Operatoröverlagring

Daniel Bosk

KTH EECS

1st October 2020

https://github.com/dbosk/intropy/classes/slides-more/examples

myfrac_base.py

```
3   class Fraction:
4       """ Class for fractions """
5       def __init__(self, nominator, denominator=1):
6           if isinstance(nominator, Fraction):
7               self.__nominator = nominator.nominator
8               self.__denominator = nominator.denominator * denominator
9           elif isinstance(nominator, int):
10              self.__nominator = nominator
11              self.__denominator = denominator
12          else:
13              raise TypeError(f"can't create fraction from {type(nominato
```

Example

```
frac = Fraction(1, 2)
int_as_frac = Fraction(2)
frac_from_frac = Fraction(Fraction(1, 2))
half_frac = Fraction(Fraction(1, 2), 2)
```

```
 3  class Fraction:

    ...

15      @property
16      def nominator(self):
17          """nominator getter"""
18          return self.__nominator
19
20      @property
21      def denominator(self):
22          """demoninator getter"""
23          return self.__denominator
24
```

Example

```
frac = Fraction(1, 2)
print(f"{frac.nominator}/{frac.denominator}")
```

```
class Class:
  @property
  def property(self):
    return self.__property

  @property.setter
  def property(self, value):
    self.__property = value

obj = Class()
print(obj.property)
obj.property = new_value
```

```
class Class:
  def get_property(self):
    return self.__property

  def set_property(self, value):
    self.__property = value

obj = Class()
print(obj.get_property())
obj.set_property(new_value)
```

```
3   class Fraction:

    ...
25      def __str__(self):
26          return f"{self.nominator}/{self.denominator}"
27
28      def __float__(self):
29          return self.nominator / self.denominator
30
```

Example

```
frac = Fraction(1, 2)
print(frac)
print(float(frac))
```

myfrac_add.py

Addition

```
3   class Fraction:
    ...
31      def __add__(self, other):
32          if isinstance(other, int):
33              return Fraction(self.nominator + other * self.denominator,
34                              self.denominator)
35          elif isinstance(other, Fraction):
36              return Fraction(self.nominator * other.denominator +
37                              other.nominator * self.denominator,
38                              self.denominator * other.denominator)
39          raise TypeError(f"can't add with type {type(other)}")
40
```

Example

```
frac_a = Fraction(1, 2)
frac_b = Fraction(1, 3)
print(frac_a + frac_b)
print(1 + frac_a) # funkar ej
```

```
3   class Fraction:

    ...
41      def __radd__(self, other):
42          return self.__add__(other)
43
```

Example

```
frac_a = Fraction(1, 2)
print(1 + frac_a)
```

myfrac_sub.py

```
3   class Fraction:

    ...

44      def __neg__(self):
45          return Fraction(-self.nominator, self.denominator)
```

Example

```
frac_a = Fraction{1, 2}
print(-frac_a)
```

```
3   class Fraction:

    ...

47      def __sub__(self, other):
48          return self + (-other)
```

Example

```
frac_a = Fraction{1, 2}
frac_b = Fraction{1, 4}
print(frac_a - frac_b)
print(1 - frac_a) # funkar ej
```

```
3   class Fraction:

    ...

50      def __rsub__(self, other):
51          return Fraction(other) - self
```

Example

```
frac_a = Fraction{1, 2}
print(1 - frac_a)
```

```
3   class Fraction:

    ...

53      def __mul__(self, other):
54          if isinstance(other, int):
55              return Fraction(self.nominator * other, self.denominator)
56          elif isinstance(other, Fraction):
57              return Fraction(self.nominator * other.nominator,
58                              self.denominator * other.denominator)
59          raise TypeError(f"can't multiply type {type(other)}")
```

Example

```
frac_a = Fraction{1, 2}
frac_b = Fraction{1, 3}
print(frac_a * frac_b)
```

```
3   class Fraction:

    ...

61      def __rmul__(self, other):
62          return self * other
```

Example

```
frac_a = Fraction{1, 2}
print(frac_a * 2)
print(2 * frac_a)
```

Exercise

- Hur kan vi implementera förkortning av bråk?
- Exempelvis $\frac{2}{6}$ bör ju skrivas $\frac{1}{3}$.