Sprawozdanie - Lista 1

October 19, 2021

Małgorzata Kowalczyk, Kamil Kowalski

Wersja podstawowa

Stworzyliśmy klasę przechowującą i umożliwiającą działania arytmetyczne i logiczne na ułamkach prostych. Poniżej prezentujemy przykładowe wywołania.

Zadanie 1. Wyświetlanie ułamków oraz podstawowe działania arytmetyczne (+,-,*,/).
W celu wykonania tego zadania, stworzyliśmy konstruktor oraz przeciążyliśmy operatory arytmetyczne.

[2]:
```
f1=Fraction(1,4)
f2=Fraction(1,2)
f3=f1+f2
print(f3)
```

```
3/4
```

[3]:
```
a=Fraction(4,7)
b=Fraction(10,3)
c=Fraction(5,9)
d=Fraction(-4,9)
e=Fraction(8,6)
f=Fraction(16,12)
```

[4]:
```
print(a*b)
```

```
40/21
```

[5]:
```
print(a/b)
```

```
6/35
```

[6]:
```
print(a-b)
```

```
-58/21
```

[7]:
```
print(Fraction(0,12)*Fraction(16,15))
```

```
0/1
```

Zadanie 2. Porównywanie ułamków (>,<,>=,<=,==,!=).

Porównywanie ułamków jest możliwe dzięki odpowiedniemu przeciążeniu operatorów logicznych.

[8]:
```
f1<f2
```

[8]:
```
True
```

[9]:
```
f1<=f2
```

[9]:
```
True
```

[10]:
```
f2>f3
```

[10]:
```
False
```

[11]:
```
d<c
```

[11]:
```
True
```

[12]:
```
d!=c
```

[12]:
```
True
```

[13]:
```
d==c
```

[13]:
```
False
```

[14]:
```
e==f
```

[14]:
```
True
```

[15]:
```
e>=f
```

[15]:
```
True
```

Zadanie 3. Wyświetlanie wartości licznika i mianownika.

Zaimplementowaliśmy również metody get_num i get_dem, które zwracają odpowiednio wartość licznika oraz mianownika.

[17]:
```
f3.get_num()
```

[17]:
```
3
```

[19]:
```
f3.get_dem()
```

[19]:
```
4
```

[20]:
```
d.get_num()
```

[20]:
```
-4
```

[21]:
```
d.get_dem()
```

[21]:
```
9
```

Zadanie 4. Obsługa błędów.

Nasz program jest także odporny na wyjątki. Zwraca je, gdy użytkownik poda mianownik równy 0 oraz gdy poda nieprawidłowe dane, takie jak słowo albo liczbę zmiennoprzecinkową.

[23]:
```
g=Fraction(5,0)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-23-674b06cfc904> in <module>
----> 1 g=Fraction(5,0)

<ipython-input-1-085438bd6e65> in __init__(self, num, den)
     16             self.denominator = den // gcd
     17         elif den == 0:
---> 18             raise ValueError("Can't divide by 0.")
     19         else:
     20             raise ValueError("Wrong data given.")

ValueError: Can't divide by 0.
```

[24]:
```
h=Fraction("e",8)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-24-470826019f02> in <module>
----> 1 h=Fraction("e",8)

<ipython-input-1-085438bd6e65> in __init__(self, num, den)
     18             raise ValueError("Can't divide by 0.")
     19         else:
---> 20             raise ValueError("Wrong data given.")
     21
     22     def __str__(self):

ValueError: Wrong data given.
```

[25]:
```
i=Fraction(3.5,6)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-25-eb7e9b136fbc> in <module>
----> 1 i=Fraction(3.5,6)

<ipython-input-1-085438bd6e65> in __init__(self, num, den)
     18             raise ValueError("Can't divide by 0.")
     19         else:
---> 20             raise ValueError("Wrong data given.")
     21
     22     def __str__(self):

ValueError: Wrong data given.
```

Zadanie 5. Porównywanie ujemnych ułamków.
Operatory porównań działają również dla ujemnych ułamków.

[26]:
```
j=Fraction(10,-3)
k=Fraction(-5, 3)
l=Fraction(8,7)
print(j>k)
print(j<k)
print(j>=k)
print(j<=k)
print(j==k)
print(j!=k)
print(l>k)
print(l>j)
```

```
False
True
False
True
False
True
True
True
```

Zadanie 6 i 7. Postać nieskracalna ułamka.
Wszystkie ułamki są wypisywane w postaci nieskracalnej. Aby to uzyskać, zaimplementowaliśmy dodatkową funkcję greatest_common_divisor(a, b), obliczającą największy wspólny dzielnik.

[27]:
```
m=Fraction(25,5)
n=Fraction(21,28)
print(m)
print(n)
```

```
5/1
3/4
```

Wersja rozszerzona

W celu dodania klasie Fraction dodatkowej funkcjonalności, stworzyliśmy funkcję float_to_fraction(x), która zamienia liczbę całkowitą lub ułamek dziesiętny na obiekt typu Fraction.

Zdjęte ograniczenie z zadania 4.
Pozbyliśmy się ograniczenia z zadania 4 i teraz możemy tworzyć ułamki poprzez podanie ułamka dziesiętnego w liczniku lub mianowniku. Dokonaliśmy tego wykorzystując funkcję float_to_fraction(x) oraz modyfikując konstruktor.

[29]:
```
print(FractionE(1.25,8))
```

```
5/32
```

[30]:
```
print(FractionE(1,5.75))
```

```
4/23
```

[31]:
```
print(FractionE(2.5,7.5))
```

```
1/3
```

[32]:
```
print(FractionE(0,5.5))
```

```
0/1
```

Operowanie na ułamkach zwykłych, dziesiętnych i liczbach całkowitych.
Dodaliśmy również możliwość wykonywania operacji na ułamkach zwykłych z dziesiętnymi lub liczbami całkowitymi. Wykorzystaliśmy do tego funkcję float_to_fraction(x) oraz przeciążyliśmy operatory arytmetyczne z prawej strony.

[33]:
```
print(FractionE(1,8) + 2)
```

```
17/8
```

[34]:
```
print(3.5 + FractionE(5.5,8))
```

```
67/16
```

[35]:
```
print(FractionE(1,8) * 8)
```

```
1/1
```

[36]:
```
print(FractionE(4,8) / FractionE(1,4))
```

```
2/1
```

Porównywanie ułamków zwykłych z dziesiętnymi.

Rozszerzona klasa umożliwia również porównywanie ułamków zwykłych z dziesiętnymi. Wykorzystaliśmy do tego funkcję float_to_fraction() oraz dodaliśmy instrukcję warunkową if podczas przeciążania operatorów logicznych.

[37]:
```
print(1.125!=FractionE(9,8))
```

```
False
```

[38]:
```
print(1>=FractionE(9,8))
```

```
False
```

[39]:
```
print(1.25<=FractionE(5,4))
```

```
True
```

[40]:
```
print(1.1256>FractionE(9,8))
```

```
True
```

[41]:
```
print(FractionE(9,8)<1.1256)
```

```
True
```

[42]:
```
print(FractionE(0,8)==0)
```

```
True
```

Zamienianie ułamków niewłaściwych na ułamki mieszane.

Ostatnią funkcjonalność, jaką dodaliśmy to umożliwienie zamiany ułamków niewłaściwych na ułamki mieszane. W tym celu stworzyliśmy funkcję mixed() wewnątrz naszej klasy.

[44]:
```
FractionE(7,3).mixed(False)
```

[44]:
```
'7/3'
```

[45]:
```
FractionE(7,3).mixed()
```

[45]:
```
'2(1/3)'
```

[46]:
```
FractionE(-4,3).mixed(False)
```

[46]:
```
'-4/3'
```

[48]:
```
FractionE(-4,3).mixed()
```

[48]:
```
'-1(1/3)'
```

[49]:
```
FractionE(0,3).mixed()
```

[49]:
```
'0/1'
```

[50]:
```
FractionE(-7,14).mixed()
```

[50]:
```
'-1/2'
```

[51]:
```
FractionE(11,7).mixed()
```

[51]:
```
'1(4/7)'
```

Github

https://github.com/github-kamilk/Lista-1