PROGRAMMATION PYTHON

— PROGRAMMATION ORIENTÉE-OBJET —

Principe

- Définir des entités (ex: Personne)
  => **Classe**

- Chaque entité peut avoir:
  - Des données qui lui sont propres (ex: nom, age)
    => **Variables d'instance**

  - Des données communes avec les autres entités du même type (ex: infos_etre_vivant)
    => **Variable de classe**

  - Des actions à effectuer (ex: SePresenter)
    => **Méthodes**

- La **classe** correspond à la définition, c'est "le modèle" et on pourra créer plusieurs objets à partir de celui-ci.

- On peut instancier la classe, ce qui permet de créer des objet. Pour cela on passe par le **constructeur**.

Exemple

Définition de la classe Personne qui possède un nom (variable d'instance), un constructeur prenant en paramètre le nom, et une méthode SePresenter.

```
# --- DEFINITION ---
class Personne:
    def __init__(self, nom):
        self.nom = nom   # crée une variable d'instance : nom
        print("Constructeur personne " + nom)

    def SePresenter(self):
        print("Bonjour, je m'appelle " + self.nom)
```

Instanciation de la classe Personne, en passant "Jean" en paramètre du constructeur, pour obtenir l'objet personne1. Puis appel de la méthode SePrésenter sur l'objet personne1

```
# --- UTILISATION ---
personne1 = Personne("Jean")   # Je cree une personne
personne1.SePresenter()
```

Important

- **Le constructeur** s'écrit toujours __init__(self)

- **self** : ce mot clef veut dire "moi-même". Il correspond à l'objet lui-même.
  - Quand on fait : personne1.SePresenter(), alors le self dans la méthode SePresenter sera personne1.

- Créez toujours vos **variables d'instance** à l'intérieur du constructeur.

Variable de classe

Une variable de classe va s'appliquer à "l'ensemble du groupe".

Exemple :

```
class Personne():
    ESPECE_ETRE_VIVANT = "Humain (Mammifère Homo sapiens)"   # variable de classe (1 pour toutes les Personnes)
```

Toute personne aura accès à cette variable commune : Personne.ESPECE_ETRE_VIVANT

Au contraire, qu'une variable d'instance est unique pour chaque objet (par exemple chaque personne possède un nom différent).

Héritage

L'héritage permet de "récupérer" le code d'une autre classe et d'en bénéficier.

Par exemple ici la classe Chat hérite de la classe EtreVivant.

- On dit que EtreVivant est la classe parent de la classe Chat.
- Ou que la classe Chat est enfant de la classe EtreVivant.

```
class EtreVivant:
    ESPECE_ETRE_VIVANT = "(être vivant non identifié)"

    def AfficherInfosEtreVivant(self):
        print("Info être vivant : " + self.ESPECE_ETRE_VIVANT)

class Chat(EtreVivant):
    ESPECE_ETRE_VIVANT = "Chat (Mammifère félin)"

chat = Chat()
chat.AfficherInfosEtreVivant()
```

Ici la méthode AfficherInfosEtreVivant n'est pas définie dans la classe Chat, mais elle a été "récupérée" par l'héritage.

- Super() correspond à l'objet "parent"