UNIVERSITÉ RENNES 2

Programmation Orientée Objet

Xavier André & Romain Tavenard

1 Rappel : organisation de votre code

Pour ce TD, vous créerez un nouveau fichier `td_poo_b.py`. Dans ce fichier, votre code sera organisé de la manière suivante :

```
# Imports
from abc import ABC, abstractmethod, abstractproperty

# Classes et Fonctions
class [...]

# Tests
[...]
```

Notamment, vous définirez vos classes fonctions en début de fichier et les appels seront listés en fin de fichier. De cette manière, vous pourrez, d'une question à l'autre, réutiliser les classes et fonctions déjà codées au besoin.

2 La classe `Point`

Soit la classe suivante :

```
class Point:
    def __init__(self, x=0, y=0):
        self.x, self.y = x, y

    def __repr__(self):
        return f'Point(x={self.x},y={self.y})'
```

1. Copiez-collez ce code et ajoutez à cette classe une méthode qui calcule la distance entre le point représenté par l'instance courante et un autre point fourni en paramètre.

UNIVERSITÉ RENNES 2

3 Création d'une nouvelle classe

3.1 La classe `Intervalle`

Définissez une classe `Intervalle` telle que le code suivant fonctionne comme attendu (c'est-à-dire que l'on rentre dans le `if` si et seulement si `a` est compris entre 10 et 20 inclus) :

```
if a in Intervalle(10, 20):
    # [...]
```

Vous aurez pour cela besoin de définir la méthode spéciale `__contains__(self, v)` qui retourne `True` si `v` est "contenu dans" l'intervalle et `False` sinon.

3.2 La classe `Fraction`

1. Définissez une classe `Fraction` qui permette de représenter une fraction rationnelle. Créez un constructeur qui possède les caractéristiques suivantes :
   - Gestion de valeurs par défaut : numérateur et dénominateur initialisés à 1 ;
   - Interdiction d'instancier une fraction ayant un dénominateur nul ;
   - Définition de trois attributs d'instance :
     - `num` : Valeur absolue du numérateur ;
     - `den` : Valeur absolue du dénominateur ;
     - `signe` : Signe de la fraction (+1 ou -1).
2. Définissez la méthode spéciale `__repr__()`, permettant d'afficher la fraction.
   Exemple d'affichage : `(-5/10)`
3. Définissez les méthodes de surcharge d'opérateurs suivantes :
   - `__neg__(self)` retourne la fraction opposée ;
   - `__add__(self, other)` retourne la fraction somme ;
   - `__sub__(self, other)` retourne la fraction différence ;
   - `__mul__(self, other)` retourne la fraction produit.