

# Découvrir et approfondir python

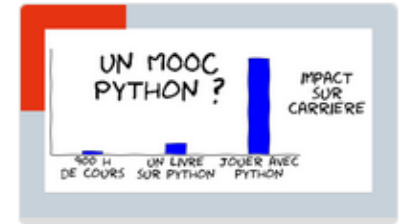


## Les classes instances et méthodes :

W3-08 Introduction aux classes.mp4

Vidéos proposées par l'INRIA sur

Python 3 : des fondamentaux aux concepts avancés du langage



Thierry PARMENTELAT et Arnaud Legoud

## Pour bien comprendre et approfondir :

a) La méthode qui crée une instance d'un objet d'une classe s'appelle ?

- le créateur
- le constructeur
- l'initiateur
- le fabricant

b) On définit une classe de la manière suivante

```
class C:
```

Puis on exécute

```
mon_objet_C = C("OBJET 1",20)
```

```
def __init__(self,a,b):  
    self.nom = a  
    self.valeur = b
```

Comment afficher la valeur de l'attribut *nom* de *mon\_objet\_C* dans la console ?

- `mon_objet_C(nom)`
- `mon_objet_C[nom]`
- `mon_objet_C.nom`
- ce n'est pas possible

c) On ajoute les deux méthodes ci-contre

à notre classe :

Comment s'appellent ces méthodes ?

```
def getNom(self):  
    return self.nom
```

```
def getValeur(self):  
    return self.valeur
```

- getter
- mutateur
- setter
- accesseur

d) On souhaite ajouter un mutateur pour pouvoir modifier l'attribut valeur de nos instances d'objets de Classe C. Quelles sont les ou les codes corrects ?

- |   |  |
|---|--|
| <input type="radio"/> <code>def setValeur(self):</code><br><code>self.valeur = new_valeur</code>  | <input type="radio"/> <code>def set_Valeur(self,v):</code><br><code>self.valeur = v</code> |
| <input type="radio"/> <code>def modification(valeur):</code><br><code>self.valeur = valeur</code> | <input type="radio"/> <code>def setValeur(self,v):</code><br><code>self.valeur = v</code>  |

e) A partir du code de définition de la classe Phrase ci-dessous, cocher les affirmations correctes :

```
class Phrase:
    """
    Docstring personnalisée
    Un petit exemple de classe avec des méthodes built-in
    """
    def __init__(self, phrase):
        self.mots = phrase.split()

    def upper(self):
        self.mots = [m.upper() for m in self.mots]

    def __str__(self):
        return "\n".join(self.mots)

    def __len__(self):
        return len(self.mots)

    def __contains__(self, mot):
        if mot in self.mots:
            return True
        else:
            return False
```

```
# Utilisation de la classe
p = Phrase("je fais un mooc sur python")
```

- les méthodes `__str__` `__len__` et `__contains__` sont des méthodes built-in
- `print(p)` appelle automatiquement la méthode `__str__` de l'objet p
- Le Docstring ne joue aucun rôle particulier
- `'je' in p` est une écriture correcte ?
- `'JE' in p` renvoie True ?
- Dans la définition de la classe *self* fait référence à l'instance de l'objet ?

