

# CS350: Data Structures Class Notes

*Class 2: Wednesday, September 3, 2008*

## **Announcements:**

We are changing classrooms. We will now meet in Olin 212.  
There are links to code discussed today on the class web page.

## **Abstract Data Types or ADT**

When programming, we use modules/modularity to partition our code for three reasons:

- It simplifies the task of coding—we can focus on individual task rather than the whole problem
- It allows us to eliminate redundancies—we can reuse code
- It allows us to isolate errors—this simplifies debugging and unit testing

The key ideas are information hiding and encapsulation.

There are two parts to what we are doing. We want to separate the “what” from the “how”. Both must be addressed, but it is easier if we do this separately. The “what” defines the external interface. The “how” is the implementation.

An ADT is composed of two parts, the data and the operations on the data. Contrasted to a data structure in a programming language, the data structure only include the data, not the operations.

E.g., an ADT for Booleans would include: TRUE, FALSE, AND, OR, and NOT.

The correspondence between an ADT and a class definition in Python should be obvious.

## **Rational Numbers**

*Data (a/b)*

numerator, denominator

*Operations*

Rational—constructor

numIs—accesser function that returns the numerator

denIs—accesser function that returns the denominator

+ or `__add__`—adds two rational numbers:  $a/b+c/d \rightarrow (ad+bc)/bd$

– or `__sub__`—subtract two rational numbers:  $a/b-c/d \rightarrow (ad-bc)/bd$

\* or `__mul__`—multiplication:  $a/b*c/d \rightarrow ac/bd$

/ or `__div__`—division:  $a/b / c/d \rightarrow ad/bc$

<, ==, >, etc. or `__cmp__`—comparison –1 for less-than, 0 for equal, +1 for greater-than

Other issues including printing, etc.

## Complex Numbers

*Data ( $a+bi$ )*

real coefficient, imaginary coefficient

*Operations*

Complex—constructor

realIs—accessor function that returns the real coefficient

imagIs—accessor function that returns the imaginary coefficient

+ or `__add__`—adds two complex numbers:  $a+bi + c+di \rightarrow a+c + (b+d)i$

– or `__sub__`—subtract two complex numbers:  $a+bi - c+di \rightarrow a-c + (b-d)i$

\* or `__mul__`—multiplication:  $a+bi * c+di \rightarrow ac-bd + (ad+bc)i$

conj—complex conjugate:  $a+bi \rightarrow a-bi$

## Homework

Following the example for rational numbers presented in class, create a Python class for complex numbers implementing the constructor, accessors, addition, multiplication, and printing. Print the class and a few examples. You do not need to email me your code.