CentraleSupélec — Département informatique
Plateau de Moulon
3 rue Joliot-Curie
F-91192 Gif-sur-Yvette cedex

1CC1000 - Systèmes d'Information et Programmation - TD : programmation orientée objet avec héritage

Présentation

Nous allons nous intéresser à la modélisation de réactions chimiques.

Par exemple, en présence de molécules de méthane et de dioxygène, une réaction de combustion va produire du dioxyde de carbone et de l'eau :

$$CH_4 + O_2 + O_2 \rightarrow CO_2 + H_2O + H_2O$$

Quelques rappels :

- Un **atome** est constitué de nucléons (les **protons** et les **neutrons**) et d'électrons. Pour cet exercice, nous ignorerons les électrons.
  Le **nombre de masse** d'un atome est son nombre de nucléons.
- Les atomes ayant le même nombre de protons sont classés par un **élément chimique** : les atomes d'hydrogène ont un proton, ceux d'oxygène huit. Le nombre de protons est nommé **numéro atomique**.
- Les atomes caractérisés par un même élément chimique peuvent différer par leur nombre de neutrons, ce sont des **isotopes**.
  Par exemple, il existe 3 isotopes de l'hydrogène : le protium n'a pas de neutron, le deutérium a un neutron et le tritium a 2 neutrons.
- Une molécule est un ensemble d'atomes (nous ne tiendrons pas compte des isomères).

Composant chimique

Nous nommerons *composant chimique* les entités que nous allons représenter, à savoir les atomes et les molécules. Un composant chimique a un nom est une masse (la masse d'un composant chimique sera le nombre de masse pour un atome, la somme des nombres de masse des atomes la constituant pour une molécule).

→ Définir la classe **`ChemicalComponent`** ; cette classe étant abstraite, elle sera une sous-classe de la classe **`ABC`** du module **`abc`**.

→ Définir le constructeur de cette classe, il recevra le nom comme argument. Ce nom ne sera pas modifiable après l'initialisation, donc il sera mémorisé dans un attribut protégé (commençant par _), et accessible comme une propriété (décorateur `@property` sur l'accesseur).

→ Définir une méthode abstraite (décorateur `@abstractmethod`) **`mass()`**.

→ Essayer de créer une instance de **`ChemicalComponent`**, vérifier qu'une erreur est déclenchée.

Élément chimique

Un élément chimique est un composant chimique constitué d'un seul atome dont on connaît le nombre de protons.

Nous pourrions mémoriser ce nombre dans un attribut non modifiable avec la même approche que celle retenue pour le nom. Cependant, pour rendre plus facile l'utilisation de notre code, nous allons proposer directement la création d'un atome d'hydrogène, plutôt que la création d'un atome avec un seul proton. Ce qui revient à dire que nous aurons une classe **`Hydrogen`** (même chose pour les autres éléments chimiques). Le nombre de protons sera une caractéristique de cette classe, et non pas d'une instance particulière de celle-ci (tous les atomes d'hydrogène ont un proton). Un attribut de classe ne peut pas être redéfini dans une sous-classe, contrairement à une méthode de classe, c'est donc la solution que nous utiliserons.

Un élément chimique définit aussi un symbole (**`H`** pour l'hydrogène, **`O`** pour oxygène…), nous utiliserons la même approche pour cette caractéristique.

→ Définir la classe **`ChemicalElement`** ; cette classe sera aussi abstraite, sous-classe de **`ChemicalComponent`**.

→ Définir le constructeur de cette classe, il recevra le nom comme argument.

→ Définir les méthodes de classe (décorateur `@classmethod`) **`atomic_number()`** et **`symbol()`**.

→ Vérifier qu'il n'est toujours pas possible de créer une instance de cette classe.

Atome

Un atome est un élément chimique dont on connaît le nombre de neutrons. Ce nombre sera non modifiable après son initialisation, mémorisé dans un attribut privé.

→ Définir la classe **`Atom`** ; cette classe sera une sous-classe de la classe **`ChemicalElement`**.

→ Définir le constructeur de cette classe, il recevra comme arguments le nom et le nombre de neutrons.

→ Définir la méthode **`neutron_number()`** et redéfinir la méthode **`mass()`** avec la bonne sémantique.

→ Vérifier qu'il est maintenant possible de créer une instance de cette classe (exemple d'instance : deutérium), afficher son nom, son nombre de neutrons.

→ Pour autant, cette classe n'est pas destinée à être instanciée directement, car son nombre de protons n'est pas encore connu : vérifier ceci en essayant d'afficher la masse de l'atome que vous avez créé.

Hydrogène

L'hydrogène est l'élément chimique ayant 1 comme numéro atomique et H comme symbole.

Il existe trois isotopes de l'hydrogène : le protium n'a pas de neutron (99,99% de l'hydrogène présent sur la Terre), le deutérium en a un (0,01%) et le tritium deux (n'existe pas à l'état naturel, instable).

→ Définir la classe **`Hydrogen`** ; cette classe sera une sous-classe de la classe **`Atom`**.

→ Définir le constructeur de cette classe, il recevra comme arguments le nom (qui sera par défaut Hydrogen) et le nombre de neutrons (0 par défaut). Si ce dernier n'est pas valide, une exception `TypeError` sera émise.

→ Redéfinir les méthodes **`atomic_number()`** et **`symbol()`** avec la bonne sémantique.

→ Créer un atome de deutérium et afficher sa masse.

→ Afficher l'atome de deutérium créé. Le résultat qui ressemble à

```
<__main__.Hydrogen object at 0x1043e6390>
```

n'est pas satisfaisant.

Nous n'allons pas essayer de l'afficher sous la forme officielle (^2_1H), nous retiendrons l'écriture `[1/2]H` : le premier nombre est le numéro atomique, le second est le nombre de masse, et la lettre est le symbole.

Quand Python veut convertir une donnée en chaîne de caractères pour l'afficher, il va utiliser la méthode **`__str__()`** qui a une définition par défaut dans la classe **`object`**, la racine de l'arbre d'héritage. Nous allons donc redéfinir cette méthode pour obtenir l'affichage souhaité.

→ Dans quelle classe est-il pertinent de redéfinir cette méthode ? Ajouter cette redéfinition.

Autres atomes

Pour la réaction chimique que nous souhaitons modéliser, nous avons besoin des atomes de carbone (symbole C, 6 protons, 6 à 8 neutrons, l'isotope le plus courant est ^{12}C) et d'oxygène (symbole O, 8 protons, 6 à 20 neutrons, l'isotope le plus courant est ^{16}O).

→ Définir les classes **`Carbon`** et **`Oxygen`**.

Molécule

Une molécule est un composant chimique constitué d'atomes, au moins deux.

→ Définir la classe **`Molecule`**, sous-classe de **`ChemicalComponent`**.

→ Définir le constructeur de cette classe, il recevra un nom et la liste d'atomes comme arguments, celle-ci sera mémorisée dans un attribut privé. Le constructeur vérifiera que la liste contient bien uniquement des atomes (sinon une exception `TypeError` sera émise) ; pour cela, il est possible d'utiliser la function

```
isinstance(une_instance, UneClass)
```

qui teste si `une_instance` est une instance de la classe `UneClasse` ou de l'une de ses sous-classes.

→ Redéfinir la méthode **`mass()`** avec la bonne sémantique.

→ Créer une molécule d'eau, et afficher sa masse.

→ Afficher cette molécule, le résultat n'est pas satisfaisant.

L'objectif est d'obtenir un affichage sous la forme :

```
H2O
```

c'est à dire afficher chaque type d'atome de la molécule, suivi s'il y en a plusieurs de ce type du nombre d'occurence. On ne cherchera pas à respecter un ordre pour les types d'atomes.

→ Définir dans la classe **`Molecule`** une méthode retournant un dictionnaire décrivant la composition d'une molécule en terme de types d'atomes.

→ Redéfinir la méthode **`__str__()`** de cette classe pour obtenir l'affichage souhaité.



Il est possible d'utiliser les classes Python comme clefs du dictionnaire.
La classe d'un objet est obtenue par la fonction **`type()`**.

Eau et autres molécules

Nous avons besoin des sortes de molécules suivantes : méthane, dioxygène, dioxyde de carbone, eau.

→ Définir les classe **`Methane`**, **`Dioxygen`**, **`CarbonDioxide`** et **`Water`**, sous-classes de **`Molecule`**.

→ Définir les constructeurs de ces classes, ils recevront un nom, avec le nom de la classe comme valeur par défaut, et la liste d'atomes comme arguments ; la valeur par défaut de la liste sera une liste des atomes constituant ces molécules. Le constructeur vérifiera que la liste contient bien uniquement les bons atomes, une exception `TypeError` sera émise dans le cas contraire.

Réaction chimique

Nous modéliserons une réaction chimiques sous la forme d'une fonction **`ChemicalReaction`**.

→ Cette fonction recevra une liste de molécules.

→ Elle cherchera dans cette liste si elle trouve des molécules permettant de provoquer une réaction chimique (nous nous limiterons à la réaction chimique de combustion mentionnée au début du sujet).

→ Si c'est le cas, elle retirera de la liste ces molécules, exécutera la réaction et retournera la liste des molécules produites. Dans le cas contraire, elle retournera **`None`**. Les molécules créées seront construites à partir des atomes des molécules à l'origine de la réaction chimique.

Nous savons comment créer les molécules produites par la réaction chimique, grace aux constructeurs, mais nous n'avons pas pour l'instant de moyen de récupérer les atomes des molécules à l'origine de la réaction chimique.

→ Ajouter dans la classe **`Molecule`** une méthode **`destroy()`** qui vide la molécule de ses atomes et retourne une liste de ceux-ci.