CSE 30 4-27-21

- pa4 posted: rational.py
  class Rational

Recall: Complex class in complex.py

built-in class complex: $a+bi$ $(i^2=-1)$

class Complex: $a+bi$ $(i^2=-1)$

Pa4:

- Rational class

file: rational.py

A rational number is a number of the form $\frac{n}{d}$ where n, and d are integers ($d \neq 0$).

$$1/3 \rightarrow \text{float } 0.3333\ldots3 \left\{\begin{array}{l}\text{decimal}\\ \text{approx.}\end{array}\right.$$

$\downarrow$ in binary

$$0.xxxxx\ldots x \left\{\begin{array}{l}\text{binare}\\ \text{approx.}\end{array}\right.$$

Rational class will store $\frac{1}{3}$ as a pair of integers: (1, 3)

i.e. Rational is an encapsulation of two integers.

suggested names:

_numer
_denom

Property decorated functions.

numer()
denom()

<u>Note</u>: $\frac{n}{d}$ should satisfy:

- $\gcd(n,d)=1$, i.e. n and d are relatively prime.

i.e $\frac{6}{4}$ represented as $\frac{3}{2}$

create fcn. _gcd(a,b) that returns GCD of a,b. helper fcn in rational.py, outside class Rational.

- $d \neq 0$:

  Rational(·, 0) should raise

  ZeroDivisionError.

- $d > 0$:

  $$-\frac{n}{d} = \frac{-n}{d} = \frac{n}{-d}$$

  Rational(n, d)

- over-load operators

  + `__add__`
  - `__sub__`
  * `__mul__`
  / `__truediv__`

  <= `__le__`

  < `__lt__`

  ⋮

  >, >=, ==, !=

look at $<$:

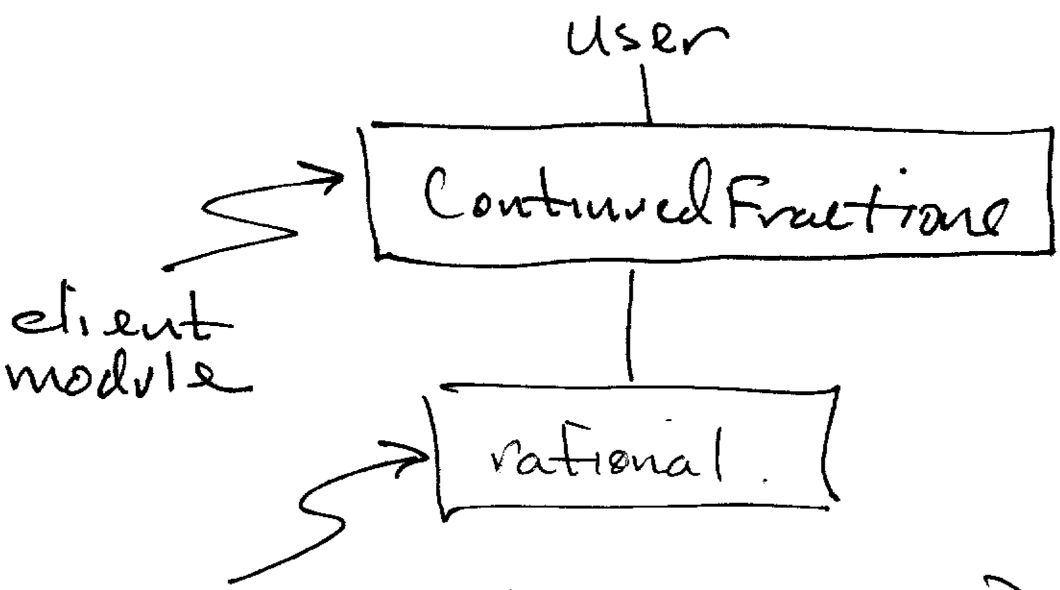
$$\frac{a}{b} < \frac{c}{d}$$

$$(ad < bc)$$ ← return this!

look at $+$:

$$\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}$$

- ContinuedFractions.py : Program.



ADT module (ADT = Abstract Data Type)

A continued fraction is an expression of the form

$$x_0 + \cfrac{1}{x_1 + \cfrac{1}{x_2 + \cfrac{1}{\ddots + \cfrac{1}{x_n}}}}$$

where $x_0, x_1, \ldots, x_n$ are integers.

notation $[x_0, x_1, \ldots, x_n]$.

looks like a list, which is how we'll represent it.

Ex.

$$\underbrace{\left(1+\cfrac{1}{\underbrace{\left(1+\cfrac{1}{\underbrace{\left(1+\cfrac{1}{\underbrace{\left(1+\cfrac{1}{\underbrace{1}_{1}}\right)}_{\frac{2}{1}}}\right)}_{\frac{3}{2}}}\right)}_{\frac{5}{3}=\frac{F_5}{F_4}}}\right)}_{\frac{8}{5}=\frac{F_6}{F_5}} = [1,1,1,1,1]$$

Fibonacci numbers : $F_0 = 0$, $F_1 = 1$,

	0,	1,	1,	2,	3,	5,	8,	13	...
ind:	0	1	2	3	4	5	6	7	...

$$\begin{array}{ccccccc} x_0 & x_1 & x_2 & & & x_n & \\ \end{array}$$

<u>note:</u> $[1, 1, 1, \ldots\ldots, 1]$ len $= n+1$

if all x's are 1's, then

$$[1, 1, \ldots, 1] = \frac{F_{n+2}}{F_{n+1}}$$

<u>Ex.</u> $[3, 7, 15, 1, 292, 1, 1, \ldots]$

$$\approx \pi.$$

Recursive defn of cont. frac.

$$[x_0, x_1, \ldots, x_n] = x_0 + \frac{1}{[x_1, x_2, \ldots, x_n]}$$

Both branches

$$[x_0, x_1, x_2, \ldots, x_n] = \begin{cases} x_0 & n=0 \\ x_0 + \dfrac{1}{[x_1, \ldots x_n]} & n \geq 1 \end{cases}$$

ContinuedFractions.py will define a function

CF(L)

that takes a list of integers L as input, returns a rational object representing the value of the cont. frac.

will use <u>decimal</u> module to create a decimal approx. to value of a cont. frac. (not float.)

- decimal module.
- built-in: map()
- string: split()

Past question.

- Rational(6) $\rightarrow \frac{6}{1}$

  Rational(6, 5) $\rightarrow \frac{6}{5}$

  Rational(12, 10) $\rightarrow \frac{6}{5}$

  Rational(6, -5) $\rightarrow \frac{-6}{5}$

  Rational(6, 0)

  raises exception