Chapitre 9

Les classes

Introduction aux Classes

Pour les étudiants en génie civil – M1 RIB

Dr. Hachimi Dahhaoui
Université de Tlemcen

Mercredi 02 Décembre 2025

Problème réel : comment organiser ces données ?

Imagine que tu dois gérer des informations sur des étudiants :

Nom	Moyenne	Groupe
Sara	14.5	M1 VOA
Yacine	12.0	M1 RIB
Lina	16.2	M1 STR

En Python, si on fait ça :

```
nom = "Sara"
moyenne = 14.5
groupe = "M1 VOA"
```

- ➡ Trop de variables séparées
- ➡ Pas organisé
- ➡ Impossible à gérer pour 30 étudiants

Comment structurer tout ça proprement ?

🎬 Problème réel : comment organiser ces données ?

🧱 Et si Python nous permettait de créer *notre propre type* ?

- ❑ Un type **Étudiant**, comme int, str, list
- ❑ Avec **ses données** (nom, moyenne…)
- ❑ Avec **ses actions** (afficher, mise à jour…)

➡ C’est exactement ce que permettent **les classes**.

🎬 Problème réel : comment organiser ces données ?

🎉 Avant d'expliquer… voici ce qu'on veut obtenir :

```
e1 = Etudiant("Sara", 14.5)
e1.afficher()
```

Résultat :

```
Sara a une moyenne de 14.5
```

➡ Simple.
➡ Beau.
➡ Organisé.

Aujourd'hui, on apprend à créer ça.

Diapo 5 – Objectifs

Ce que vous serez capables de faire :

- ❑ Comprendre ce qu’est un objet
- ❑ Créer vos propres types (classes)
- ❑ Construire des objets
- ❑ Leur faire exécuter des actions
- ❑ Organiser vos projets Python proprement

Diapo 6 – Pourquoi c’est indispensable (Génie Civil)

Vous devrez gérer :

- ❑ des **matériaux**
- ❑ des **poutres**
- ❑ des **sols**
- ❑ des **échantillons**
- ❑ des **capteurs**
- ❑ des **mesures**

➜ Chacun avec ses **données**
➜ Et ses **méthodes** (calculs, affichage…)

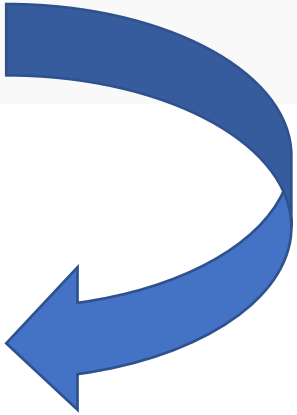
Les classes Python permettent exactement ça.

Diapo 7 – Tout est objet en Python (Code)

```python
x = 5
nom = "Sara"
L = [1, 2, 3]

print(type(x))
print(type(nom))
print(type(L))
```

```
<class 'int'>
<class 'str'>
<class 'list'>
```



Diapo 8 – Tout est objet en Python (Code)

- ❑ Python ne manipule jamais “juste des valeurs”
- ❑ Python manipule toujours des objets
- ❑ Même des choses simples comme 5, "Sara" ou [1,2,3] sont des objets

👉 C’est la base pour comprendre ensuite les classes,
puisque créer une classe = **créer un nouveau type d’objet.**

Diapo 9 – Noms et Références (Code)

```
x = 10
y = x


print("x =", x)
print("y =", y)
```

```
x = 10
y = 10
```

➡ x et y pointent vers le même objet.

Diapo 10 – Portées (Scopes)

Un nom peut exister dans différents espaces :

- ❑ Portée **locale**
- ❑ Portée **globale**
- ❑ Portée **builtin**
- ❑ Portée **enclosing** (fonctions imbriquées)

Règle de recherche Python : **LEGB**

Diapo 11 – Exemple de portée (Code)

```python
x = 10  # globale

def f():
    x = 99  # locale
    print("dans f, x =", x)

f()
print("dans le programme principal, x =", x)
```

```
dans f, x = 99
dans le programme principal, x = 10
```

Diapo 12 – Pourquoi utiliser des classes ?

Sans classe :

```
nom = "Sara"
moyenne = 14.5
groupe = "M1 VOA"
```

➡ Les données sont séparées.
➡ Pas d'organisation.
Les classes regroupent **données + fonctions**.

Diapo 13 – Première classe vide (Code)

```
class Etudiant:
    pass

e = Etudiant()
print(e)
```

```
<__main__.Etudiant object at 0x...>
```

Diapo 14 – Ajouter un constructeur (Code)

```python
class Etudiant:
    def __init__(self, nom, moyenne):
        self.nom = nom
        self.moyenne = moyenne

e1 = Etudiant("Sara", 14.5)

print(e1.nom)
print(e1.moyenne)
```

```
Sara
14.5
```

Diapo 15 – Attributs d’instance (Code)

```python
class Etudiant:
    def __init__(self, nom, moyenne):
        self.nom = nom
        self.moyenne = moyenne

e1 = Etudiant("Sara", 14.5)
e2 = Etudiant("Yacine", 12.0)

print("e1 :", e1.nom, e1.moyenne)
print("e2 :", e2.nom, e2.moyenne)
```

```
e1 : Sara 14.5
e2 : Yacine 12.0
```

➡ Chaque objet possède ses propres valeurs.

Diapo 16 – Attributs de classe (Code)

```python
class Etudiant:
    universite = "Université de Tlemcen"  # variable de classe

    def __init__(self, nom):
        self.nom = nom  # variable d'instance

e1 = Etudiant("Sara")
e2 = Etudiant("Yacine")

print(e1.nom, "-", e1.universite)
print(e2.nom, "-", e2.universite)
```

```
Sara - Université de Tlemcen
Yacine - Université de Tlemcen
```

Partagé entre tous les objets.

Diapo 17 – Méthodes d'instance (Code)

Dans cette classe, on a :

python Copier le code

```python
def afficher(self):
    print(f"{self.nom} a une moyenne de {self.moyenne}")
```

Cette méthode utilise `self` pour accéder aux données de l'objet.

Elle sait donc afficher **le nom** et **la moyenne** de n'importe quel étudiant.

Diapo 18 – Méthode qui modifie l’objet (Code)

```python
class Etudiant:
    def __init__(self, nom, moyenne):
        self.nom = nom
        self.moyenne = moyenne

    def maj_moyenne(self, nouvelle_moy):
        self.moyenne = nouvelle_moy

e1 = Etudiant("Sara", 14.5)
e1.maj_moyenne(16)
print(e1.moyenne)
```

```
16
```

Diapo 19 – Exemple complet : Rectangle (Code)

```python
class Rectangle:
    def __init__(self, longueur, largeur):
        self.longueur = longueur
        self.largeur = largeur

    def surface(self):
        return self.longueur * self.largeur

r = Rectangle(5, 3)
print("Surface =", r.surface())
```

```
Surface = 15
```

Diapo 20 – Mini-exercice 1

Créer une classe Point avec :

- ❑ X
- ❑ Y
- ❑ une méthode afficher()

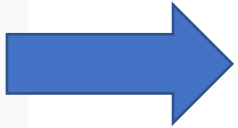
Créer un point (3,7) et l'afficher.

Diapo 21 – Mini-exercice 1 (Correction)

```python
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def afficher(self):
        print(f"Point({self.x}, {self.y})")

p = Point(3, 7)
p.afficher()
```



```
Point(3, 7)
```

Diapo 22 – Mini-exercice 2

Créer une classe Etudiant avec :

- ❑ Nom
- ❑ Moyenne
- ❑ méthode admis() → True si moyenne ≥ 10

Tester avec un étudiant.

Diapo 23 – Mini-exercice 2 (Correction)

```python
class Etudiant:
    def __init__(self, nom, moyenne):
        self.nom = nom
        self.moyenne = moyenne

    def admis(self):
        return self.moyenne >= 10

e = Etudiant("Sara", 14)
print(e.nom, "admis ?", e.admis())
```

```
Sara admis ? True
```

Diapo 24 – Conclusion

Aujourd'hui, nous avons appris :

- Objets et références
- Portées et espaces de nommage
- Classes simples
- Constructeur `__init__`
- Variables d'instance / classe
- Méthodes d'instance
- Exercices pratiques

Prochaine séance :

➡️ Héritage, variables privées, générateurs.

Merci !

Des questions ?

Contact : hachimi.dahhaoui@univ-tlemcen.dz

Fin du Chapitre 9

À très bientôt pour la suite !