

# PYTHON № 10 : INVITATION AUX CLASSES

---

## 1 une Classe, c'est quoi (vision simplifiée) ?

### 1.1 explication

- c'est 1 **type** d'objet, qui a des **attributs** et des **méthodes**
- ce n'est pas obligatoire mais 3 choses sont intéressantes à faire :
  - mettre un **constructeur `__init__`** : c'est 1 méthode particulière qui permet de créer 1 instance de la classe
  - créer 1 méthode **`del`** qui permet de supprimer les instances inutiles
  - renseigner la méthode **`help()`** qui permet d'obtenir des explications sur la classe
- ensuite, on ajoute des attributs et des méthodes pour travailler
- attribut et méthode peuvent être globale (pour la classe) ou locale (pour l'instance)

### 1.2 exemple de classe

---

```
1 # création d'une classe Personne
2 class Personne:
3     """Classe définissant une personne caractérisée par :
4         - son nom
5         - son prénom
6         - son âge
7         - son lieu de résidence"""
8
9     def __init__(self): # Notre méthode constructeur
10        """Constructeur de notre classe. Chaque attribut va être instancié
11        avec une valeur par défaut... original"""
12
13        self.nom = "Smith"
14        self.prenom = "John" # Quelle originalité
15        self.age = 28 # Cela n'engage à rien
16        self.lieu = "London"
17
18 moi = Personne()
19 print(moi.nom, moi.prenom, moi.age, moi.lieu)
```

---

pour l'instant, tous les objets créés comportent les mêmes infos :

```
Smith John 28 London
```

### 1.3 insertion d'information à la création

```
1 # création d'une classe Personne
2 class Personne:
3     """Classe définissant une personne caractérisée par :
4         - son nom
5         - son prénom
6         - son âge
7         - son lieu de résidence"""
8
9     def __init__(self, nom, age, lieu):
10        """Constructeur de notre classe"""
11        self.nom = nom
12        self.prenom = "John"
13        self.age = age
14        self.lieu = lieu
15
16 moi = Personne("Dubert",25,"New-York")
17 print(f"Bonjour, je suis {moi.prenom} {moi.nom} et j'ai {moi.age} ans. J'←
    habite à {moi.lieu}.")
```

en créant une personne, j'indique aussi des infos sur elle :

```
Bonjour, je suis John Dubert et j'ai 25 ans.
J'habite à New-York .
```

### 1.4 attribut pour la classe

- pour l'instant, les attributs créés étaient associées à l'instance créée
- on peut aussi créer des attributs pour la classe elle-même
- voici par un attribut associé à la classe Compteur qui compte le nombre d'objets créés

---

```
1 class Compteur:
2     """Cette classe possède un attribut de classe qui s'incrémente à ↵
3         chaque
4         fois que l'on crée un objet de ce type"""
5
6     objets_crees = 0 # Le compteur vaut 0 au départ
7     def __init__(self):
8         """À chaque fois qu'on crée un objet, on incrémente le compteur"""
9         Compteur.objets_crees += 1
```

---

## 2 exemple complet

---

```
1 class TableauNoir:
2     """Classe définissant une surface sur laquelle on peut écrire,
3         que l'on peut lire et effacer, par jeu de méthodes. L'attribut modifié
4         est 'surface'"""
5
6     def __init__(self):
7         """Par défaut, notre surface est vide"""
8         self.surface = ""
9
10    def ecrire(self, message_a_ecrire):
11        """Méthode permettant d'écrire sur la surface du tableau.
12            Si la surface n'est pas vide, on saute une ligne avant de rajouter
13            le message à écrire"""
14        if self.surface != "":
15            self.surface += "\n"
16        self.surface += message_a_ecrire
17
18    def lire(self):
19        """Cette méthode se charge d'afficher, grâce à print,
20            la surface du tableau"""
21        print(self.surface)
22
23    def effacer(self):
24        """Cette méthode permet d'effacer la surface du tableau"""
25        self.surface = ""
26
27 a = TableauNoir()
28 a.ecrire("texte_a_ecrire_1")
29 a.ecrire("texte_a_ecrire_2")
30 a.lire()
31 a.effacer()
32 a.lire()
```

---

ce qui donne :

```
texte_a_ecrire_1  
texte_a_ecrire_2  
▶ []
```

### 3 exercices

- [ex 1](#) : palindrome - méthode de classe
- [ex 2](#) : palindrome - méthode d'instance
- [ex 3](#) : puzzle
- [ex 4](#) : logger
- [ex 5](#) : héritage "simple"
- [ex 6](#) : puzzle
- [ex 7](#) : héritage multiple - cas réel