



TP – CHAPITRE 1 - Programmation orienté objet



Sommaire

TP1 - Exemple de classe.....	2
TP2 - Autre exemple de classe.....	3
TP3 - Autre exemple de classe.....	4
TP4 - Attribut de classe.....	5
TP5 - Encapsulation et attributs privés.....	5
TP6 - Rectangle avec turtle.....	6
TP7 - Rectangles avec turtle.....	6
TP8 - Rectangles et attribut de classe.....	6
TP9 - Héritage.....	7
TP10 - Héritage.....	7
TP11 - Héritage avec turtle.....	9
TP12 - Deuxième attribut de classe avec turtle.....	10
TP13 - Mini-projet 1 : Jeux de 52 cartes.....	10

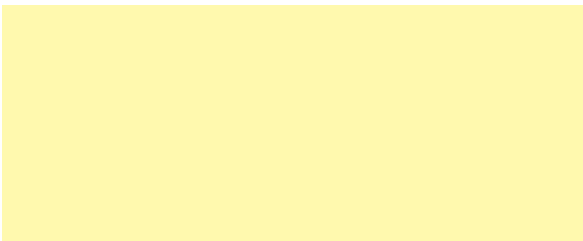
TP1 - Exemple de classe

- Sur votre compte, créer un dossier **NSI** , puis un sous-dossier **TP-Chapitre1** dans lequel on rangera les TP de ce chapitre.
- Récupérer le fichier **TP1 .py** donné en ressource et l'enregistrer dans le dossier **TP-Chapitre1** sur votre compte.
- Lire le code, puis remplir les zones ci-dessous, avec les attributs et les méthodes de la classe **Menu**.

Attributs :



Méthodes :



Lever la main pour valider ce TP.

TP2 - Autre exemple de classe

- Récupérer le fichier **TP2.py** donné en ressource et l'enregistrer dans le dossier **TP-Chapitre1** sur votre compte.
- Compléter le **pass** dans le constructeur **__init__()**.
- Compléter le **pass** dans la méthode **afficher()**.
- Compléter le **pass** dans la méthode **grandir()**.
- L'affichage attendu dans la console est suivant.

```
>>> %Run TP2.py
Espèce : sapin
Age : 5 ans
Taille : 3 mètres
Croissance annuelle : 0.5 mètres par an

Espèce : sapin
Age : 7 ans
Taille : 4.0 mètres
Croissance annuelle : 0.5 mètres par an
```



Lever la main pour valider ce TP.

TP3 - Autre exemple de classe

- Dans Thonny, créer un nouveau fichier **TP3.py** que l'on enregistrera dans le dossier **TP-Chapitre1** .
- En vous inspirant du TP précédent, créer une classe **Compte** avec les caractéristiques suivantes :

Classe Compte :

Attributs:

numero
solde

Méthodes :

afficher_solde(self)
credit(self, somme)
debit(self, somme)

- Pour voir si le programme fonctionne correctement, vérifier dans la console que l'on obtient bien les résultats affichés ci-après, en rajoutant après la définition de la classe les lignes de code ci-dessous.

```
24 compte1 = Compte(274569183746, 517.27)
25 compte1.afficher_solde()
26 compte1.credit(1000.00)
27 compte1.afficher_solde()
28 compte1.debit(500.00)
29 compte1.afficher_solde()
```

Console ×

```
>>> %Run TP3.py
```

```
Le solde du compte numéro 274569183746 est de 517.27
Le solde du compte numéro 274569183746 est de 1517.27
Le solde du compte numéro 274569183746 est de 1017.27
```

TP4 - Attribut de classe

- Dupliquer le fichier **TP2.py** en **TP4.py** .
- Rajouter un attribut de classe **compteur** pour pouvoir dénombrer le nombre d'instances de la classe **Arbre**.
- Rajouter une ligne dans le constructeur pour que le compteur augmente de 1 à chaque instantiation.
- Après la définition de la classe, supprimer les lignes de codes et les remplacer par quatre instantiations de 4 objets : **arbre1**, **arbre2**, **arbre3** et **arbre4**, en respectant les caractéristiques du fichier **arbres.csv** donné en ressources.
- En dernière ligne de code, afficher la valeur de l'attribut de classe **compteur**.
- Le retour attendu dans la console est :

```
>>> %Run TP4.py  
Le nombre d'instances de la classe Arbre est 4 .
```

 **Lever la main pour valider ce TP.**

TP5 - Encapsulation et attributs privés

- Dupliquer le fichier **TP4.py** en **TP5.py** .
- Mettre tous les attributs d'instance en privé, en rajoutant 2 underscores __ avant, dans toutes les méthodes de la classe.
- Rajouter à la fin du code les deux lignes ci-dessous et vérifier que l'attribut d'instance **arbre** n'a pas pu être modifié ainsi, donc qu'il est bien protégé.

```
arbre1.__espece = 'pin'  
arbre1.afficher()
```

- Le retour attendu dans la console est suivant.

```
Espèce : sapin  
Age : 5 ans  
Taille : 3 mètres  
Croissance annuelle : 0.5 mètres par an
```

 **Lever la main pour valider ce TP.**

TP6 - Rectangle avec turtle

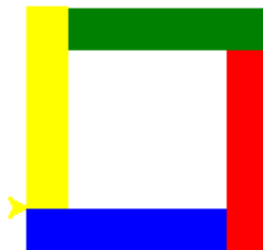
- Récupérer le fichier **TP6 .py** donné en ressource et l'enregistrer dans le dossier **TP-Chapitre1** sur votre compte.
- Compléter le **pass** dans la méthode **afficher()** de la classe **Rectangle**, afin de compléter le tracé du rectangle. Pour plus de précision sur le module turtle consulter le site : <https://docs.python.org/fr/3/library/turtle.html>.
- Le retour attendu dans la fenêtre qui s'affiche est suivant :



 **Lever la main pour valider ce TP.**

TP7 - Rectangles avec turtle

- Dupliquer le fichier **TP6 .py** en **TP7 .py** .
- A la fin du code créer trois autres instantiations de la classe **Rectangle** nommées **rectangle2**, **rectangle3** et **rectangle4**, afin d'obtenir la fenêtre qui s'affiche ci-dessous.



 **Lever la main pour valider ce TP.**

TP8 - Rectangles et attribut de classe

- Dupliquer le fichier **TP7 .py** en **TP8 .py** .
- Rajouter un attribut de classe **compteur** pour pouvoir dénombrer le nombre d'instances de la classe **Rectangle**.
- Rajouter une ligne dans le constructeur pour que le compteur augmente de 1 à chaque instantiation.
- A la fin du code, afficher la valeur de l'attribut de classe **compteur**.
- Le retour attendu dans la console est :

```
>>> %Run TP8.py
```

```
Le nombre d'instances de la classe Rectangle est 4 .
```

 **Lever la main pour valider ce TP.**

TP9 - Héritage

- Récupérer le fichier **TP9 .py** donné en ressource et l'enregistrer dans le dossier **TP-Chapitre1** sur votre compte.
- Compléter le **pass** dans le constructeur `__init__()` de la classe **Animaux**.
- Compléter le **pass** dans l'accesseur `get_vitesse()` de la classe **Animaux**.
- Compléter le **pass** dans la méthode `freiner()` de la classe **Animaux**.
- Compléter le **pass** dans le constructeur `__init__()` de la classe **Voiture**.
- Compléter le **pass** dans la méthode `est_une_voiture()` de la classe **Voiture**.
- Compléter le **pass** dans le constructeur `__init__()` de la classe **Moto**.
- Le retour attendu dans la console est suivant.

```
>>> %Run TP6.py
Peugeot : True ; vitesse : 80 km/h
Honda : False ; vitesse : 120 km/h
```



Lever la main pour valider ce TP.

TP10 - Héritage

- Créer un nouveau fichier **TP10 .py** dans le dossier **TP-Chapitre1** sur votre compte.
- Créer une première classe Mère nommée **Personne** dont les caractéristiques sont données ci-dessous.

Classe Personne :

Attributs d'instance :

`__nom`
`__prenom`

Accesseurs :

`get_nom()`
`get_prenom()`

Méthode polymorphe :

`presentation()`

- Créer ensuite une classe Fille nommée **Professeur** qui hérite de la classe mère **Personne** dont les caractéristiques sont données ci-dessous.

Classe Professeur :

Attributs hérités :

__nom

__prenom

Autre attribut :

__matiere

Accesseurs hérités :

get_nom()

get_prenom()

Autre accesseur :

get_matiere()

Méthode polymorphe :

presentation()

- Créer ensuite une classe Fille nommée **Eleve** qui hérite de la classe mère **Personne** dont les caractéristiques sont données ci-dessous.

Classe Eleve :

Attributs hérités :

__nom

__prenom

Autre attribut :

__classe

Accesseurs hérités :

get_nom()

get_prenom()

Autre accesseur :

get_classe()

Méthode polymorphe :

presentation()

- Créer une instance nommée **personne1** de la classe **Professeur** et une instance nommée **personne2** de la classe **Eleve** . Puis appeler pour ces deux instances, la méthode **présentation()** afin d'obtenir dans la console l'affichage suivant :

```
>>> %Run TP7.py
```

```
Bonjour je m'appelle Jean Dupont ; je suis professeur en Mathématiques  
Bonjour je m'appelle Jules Morel ; je suis élève en classe de TA
```



Lever la main pour valider ce TP.

TP11 - Héritage avec turtle

- Dupliquer le fichier **TP11.py** en **TP12.py** .
- Rédiger une classe nommée **Carre** qui hérite de la classe **Rectangle** et qui possède les caractéristique ci-dessous :

Classe Carre :

Attributs hérités :

abscisse
ordonnee
couleur

Autre attribut :

cote

Méthodes hérités :

afficher()

- Après l'instanciation des 4 rectangles, créer une instance de la classe **Carre** nommée **carre1**, puis l'afficher avec la méthode **afficher()** de façon à obtenir la fenêtre ci-dessous :



- En regardant dans la console, on voit que le nombre d'instance de la classe rectangle est maintenant passé à .



Lever la main pour valider ce TP.

TP12 - Deuxième attribut de classe avec turtle

- Dupliquer le fichier **TP11 .py** en **TP11 .py** .
- Rajouter un attribut de classe **compteur_couleur** dans la classe **Rectangle** pour pouvoir dénombrer le nombre de couleurs différentes utilisées pour les instances de la classe **Rectangle**.
- Rajouter un attribut de classe **liste_couleur** initialisé à une liste vide, dans la classe **Rectangle** pour pouvoir stocker les noms des couleurs déjà utilisées pour les instances de la classe **Rectangle**.
- A la fin du constructeur de la classe **Rectangle** rajouter une structure conditionnelle, pour rajouter la couleur utilisée dans la **liste_couleur** si c'est une nouvelle couleur et pour augmenter de 1 le **compteur_couleur**.
- Modifier les couleurs utilisées afin d'obtenir la figure ci-dessous :



- Modifier le code de la ligne qui affiche dans la console avec la fonction **print**, pour obtenir l'affichage ci-dessous dans la console :

```
>>> %Run TP12.py  
Le nombre de couleurs différentes est 3 .
```

- Modifier les couleurs utilisées et vérifier que le nombre de couleurs dans la console correspond bien au nombre de couleurs utilisées.



Lever la main pour valider ce TP.

TP13 - Mini-projet 1 : Jeux de 52 cartes

Voir le cahier des charges dans le fichier : mini_projet_1.pdf
