P2 Programmation Orientée Objet

Objectifs et sommaire :

•• Objectifs

faCheck Connaître le vocabulaire lié à la P.O.O : classes, instances, attributs, méthodes, objets.
faCheck Accéder aux attributs et méthodes d'une classe
faCheck Écrire la définition d'une classe
faCheck Résoudre un problème nécessitant la mise en oeuvre de la P.O.O

•• Sommaire

Vocabulaire

Définition

La programmation orientée objet (POO), ou programmation par objet, est un paradigme de programmation informatique.
Elle consiste en la définition et l'interaction de briques logicielles appelées objets.
Chaque objet représente un concept

Définition `attributs & méthodes`

Un objet est composé d'**`attributs`** et de **`méthodes`** :

— Les **`attributs`** sont les propriétés statiques de l'objet, à la manière des variables. Ce sont de données liées à l'objet
— Les **`méthodes`** définissent les interactions de l'objet. Ce sont des fonctions liées à l'objet

Exercice 1 ★

On considère l'objet `Sorcier` qui représente les sorciers.
Classe les différents éléments selon qu'il représente des `attributs` ou des `méthodes`.

âge
lancer un sort
prenom
nom
apprendre un sort

Attributs	Méthodes

Remarque

Le nom choisi pour les méthodes est souvent un verbe et représente généralement une action.

Définition `classes & instances`

Les `classes` sont les modèles générique qui permettent de définir les propriétés de chaque objet issue d'une même classe.
Un objet est une `instance` d'une classe.

Exercice 2 La classe Voiture

On considère la classe Voiture, qui représente une voiture.
Indiquer à l'aide d'une croix, le type d'élément auquel réfère chacun de ces noms.

	instance	attribut	méthode
nombre de roues			
clio			
vitesse maximum			
accélerer			
nombre de portes			
Puma			
freiner			
tourner			

Implémentation Python

•Instanciation d'un objet

Définition

Pour créer un objet, il faut l'`instancier`.
Comme pour les variables, on utilisera le signe =.

`nom_instance = Nom_de_la_classe()`

Exemple

Si 'lon dispose d'une classe `Sorcier`, on peut instancier des personnes à l'aide de cette classe.

```
1 sorcier_1 = Sorcier()
2 sorcier_2 = Sorcier()
```

Ici on instancie 2 objets de la classe `Sorcier` ayant pour références respectives `sorcier_1` et `sorcier_2`.

Exercice 3 ★

Quelle instruction permet d'instancier un troisième sorcier dont le nom de l'instance est `sorcier_3` ?

.....

•Manipuler les attributs d'une instance

i Un attribut est une variable lié à un objet, il est donc possible d'acceder, de modifier et d'ajouter des attributs à une instance.

•• Accéder un attribut

Définition

Une fois l'instance définie, on pourra accéder à la valeurs de ces attributs à l'aide des instructions du type :

nom_de_l_instance . nom_de_l_attribut

Exemple

On peut donc connaître le nom du `sorcier_1` grâce à l'instruction :

```
1 sorcier_1.nom
```

Exercice 4

Quelle instruction permet d'obtenir l'attribut `age` du `sorcier_3` ?

. . . . . .

•• Modifier un attribut

Définition

Une fois l'instance définie, on pourra modifier ces attributs à l'aide des instructions du type :

nom_de_l_instance . nom_de_l_attribut = nouvelle_valeur

Exemple

On peut donc modifier l'âge du `sorcier_1` et lui affecter la valeur 15 grâce à l'instruction :

```
1  sorcier_1.age = 15
```

Exercice 5 ★

Quelle instruction permet de modifier le nom du `sorcier_2` et de lui affecter la valeur Ron.

. . . . . .

•Utiliser une méthode

i Une méthode est une fonction liée à l'objet, il est donc possible de l'appeler à partir d'une instance

Définition

Une fois l'instance définie, on pourra appeler ces méthodes à l'aide des instructions du type :

nom_de_l_instance . nom_de_l_attribut = nouvelle_valeur

! Il faut penser à mettre les parenthèses

Exercice 6 ★

Quelle instruction permet d'utiliser la méthode `lancer` avec l'instance `sorcier_1` pour lancer le sort `"Abracadabra"` ?

. . . . . .

Définir une classe

i Avant d'instancier des objets, il faut définir la classe `Class` de l'objet.
C'est le moule qui permet de fabriquer ensuite nos objets.

•Le mot clé `class`

Le mot clé `class` permet de donner un nom à notre classe. Par la suite, l'instanciation d'un objet se fera à l'aide de ce nom.

`class` **Nom_de_la_classe**`:`

i Par convention, les noms de `class` commence par une **majuscule**.
On respectera les règles de nommage établies pour les variables.

Exemple

Le mot clé class

```
1 class Sorcier:
2     pass
```

•Le constructeur `__init__(self)`

i Le fait de devoir déclarer l'instance puis dans un second temps de définir les valeurs des attibuts n'est pas pratique.
Les objets disposent donc d'un `constructeur` qui va permettre de faire la création de l'instance et la définition des valeurs des attributs en une seule étape.

La méthode `__init__` est la méthode de construction de l'objet qui sera automatiquement exécutée au moment de la création d'une instance.
Nous utiliserons dans ce cours, parce que c'est la dénomination courante, le mot `self` comme premier argument de toutes méthodes et donc de la méthode `__init__`.

`def __init__(self):`

Exemple

```
1 class Sorcier:
2
3     def __init__(self):
4         self.prenom = "Harry"
5         self.nom = "Potter"
6         self.age = 11
7         self.__sorts = []
```

`self` représente l'instance de l'objet.

Exemple

Maintenant, si l'on instancie à nouveaudeux sorcier, à l'aide de la classe `Sorcier`.
On pourra afficher directement leur prénom (`prenom`)

```
1  sorcier1 = Sorcier()
2  sorcier2 = Sorcier()
3
4  print(sorcier1.prenom)
5  print(sorcier2.prenom)
```

On remarquera que les deux istances étant issue de la classe `Sorcier`, elles possèdent les mêmes attributs.
Ainsi `sorcier1.prenom` et `sorcier2.prenom` vont renvoyer `"Harry"`.

Ce n'est pas forcement le comportement voulu.

•Paramètres de la méthode `__init__`

Pour pouvoir créer différents sorciers, on va placer en paramètre les attributs que l'on souhaite attribuer à chaque instance.

```
1  class Sorcier:
2
3      def __init__(self,prenom, nom, age):
4          self.prenom = prenom
5          self.nom = nom
6          self.age = age
7          self.__sorts = []
```

Ce qui va changer la déclaration d'une instance :

```
1  harry = Sorcier("Harry","Potter","Griffondor",11)
```

Exercice 7

Quelle instruction permettra de définir `Hermione` une instance de l'objet sorcier sachant que Hermione à 10 ans s'appelle Hermione Granger.

.....

.....

.....

•Créer une méthode

Il est possible de créer d'autres méthodes sur un objet.
⚠Le premier paramètre reste le mot clé `self`

```
1  class Sorcier():
2
3      def __init__(self,prenom, nom, maison, age):
```

```
        self.prenom = prenom
        self.nom = nom
        self.maison = maison
        self.age = age
        self.sorts = []

    def presentation(self):
        print(f"""Bonjour je m'appelle {self.prenom} {self.nom}  et j'ai {self.age}""")
```

Exercice 8 ★

Quel sera l'affichage obtenu après l'éxécution de ce code ?

```
harry = Sorcier("Harry","Potter",11)
harry.presentation()
```

.....

.....

.....

Exercice 9 ★

Ecrire une méthode **`apprendre`** qui prend en paramètre **`sort`** le nom d'un sort et qui l'ajoute à la liste des sorts de l'instance **`Sorcier`** concernée.

.....

.....

.....

.....

.....

.....

Encapsulation

i On peut vouloir que certains attributs ne soient pas directement accessible aux utilisateur.
On peut aussi vouloir vérifier que les valeurs données à certains attributs respect des régles

Définition `Encapsulation`

L'encapsulation est le regroupement de données et de méthodes dans une classe afin que vous puissiez masquer les informations et restreindre l'accès de l'extérieur.

Définition `Attribut privé`

Les attributs privés sont des attributs auxquels on ne peut pas accéder en dehors de la classe (c'est à dire la déclaration de la classe). Les accès et les modifications peuvent être possible à condition d'avoir été prévus.

En Python, les attributs privés sont précédés d'un ou deux espaces soulignés.

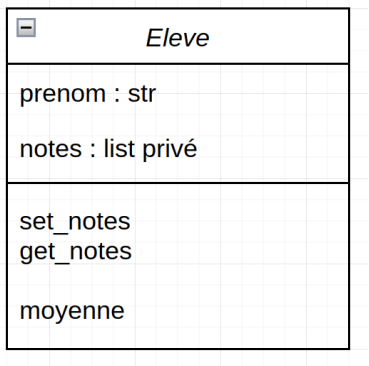
`self.__nom_de_l_attribut`

Définition `Accesseurs et mutateur`

- Les **accesseurs** sont des méthodes permettant d'accéder aux données d'un objet, parfois `getter` (appellation d'origine anglophone)
- Les **mutateur** méthodes permettant d'accéder aux données d'un objet, parfois `setter`

Exercice 10 ★

Créer une classe élève (Eleve) dont les spécificités sont données dans le schéma ci-dessous.



Les notes ne peuvent qu'être comprise entre 0 et 20.