CSC148-Section:L0301 Week#2-Friday

Instructed by

AbdulAziz Al-Helali

a.alhelali@mail.utoronto.ca

Office hours: Wednesday 11-1, BA2230.

Slides adapted from Professor Danny Heap and Jacqueline Smith slides winter17

Computer Science
UNIVERSITY OF TORONTO

Outline

- Announcements
- Build class Point. . .
  in that **deeply wrong** way
- Continue Rational class
  - float and __eq__ __lt___

Announcements

- Make sure your teach.cs account works
  - If you take the lab quiz on paper
  - Your Account will **may not work** if
    - Immediately you have been added to the course
    - Or changed your password

- Accessibility Services Note-Taking Program

UNIVERSITY OF TORONTO

ACCESSIBILITY SERVICES

student life.

Accessibility Services Note-Taking Program

Volunteer Note-Takers Needed!

Why volunteer as a note-taker?

- Help your fellow students living with a disability to achieve academic success
- Many volunteers report that while helping others their own note-taking skills improve
- You will receive a Certificate of Appreciation upon the completion of the semester
- You will receive CCR (Co-Curricular Record) recognition

Does it take up a lot of my time?

Volunteering as a note-taker is easy and involve almost no extra work:

- Continue to attend classes regularly and take lecture notes
- Upload your notes to our secure website consistently
- Inform us if you drop any classes that you volunteered

How to volunteer as a note-taker?

Step 1: Register as a volunteer note-taker online at:

https://sites.studentlife.utoronto.ca/accessibility/vollogin.aspx

Step 2: Select your course and click register

Step 3: Upload your notes after every class

- Typed notes can be submitted online
- Legible hand-written notes can be scanned and uploaded at our office (or you can do it at home if you have a scanner)

Thank you for your generous help!

Email us at: as.notetaking@utoronto.ca or call 416-978-6186 if you have questions or require any assistance.

Our Address: Accessibility Services

455 Spadina Avenue, 4th Floor, Room 400

Office Hours: 9:30 AM - 4:30 PM (Monday to Friday)

Daily Office closure 12:30 PM - 1:30 PM

build class Point. . .

Do not use it

in that deeply wrong, but informative, way

```
>>> class Point:
... pass
...
>>> def initialize(point, x, y):
... point.x = x
... point.y = y
...
>>> def distance(point):
... return (point.x**2 + point.y**2) ** (1 / 2)
...
```

```
>>> Point.__init__ = initialize
>>> Point.distance = distance
>>> p2 = Point(12, 5)
>>> p2.distance()
13.0
>>>
```

Linking module function to class method

```
>>> from turtle import Turtle
>>> t=Turtle()
>>> t.wings
Traceback (most recent call last):
  File "<input>", line 1, in <module>
AttributeError: 'Turtle' object has no attribute 'wings'
>>> t.wings="object wings"
>>> Turtle.wings="Class wings"
>>> t.wings
'object wings'
>>> Turtle.wings
'Class wings'
>>> e = Turtle()
>>> e.wings
'Class wings'
```

Weird things

- what happens if, after declaring Point, you try

  print(Point.x)

  OR

  Point.y = 17

- methods can be invoked in two equivalent ways:

  p = Point(3, 4)

  p.distance_to_origin()

  5.0

  Point.distance_to_origin(p)

in each case the first parameter, conventionally self, refers to the instance named p

Computer Science
UNIVERSITY OF TORONTO

Exercise: build Rational class

Here is a description of rational numbers, the fractions we learned in grade school:

> Rational numbers are ratios of two integers p/q, where p is called the numerator and q is called the denominator. The denominator q is non-zero. Operations on rationals include addition, multiplication, and comparisons: $>$, $<$, $\geq$, $\leq$, $=$.

http://www.teach.cs.toronto.edu/~csc148h/winter/lecturedata/Danny/W1/rational_exercise.pdf

Exercise: build Rational class

Here is a description of rational numbers, the fractions we learned in grade school:

> Rational numbers are ratios of two integers p/q, where p is called the numerator and q is called the denominator. The denominator q is non-zero. Operations on rationals include addition, multiplication, and comparisons: $>$, $<$, $\geq$, $\leq$, $=$.

Define a class API:

1. choose a class name and write a brief description in the class docstring.
2. write some examples of client code that uses your class
3. decide what services your class should provide as public methods, for each method declare an API (examples,header, type contract, description)
4. decide which attributes you class should provide without calling a method, list them in the class docstring

Implement the class:

1. body of special methods __init__, __eq__, and __str__
2. body of other methods
3. testing (more on this later)

http://www.teach.cs.toronto.edu/~csc148h/winter/lecturedata/Danny/W1/rational_exercise.pdf

Exercise: build Rational class

- Rational
  - num: int
  - denum: int

__init__ called when >>> r1=Rational(2,3)

__eq__ called when >>> r1 == r2

also when we have list L=[r1,r2]

>>> r1 in L

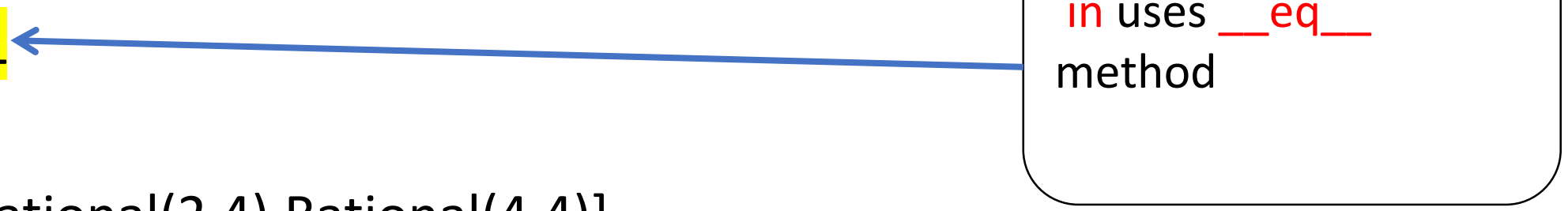
__str__ called when >>>print(r1)

__lt__ called when >>> r1 < r2

Computer Science
UNIVERSITY OF TORONTO

Exercise: build Rational class

```
>>> from rational_api_imp import *
>>> L=[Rational(1,2)]
>>> r=Rational(1,2)
>>> r in L
True
>>> L=[Rational(2,4),Rational(4,4)]
>>> r in L
True
```



Exercise: build Rational class

Check the course website for the implementation of Rational class