



Présentation ADRAR Pôle Numérique

Suivez-nous... LinkedIn  et Twitter  @Adrar_Numerique



ADRAR\o/ **PÔLE NUMÉRIQUE**

PÔLE NUMÉRIQUE DU CENTRE DE FORMATION ADRAR

- > SUPPORT, ADMINISTRATION SYSTEMES & RESEAUX
- > DEVELOPPEMENT D'APPLICATIONS WEB & MOBILES
- > WEBDESIGN & DEVELOPPEMENT INTERFACES
- > TRANSFORMATION NUMERIQUE DES ENTREPRISES

<http://www.adrar-numerique.com>

Suivez-nous sur LinkedIn  Twitter  @Adrar_Numerique

... et sur le web : www.adrar-numerique.com



Suivez-nous sur LinkedIn Twitter  @Adrar_Numerique... et sur le web : www.adrar-numerique.com

LA PROGRAMMATION ORIENTEE

OBJET : POO

Suivez-nous sur LinkedIn  Twitter  @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

La programmation orientée objet (POO) est une méthode de structuration de code permettant de rendre votre programme mieux organisé et plus lisible. Comme vous le savez en python TOUT est objet, c'est donc un langage propice à l'apprentissage de la POO. Quelques définitions :

- Classe : Ensemble de code regroupant des variables et des méthodes permettant de créer des objets.
- Objet : Instance d'une classe disposant d'attributs et de méthodes.
- Constructeur : Méthode qui va être utilisée pour instancier un objet.
- Assesseur (getter) : Méthode qui va être utilisée pour afficher un attribut.
- Mutateur (setter) : Méthode qui va être utilisée pour modifier un attribut.
- Héritage : Principe voulant qu'une classe hérite des propriétés de son parent. De base toutes les classes héritent de la classe Object

Suivez-nous sur LinkedIn 

Twitter  @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

METHODES SPECIALES

Suivez-nous sur LinkedIn  Twitter  @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

Les méthodes spéciales sont utilisées pour définir le comportement de l'objet lors d'utilisation de fonctions built in ou opérations tels que print, +, -, boucle for, etc.

On définit ces comportements en déclarant des fonctions avec des noms spécifiques commençant et terminant par « __ ».

`__add__`: définit le comportement lors d'un « + »

`__sub__`: définit le comportement lors d'un « - »

`__mul__`: définit le comportement lors d'un « * »

`__truediv__`: définit le comportement lors d'un « / »

`__pow__`: définit le comportement lors d'un « ** »

`__len__`: définit le comportement lors de l'utilisation de la fonction `len()`

`__repr__`: définit le comportement lors de l'utilisation de la fonction `print()`

`__getitem__`: définit le comportement lors de l'utilisation de la syntaxe `objet[clé]`

`__setitem__`: définit le comportement lors de l'utilisation de la syntaxe `objet[clé] = valeur`

`__contains__`: définit le comportement lors de l'utilisation d'un « if elem in objet »

`__iter__` et `__next__`: définissent le comportement lors d'un « for i in objet »

Suivez-nous sur LinkedIn Twitter  @Adrar_Numerique... et sur le web : www.adrar-numerique.com

Prenons en exemple une classe Piece représentant une pièce d'une maison.

Classe : Piece

Attributs :

Nom: str

Longueur : int

Largeur : str

Meubles : list

Suivez-nous sur LinkedIn  Twitter  @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

Commençons par déclarer le constructeur de notre classe :

```
class Piece:  
    def __init__(self, nom, longueur, largeur):  
        self.nom = nom  
        self.longueur = longueur  
        self.largeur = largeur  
        self.meubles = []
```


Suivez-nous sur LinkedIn  Twitter  @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

Définissons ensuite les méthodes `__len__` et `__repr__` :

```
def __repr__(self):
    affiche = f"Bienvenue dans le {self.nom}. Il y a {len(self)} meubles :\n"
    for i in self.meubles:
        affiche += i+"\n"
    return affiche

# len est appelé lors d'une méthode "len(objet)"
def __len__(self):
    return len(self.meubles)
```

Nous pouvons maintenant faire un `print()` de notre objet:

```
salon = Piece("salon", 20, 30)
print(salon)
```

```
Bienvenue dans le salon. Il y a 0 meubles :
```

Suivez-nous sur LinkedIn  Twitter  @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

Définissons ensuite les méthodes spéciales d'addition et de soustraction :

```
# add est appelé lors d'un "objet + item"
def __add__(self, item):
    self.meubles.append(item)

# sub est appelé lors d'un "objet - meuble"
def __sub__(self, item):
    self.meubles.pop(self.meubles.index(item))
```

Nous pouvons maintenant ajouter ou enlever des meubles :

```
salon + "table"
salon + "sofa"
print(salon)
salon - "table"
print(salon)
```

```
Bienvenue dans le salon. Il y a 2 meubles :
table
sofa

Bienvenue dans le salon. Il y a 1 meubles :
sofa
```

Suivez-nous sur LinkedIn Twitter @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

Définissons ensuite les méthodes contains, getitem et setitem:

```
# contains est appelé lors d'un "if item in objet"
def __contains__(self, item):
    return item in self.meubles

# getitem est utilisé lors d'un objet[item]
def __getitem__(self, item):
    return self.meubles[item]

# setitem est utilisé lors d'un "objet[cle] = item"
def __setitem__(self, cle, item):
    self.meubles[cle] = item
```

```
print(salon)
if "table" in salon:
    print("j'ai une table")
else:
    print("je n'ai pas de table")
print(salon[0])
salon[1] = "chaise"
print(salon)
```

```
Bienvenue dans le salon. Il y a 2 meubles :
table
sofa

j'ai une table
table
Bienvenue dans le salon. Il y a 2 meubles :
table
chaise
```

Suivez-nous sur LinkedIn  Twitter  @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

Et enfin définissant le comportement de notre objet lorsqu'on essaye d'itérer dedans :

```
# iter et next sont utilisé lors d'une boucle for
def __iter__(self):
    self.n = 0
    return self

def __next__(self):
    if self.n < len(self.meubles):
        indice = self.n
        self.n += 1
        return self.meubles[indice]
    else:
        raise StopIteration
```

```
for i in salon:
    print(i)
```

```
table
chaise
```

Suivez-nous sur LinkedIn Twitter  @Adrar_Numerique... et sur le web : www.adrar-numerique.com

EXERCICE

Classe : Liste de courses

Attributs :

Date création : int

A acheter : list

Acheté : list

Créez la classe Liste de courses et implémentez la méthodes spéciales add, sub, contains, getitem, setitem, iter, next et repr