Lista 1

Wykonał: Sergei Raikov
Numer albumu: 255908

Opis zadania

Zadanie polega na stworzeniu klasy ulamkow z mozliwoscia porownywania, dodawania, mnozenia, dzielenia, odejmowania pomiedzy soba

In [27]:

```
class Frac():
    """
    n - licznik
    d - mianownik
    c - całe częsci
    Klasa tworzy ułamki, daje możliwość porównywania z typem int.
    także da się porównać ułamki pomiędzy sobą
    Ułamki można dodawać odejmować dzielić oraz mnożyć
    """
    def __init__(self,n, d):
        self.n = n
        self.d = d
        if not d != 0: raise ZeroDivisionError
        if type(n) != int: raise ValueError
        if type(d) != int: raise ValueError
        if self.d < 0:
            self.n = -self.n
            self.d = -self.d
        greatest = gcd(abs(self.n), abs(self.d))
        self.n = (int(self.n/greatest))
        self.d = int(self.d/greatest)


    def get_num(self):
        """
        Metoda zwracajaca licznik ulamku
        """
        return self.n

    def get_den(self):
        """
        Metoda zwracajaca mianownik ulamku
        """
        return self.d


    def __neg__(self):
        """
        Metoda umozliwiajaca na negowanie ulamku
        po stworzeniu obiektu
        """
        return Frac(-self.n , self.d)


    def __str__(self):
        """
        Metoda pozwalajaca na reprezencje ulamku w postaci napisu
        """
        return str(self.n) + "/" + str(self.d)


    def __eq__(self, x):
        """
        Metoda pozwalajaca na sprawdzenie rownosci (==) ulamkow
        @x - ulamek z ktorym porownujemy
        Zwraca True albo False
        """
        x = check_int(x)
        a = self.__str__()
        b = x.__str__()
        return a == b
```

```
        x = check_int(x)
        return not y.__eq__(x)

    def __lt__(self, x):
        """
        Metoda pozwalajaca na sprawdzenie mneijszosci (<) ulamkow
        @x - ulamek z ktorym porownujemy
        Zwraca True albo False
        """
        y = self
        x = check_int(x)
        return y.get_num() * x.get_den() < y.get_den() * x.get_num()

    def __le__(self, x):
        """
        Metoda pozwalajaca na sprawdzenie mneijszosci i rownosci (<=) ulamk
        @x - ulamek z ktorym porownujemy
        Zwraca True albo False
        """
        y = self
        x = check_int(x)
        return not y.__gt__(x)

    def __gt__(self, x):
        """
        Metoda pozwalajaca na sprawdzenie wieksosci i rownosci (>=) ulamkow
        @x - ulamek z ktorym porownujemy
        Zwraca True albo False
        """
        y = self
        x = check_int(x)
        return y.get_num() * x.get_den() > y.get_den() * x.get_num()

    def __ge__(self, x):
        """
        Metoda pozwalajaca na sprawdzenie wiekszosci (>) ulamkow
        @x - ulamek z ktorym porownujemy
        Zwraca True albo False
        """
        y = self
        x = check_int(x)
        return not y.__lt__(x)

    def __add__(self, x):
        """
        Metoda pozwalajaca na dodawanie ulamkow
        @x - ulamek do dodania
        Zwraca obiekt klasy Frac
        """
        y = self
        x = check_int(x)
        n = y.get_num() * x.get_den() + x.get_num() * y.get_den()
        d = y.get_den() * x.get_den()
        return Frac(n, d) #Tworzymy obiekt naszej klasy z nowymi danymi

    def __sub__(self, x):
        """
```

```
        Zwraca obiekt klasy Frac
        """
        y = self
        x = check_int(x)
        n = y.get_num() * x.get_num()
        d = y.get_den() * x.get_den()
        return Frac(n, d)

    def __rmul__(self, x):
        """
        Metoda pozwalajaca na mnozenie ulamkow przez int z prawej strony
        @x - ulamek do dodania
        Zwraca obiekt klasy Frac
        """
        y = self
        x = check_int(x)
        n = y.get_num() * x.get_num()
        d = y.get_den() * x.get_den()
        return Frac(n, d)

    def __truediv__(self, x):
        """
        Metoda pozwalajaca na dzielenie ulamkow
        @x - ulamek do dodania
        Zwraca obiekt klasy Frac
        """
        y = self
        x = check_int(x)
        n = y.get_num() * x.get_den()
        d = y.get_den() * x.get_num()
        return Frac(n, d)

def check_int(x):
    """
    Funkcja spradzajaca czy argument jest liczba typu int
    Jesli jest to przeksztalca ta liczbe w obiekt klasy Frac
    Umozliwia to mnozenie oraz porownywanie ulamkow z liczbami typu int
    """
    if isinstance(x, int):
        x = Frac(x, 1)
    return x

def gcd(a, b):
    """
    Znajduje najwiekszy wspolny dzielnik 2 liczb
    Umozliwia to skracanie ulamkow
    """
    if b > 0:
        return gcd(b, a%b)
```

Opis wynikow zadania

Zostala stworzona klasa Frac ktora pozwala na tworzenie ulamkow, na ktorych mozemy wykonywac podstawowe dzialania algebraiczne (dodawanie, mnozenie i td).

Rowniez wszystkie ulamki mozna porownywac pomiedzy soba a nawet z liczbami calkowitymi

Przy tworzeniu obiektow klasy (ulamkow) podajemy 2 argumenty (n, d) licznik i mianownik. Obie te liczby powinne byc liczbami calkowitymi

TRZEBA ROWNIEZ PAMIETAC ZE NIE DZIELIMY PRZEZ ZERO.

Jak bedziemy bardzo chcieli to dostaniemy wyjatek, rowniez jak w powyzszym przypadku przy podawaniu argumentow roznych od typu int (liczb calkowitych).

Kazdy z argumentow moze byc ujemny, w przypadku jak obydwa beda ujemne to dastaniemy ulamek dodatni

Zostaly tez napisane funkcje wspomagajace: check_int(x) ktora przyjmuje jeden argument i w przypadku jak ten argument bedzie liczba calkowita to przeksztalca go w ulamek klasy Frac. Zrobione to zostalo aby umozliwic dzialania algebraiczne oraz porownywanie pomiedzy liczbami calkowitymi oraz ulamkami. Druga funkcja gcd(a,b) przyjmuje dwie liczby i znajduje najwiekszy wspolny dzielnik tych dwoch liczb, daje nam to mozliwosc skracania ulamkow.

Dodatkowe komponenty klasy

Podstawowe dzialania algebraiczne z liczbami calkowitymi. Negowanie ulamkow (zmiana znaku). Porownywanie ulamkow z liczbami calkowitymi

In [20]:
```
f1 = Frac(1, 2)
f2 = Frac(-2, 4)
f3 = Frac(3, 7)
f4 = Frac(2, 12)
f5 = Frac(-4, 7)
f6 = Frac(4, -7)
f7 = Frac(256, -512)
f8 = Frac(-2, -5)
```

In [4]:
```
print(f1 + f2)
```

```
0/1
```

In [5]:
```
print(f1 - f2)
```

```
1/1
```

In [6]:
```
print(f1 - -f2)# poniewaz -f2 jest -(-(1/2))
```

```
0/1
```

In [12]:
```
print(-f1)
```

```
-1/2
```

In [8]:
```
print(f1*f3)
```

```
3/14
```

In [11]:
```
print(f1 / f2)
```

```
-1/1
```

In [9]:
```
print(f1 * 1) # dodane zostalo mnozenie przez liczby calkowite
```

```
1/2
```

In [10]:
```
print(2 * f2)
```

```
-1/1
```

In [13]:
```
print((f2 + 1) == f1)
```

```
True
```

In [14]:
```
-f1 <= f2
```

Out[14]:
```
True
```

In [15]:
```
print(f1 >= f2)
```

```
True
```

In [16]:
```
print(f1 > f2)
```
True

In [17]:
```
print(f1 < f2)
```
False

In [164…
```
print(12 * f1)
```
6/1

In [161…
```
f1 > f2
```
Out[161… True

In [94]:
```
f1 < -f2
```
Out[94]: False

In [80]:
```
print(f2*f7)
```
1/4

In [81]:
```
print(f2/f7)
```
1/1

In [83]:
```
print(f1>=f7)
```
True

In [87]:
```
print(f1*f4/f7)
```
-1/6

In [84]:
```
print(f1<=f2)
```
False

In [18]:
```
print(f1*f2/f3+f4-f5)
```
13/84

In [19]:
```
print(f1*f2-f3+f7/f5)
```
11/56

Wieksza czesc testow zostala sprawdzona na prawidlowosc recznie badz za pomoca