, float, str, bool...`

```
In : maListe = [1, 2.5, 'abc', False]
```

```
In : maListe
```

```
Out: [1, 2.5, 'abc', False]
```

```
In : type(maListe)
```

```
Out: list
```

Les listes peuvent être hétérogènes

Les éléments d'une liste peuvent être de types différents.

`int, float, str, bool...`

```
In : maListe = [1, 2.5, 'abc', False]
In : maListe
Out: [1, 2.5, 'abc', False]
In : type(maListe)
Out: list
```

Exercice ►

Définir une variable `monIdentite` dont la valeur est une liste contenant dans l'ordre : votre nom, votre prénom et votre année de naissance.



Les listes peuvent être hétérogènes

Les éléments d'une liste peuvent être de types différents.

`int, float, str, bool...`

```
In : maListe = [1, 2.5, 'abc', False]
In : maListe
Out: [1, 2.5, 'abc', False]
In : type(maListe)
Out: list
```

Exercice ► Définir une variable `monIdentite` dont la valeur est une liste contenant dans l'ordre : votre nom, votre prénom et votre année de naissance.



```
monIdentite = ["<nom>", "<prénom>", <aaaa>]
```

Les listes peuvent être hétérogènes

Les éléments d'une listes peuvent être eux-mêmes des listes.

Exemple

```
In : maListe2 = ['MARTIN', [14.5, 8.5, 13.0]]
```

```
In : maListe2
```

```
Out: ['MARTIN', [14.5, 8.5, 13.0]]
```

Concaténation

- L1+L2 fabrique la nouvelle liste obtenu par concaténation de L1 suivi de L2.

```
>>> deb=[1,4]
>>> fin=[3,10]
>>> mixte=deb+fin
>>> mixte
[1,4,3,10]
```

- mettre le résultat dans une variable pour le réutiliser.

Concaténation

- $L1+L2$ fabrique la nouvelle liste obtenu par concaténation de $L1$ suivi de $L2$.

```
>>> deb=[1,4]
>>> fin=[3,10]
>>> mixte=deb+fin
>>> mixte
[1,4,3,10]
```

- mettre le résultat dans une variable pour le réutiliser.
- $n*L1$ fabrique la liste obtenue en concaténant $L1$ avec lui-même un nombre entier $n > 0$ de fois.

```
>>> 3*[True]
[True,True,True]
```

Par compréhension

- `[f(k) for k in range(a,b)]` fabrique la liste des images par `f` des entiers entre `a` inclus et `b` exclu

```
>>> essai=[k**2+1 for k in range(1,4)]
```

```
>>> essai
```

```
[2, 5, 10]
```

```
>>> [x-3 for x in essai]
```

```
[-1, 2, 7]
```

Parcours d'une liste

- Directement par les éléments

```
for elt in MaListe:
```

La variable `elt` prend successivement toutes les valeurs de la liste

Parcours d'une liste

- Directement par les éléments

```
for elt in MaListe:
```

La variable `elt` prend successivement toutes les valeurs de la liste

- par les indices

Nombre d'éléments d'une liste

`len(L)` = nombre d'éléments de la liste L

`len` pour *length* = longueur

Exemple

```
In : maListe = [1, 2.5, 'abc', False]
```

```
In : len(maListe)
```

```
Out: 4
```

Nombre d'éléments d'une liste

`len(L)` = nombre d'éléments de la liste `L`

`len` pour *length* = longueur

Exemple

```
In : maListe = [1, 2.5, 'abc', False]
```

```
In : len(maListe)
```

```
Out: 4
```

Exercice ►

```
maListe2 = ['MARTIN', [14.5, 8.5, 13.0]]
```

Combien y a-t-il d'éléments dans la liste `maListe2`?



Nombre d'éléments d'une liste

`len(L)` = nombre d'éléments de la liste `L`

`len` pour *length* = longueur

Exemple

```
In : maListe = [1, 2.5, 'abc', False]
```

```
In : len(maListe)
```

```
Out: 4
```

Exercice ►

```
maListe2 = ['MARTIN', [14.5, 8.5, 13.0]]
```

Combien y a-t-il d'éléments dans la liste `maListe2`?

2 seulement.

La liste à l'intérieur de la liste compte pour un seul élément.



Liste à un élément

- Exemple**

```
In : maListe3 = [42]
```

```
In : type(maListe3)
```

```
Out: list
```

```
In : len(maListe3)
```

```
Out: 1
```

Liste à un élément

- Exemple**

```
In : maListe3 = [42]
```

```
In : type(maListe3)
```

```
Out: list
```

```
In : len(maListe3)
```

```
Out: 1
```

- Liste ayant un seul élément $\neq$ cet élément.**

```
In : maListe3 == 42
```

```
Out: False
```

Liste vide

- **Liste vide = liste à zéro élément**

Syntaxe : `[]`

```
In : maListe4 = []
```

```
In : len(maListe4)
```

Liste vide

- **Liste vide = liste à zéro élément**

Syntaxe : `[]`

```
In : maListe4 = []
```

```
In : len(maListe4)
```

```
Out: 0
```

```
In : type(maListe4)
```

Liste vide

- **Liste vide = liste à zéro élément**

Syntaxe : `[]`

```
In : maListe4 = []
```

```
In : len(maListe4)
```

```
Out: 0
```

```
In : type(maListe4)
```

```
Out: list
```

Liste vide

- **Liste vide = liste à zéro élément**

Syntaxe : `[]`

```
In : maListe4 = []
```

```
In : len(maListe4)
```

```
Out: 0
```

```
In : type(maListe4)
```

```
Out: list
```

- Sert à **initialiser** des variables de type `list` **en vue d'ajouter des éléments** par la suite.

Extraction d'un élément

- **Listes ordonnées** : éléments numérotés par des **indices**.

Attention ! Numérotation à partir de 0 !

```
In : joursOuvres = ['lundi', 'mzrdi',  
                    'mercredi', 'jeudy', 'vendredy']
```

```
In : joursOuvres[0]
```

Extraction d'un élément

- **Listes ordonnées** : éléments numérotés par des **indices**.

Attention ! Numérotation à partir de 0 !

```
In : joursOuvres = ['lundi', 'mzrdi',  
                    'mercredi', 'jeudy', 'vendredy']
```

```
In : joursOuvres[0]  
Out: 'lundi'
```

Extraction d'un élément

- **Listes ordonnées** : éléments numérotés par des **indices**.

Attention ! Numérotation à partir de 0 !

```
In : joursOuvres = ['lundi', 'mzrdi',  
                    'mercredi', 'jeudy', 'vendredy']
```

```
In : joursOuvres[0]  
Out: 'lundi'
```

```
In : jour = joursOuvres[2]
```

```
In : jour
```

Extraction d'un élément

- **Listes ordonnées** : éléments numérotés par des **indices**.

Attention ! Numérotation à partir de 0 !

```
In : joursOuvres = ['lundi', 'mzrdi',  
                    'mercredi', 'jeudy', 'vendredy']
```

```
In : joursOuvres[0]  
Out: 'lundi'
```

```
In : jour = joursOuvres[2]
```

```
In : jour  
Out: 'mercredi'
```

```
In : type(jour)  
Out: str
```

Extraction d'un élément

Exercice ►

Extraire votre prénom de la liste `monIdentite` et stocker le résultat dans une variable `monPrenom`.



Extraction d'un élément

Exercice ► Extraire votre prénom de la liste `monIdentite` et stocker le résultat dans une variable `monPrenom`.



```
monPrenom = monIdentite[1]
```

Indices accessibles

Attention

Ne pas utiliser d'indices trop grands !

```
In : L = ['bal', 'tha', 'zar']
```

```
In : L[3]
```

Indices accessibles

Attention

Ne pas utiliser d'indices trop grands !

```
In : L = ['bal', 'tha', 'zar']
```

```
In : L[3]
```

```
IndexError: list index out of range
```

Indices accessibles

Attention

Ne pas utiliser d'indices trop grands !

```
In : L = ['bal', 'tha', 'zar']
```

```
In : L[3]
```

```
IndexError: list index out of range
```

Exercice ►

Comment extraire le dernier élément d'une liste L, contenant n éléments, à l'aide d'un indice positif ?



Indices accessibles

Attention

Ne pas utiliser d'indices trop grands !

```
In : L = ['bal', 'tha', 'zar']
```

```
In : L[3]
```

```
IndexError: list index out of range
```

Exercice ►

Comment extraire le dernier élément d'une liste L , contenant n éléments, à l'aide d'un indice positif ?



Dans une liste à n éléments, ces éléments sont numérotés par ordre croissant $0, 1, \dots, n - 1$.

Donc : $L[n - 1]$ ou $L[\text{len}(L) - 1]$

Extraction d'une sous-liste

Syntaxe : `liste[indiceDebut:indiceFin]`

Extraction d'une sous-liste

Syntaxe : `liste[indiceDebut:indiceFin]`

Attention Comme pour **range** :

- Numérotation à partir de 0,
- Indice de fin **exclu**.

```
In : joursOuvres[1:3]
```

Extraction d'une sous-liste

Syntaxe : `liste[indiceDebut:indiceFin]`

Attention Comme pour `range` :

- Numérotation à partir de 0,
- Indice de fin **exclu**.

```
In : joursOuvres[1:3]
      # éléments d'indice 1 et 2
Out: ['mzrdi', 'mercredi']
```

```
In : sousliste = joursOuvres[1:3]
In : type(sousliste)
Out: list
```

Exercice ► Extraire une liste contenant vos nom et prénom de la liste `monIdentite`.



Extraction d'une sous-liste

Syntaxe : `liste[indiceDebut:indiceFin]`

Attention Comme pour `range` :

- Numérotation à partir de 0,
- Indice de fin **exclu**.

```
In : joursOuvres[1:3]
      # éléments d'indice 1 et 2
Out: ['mzrdi', 'mercredi']
```

```
In : sousliste = joursOuvres[1:3]
In : type(sousliste)
Out: list
```

Exercice ► Extraire une liste contenant vos nom et prénom de la liste `monIdentite`.



```
monNomEtPrenom = monIdentite[0:2]
```

Extraction de sous-liste

- Indice de début et de fin **non obligatoires**.

à partir de l'indice 3 :

```
In : joursOuvres[3:]  
Out: ['jeudy', 'vendredy']
```

jusqu'à l'indice 3 exclu :

```
In : joursOuvres[:3]  
Out: ['lundi', 'mzrdi',  
      'mercredi']
```

Extraction de sous-liste

- Indice de début et de fin **non obligatoires**.

à partir de l'indice 3 :

```
In : joursOuvres[3:]  
Out: ['jeudy', 'vendredy']
```

jusqu'à l'indice 3 **exclu** :

```
In : joursOuvres[:3]  
Out: ['lundi', 'mzrdi',  
      'mercredi']
```

- Ni indice de début, ni indice de fin
= **extraction de la liste complète**

```
In : joursOuvres[:]  
Out: ['lundi', 'mzrdi', 'mercredi',  
      'jeudy', 'vendredy']
```

Modification des éléments d'une liste

- **Listes = objets modifiables**

Possible de modifier leur contenu après leur création.

Modification des éléments d'une liste

- **Listes = objets modifiables**

Possible de modifier leur contenu après leur création.

- **Modification d'un élément**

```
In : joursOuvres
Out: ['lundi', 'mzrdi', 'mercredi',
      'jeudy', 'vendredy']
```

```
In : joursOuvres[1] = 'mardi'
```

```
In : joursOuvres
Out: ['lundi', 'mardi', 'mercredi',
      'jeudy', 'vendredy']
```

Modification des éléments d'une liste

- **Modifier tout une sous-liste d'un seul coup :**

```
In : joursOuvres[3:] = ['jeudi', 'vendredi']
```

```
In : joursOuvres
```

```
Out: ['lundi', 'mardi', 'mercredi',  
      'jeudi', 'vendredi']
```

```
In : joursOuvres[1:3] = ['soirée', 'matinée',  
                        'journée']
```

```
In : joursOuvres
```

```
Out: ['lundi', 'soirée', 'matinée',  
      'journée', 'jeudi', 'vendredi']
```

Modification des éléments d'une liste

- **Modifier tout une sous-liste d'un seul coup :**

```
In : joursOuvres[3:] = ['jeudi', 'vendredi']
```

```
In : joursOuvres
```

```
Out: ['lundi', 'mardi', 'mercredi',  
      'jeudi', 'vendredi']
```

```
In : joursOuvres[1:3] = ['soirée', 'matinée',  
                        'journée']
```

```
In : joursOuvres
```

```
Out: ['lundi', 'soirée', 'matinée',  
      'journée', 'jeudi', 'vendredi']
```

Exercice ►

Dans la liste `monIdentite`, remplacez l'année de naissance par votre âge.



Modification des éléments d'une liste

- **Modifier tout une sous-liste d'un seul coup :**

```
In : joursOuvres[3:] = ['jeudi', 'vendredi']
```

```
In : joursOuvres
```

```
Out: ['lundi', 'mardi', 'mercredi',  
      'jeudi', 'vendredi']
```

```
In : joursOuvres[1:3] = ['soirée', 'matinée',  
                        'journée']
```

```
In : joursOuvres
```

```
Out: ['lundi', 'soirée', 'matinée',  
      'journée', 'jeudi', 'vendredi']
```

Exercice ►

Dans la liste `monIdentite`, remplacez l'année de naissance par votre âge.

```
monIdentite[2] = <17|18|19>
```



Recopier l'intégralité d'une liste

```
In : joursOuvres = ['lundi', 'mardi', 'mercredi',  
                    'jeudi', 'vendredi']
```

```
In : joursOuvresAnglais = joursOuvres
```

```
In : joursOuvresAnglais[0] = 'monday'
```

```
In : joursOuvresAnglais
```

Recopier l'intégralité d'une liste

```
In : joursOuvres = ['lundi', 'mardi', 'mercredi',  
                    'jeudi', 'vendredi']
```

```
In : joursOuvresAnglais = joursOuvres
```

```
In : joursOuvresAnglais[0] = 'monday'
```

```
In : joursOuvresAnglais  
Out: ['monday', 'mardi', 'mercredi',  
      'jeudi', 'vendredi']
```

```
In : joursOuvres
```

Recopier l'intégralité d'une liste

```
In : joursOuvres = ['lundi', 'mardi', 'mercredi',  
                    'jeudi', 'vendredi']
```

```
In : joursOuvresAnglais = joursOuvres
```

```
In : joursOuvresAnglais[0] = 'monday'
```

```
In : joursOuvresAnglais  
Out: ['monday', 'mardi', 'mercredi',  
      'jeudi', 'vendredi']
```

```
In : joursOuvres  
Out: ['monday', 'mardi', 'mercredi',  
      'jeudi', 'vendredi']
```

→ joursOuvres a été modifié aussi !

Recopier l'intégralité d'une liste

Pourquoi ? Après `joursOuvresAnglais = joursOuvres`,
`joursOuvresAnglais` et `joursOuvres` désignent la
même liste.

Recopier l'intégralité d'une liste

Pourquoi ? Après `joursOuvresAnglais = joursOuvres`,
`joursOuvresAnglais` et `joursOuvres` désignent la
même liste.

Conséquence Toute modification de l'une se répercute sur l'autre.

Recopier l'intégralité d'une liste

Pourquoi ? Après `joursOuvresAnglais = joursOuvres`,
`joursOuvresAnglais` et `joursOuvres` désignent la
même liste.

Conséquence Toute modification de l'une se répercute sur l'autre.

Solution Extraction complète avant affectation :
`joursOuvresAnglais = joursOuvres[:]`

Recopier l'intégralité d'une liste

Solution Extraction complète avant affectation :

joursOuvresAnglais = joursOuvres[:]

```
In : joursOuvres = ['lundi', 'mardi',  
                    'mercredi', 'jeudi',  
                    'vendredi']  
In : joursOuvresAnglais = joursOuvres[:]  
In : joursOuvresAnglais[0] = 'monday'  
In : joursOuvresAnglais
```

Recopier l'intégralité d'une liste

Solution Extraction complète avant affectation :

joursOuvresAnglais = joursOuvres[:]

```
In : joursOuvres = ['lundi', 'mardi',  
                    'mercredi', 'jeudi',  
                    'vendredi']  
In : joursOuvresAnglais = joursOuvres[:]  
In : joursOuvresAnglais[0] = 'monday'  
In : joursOuvresAnglais  
Out: ['monday', 'mardi', 'mercredi', 'jeudi',  
      'vendredi']  
In : joursOuvres
```

Recopier l'intégralité d'une liste

Solution Extraction complète avant affectation :

joursOuvresAnglais = joursOuvres[:]

```
In : joursOuvres = ['lundi', 'mardi',  
                    'mercredi', 'jeudi',  
                    'vendredi']  
  
In : joursOuvresAnglais = joursOuvres[:]  
In : joursOuvresAnglais[0] = 'monday'  
In : joursOuvresAnglais  
Out: ['monday', 'mardi', 'mercredi', 'jeudi',  
      'vendredi']  
  
In : joursOuvres  
Out: ['lundi', 'mardi', 'mercredi', 'jeudi',  
      'vendredi']
```

Recopier l'intégralité d'une liste

Solution Extraction complète avant affectation :

joursOuvresAnglais = joursOuvres[:]

```
In : joursOuvres = ['lundi', 'mardi',  
                    'mercredi', 'jeudi',  
                    'vendredi']  
  
In : joursOuvresAnglais = joursOuvres[:]  
In : joursOuvresAnglais[0] = 'monday'  
In : joursOuvresAnglais  
Out: ['monday', 'mardi', 'mercredi', 'jeudi',  
      'vendredi']  
  
In : joursOuvres  
Out: ['lundi', 'mardi', 'mercredi', 'jeudi',  
      'vendredi']
```

À retenir

Toujours `maCopie = maListe[:]`

Jamais `maCopie = maListe`.

Modification

- **Ajout d'un élément à la fin : méthode append**

```
In : jours = ['lundi', 'mardi']
```

```
In : jours
```

```
Out: ['lundi', 'mardi']
```

```
In : jours.append('mercredi')
```

```
In : jours
```

```
Out: ['lundi', 'mardi', 'mercredi']
```

Modification

- **Ajout d'un élément à la fin : méthode append**

```
In : jours = ['lundi', 'mardi']  
In : jours  
Out: ['lundi', 'mardi']  
  
In : jours.append('mercredi')  
  
In : jours  
Out: ['lundi', 'mardi', 'mercredi']
```

- **Appel d'une méthode :** `nomListe.append(argument1)`

Attention Ne pas utiliser `L1 = L1 + [a]` à la place de `L1.append(a)` ! Même résultat mais coût en temps et mémoire très différents !

Exercice ► Écrire une fonction d'en-tête $u(a, n)$ qui renvoie les termes $u_0, u_1, \dots, u_n$ de la suite récurrente vérifiant $u_0 = a$ et $u_{n+1} = \frac{u_n}{1+u_n}$.

Exercice ► Écrire une fonction d'en-tête $begaie(L)$ qui renvoie une liste où chaque objet de L est répété deux fois :

```
In : shadoks = ['ga', 'bu', 'zo', 'meu']  
In : begaie(shadoks)  
Out: ['ga', 'ga', 'bu', 'bu', 'zo', 'zo',  
      'meu', 'meu']
```