



avec python<sup>TM</sup>

- Attributs privés,
- Propriétés,
- Setters et getters
- Try...except
- raiseError

# Encapsulation



- > Principe en programmation.
- > Concept fondamental en programmation objet.
- > Consiste en le regroupement de données, valeurs, et fonctions dans un bloc pour permettre la lecture et manipulation de ces données.
- > En programmation objet, on fait de l'encapsulation en utilisant des classes et des instances.

# Attributs privés



- > On peut vouloir restreindre l'accès à certaines valeurs hors de la classe.
- > En Python, la désignation d'un attribut comme étant privé est fait simplement en commençant son nom par un "\_" (un seul underscore)

```
class Employe:
    ... def __init__(self, nom, prenom, salaire):
    ...     self.nom = nom
    ...     self.prenom = prenom
    ...     self._salaire = salaire
```

Attributs publiques

Attribut privé

# Attributs privés



- Par standard, on n'appelle jamais les attributs commençant par un "\_" hors de la classe.

```
class Employe:
    ...def __init__(self,nom,prenom,salaire):
    ...    self.nom = nom
    ...    self.prenom = prenom
    ...    self._salaire = salaire

chimiste = Employe("Belatekallim","Tapputi","45k")

print(chimiste.nom)
print(chimiste.prenom)
print(chimiste.salaire)
```

PROBLÈMES

1

SORTIE

CONSOLE DE DÉBOGAGE

TERMINAL

.NET

Belatekallim

Tapputi

Traceback (most recent call last):

File "c:\Users\pierre-paul.gallant\Cégep Édouard-Montpe

\*Dans la plupart des langages on peut indiquer qu'un attribut est privé avec un mot-clé.  
Cet attribut n'est alors aucunement accessible hors de la classe.



# Propriétés ( getters )

- > Si on veut quand même avoir accès au salaire de l'employé, on va devoir utiliser un décorateur pour créer une propriété.
- > Une propriété se comporte généralement comme un attribut mais est définie à l'aide d'une méthode.

```
....@property  
....def salaire(self):  
....|....return self._salaire
```

```
print(chimiste.nom)  
print(chimiste.prenom)  
print(chimiste.salaire)
```

PROBLÈMES

SORTIE

CONSOLE DE DÉBOGAGE

TERMINAL

.NE

```
Belatekallim  
Tapputi  
45k
```



# Propriétés ( getters )

- > Dans cet exemple, la différence entre l'attribut et la propriété est:
- > nom et prénom sont des attributs et on peut changer leur valeur.
- > salaire est une propriété et sa valeur ne peut pas être changée.

```
chimiste.nom = "Tapi"  
print(chimiste.nom)
```

```
chimiste.prenom = "Bela"  
print(chimiste.prenom)
```

PROBLÈMES	SORTIE	CONSOLE DE DÉBOGAGE	<u>TERMINAL</u>
			Tapi Bela

```
chimiste.salaire = "50k"  
print(chimiste.salaire)
```

PROBLÈMES	SORTIE	CONSOLE DE DÉBOGAGE	TERMINAL
			Traceback (most recent call last): File "c:\Users\pierre-paul.gallant\Cégep Édouard (Pilote réseau)\R21\exemple.py", line 22, in <module> chimiste.salaire = "50k"



# setters

- > Les setters nous permettent de changer les valeurs des propriétés.
- > Il faut encore utiliser un décorateur et le comportement des setters sera défini par une méthode.

```
....@salaire.setter
....def salaire(self,nvx_salaire):
....|....if nvx_salaire > self._salaire:
....|....|....self._salaire = nvx_salaire
```

- > **N.B.** la syntaxe de ce décorateur est légèrement différente.  
@nom\_de\_la\_propriété.setter



# setters

- > Ce setter contrôle la façon dont on change la valeur du salaire.
- > Si la nouvelle valeur est inférieure à l'ancienne, la valeur du salaire n'est pas changée.

```
....@salaire.setter  
....def salaire(self,nvx_salaire):  
....|....if nvx_salaire > self._salaire:  
....|....|....self._salaire = nvx_salaire
```

```
chimiste = Employe("Belatekallim","Tapputi",45000)
```

```
chimiste.salaire = 30000  
print(chimiste.salaire)  
chimiste.salaire = 60000  
print(chimiste.salaire)
```

PROBLÈMES    SORTIE    CONSOLE DE DÉBOGAGE    TERMINAL    .NET I

45000  
60000





# setters

- > Ce setter contrôle la façon dont on change la valeur du salaire.
- > Si la nouvelle valeur est inférieure à l'ancienne, la valeur du salaire n'est pas changée.

```
....@salaire.setter  
....def salaire(self,nvx_salaire):  
....|....if nvx_salaire > self._salaire:  
....|....|....self._salaire = nvx_salaire
```

```
chimiste = Employe("Belatekallim","Tapputi",45000)
```

```
chimiste.salaire = 30000  
print(chimiste.salaire)  
chimiste.salaire = 60000  
print(chimiste.salaire)
```

PROBLÈMES

SORTIE

CONSOLE DE DÉBOGAGE

TERMINAL

.NET I

45000

60000



# proprieties delete

- > Les propriétés ne peuvent pas être supprimées normalement.
- > Le décorateur `@nom_de_la_propriété.delete` est nécessaire pour supprimer une propriété

```
....@salaire.delete
....def salaire(self):
....|....del self._salaire
```



# proprieties deleter

## > Sans deleter :

```
30
31 del chimiste.salaire
32 print(chimiste.salaire)
```

```
PROBLÈMES  SORTIE  CONSOLE DE DÉBOGAGE  TERMINAL

in <module>
  del chimiste.salaire
AttributeError: can't delete attribute 'salaire'
```

> Ne peut pas exécuter la ligne 31, ne peut pas supprimer l'attribut "salaire"

## > avec deleter :

```
23     ....@salaire.deleter
24     ....def salaire(self):
25     ....     ....del self._salaire
26
27     chimiste = Employe("Belatekallim", "Tapputi", 45000)
28
29     del chimiste.salaire
30     print(chimiste.salaire)
31
```

```
Traceback (most recent call last):
  File "c:\Users\pierre-paul.gallant\Cégep Édouard-Montpetit\CMT-420_Intro
in <module>
    print(chimiste.salaire)
  File "c:\Users\pierre-paul.gallant\Cégep Édouard-Montpetit\CMT-420_Intro
in salaire
    return self._salaire
AttributeError: 'Employe' object has no attribute '_salaire'. Did you me
```

> Ne peut pas faire l'impression, l'attribut a été supprimé.



# Property deleters

- > Puisque le comportement des deleters est déterminé par une fonction, on décide ce qui se passe lorsqu'on supprime un attribut.

```
....@salaire.delete  
....def salaire(self):  
....    self._salaire = 0
```

```
chimiste = Employe("Belatekallim", "Tapputi", 45000)
```

```
del chimiste.salaire  
print(chimiste.salaire)
```

- > Maintenant, supprimer la valeur "salaire" ne retire pas l'attribut, mais plutôt il réinitialise le salaire à 0.

PROBLÈMES   SORTIE   CONSOLE DE DÉBOGAGE   TERMINAL   .NET INTERACTIVE

0



# raise Error & Try ... except

- > Reprenons l'exemple de salaire.setter
- > La valeur du salaire change seulement lorsqu'on rentre un chiffre supérieur à l'ancien salaire.
- > Dans cet exemple il n'y a pas d'indications que le salaire n'a pas été modifié lorsqu'on passe un chiffre inférieur au salaire original.

```
....@salaire.setter  
....def salaire(self,nvx_salaire):  
....|....if nvx_salaire > self._salaire:  
....|....|....self._salaire = nvx_salaire
```

```
chimiste = Employe("Belatekallim","Tapputi")
```

```
chimiste.salaire = 30000  
print(chimiste.salaire)  
chimiste.salaire = 60000  
print(chimiste.salaire)
```

PROBLÈMES	SORTIE	CONSOLE DE DÉBOGAGE	TERMINA
	45000		
	60000		



# raise Error

- > Le mot-clefs **raise** nous permet de soulever une erreur dans les situations où le code fonctionne mais une erreur de logique a lieu.
- > L'erreur interrompt l'exécution du code et nous affiche un message de notre choix.

```
....@salaire.setter
....def salaire(self,nvx_salaire):
....    if nvx_salaire > self._salaire:
....        self._salaire = nvx_salaire
....    else:
....        raise ValueError("Le salaire doit être un chiffre.")
```

```
chimiste = Employe("Belatekallim","Tapputi",45000)
```

```
chimiste.salaire = "15k"
```

42

PROBLÈMES   SORTIE   CONSOLE DE DÉBOGAGE   TERMINAL   .NET INTERACTIVE   JUPY

Traceback (most recent call last):

```
File "c:\Users\pierre-paul.gallant\Cégep Édouard-Montpetit\CMT-420_In
in <module>
```

```
    chimiste.salaire = "15k"
```

```
File "c:\Users\pierre-paul.gallant\Cégep Édouard-Montpetit\CMT-420_In
in salaire
```

```
    if nvx_salaire > self._salaire:
```

# Try ... except



- > On peut aussi vouloir le comportement opposé.
- > Si on anticipe une erreur possible, on peut l'attraper à l'aide des mots-clefs **try** et **except** pour éviter d'interrompre l'exécution du programme lorsqu'une erreur survient.

```
.....@salaire.setter
.....def salaire(self,nvx_salaire):
.....    try:
.....        if nvx_salaire > self._salaire:
.....            self._salaire = nvx_salaire
.....    except TypeError:
.....        print("Vous devez entrer un montant en chiffres.")
```

# Try ... except

- Maintenant, le programme n'est pas interrompu lorsqu'on entre le mauvais type de données.

```
....@salaire.setter
....def salaire(self,nvx_salaire):
....    try:
....        if nvx_salaire > self._salaire:
....            self._salaire = nvx_salaire
....    except TypeError:
....        print("Vous devez entrer un montant en chiffres.")
```

```
chimiste = Employe("Belatekallim","Tapputi",45000)
```

```
chimiste.salaire = "15k"
print("Le programme n'est pas interrompu")
```

PROBLÈMES   SORTIE   CONSOLE DE DÉBOGAGE   TERMINAL   .NET INTERACTIVE

```
Vous devez entrer un montant en chiffres.
Le programme n'est pas interrompu
```