BAC Pro CIEL	Python – Module 6 Les listes	Année 2025/2026

A. Pourquoi utiliser des listes

Les listes permettent de stocker plusieurs valeurs comme des données de capteur, des adresses IP, des utilisateurs, etc...

B. Créer une liste

Pour créer une liste on doit :

- ⇨ Définir un nom de variable
- ⇨ Mettre les éléments de la liste entre crochet « [] »
- ⇨ Séparer les éléments de la liste par des virgules « , »

```
nombre = [1,2,3,4]
fruits = ["pomme","banane","cerise"]
vide = []
```

C. Accéder aux éléments

```
fruits = ["pomme","banane","cerise","poire","citron","melon"]
print(fruits[0])
print(fruits[3])
```

Résultat :

```
pomme
poire
```

D. Modifier une liste

```
fruits = ["pomme","banane","cerise","poire","citron","melon"]
print(fruits)
fruits[1] = "kiwi"
print(fruits)
```

Résultat :

```
['pomme', 'banane', 'cerise', 'poire', 'citron', 'melon']
['pomme', 'kiwi', 'cerise', 'poire', 'citron', 'melon']
```

E. Ajouter des éléments

1) append()

```
fruits = ["pomme","banane","cerise","poire","citron","melon"]
print(fruits)
fruits.append("orange")
print(fruits)
```

Résultat :

```
['pomme', 'banane', 'cerise', 'poire', 'citron', 'melon']
['pomme', 'banane', 'cerise', 'poire', 'citron', 'melon', 'orange']
```

2) insert()

```
fruits = ["pomme","banane","cerise","poire","citron","melon"]
print(fruits)
fruits.insert(2,"kiwi")
print(fruits)
```

Résultat :

```
['pomme', 'banane', 'cerise', 'poire', 'citron', 'melon']
['pomme', 'banane', 'kiwi', 'cerise', 'poire', 'citron', 'melon']
```

F. Supprimer des éléments

1) remove()

Utilisé lorsque l'on connait la donnée à supprimer.

```
fruits = ["pomme","banane","cerise","poire","citron","melon"]
print(fruits)
fruits.remove("cerise")
print(fruits)
```

Résultats :

```
['pomme', 'banane', 'cerise', 'poire', 'citron', 'melon']
['pomme', 'banane', 'poire', 'citron', 'melon']
```

2) pop()

Utilisé pour supprimer le dernier élément de la liste.

```
fruits = ["pomme","banane","cerise","poire","citron","melon"]
print(fruits)
fruits.pop()
print(fruits)
```

Résultat :

```
['pomme', 'banane', 'cerise', 'poire', 'citron', 'melon']
['pomme', 'banane', 'cerise', 'poire', 'citron']
```

3) del

Utilisé pour supprimer un élément de la liste avec son index.

```
fruits = ["pomme","banane","cerise","poire","citron","melon"]
print(fruits)
del fruits[2]
print(fruits)
```

Résultat :

```
['pomme', 'banane', 'cerise', 'poire', 'citron', 'melon']
['pomme', 'banane', 'poire', 'citron', 'melon']
```

G. Parcourir une liste avec for

```
fruits = ["pomme","banane","cerise","poire","citron","melon"]
for f in fruits :
    print("J'aime ce fruit : ",f)
```

Résultat :

```
J'aime ce fruit :  pomme
J'aime ce fruit :  banane
J'aime ce fruit :  cerise
J'aime ce fruit :  poire
J'aime ce fruit :  citron
J'aime ce fruit :  melon
```

H. Taille d'une liste

```
fruits = ["pomme","banane","cerise","poire","citron","melon"]
print(len(fruits))
```

Résultat :

```
6
```

I. Vérifier la présence d'un élément

```
fruits = ["pomme","banane","cerise","poire","citron","melon"]
if "poire" in fruits :
    print("Fruit trouvé")
else :
    print("Fruit non trouvé")
```

Résultat :

```
Fruit trouvé
```

J. Retrouver l'index d'un élément

La méthode index() permet de retrouver la position d'un élément dans une liste.

```
fruits = ["pomme","banane","cerise","poire","citron","melon"]
position = fruits.index("poire")
print(position)
```

Résultat :

```
3
```

Attention, si l'élément n'existe pas dans la liste, Python va générer une erreur. Il est fortement recommandé de chercher au préalable la présence du mot avant de récupérer son index.

La recherche d'index est très utile pour les listes parallèles.

Par exemple :

```
fruits = ["pomme","banane","cerise","poire","citron","melon"]
quantite = [5,12,8,3,2,1]
position = fruits.index("poire")
print(f"Il y a {quantite[position]} {fruits[position]}")
```

Résultat :

```
Il y a 3 poire
```

K. Trier une liste

```
notes = [12,5,18,9,11,6,1,15,13]
notes.sort()
print(notes)
```

Résultat :

```
[1, 5, 6, 9, 11, 12, 13, 15, 18]
```

L. Erreurs fréquentes

Index hors liste

```
fruits = ["pomme","banane"]
print(fruits[5])
```

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

Oublier les crochets

```
fruits = "pomme","banane"
```

⇨ Python crée un tuple et non une liste

Confondre append() et insert :

```
fruits = ["pomme","banane"]
fruits.append("kiwi")
```

ajoute à la fin uniquement

Modifier un index inexistant

```
fruits = ["pomme","banane"]
fruits[10] = "kiwi"
```

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

Utiliser une parenthèse au lieu des crochets

```
fruits = ["pomme","banane"]
print(fruits(0))
```

```
TypeError: 'list' object is not callable
```

M.A retenir

une liste stocke plusieurs valeurs

les listes utilisent les crochets []

les index commencent à 0

append() ajoute un élément

remove() supprime un élément

len() donne la taille

for permet de parcourir une liste

une liste peut être modifiée

N. Synthèse