Dans l'exemple suivant, on va créer une classe personne en lui cédant des attributs et leurs valeurs usine.

- **Syntaxe :**

```
class Personne :
    CIN = "CIN de la personne"
    nom = "Nom de la personne"
    age = "Age de la personne"
    adresse = "Adresse de la personne"
```

1. Création d'un objet :

Après la création d'une classe, la dernière étape qui parvient sera de tester notre classe et de créer des objets de cette classe.

La création d'un objet est déterminée telle que : ***objet = NomDeClass ().***
Ainsi, pour faire appel à un attribut associé à un objet : *objet.attribut* , et pour le modifier il suffit de l'attribuer une nouvelle valeur : ***objet.attribut = nouvelleValeurDAttribut .***
D'autre part, pour supprimer la valeur d'un attribut associé à un objet, on utilise le mot-clé *del* tel que : *del objet.attribut.*

Exemple 1 :

En reprenant l'exemple proposé dans la partie précédente, on fera appel à un objet contenant les valeurs définies par défaut lors de la création de la classe.

- **Syntaxe :**

```
python
class Personne :
    CIN = "CIN de la personne"
    nom = "Nom de la personne"
    age = "Age de la personne"
    adresse = "Adresse de la personne"

personne_usine = Personne()
print(personne_usine.CIN)
print(personne_usine.nom)
print(personne_usine.age)
print(personne_usine.adresse)
```

- **Résultat de l'exécution :**

```
class Personne :
    CIN = "CIN de la personne"
    nom = "Nom de la personne"
    age = "Age de la personne"
    adresse = "Adresse de la personne"

personne_usine = Personne()

print(personne_usine.CIN)
print(personne_usine.nom)
print(personne_usine.age)
print(personne_usine.adresse)
```

```
CIN de la personne
Nom de la personne
Age de la personne
Adresse de la personne
```

Exemple 2 :

Ainsi, pour affecter des valeurs aux attributs de notre objet de la classe précédente :

- **Syntaxe :**

```python
class Personne :
    CIN = "CIN de la personne"
    nom = "Nom de la personne"
    age = "Age de la personne"
    adresse = "Adresse de la personne"
personne1 = Personne()
personne1.CIN = "A123456"
personne1.nom = "Ismail BOUZIANE"
personne1.age = 24
personne1.adresse = "Apt 1, rue 1, ville 1"
print(personne1.CIN)
print(personne1.nom)
print(personne1.age)
print(personne1.adresse)
```

- **Résultat de l'exécution :**

```
class Personne :
    CIN = "CIN de la personne"
    nom = "Nom de la personne"
    age = "Age de la personne"
    adresse = "Adresse de la personne"

personne1 = Personne()
personne1.CIN = "A123456"
personne1.nom = "Ismail BOUZIANE"
personne1.age = 24
personne1.adresse = "Apt 1, rue 1, ville 1"

print(personne1.CIN)
print(personne1.nom)
print(personne1.age)
print(personne1.adresse)
```

```
A123456
Ismail BOUZIANE
24
Apt 1, rue 1, ville 1
```

2. Exercice d'application :

Réappliquez l'exemple précédent pour créer une classe *Produit* sur Python ayant comme attributs : *référence*, *désignation*, *prix* et *quantité*. Instanciez ces attributs, ainsi que créez des exemples d'objets produits.

Solution de l'exercice d'application :

- **Syntaxe :**

```python
class Produit :
    référence = "référence du produit"
    désignation = "désignation du produit"
    prix = "prix du produit"
    quantité = "quantité du produit"
produit_ini = Produit()
produit1 = Produit()
produit1.référence = "ABC123"
produit1.désignation = "SMART TV"
produit1.prix = 3999
produit1.quantité = 5
print("l'instanciation des attributs :")
print(produit_ini.référence)
print(produit_ini.désignation)
```

```
print(produit_ini.prix)
print(produit_ini.quantité)
print("Exemple d'objet de la classe Produit :")
print(produit1.référence)
print(produit1.désignation)
print(produit1.prix)
print(produit1.quantité)
```

- **Résultat de l'exécution :**

```
class Produit :
    référence = "référence du produit"
    désignation = "désignation du produit"
    prix = "prix du produit"
    quantité = "quantité du produit"

produit_ini = Produit()

produit1 = Produit()
produit1.référence = "ABC123"
produit1.désignation = "SMART TV"
produit1.prix = 3999
produit1.quantité = 5

print("l'instanciation des attributs :")
print(produit_ini.référence)
print(produit_ini.désignation)
print(produit_ini.prix)
print(produit_ini.quantité)

print("Exemple d'objet de la classe Produit :")
print(produit1.référence)
print(produit1.désignation)
print(produit1.prix)
print(produit1.quantité)
```

```
l'instanciation des attributs :
référence du produit
désignation du produit
prix du produit
quantité du produit
Exemple d'objet de la classe Produit :
ABC123
SMART TV
3999
5
```

Le constructeur de classe ou la méthode __init__ :

Le constructeur ou bien la méthode __init__ permet d'initialiser les valeurs par défaut des attributs d'une classe, elle est considérée comme la méthode la plus importante d'une classe, on l'appelle lors de la création d'un objet de classe en utilisant le nom de la classe comme fonction.

Cette méthode est représentée sous forme d'une fonction ayant par convention comme premier paramètre l'attribut qu'on appellera « *self* », elle peut disposer d'aucun ou de plusieurs paramètres.

Le paramètre « *self* » :

Le paramètre « *self* » permet de faire référence à l'adresse mémoire de l'instance d'un objet de classe déclaré, ainsi, il est utilisé pour accéder aux attributs appartenant à la classe.
Toute méthode de classe, et non pas seulement la méthode *__init__*, doit contenir ce paramètre.
Remarque : il n'est pas nécessaire d'écrire *self* tel qu'il est, vous pouvez l'appeler comme vous le souhaitez, mais à condition qu'il soit le premier paramètre de toute méthode de la classe:

Exemple 1 de la méthode __init__ :

En progressant l'exemple de la partie précédente par la méthode __inti__ on aura :

- **Syntaxe :**

```
python
class Personne :
    CIN = "CIN de la personne"
    nom = "Nom de la personne"
    age = "Age de la personne"
    adresse = "Adresse de la personne"

    def __init__ (self, CIN, nom, age, adresse) :
        self.CIN = CIN
```

```
        self.nom = nom
        self.age = age
        self.adresse = adresse
personne1 = Personne("A123456","Ismail BOUZIANE",24,"Apt 1, rue 1, ville 1")
print(personne1)
print(personne1.CIN)
print(personne1.nom)
print(personne1.age)
print(personne1.adresse)
```

Cette méthode ne renvoie aucune valeur, elle est exécutée automatiquement lors de la création d'un objet de la classe.

- **Résultat de l'exécution :**

```
class Personne :
    CIN = "CIN de la personne"
    nom = "Nom de la personne"
    age = "Age de la personne"
    adresse = "Adresse de la personne"

    def __init__ (self, CIN, nom, age, adresse) :
        self.CIN = CIN
        self.nom = nom
        self.age = age
        self.adresse = adresse

personne1 = Personne("A123456","Ismail BOUZIANE",24,"Apt 1, rue 1, ville 1")
print(personne1)
print(personne1.CIN)
print(personne1.nom)
print(personne1.age)
print(personne1.adresse)
```

```
<__main__.Personne object at 0x7f96443e32e8>
A123456
Ismail BOUZIANE
24
Apt 1, rue 1, ville 1
```

Exemple 2 de la méthode __init__ :

On progresse l'exercice de l'application de la partie précédente (class Produit) en intégrant la méthode __init__.

- **Syntaxe :**

```
python
class Produit :
    référence = "référence du produit"
    désignation = "désignation du produit"
    prix = "prix du produit"
    quantité = "quantité du produit"

    def __init__ (self, référence, désignation, prix, quantité) :
        self.référence = référence
        self.désignation = désignation
        self.prix = prix
        self.quantité = quantité

produit1 = Produit("ABC123", "SMART TV", 3999, 5)
print(produit1)
print(produit1.référence)
print(produit1.désignation)
print(produit1.prix)
print(produit1.quantité)
```

- **Résultat de l'exécution :**

```
class Produit :
    référence = "référence du produit"
    désignation = "désignation du produit"
    prix = "prix du produit"
    quantité = "quantité du produit"

    def __init__ (self, référence, désignation, prix, quantité) :
        self.référence = référence
        self.désignation = désignation
        self.prix = prix
        self.quantité = quantité

produit1 = Produit("ABC123", "SMART TV", 3999, 5)
print(produit1)
print(produit1.référence)
print(produit1.désignation)
print(produit1.prix)
print(produit1.quantité)
```

```
<__main__.Produit object at 0x7f96443e34a8>
ABC123
SMART TV
3999
5
```

3. Définir des attributs privés :

Afin de protéger l'accès à un ou plusieurs attributs d'une classe, on utilise les attributs privés, pour en créer, on ajoute deux *underscore* « __ » avant le nom de l'attribut.
Remarque : il est possible d'attribuer l'accès à ce paramètre qu'on ont veut, en utilisant les accesseurs ou les getters (on introduira ces notions dans une autre partie).

Exemple d'attribut privé :

On ajoute à l'exemple précédent, un attribut compte bancaire « cpt_bancaire » qu'on définira comme attribut privé :

- **Syntaxe :**

```
python
class Personne :
    CIN = "CIN de la personne"
    nom = "Nom de la personne"
    age = "Age de la personne"
    adresse = "Adresse de la personne"

    def __init__ (self, CIN, nom, age, adresse, cpt_bancaire) :
        self.CIN = CIN
        self.nom = nom
        self.age = age
        self.adresse = adresse
        self.__cpt_bancaire = cpt_bancaire

personne1 = Personne("A123456","Ismail BOUZIANE",24,"Apt 1, rue 1, ville 1","1324456")
print(personne1)
print(personne1.CIN)
print(personne1.nom)
print(personne1.age)
print(personne1.adresse)
print(personne1.__cpt_bancaire)
```

Il n'est alors plus possible d'avoir accès à l'attribut « cpt_bancaire » depuis l'extérieur de la classe « Personne » :

- **Résultat de l'exécution :**

```python
class Personne :
    CIN = "CIN de la personne"
    nom = "Nom de la personne"
    age = "Age de la personne"
    adresse = "Adresse de la personne"

    def __init__ (self, CIN, nom, age, adresse, cpt_bancaire) :
        self.CIN = CIN
        self.nom = nom
        self.age = age
        self.adresse = adresse
        self.__cpt_bancaire = cpt_bancaire

personne1 = Personne("A123456","Ismail BOUZIANE",24,"Apt 1, rue 1, ville 1","1324456")
print(personne1)
print(personne1.CIN)
print(personne1.nom)
print(personne1.age)
print(personne1.adresse)
print(personne1.__cpt_bancaire)
```

```
<__main__.Personne object at 0x7f1a38e67278>
A123456
Ismail BOUZIANE
24
Apt 1, rue 1, ville 1

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-1-1a3bc0856fed> in <module>
     18 print(personne1.age)
     19 print(personne1.adresse)
---> 20 print(personne1.__cpt_bancaire)

AttributeError: 'Personne' object has no attribute '__cpt_bancaire'
```

4. Exercice global :

On souhaite réaliser une classe issue d'un outil classique utilisé régulièrement dans la vie ordinaire : une voiture, tout en suivant la procédure suivante :

1. Créer une classe **Voiture** ayant les attributs suivants et leur définir des valeurs par défaut :

- ·**modèle** : qui indique la marque et le modèle de la voiture (Mercedes, Audi…)
- ·**annéeCréation** : indiquant l'année de création de la voiture (2004, 2005…)
- ·**alimentation** : type d'alimentation de la voiture (électrique, diesel, essence…)
- ·**nbPlaces** : indiquant la capacité de la voiture (4 places, 2 places...)
- ·**puissance** : indiquant la puissance en cheval de la voiture.
- ·**coûtFabri** : on considérera ce paramètre étant **privé**, représentant le coût de fabrication de la voiture.

Tester votre classe et créer un objet contenant les valeurs par défaut.

2. Redéfinir votre classe en utilisant le constructeur ou la méthode __init__ de classe **Voiture** tel que :

```
python
__init__ (self, modèle, annéeCréation, puissance, nbPlaces, alimentation, coûtFabri).
```

Tester votre classe et créer un exemple d'objet.

Solution de l'exercice global :

Solution de la première question :

- **Syntaxe :**

```
python
# Réponse de la première question : Créer une classe Voiture et définir des valeurs par défaut à ses attributs :

class Voiture :
modèle = "Le modèle de la voiture"
annéeCréation = "L'année de création de la voiture"
alimentation  = "Le type d'alimentation de la voiture"
nbPlaces = "La capacité de la voiture"
```

```
puissance = "La puissance en cheval de la voiture"
coûtFabri = "Le coût de fabrication de la voiture"
# Tester les valeurs par défaut des attributs :
voitureParDéfaut = Voiture()
print(voitureParDéfaut.modèle)
print(voitureParDéfaut.annéeCréation)
print(voitureParDéfaut.puissance)
print(voitureParDéfaut.nbPlaces)
print(voitureParDéfaut.alimentation)
```

- **Résultat de l'exécution :**

```
# Réponse de la première question : Créer une classe Voiture et définir des valeurs par défaut à ses attributs :

class Voiture :
    modèle = "Le modèle de la voiture"
    annéeCréation = "L'année de création de la voiture"
    alimentation  = "Le type d'alimentation de la voiture"
    nbPlaces = "La capacité de la voiture"
    puissance = "La puissance en cheval de la voiture"
    coûtFabri = "Le coût de fabrication de la voiture"

# Tester les valeurs par défaut des attributs :
voitureParDéfaut = Voiture()
print(voitureParDéfaut.modèle)
print(voitureParDéfaut.annéeCréation)
print(voitureParDéfaut.puissance)
print(voitureParDéfaut.nbPlaces)
print(voitureParDéfaut.alimentation)
```

```
Le modèle de la voiture
L'année de création de la voiture
La puissance en cheval de la voiture
La capacité de la voiture
Le type d'alimentation de la voiture
```

Solution de la deuxième question :

- **Syntaxe :**

```
python
# Réponse de la deuxième question : Redéfinir la classe avec le constructeur ou la méthode __init__ de classe Voiture :
class Voiture :

modèle = "Le modèle de la voiture"
annéeCréation = "L'année de création de la voiture"
alimentation  = "Le type d'alimentation de la voiture"
nbPlaces = "La capacité de la voiture"
puissance = "La puissance en cheval de la voiture"
coûtFabri = "Le coût de fabrication de la voiture"
def __init__ (self, modèle, annéeCréation, puissance, nbPlaces, alimentation, coûtFabri) :
self.modèle = modèle
self.annéeCréation = annéeCréation
self.puissance = puissance
self.nbPlaces = nbPlaces
self.alimentation = alimentation
self.__coûtFabri = coûtFabri
# Tester un exemple d'objet :
voiture1 = Voiture("Mercedes", "2004", 200, 4, "Diesel", 123456)
print(voiture1.modèle)
print(voiture1.annéeCréation)
print(voiture1.puissance)
print(voiture1.nbPlaces)
print(voiture1.alimentation)
# Bonus : redéfinition de la valeur de l'attribut modéle :
voiture1.modèle = "Audi"
print(voiture1.modèle)
# Bonus : utilisation de l'instruction del :
```

```
del voiture1.puissance
print(voiture1.puissance)
del voiture1.alimentation
print(voiture1.alimentation)
```

- **Résultat de l'exécution :**

```
# Réponse de la deuxième question : Redéfinir la classe avec le constructeur ou la méthode __init__ de classe Voiture :

class Voiture :

    modèle = "Le modèle de la voiture"
    annéeCréation = "L'année de création de la voiture"
    alimentation  = "Le type d'alimentation de la voiture"
    nbPlaces = "La capacité de la voiture"
    puissance = "La puissance en cheval de la voiture"
    coûtFabri = "Le coût de fabrication de la voiture"

    def __init__ (self, modèle, annéeCréation, puissance, nbPlaces, alimentation, coûtFabri) :
        self.modèle = modèle
        self.annéeCréation = annéeCréation
        self.puissance = puissance
        self.nbPlaces = nbPlaces
        self.alimentation = alimentation
        self.__coûtFabri = coûtFabri

# Tester un exemple d'objet :
voiture1 = Voiture("Mercedes", "2004", 200, 4, "Diesel", 123456)
print(voiture1.modèle)
print(voiture1.annéeCréation)
print(voiture1.puissance)
print(voiture1.nbPlaces)
print(voiture1.alimentation)

# Bonus : redéfinition de la valeur de l'attribut modèle :
voiture1.modèle = "Audi"
print(voiture1.modèle)

# Bonus : utilisation de l'instruction del :
del voiture1.puissance
print(voiture1.puissance)
del voiture1.alimentation
print(voiture1.alimentation)
```

```
Mercedes
2004
200
4
Diesel
Audi
La puissance en cheval de la voiture
Le type d'alimentation de la voiture
```