

Examples

Tue Herlau
tuhe@dtu.dk

November 7, 2022

1 Working with fractions (fractions.py)

In this problem, you have to implement the methods in the following class:

```
1  # cpp_course/fractions.py
2  class Fraction:
3      def __init__(self, n, m):
4          self.n = n
5          self.m = m
6
7      def __add__(self, other):
8          """ Computes  $f = f_1 + f_2$  (where  $f_1$  and  $f_2$  are both Fraction-objects and
9               $\hookrightarrow f_1=self, f_2=other$ ), see (Assignment 6, eq. (1)) and (Har20) """
10         return f
11
12     def __mul__(self, other):
13         # Overwrite to implement  $f = f_1 * f_2$ 
14         return f
15
16     def __truediv__(self, other):
17         # Overwrite to implement  $f = f_1/f_2$ , or more specifically  $self/other$ .
18         return f
19
20     def __str__(self):
21         """ Creates a string representation. You can use it as
22              $\hookrightarrow$  `print(str(Fraction(1,2)))` to output  $1/2$  """
23         return f"{self.n} / {self.m}"
```

When done, you should be able to create two fractions objects and add them using the `+`-operator, which will be transformed to a call of the form `f1.__add__(f2)` as so:

```
1  # cpp_course/fractions.py
2  f1 = Fraction(1, 2) # Represents 1/2
```

```

3     f2 = Fraction(3, 5) # Represents 3/5
4     print(f"Result of {f1} + {f2} is", f1+f2)

```

This fragment will produce the terminal output:

```

1 Result of 1 / 2 + 3 / 5 is 11 / 2

```

You may find it convenient to use the formula (see [Ber07]):

$$\frac{a}{b} + \frac{n}{m} = \frac{am + bn}{bm} \quad (1)$$

When done, implement the `from_string` function, that can simplify string expressions to fractions:

```

1 # cpp_course/fractions.py
2 s = " 1 / 4 * 1 / 2"
3 print("Result of", s, "is", from_string(s))
4 s = "5 / 2 div 10 / 3"
5 print("Result of", s, "is", from_string(s))

```

This fragment will produce the following terminal output:

```

1 Result of 1 / 4 * 1 / 2 is 1 / 8
2 Result of 5 / 2 div 10 / 3 is 15 / 20

```

```

1 >>>
2 >>> f1 = Fraction(1, 2) # Represents 1/2
3 >>> f2 = Fraction(3, 5) # Represents 3/5
4 >>> print(f"Result of {f1} + {f2} is", f1 + f2)
5 Result of 1 / 2 + 3 / 5 is 11 / 2
6 >>>
7 >>> # Now do some compound tests:
8 >>> s = " 1 / 4 * 1 / 2"
9 >>> print("Result of", s, "is", from_string(s))
10 Result of 1 / 4 * 1 / 2 is 1 / 8
11 >>> s = "5 / 2 div 10 / 3"
12 >>> print("Result of", s, "is", from_string(s))
13 Result of 5 / 2 div 10 / 3 is 15 / 20

```

References

[Ber07] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II*. Athena Scientific, 3rd edition, 2007.