Programmation orientée objet

March 11, 2025

Pour ce devoir maison, merci de renvoyer un fichier ipynb avec l'ensemble des cellules qui sont exécutés afin de faciliter la correction

1 Nombres rationnels

Un nombre rationnel est, en mathématiques, un nombre qui peut s'exprimer comme le quotient de deux entiers relatifs a et b.

On représentera donc les nombres rationnels par les deux nombres.

1. Écrire la classe Rational avec sa méthode `__init__` qui prend a et b comme argument et en fait des attributs.
2. Écrire la méthode `__repr__` et la méthode `__str__`. La première doit renvoyer une chaine de caractère comme `Rational(a, b)` alors que la seconde affichera `a/b`
3. Ajouter une méthode `evalf` qui donne une approximation sous forme d'un nombre à virgule flottant de a/b
4. Deux nombres rationnels représentés par $(a_1,\ b_1)$ et $(a_2,\ b_2)$ sont égaux si et seulement si $a_1b_2 = a_2b_1$). De la même façon que l'addition deux deux nombres fait appel à la méthode `__add__`, le test d'égalité fait appel à la méthode `__eq__`. Écrire la méthode `__eq__` et tester (par exemple) que `Rational(2, 4) == Rational(-1, -2)`.
5. Écrire et tester les méthodes `__mul__`, `__add__`, `__sub__`, `__truediv__`
6. Pour simplifier une fraction, il est possible de diviser le numérateur et le dénominateur par le PGCD. Le PGCD s'obtient à l'aide de la fonction suivante :

[1]:
```
def pgcd(a, b):
    while b>0:
        a, b = b, a % b
    return abs(a)
```

Écrire une fonction `simplifie(a, b)` qui simplifie le quotient a/b et l'intégrer dans le `__init__` de la classe afin de simplifier la fraction au moment de sa création.

7. Il est possible d'approximer le nombre π à l'aide de la formule suivante :

$$\pi = \sum_{k=0}^{\infty} u_k$$

avec

$$u_0 = 2$$

et

$$u_k = u_{k-1} \times \frac{k}{2k+1}$$

C'est à dire que $u_1 = \frac{2}{3}$, $u_2 = \frac{4}{15}$, $u_3 = \frac{4}{35}$, ...

Calculer une fraction qui approxime π en prenant les 100 premiers termes de la somme (on s'arrêtera donc à $k = 99$).

[2]:
```python
def pgcd(a, b):
    while b>0:
        a, b = b, a % b
    return abs(a)

def simplifie(a, b):
    if b<0:
        a, b = -a, -b
    factor = pgcd(a, b)
    return a//factor, b//factor

class Rational():
    def __init__(self, a, b):
        a, b = simplifie(a, b)
        self._a = a
        self._b = b

    def __repr__(self):
        return f"Rational({self._a}, {self._b})"

    def __str__(self):
        return f"{self._a}/{self._b}"

    def __eq__(self, other):
        if isinstance(other, int):
            other = Rational(other, 1)
        if isinstance(other, Rational):
            return self._a*other._b == self._b*other._a
        return NotImplemented

    def __mul__(self, other):
        if isinstance(other, int):
            other = Rational(other, 1)
        if isinstance(other, Rational):
            return Rational(self._a*other._a, self._b*other._b)
        return NotImplemented
```

```
    def __add__(self, other):
        if isinstance(other, int):
            other = Rational(other, 1)
        if isinstance(other, Rational):
            return Rational(self._a*other._b + other._a*self._b, self._b*other.
↪_b)
        return NotImplemented


    def __truediv__(self, other):
        if isinstance(other, int):
            other = Rational(other, 1)
        if isinstance(other, Rational):
            return Rational(self._a*other._b, self._b*other._a)

    def evalf(self):
        return self._a/self._b
```

[3]:
```
u = Rational(2, 1)
s = Rational(2, 1)
for k in range(1, 100):
    u = u * Rational(k, 2*k + 1)
    s = s + u
print(s)
```

```
16392931014549156490602385775691484945995869356887225439257329140356518912858771
161088/52180320054598741435560916892766306842882794366515661235898363539176054774
23845762175
```

Voici des exemples de fonctionnement de la librairie :

[4]:
```
q1 = Rational(21, 24)
q2 = Rational(13, 25)

print("Valeur de q1 (après simplification, question 6):", q1)
print("Représentation de q2 :", repr(q2))
print("Test de la méthode __eq__ :", q1==q2)

q3 = Rational(1, 2)/q2

print("Autre test", q2==Rational(1, 2)/q3)
print("Test de evalf", q2.evalf())
```

```
Valeur de q1 (après simplification, question 6): 7/8
Représentation de q2 : Rational(13, 25)
```

```
Test de la méthode __eq__ : False
Autre test True
Test de evalf 0.52
```

2 Gestion d'un magasin en ligne

On souhaite gérer un magasin en ligne. Ce magasin est composé de produit. L'inventaire permet de connaitre la quantité de chaque produit. Un client va créer un panier qui contient un ensemble de produit.

On va donc devoir créer les classes suivantes :

- `Produit` pour décrire un produit. On pourra lui donner comme attribut un nom, une couleur, le prix ainsi que le numéro unique de référence.
- `Inventaire`, contient une collection de produit ainsi que la quantité disponible.
- `Panier`. On crée un panier vide et on ajoute au fur et à mesure des produits.

Nous avons pris exemple sur le magasin décathlon. La référence, la couleur et le prix se trouvent sur la page de chaque produit.

Nous vous demandons de faire fonctionner votre code sur votre propre exemple en choisissant quelques produits (entre 5 et 10). Vous pouvez choisir un magasin autre que décathlon.

Ce sujet d'exercice ne rentre pas dans les détails, c'est à vous de construire les classes et les bonnes méthodes. Il est proche de l'exercice sur la bibliographie (correction sur la page github, à la fin).

N'oubliez pas les méthodes `__repr__` pour chaque classe. L'objectif est que le code suivant fonctionne.

[5]:
```python
class Produit:
    def __init__(self, reference, nom, couleur, prix):
        self.reference = reference
        self.nom = nom
        self.couleur = couleur
        self.prix = prix

    def __repr__(self):
        return f"Produit({self.reference!r}, {self.nom!r}, {self.prix!r}, {self.
 ↪couleur!r})"


class Inventaire:
    def __init__(self, item_list):
        self.item_list = item_list

    def get_product_from_ref(self, ref):
        for prod, _ in self.item_list:
            if prod.reference==ref:
                return prod
```

```
    def __repr__(self):
        return f"Inventaire({self.item_list!r})"

    def __str__(self):
        out = []
        out.append(f"{'Description':<35} | {'Référence':<10} | {'Quantite':
↪>10}")
        out.append('-'*71)
        for item, quantite in self.item_list:
            out.append(f"{item.nom:<35} | {item.reference:<10} | {quantite:
↪>10}")
            out.append(f"{item.couleur:<35} | {'':<10} | {'':>10}")
            out.append(f"{'':<35} | {'':<10} | {'':>10}")
        return '\n'.join(out)

    def est_disponible(self, panier):
        for prod, qt in self.item_list:
            if panier.item_list.get(prod.reference, ("", 0))[1]>qt:
                return False
        return True

    def enleve(self, panier):
        for i, (prod, qt) in enumerate(self.item_list):
            quantite_a_suprimer = panier.item_list.get(prod.reference, ("",␣
↪0))[1]
            self.item_list[i] = (prod, qt-quantite_a_suprimer)


class Panier:
    def __init__(self):
        self.item_list = {}

    def ajoute(self, produit, quantite):
        ref = produit.reference
        if ref not in self.item_list:
            self.item_list[ref] = (produit, 0)
        self.item_list[ref] = (self.item_list[ref][0],
                               self.item_list[ref][1] + quantite)

    def supprime(self, produit):
        del self.item_list[produit.reference]

    def calculate_total(self):
        out = 0
        for item, quantite in self.item_list.values():
            out += item.prix*quantite
        return out
```

```
    def __repr__(self):
        return f"Panier({self.item_list})"


    def __str__(self):
        out = []
        out.append(f"{'Description':<35} | {'Référence':<10} | {'Quantite':>10}␣
↪| {'Prix':>10}")
        out.append('-'*len(out[0]))
        for item, quantite in self.item_list.values():
            out.append(f"{item.nom:<35} | {item.reference:<10} | {quantite:>10}␣
↪| {item.prix*quantite:10.02f}")
            out.append(f"{item.couleur:<35} | {'':<10} | {'':>10} | {'':>10}")
            out.append(f"{'':<35} | {'':<10} | {'':>10} | {'':>10}")
        out.append('-'*len(out[0]))
        out.append('-'*len(out[0]))
        out.append(f"{'Total':<35} | {'':<10} | {'':>10} | {self.
↪calculate_total():10.02f}")
        return '\n'.join(out)
```

[6]:
```
polaire_rando_gris_asphalte = Produit('8841686',
                                      'Veste polaire chaude de randonnée',
                                     40,
                                     'gris asphalte')

polaire_rando_gris_asphalte = Produit('8526050',
                                      'Veste polaire chaude de randonnée',
                                      'noir fumé',
                                     40)

polaire_chasse_vert_kaki = Produit('8753993',
                                   'Polaire de chasse chaude',
                                  'vert kaki',
                                  55)

chaussettes_invisibles_blanc = Produit('8933890',
                                       'Chaussettes invisibles',
                                       'blanc',
                                       10)

short_tennis_orange = Produit('8916827',
                              'Short de tennis homme respirant',
                              'orange abricot',
                              15)
```

```
inventaire = Inventaire([
    (polaire_rando_gris_asphalte, 2),
    (polaire_rando_gris_asphalte, 4),
    (polaire_chasse_vert_kaki, 2),
    (chaussettes_invisibles_blanc, 30),
    (short_tennis_orange, 2)
])

print(inventaire)
```

```
Description                          | Référence  |   Quantite
-------------------------------------------------------------------------
Veste polaire chaude de randonnée    | 8526050    |          2
noir fumé                            |            |
                                     |            |
Veste polaire chaude de randonnée    | 8526050    |          4
noir fumé                            |            |
                                     |            |
Polaire de chasse chaude             | 8753993    |          2
vert kaki                            |            |
                                     |            |
Chaussettes invisibles               | 8933890    |         30
blanc                                |            |
                                     |            |
Short de tennis homme respirant      | 8916827    |          2
orange abricot                       |            |
                                     |            |
```

[7]:
```
panier = Panier()
# ajoute 5 vestes polaires kaki, ...
panier.ajoute(polaire_chasse_vert_kaki, 5)
panier.ajoute(chaussettes_invisibles_blanc, 4)
panier.ajoute(short_tennis_orange, 2)

# Indique si tout le pannier est disponible
print("Le panier est-il disponible ?", inventaire.est_disponible(panier))

# Suprime toutes les vestes polaires kaki
panier.supprime(polaire_chasse_vert_kaki)

print("Le panier est-il disponible ?", inventaire.est_disponible(panier))
```

```
Le panier est-il disponible ? False
Le panier est-il disponible ? True
```

[8]:
```
print(panier)
```

```
Description                          | Référence  |   Quantite |       Prix
```

```
-------------------------------------------------------------------------
Chaussettes invisibles              | 8933890    |          4 |      40.00
blanc                               |            |            |
                                    |            |            |
Short de tennis homme respirant     | 8916827    |          2 |      30.00
orange abricot                      |            |            |
                                    |            |            |
-------------------------------------------------------------------------
-------------------------------------------------------------------------
Total                               |            |            |      70.00
```

[9]:
```
# Met à jour l'inventaire en enlevant les produit du panier
inventaire.enleve(panier)
print(inventaire)
```

```
Description                         | Référence  |   Quantite
-------------------------------------------------------------------------
Veste polaire chaude de randonnée   | 8526050    |          2
noir fumé                           |            |
                                    |            |
Veste polaire chaude de randonnée   | 8526050    |          4
noir fumé                           |            |
                                    |            |
Polaire de chasse chaude            | 8753993    |          2
vert kaki                           |            |
                                    |            |
Chaussettes invisibles              | 8933890    |         26
blanc                               |            |
                                    |            |
Short de tennis homme respirant     | 8916827    |          0
orange abricot                      |            |
                                    |            |
```