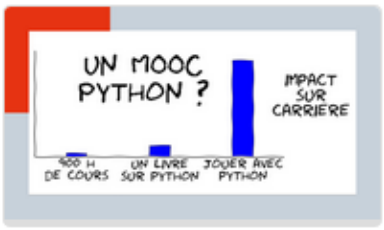
Découvrir et approfondir python

Les classes instances et méthodes :

V W3-08 Introduction aux classes.mp4

Vidéos proposées par l'INRIA sur FUN MOOC

Python 3 : des fondamentaux aux concepts avancés du langage



Thierry PARMENTELAT et Arnaud Legoud

Pour bien comprendre et approfondir :

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

O le créateur **O** le constructeur

O l'initiateur **O** le fabricateur

b) On défini une classe de la manière suivante

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

Puis on exécute

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

Comment afficher la valeur de l'attribut *nom* de *mon_objet_C* dans la console ?

O mon_objet_C(nom) **O** mon_objet_C[nom]

O mon_objet_C.nom **O** ce n'est pas possible

c) On ajoute les deux méthodes ci-contre à notre classe :

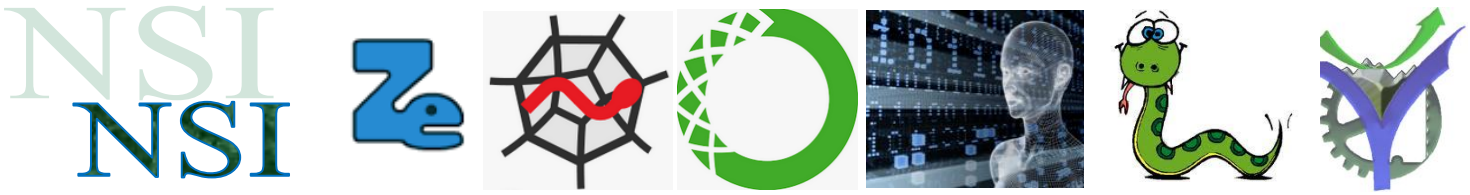
Comment s'appelle ces méthodes ?

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

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

O getter **O** mutateur

O setter **O** 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 ?

O
```
def setValeur(self):
    self.valeur = new_valeur
```

O
```
def set_Valeur(self,v):
    self.valeur = v
```

O
```
def modification(valeur):
    self.valeur = valeur
```

O
```
def setValeur(self,v):
    self.valeur = v
```

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")
```

O les méthodes __str__ __len__ et __contains__ sont des méthodes built-in

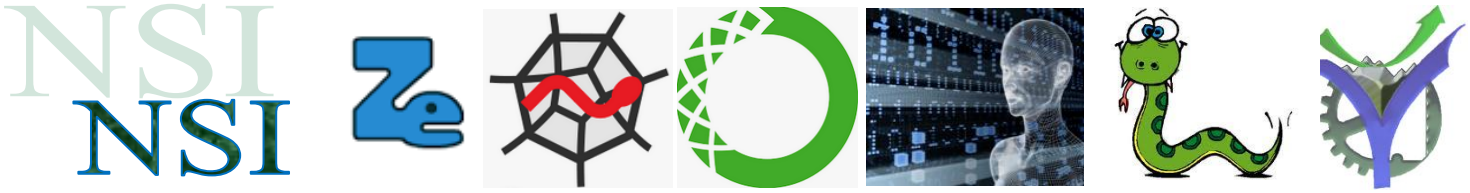
O `print(p)` appelle automatiquement la méthode __str__ de l'objet p

O Le Docstring ne joue aucun rôle particulier

O `'je' in p` est une écriture correcte ?

O `'JE' in p` renvoie True ?

O Dans la définition de la classe *self* fait référence à l'instance de l'objet ?



Corrigé :

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

O le créateur **O** le constructeur

O l'initiateur **O** le fabricateur

---

b) On défini une classe de la manière suivante

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

Puis on exécute

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

Comment afficher la valeur de l'attribut *nom* de *mon_objet_C* dans la console ?

O mon_objet_C(nom) **O** mon_objet_C[nom]

O mon_objet_C.nom **O** ce n'est pas possible

---

c) On ajoute les deux méthodes ci-contre à notre classe :

Comment s'appelle ces méthodes ?

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

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

O getter **O** mutateur

O setter **O** accesseur

---

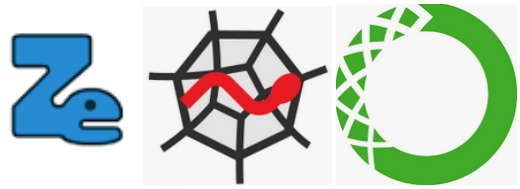
d) On souhaite ajouter un mutateur pour pouvoir modifier l'attribut valeur de nos instances d'objets de Classe C. Quelles sont les ou les codes corrects ?

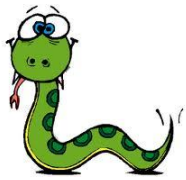
O
```
def setValeur(self):
    self.valeur = new_valeur
```

O
```
def set_Valeur(self,v):
    self.valeur = v
```

O
```
def modification(valeur):
    self.valeur = valeur
```

O
```
def setValeur(self,v):
    self.valeur = v
```






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")
```

O les méthodes __str__ __len__ et __contains__ sont des méthodes built-in

O `print(p)` appelle automatiquement la méthode __str__ de l'objet p

O Le Docstring ne joue aucun rôle particulier

O `'je' in p` est une écriture correcte ?

O `'JE' in p` renvoie True ?

O Dans la définition de la classe, *self* fait référence à l'instance de l'objet ?