Informatique 2: Travaux Pratiques

Programmation Orientée Objet 1/3 – Classes et Objets

Exercices de base

Le but de cette séance est de se familiariser avec la création de classes et la manipulation d'instances.
Procéder par étape et tester votre code (dans un bloc main) pour chaque étape. Un dossier contenant des fichiers de test vous est fourni sur Moodle pour que vous puissiez tester votre code.
Temps mentionné (🕓) à titre strictement indicatif.

Fraction

Le but de ces exercices est de construire une classe **Fraction** complète pour manipuler des fractions. Une fraction contient un numérateur et un dénominateur (non nul).

Question 1: (🕓 *5 minutes*) Dans un module *fraction.py*, définir une classe **Fraction** (vide) et un bloc **main** (vide) pour effectuer les tests de cette classe.

Question 2: (🕓 *10 minutes*) Déterminer quels attributs doit contenir un objet **Fraction**. Définir un constructeur (méthode **__init__**) qui prendra en paramètres des valeurs pour initialiser les attributs de l'objet **Fraction**.

> **⚠ Attention**
>
> Effectuer des tests dans le bloc **main** ! (création d'instances)

Question 3: (🕓 *5 minutes*) Ajouter des valeurs par défaut pour les arguments du constructeur. Déterminer des valeurs pertinentes (par exemple, la valeur par défaut devrait être une fraction nulle ; si un seul argument est passé, la fraction devrait être égale à l'entier correspondant).

Question 4: (🕓 *5 minutes*) Lever une exception dans le cas où le dénominateur est égal à 0.

> **⚠ Attention**
>
> Les exceptions doivent être interceptées au plus tard dans le bloc **main**.

Question 5: (🕓 *5 minutes*) Convertir les arguments en entiers (avec **int**) avant de les affecter aux attributs de telle sorte qu'une exception soit levée si les arguments ne sont pas d'un type qui peut être convertit en entier (vous n'avez pas à lever d'exception vous-même, **int** s'en chargera).

Question 6: (🕓 *5 minutes*) Définir la méthode d'instance **__str__** pour produire une représentation textuelle lisible d'un objet **Fraction**. Test en utilisant **print**

> **>_ Exemple**
>
> **'4/3'** pour l'objet Fraction(4,3).

Question 7: (🕓 *5 minutes*) Définir une méthode d'instance **value**, qui renvoie la valeur de l'objet **Fraction** sous forme de nombre décimal de type **float**.

Attention

— Définir la fonction *value* **SANS** les double *underscore* avant et après.
— Tester en s'assurant que `print(Fraction(5, 3).value())` renvoie un 1.66 (*float*)

Question 8: (*5 minutes*) Rendre les attributs de la classe privés (utiliser la fonction Refactor → Rename de PyCharm). **Tester** et noter les différences.

Question 9: (*10 minutes*) Générer les *getters* et les *setters* pour les attributs d'instance de la classe **Fraction** afin de les rendre à nouveau accessibles.

Question 10: (*15 minutes*) Définir une méthode d'instance "en place" **simplify**, qui réduit la fraction (en modifiant l'objet ; elle ne renvoie rien). Le numérateur et le dénominateur doivent être premiers entre eux. Le dénominateur doit être strictement positif.

>_Exemple

`'4/6'` devient `'2/3'` et `'1/–2'` devient `'–1/2'`.

Question 11: (*5 minutes*) Modifier les méthodes d'instance pour utiliser la méthode **simplify** après chaque modification de l'objet de telle sorte qu'un objet soit toujours représenté sous sa forme simplifiée.

Question 12: (*10 minutes*) Définir une méthode d'instance **add** qui prend en paramètre une instance de la classe **Fraction** et l'ajoute à l'instance considérée. La méthode **add** modifie l'objet directement et ne renvoie aucune valeur.

Question 13: (*5 minutes*) Définir une méthode d'instance **plus** qui prend en paramètre une instance de la classe **Fraction** et renvoie le résultat de l'addition de l'instance considérée et de l'objet passé en paramètre. Cette méthode ne doit pas modifier l'instance.

Question 14: (*10 minutes*) Définir la méthode d'instance **equal** qui prend en argument une instance de la classe **Fraction** et qui renvoie un booléen indiquant si l'instance considérée est égale à l'argument.

Attention

Les fractions `'–1/2'` et `'2/–4'` sont égales !

Location

Question 15: (*25 minutes*) Le but de cet exercice est de définir une classe **Location** représentant des lieux décrits par leurs coordonnées géographiques. projet

1. Définir une classe **Location** vide.
2. Définir son constructeur qui prendra en argument une latitude (`float`, comprise entre *-90* et *+90* ; lever une exception en cas de problème) et une longitude (`float`, comprise entre *-180* et *+180* ; lever une exception en cas de problème). Utiliser des attributs privés.
3. Définir la méthode d'instance **__str__** pour représenter une localisation sous forme textuelle, par exemple `"Location (latitude, longitude)"`.
4. Définir une méthode d'instance qui calcule la distance entre deux instances de la classe **Location** (en kilomètres) en utilisant la formule de la distance du grand cercle `https://fr.wikipedia.org/wiki/Distance_du_grand_cercle`.

   Attention

   La formule de la distance du grand cercle utilise des latitudes et longitudes en *radians*. Utiliser le module **math** pour les opérations mathématiques.

5. Tester en calculant la distance entre Paris et Lausanne

>_Exemple

```
geneve = Location(46.2044, 6.1432)
lausanne = Location(46.517738, 6.632233)
print(geneve.distance(lausanne)) affiche 51.20725889214411
```