# TD5 – Structures itératives
# `for... , while …`

## Table des matières

# 1 - Exercice 1 : Programme adresse IP

Une adresse IP est composée de 4 octets, chacun étant un entier compris entre 0 et 255.

### Que fait le programme suivant ?

Un stagiaire a écrit ce programme. Décrivez ce qu'il fait dans son ensemble puis étape par étape.

```
a = input("Entrez une adresse IP (format : xxx.xxx.xxx.xxx) : ")
b = a.split('.')
if len(b) != 4:
    print("La chaîne doit comporter 4 parties.")
else:
    c = 0
    for d in b:
        c += int(d)
    print(f"Le résultat final est : {c}")
```

**Modifiez ce programme pour qu'il respecte les bonnes pratiques de programmation en Python (nommage des variables, commentaires utiles...)**

# 2 - Exercice 2 : Manipulation de séquences

Vous disposez d'une liste de fruits. Le but est de manipuler cette liste avec des boucles `for` pour différentes opérations.

**Questions :**

1. Écrire un programme qui contient la liste suivante : `fruits = ['pomme', 'banane', 'orange', 'kiwi', 'raisin']`.
2. Utilisez une boucle `for` pour afficher chaque fruit avec son numéro dans la liste (exemple : `1. pomme`).
3. Ajoutez une fonction pour demander à l'utilisateur d'entrer un fruit, puis vérifiez si ce fruit est dans la liste des fruits.
4. Si le fruit est dans la liste, affichez "Fruit trouvé", sinon affichez "Fruit non trouvé".

```
1. pomme
2. banane
3. orange
4. kiwi
5. raisin
Entrez le nom d'un fruit à chercher : kiwi
Fruit trouvé
```

*Figure 1: Fruit existant dans la liste*

```
1. pomme
2. banane
3. orange
4. kiwi
5. raisin
Entrez le nom d'un fruit à chercher : poire
Fruit non trouvé
```

*Figure 2: Le fruit saisi n'est pas dans la liste*

## 3 - Exercice 3 : Simulation d'un mot de passe sécurisé

Dans cet exercice, vous allez simuler la création d'un mot de passe sécurisé. Le mot de passe doit avoir une longueur minimale de 8 caractères, contenir au moins une lettre majuscule et un chiffre.

**Questions :**

Écrire un programme qui demande à l'utilisateur de saisir un mot de passe et qui vérifie que celui-ci respecte les règles de sécurité suivantes :

1. Le mot de passe doit contenir au moins 8 caractères.
2. Le mot de passe doit contenir au moins une lettre majuscule.
3. Le mot de passe doit contenir au moins un chiffre.

## Consignes :

- Si le mot de passe ne respecte pas l'une des conditions, le programme doit afficher un message indiquant pourquoi le mot de passe est invalide et redemander un nouveau mot de passe.
- Le programme s'arrête lorsque le mot de passe respecte toutes les règles et affiche alors le message : "Mot de passe accepté".
- Utilisez des conditions pour vérifier chacune des règles (longueur du mot de passe, présence d'une majuscule, présence d'un chiffre).

## Exemple de sortie attendue :

```
Entrez un mot de passe : 1234567
Mot de passe non valide car il doit contenir au moins 8 caractères.
Mot de passe non valide car il doit contenir au moins une majuscule.

Entrez un nouveau mot de passe : abcd
Mot de passe non valide car il doit contenir au moins 8 caractères.
Mot de passe non valide car il doit contenir au moins une majuscule.
Mot de passe non valide car il doit contenir au moins un chiffre.

Entrez un nouveau mot de passe : Abcdefghi
Mot de passe non valide car il doit contenir au moins un chiffre.

Entrez un nouveau mot de passe : Abcdefgh1
Mot de passe accepté
```

## Pour aller plus loin

# 4 - Projet « Gestion d'une liste de fruits »

Dans ce projet, nous allons reprendre la gestion d'une liste de fruits (exercice 2) avec les fonctionnalités suivantes :

1. Ajouter un fruit à la liste.
2. Supprimer un fruit de la liste (si présent).
3. Vérifier si un fruit est dans la liste.
4. Terminer le programme en entrant une chaîne spéciale (ex : -1).

### Consignes :

1. Étape 1 : Afficher le menu des options possibles.
2. Étape 2 : En fonction du choix de l'utilisateur, exécuter l'action appropriée (ajout, suppression, vérification ou fin du programme).
3. Étape 3 : Répéter l'affichage du menu tant que l'utilisateur n'a pas demandé la fin du programme.

### Exemple 1 d'exécution :

```
Liste actuelle des fruits : ['pomme', 'banane', 'orange']
1. Ajouter un fruit
2. Supprimer un fruit
3. Vérifier si un fruit est dans la liste
4. Terminer
Choix : 1
Entrez le fruit à ajouter : mangue
mangue a été ajouté à la liste.

Liste actuelle des fruits : ['pomme', 'banane', 'orange', 'mangue']
1. Ajouter un fruit
2. Supprimer un fruit
3. Vérifier si un fruit est dans la liste
4. Terminer
Choix : 2
Entrez le fruit à supprimer : poire
poire n'est pas dans la liste.

Liste actuelle des fruits : ['pomme', 'banane', 'orange', 'mangue']
1. Ajouter un fruit
2. Supprimer un fruit
3. Vérifier si un fruit est dans la liste
4. Terminer
Choix : 2
Entrez le fruit à supprimer : pomme
pomme a été supprimé de la liste.

Liste actuelle des fruits : ['banane', 'orange', 'mangue']
1. Ajouter un fruit
2. Supprimer un fruit
3. Vérifier si un fruit est dans la liste
4. Terminer
Choix :
```

**Exemple 2 d'exécution :**

```
Liste actuelle des fruits : ['banane', 'orange', 'mangue']
1. Ajouter un fruit
2. Supprimer un fruit
3. Vérifier si un fruit est dans la liste
4. Terminer
Choix : 3
Entrez le fruit à vérifier : poire
'poire' n'est pas dans la liste.

Liste actuelle des fruits : ['banane', 'orange', 'mangue']
1. Ajouter un fruit
2. Supprimer un fruit
3. Vérifier si un fruit est dans la liste
4. Terminer
Choix : 3
Entrez le fruit à vérifier : orange
'orange' est bien dans la liste.

Liste actuelle des fruits : ['banane', 'orange', 'mangue']
1. Ajouter un fruit
2. Supprimer un fruit
3. Vérifier si un fruit est dans la liste
4. Terminer
Choix : 4
Voulez-vous vraiment quitter ? (o/n) : n
Retour au menu.

Liste actuelle des fruits : ['banane', 'orange', 'mangue']
1. Ajouter un fruit
2. Supprimer un fruit
3. Vérifier si un fruit est dans la liste
4. Terminer
Choix : 4
Voulez-vous vraiment quitter ? (o/n) : o
Fin du programme.
```

# Petit mémento sur des instructions utiles

## 1- Entrées et Sorties

- **`input()`**
  `input("message")`
  Demande une saisie de l'utilisateur sous forme de chaîne de caractères. Le texte entre guillemets est un message optionnel affiché à l'utilisateur.

- **`print()`**
  `print(objet)`
  Affiche l'objet passé en paramètre (texte, variable, etc.) à l'écran. Peut accepter plusieurs objets séparés par des virgules.

## 2- Manipulation de Chaînes

- **`split()`**
  `chaine.split(séparateur)`
  Sépare la chaîne de caractères en sous-chaînes selon le séparateur fourni (par défaut un espace). Retourne une liste des sous-chaînes.

- **`len()`**
  `len(objet)`
  Renvoie le nombre d'éléments dans un objet (longueur d'une chaîne, nombre d'éléments dans une liste, etc.).

- **`lower()`**
  `chaine.lower()`
  Convertit tous les caractères de la chaîne en minuscules.

## 3- Structures alternatives

```
- if condition:
      bloc_d_instructions
  elif autre_condition:
      bloc_d_instructions_alternative
  else:
      bloc_d_instructions_alternative
```

`if`
Vérifie si la condition est vraie. Si oui, exécute les instructions du bloc. Peut être suivi de `elif` ou `else` pour tester d'autres conditions.

`Elif`
Permet de tester une nouvelle condition si la première est fausse

`Else`
S'exécute lorsque les condition du `if` et `elif` (si elle existe) sont fausses.

## 4- Structures itératives

- ```
  for élément in séquence:
      bloc_d_instructions
  else:
      bloc_d_instructions_alternative
  ```
  Parcourt chaque élément d'une séquence (liste, chaîne, etc.). Si la boucle se termine sans qu'un `break` soit rencontré, le bloc `else` est exécuté.

- ```
  for élément in séquence:
      bloc_d_instructions
      if condition_rencontree:
          break
  else:
      bloc_si_pas_de_break
  ```
  Le bloc `else` s'exécute uniquement si la boucle se termine normalement, c'est-à-dire sans rencontrer d'instruction `break`. Si un `break` est utilisé pour quitter la boucle, le bloc `else` est ignoré.

- ```
  while condition:
      bloc_d_instructions
  ```
  Répète les instructions tant que la condition est vraie.

## 5- Conversions et types

- **`int()`**
  `int(valeur)`
  Convertit la valeur donnée en un entier. La valeur doit être un nombre ou une chaîne contenant un nombre entier.
- **`str()`**
  `str(valeur)`
  Convertit la valeur en chaîne de caractères.

## 6- Opérations sur les listes

- **`append()`**
  `liste.append(élément)`
  Ajoute un élément à la fin de la liste.
- **`remove()`**
  `liste.remove(élément)`
  Supprime la première occurrence de l'élément donné dans la liste. Génère une erreur si l'élément n'existe pas.
- **`in`**
  `élément in séquence`
  Vérifie si un élément est présent dans une séquence (liste, chaîne, etc.). Renvoie `True` si l'élément est trouvé, sinon `False`.
  ```

## 7- Fonctions sur les caractères et les séquences

- `isupper()`
  `caractère.isupper()`
  Renvoie `True` si le caractère est une lettre majuscule, sinon `False`.

- `isdigit()`
  `caractère.isdigit()`
  Renvoie `True` si le caractère est un chiffre, sinon `False`.

- `enumerate(iterable, start=0)`
  `iterable` : une séquence (comme une liste, un tuple, une chaîne de caractères, etc.).
  `start` (optionnel) : l'indice à partir duquel commence la numérotation (par défaut 0).

  `enumerate` renvoie un objet qui produit des paires (index, élément) lorsque vous itérez sur une séquence. Cela permet de parcourir les éléments d'une séquence tout en ayant accès à leur index.

  **Exemple**

```
mot = "Python"
for index, lettre in enumerate(mot, start=1):
    print(f"Lettre {index}: {lettre}")
```

  **Résultat**

```
Lettre 1: P
Lettre 2: y
Lettre 3: t
Lettre 4: h
Lettre 5: o
Lettre 6: n
```

## 8- Formats de chaînes

- `F-string`
  `f"texte {variable}"`
  Insère la valeur d'une variable ou d'une expression directement dans une chaîne de caractères. Le préfixe `f` est utilisé avant les guillemets.

## 9- Système

- `break`
  Interrompt l'exécution d'une boucle (`for` ou `while`) et passe à l'instruction suivante en dehors de la boucle.