Computer Science 125
29 October 2020

1. **Python `Fraction` class from the text:**

```
def gcd(m, n):
    while m % n != 0:
        oldm = m
        oldn = n
        m = oldn
        n = oldm % oldn
    return n
```

m	n	oldm	oldn
9	12	9	12
12	9	12	9
9	3		

```
class Fraction:
    def __init__(self, top, bottom):
        self.num = top          # the numerator is on top
        self.den = bottom       # the denominator is on the bottom

    def __str__(self):
        return str(self.num) + "/" + str(self.den)

    def simplify(self):
        common = gcd(self.num, self.den)
        self.num = self.num // common
        self.den = self.den // common
```

2. **For ~~Python objects~~, what is the difference between "shallow equality" and "deep equality"?**

whether the objects have the same numerical value

whether the objects have the same memory location

3. **Implement a "deep equality" operator for the `Fraction` class.**

implement: `__eq__(self, other):`

Fractions to be compared

Want: return True or False

return:

`self.num * other.den == self.den * other.num`

self.num ? other.num

$\frac{a}{b} \stackrel{?}{=} \frac{c}{d}$ ⟷ equivalent

$ad \stackrel{?}{=} cb$

$\frac{self.num}{self.den} \stackrel{?}{=} \frac{other.num}{other.den}$

self.num * other.den == self.den * other.num

4. Implement a < operator for the `Fraction` class.

implement: `__lt__(self, other):` evaluate self < other, return True or False

return (self.num * other.den < self.den * other.num)

$\frac{a}{b} \stackrel{?}{<} \frac{c}{d}$ ⟷ equivalent

$ad \stackrel{?}{<} cb$

other comparison methods:

`__le__`	$\leq$
`__ne__`	$!=$
`__gt__`	$>$
`__ge__`	$\geq$

5. Implement a * operator for the `Fraction` class.

implement `__mul__(self, other):` return the product of self and other

$\frac{a}{b} \cdot \frac{c}{d} = \frac{ac}{bd}$