Wrocław, 19.10.2021r.

Lista zadań nr 1

Julia Mazur, 262296

Poniżej przedstawiam wyniki otrzymane po wykonaniu konkretnych zadań z listy:

Zadanie 1.
Stworzyłam klasę Fraction reprezentującą ułamki zwykłe oraz zaimplementowałam podstawowe operatory arytmetyczne:

```
f1 = Fraction(1,4)
f2 = Fraction(1,2)
```

```
print(f1+f2)
print(f1-f2)
print(f1*f2)
print(f1/f2)
```

```
3/4
-1/4
1/8
1/2
```

Zadanie 2.
Wprowadziłam możliwość porównywania ułamków:

```
print(f1==f2)
print(f1!=f2)
print(f1>f2)
print(f1<f2)
print(f1>=f2)
print(f1<=f2)
```

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

Zadanie 3.
Dodałam metody getNum oraz getDem:

```
print(f1.getNum())
print(f1.getDem())
```

```
1
4
```

Zadanie 4.

Jeśli podany przez użytkownika licznik lub mianownik ułamka jest niecałkowity to zgłaszany jest wyjątek:

```
print(Fraction(1.5,3))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-3eb22c955e11> in <module>
----> 1 print(Fraction(1.5,3))

~\lista1_dodatki.py in __init__(self, numerator, denominator)
     63         #---------checking-given-data---------------------------------
     64         if not isinstance(numerator, int) or not isinstance(denominator, int):
---> 65             raise TypeError('numbers for the fraction should be integers')
     66         if denominator == 0:
     67             raise ValueError('division by zero')

TypeError: numbers for the fraction should be integers
```

Zadanie 5.

Operatory porównań działają poprawnie w przypadku, gdy podany mianownik jest liczbą ujemną:

```
g = Fraction(1,-4)
f = Fraction(-2,-3)
```

```
print(f==g)
print(f!=g)
print(f>g)
print(f<g)
print(f>=g)
print(f<=g)
```

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

Zadanie 6.

Ułamek jest wypisywany na ekran w postaci nieskracalnej:

```
a = Fraction(7,21)
print(a)
```

```
1/3
```

Zadanie 7.

Ułamek jest przechowywany w postaci nieskracalnej:

```
b = Fraction(7,21)
print(b.getNum())
print(b.getDem())
```

```
1
3
```

Dodatkowo:

Poza częścią obowiązkową, stworzyłam nową klasę (FractionExtended), która:

1. pozwala na podawanie licznika i mianownika ułamka jako liczb rzeczywistych,

```
print(FractionExtended(1.5, 3))
print(FractionExtended(3, 3.5))
print(FractionExtended(3, 3.55))
```

```
1/2
6/7
60/71
```

2. umożliwia wykonywanie operacji pomiędzy ułamkami zwykłymi i liczbami rzeczywistymi (zarówno podstawowych operacji arytmetycznych jak i porównywania ich wartości),

```
c = FractionExtended(1,2)
print(c+3)
print(3.0+c)
print(c-3)
print(3.0-c)
print(c*3)
print(3.0*c)
print(c/3.0)
print(3/c)
```

```
7/2
7/2
-5/2
5/2
3/2
3/2
1/6
6/1
```

```
f = FractionExtended(1.5, 3.1)
g = FractionExtended(3.55, 3)
```

```
print(f == g)
print(f == f)
print(f != g)
print(f != f)
print('-------------------------')
print(f < g)
print(f > g)
print(f <= g)
print(f >= g)
```

```
False
True
True
False
-------------------------
True
False
True
False
```

```
print(f < 1)
print(f > 1.5)
print(f <= 0.2)
print(f >= 0)
print('-------------------------')
print(1 > f)
print(1.5 < f)
print(0.2 >= f)
print(0 <= f)
```

```
True
False
False
True
-------------------------
True
False
False
True
```

3. pozwala na wyświetlenie wartości w postaci ułamka niewłaściwego lub liczby mieszanej (służy do tego metoda mixed).

```
print(f.mixed(True))
print(g.mixed(True))
print(f.mixed(False))
print(g.mixed(False))
```

```
0;15/31
1;11/60
15/31
71/60
```