L3 TP AIS — PYTHON

Séance 2 · Listes et boucles

Fil rouge : auditer un parc de serveurs · Durée indicative : 2 h

Du serveur au parc de serveurs

À la séance 1, un script examinait une seule machine. Dans une infrastructure, on doit souvent appliquer la même vérification à plusieurs serveurs : une liste et une boucle permettent de le faire sans recopier le code.

Séance 1	Séance 2	Vers la séance 3
un serveur	plusieurs serveurs	organiser le code et lire un inventaire

Objectifs

- créer et parcourir une liste ;
- utiliser une boucle for et une boucle while ;
- compter des résultats avec un compteur ;
- ajouter un élément à une liste ;
- produire un bilan simple d'audit.

Fil rouge — Nous allons transformer audit_serveur.py en un outil qui vérifie une petite liste de machines.

1. Les listes

Une liste stocke plusieurs valeurs dans un ordre donné. Les éléments sont séparés par des virgules, entre crochets.

```
serveurs = ["web-01", "db-01", "api-01"]
print(serveurs)
print(len(serveurs))
```

Accéder à un élément

```
print(serveurs[0])  # web-01
print(serveurs[1])  # db-01
```

Attention — Le premier élément porte l'indice 0. Une liste de 3 éléments a les indices 0, 1 et 2.

Manipuler une liste

```
serveurs.append("cache-01")
serveurs[0] = "web-prod-01"
```

Action	Exemple
ajouter à la fin	serveurs.append("cache-01")
modifier un élément	serveurs[0] = "web-prod-01"
connaître la taille	len(serveurs)

2. Répéter avec for

La boucle for parcourt une collection élément par élément. À chaque tour, la variable serveur prend la valeur d'un élément de la liste.

```
serveurs = ["web-01", "db-01", "api-01"]

for serveur in serveurs:
    print("Audit de", serveur)
```

Lire le déroulé

Tour	Valeur de serveur	Affichage
1	"web-01"	Audit de web-01
2	"db-01"	Audit de db-01
3	"api-01"	Audit de api-01

Exercice rapide

Créez une liste de trois services (par exemple ssh, nginx, postgresql), puis affichez « Vérification de … » pour chacun.

Indentation — Comme avec if, le bloc qui se répète est indenté de 4 espaces.

3. Compter et synthétiser

Un compteur est une variable numérique que l'on met à jour au fil de la boucle. Initialisez-le avant la boucle.

Compter les éléments avec len()

Quand une liste est déjà complète, len(...) donne directement son nombre d'éléments :

```
serveurs = ["web-01", "db-01", "api-01"]
nb_serveurs = len(serveurs)
print(nb_serveurs)  # 3
```

À distinguer — len(serveurs) compte tous les éléments de la liste. Un compteur permet de compter seulement certains événements, par exemple les serveurs en alerte.

Créer un compteur

```
serveurs = ["web-01", "db-01", "api-01"]
nb_audites = 0

for serveur in serveurs:
    print("Audit de", serveur)
    nb_audites = nb_audites + 1

print("Serveurs audités :", nb_audites)
```

Compter des alertes

```
charges = [0.35, 0.92, 0.61]
nb_alertes = 0

for charge in charges:
    if charge > 0.85:
        nb_alertes = nb_alertes + 1
```

Piège courant — Si nb_alertes = 0 est placé dans la boucle, le compteur repart à zéro à chaque tour.

4. La boucle while

while répète un bloc tant qu'une condition est vraie. Elle est utile quand on ne connaît pas à l'avance le nombre de répétitions.

```
tentative = 1

while tentative <= 3:
    print("Tentative", tentative)
    tentative = tentative + 1
```

Règle essentielle

La condition doit évoluer dans le bloc. Sinon, la boucle peut ne jamais s'arrêter : c'est une boucle infinie.

Exercice rapide

Écrivez une boucle qui affiche les numéros de contrôle de 1 à 5.

Choisir la bonne boucle — for : parcourir une liste connue. while : répéter tant qu'une condition reste vraie.

5. TP guidé · Audit d'un parc

Contexte : vous disposez de deux listes alignées. L'élément d'indice 0 de charges correspond au serveur d'indice 0 de serveurs.

```
serveurs = ["web-01", "db-01", "api-01", "cache-01"]
charges = [0.35, 0.92, 0.61, 0.88]
```

Étape 1 — Préparer le bilan

1. Créer nb_audites et nb_alertes, initialisés à 0.
2. Créer une liste vide serveurs_en_alerte.
3. Afficher un titre : « Début de l'audit du parc ».

Étape 2 — Parcourir les indices

Utilisez range(len(serveurs)) pour obtenir les indices 0, 1, 2, 3. À chaque tour, récupérez le nom et la charge correspondants.

```
for indice in range(len(serveurs)):
    serveur = serveurs[indice]
    charge = charges[indice]
```

TP guidé · suite

Étape 3 — Décider

4. Afficher le nom du serveur et sa charge en pourcentage.
5. Si la charge est supérieure à 0.85, afficher « ALERTE », augmenter nb_alertes et ajouter le serveur à serveurs_en_alerte.
6. Sinon, afficher « OK ».
7. Augmenter nb_audites à chaque tour.

Étape 4 — Produire le bilan

Après la boucle, afficher le nombre de serveurs audités, le nombre d'alertes et la liste des serveurs en alerte.

Attendu pour les données fournies	Valeur
serveurs audités	4
alertes	2
serveurs en alerte	db-01, cache-01

Indice — Pour ajouter un serveur à la liste : serveurs_en_alerte.append(serveur).

6. Défi · Saisie contrôlée

Demandez un identifiant jusqu'à ce que l'utilisateur saisisse admin. Comptez le nombre de tentatives.

Contraintes

- Utiliser une boucle while.
- Initialiser le compteur avant la boucle.
- Afficher « Accès accordé » à la fin.
- Bonus : arrêter après 3 essais et afficher « Compte verrouillé ».

Défi bonus · Inventaire évolutif

Partir d'une liste vide. Demander trois noms de serveurs à l'utilisateur et les ajouter avec append(), puis les afficher avec for.

À expliquer — Pourquoi une boucle est-elle plus adaptée que trois instructions input() recopiées ?

7. Travail personnel

Exercice 1 — Ports à vérifier

Parcourez [22, 80, 443, 8080]. Affichez « service web » pour 80 ou 443, sinon « autre port ».

Exercice 2 — Disques

Parcourez des pourcentages d'espace libre. Comptez les valeurs strictement inférieures à 15 et affichez le total.

Exercice 3 — Tentatives

Avec while, demander un code jusqu'à obtenir « AIS2026 ». Compter les essais.

Débogage

Ce programme doit compter les charges élevées. Repérez au moins trois erreurs.

```
charges = [0.2, 0.91, 0.87]
nb_alertes = 0

for charge in charges
    if charge > 0.85:
        nb_alertes = 0

print(nb_alertes)
```

Pour la séance 3 — Nous découperons l'audit en fonctions et lirons un inventaire depuis un fichier. Conservez votre script audit_parc.py.