Welcome to CSC148!
Introduction to Computer Science

Amir H. Chinaei, Winter 2016

ahchinaei@cs.toronto.edu
http://www.cs.toronto.edu/~ahchinaei/

Office hours: W 4:00 - 5:45 BA4222

Today

- Course Outline (bird's-eye view)
  - What this course is about
- Logistics
  - Course organization, information sheet
  - Assignments, grading scheme, *etc.*
- Introduction to
  - Object Oriented Design/Programming

What is this course about?

- Design vs Programming concepts
  - CSC108 ~ Basic constructs in Python
    - elementary data types, statements, control flow, functions, using classes, …
  - **CSC148 ~ Basic design concepts** (using Python)
    - Advanced data types, designing classes, …
- Prerequisite
  - Need to have solid background from CSC108
  - otherwise, sign up **now** for the ramp-up session by mailing: csc148w16rampup@cs.toronto.edu
    - **Sat Jan 16th, 10am-4pm in WB116**

Bird's-eye view

- Abstract(Advanced) data types (aka classes)
  - building blocks of design (of an oop)
- Hence, you need *data structures*
  - to implement those ADTs
  - Linked data structures
- Design properties (concepts and habits)
  - encapsulation (hiding details of implementation),
  - (cohesion, decoupling, exceptions, etc.)
  - thinking/writing/testing
  - Learning how to learn!
- Advancing your programming tool-box (skills)
  - Recursion, a powerful technique for some problems
- Analyzing the efficiency of programs

Logistics (1/2)

- Prerequisite knowledge
  - Knowledge of csc108 is a must
  - Have a doubt? Send the email to csc148w16rampup@cs.toronto.edu **now**
- Course components
  - Lectures: concepts
  - Labs: practice, more details, problem solving
  - Exercises, SLOG, and assignments: mastering your skills
  - Readings: preparing you for above

Logistics (2/2)

- For important information on
  - Lecture and lab time/location/material
  - Contact information of course staff
  - Office hour and location
  - Exercises/Assignments/Readings specification/solution
  - Deadlines and evaluation
  - Communication and announcements
  - Software tools, information sheet
- Follow the course web page, regularly

  http://www.cdf.toronto.edu/~csc148h/winter/

Getting ready

- Make sure to have Python 3 or a later version
- Download PyCharm IDE (community version)
- Make sure your assignments work on your CDF account too as we grade it there

Instructions and troubleshooting available in the course page

Python (review)

A datum has 3 components

- id (a reference/alias to its address in memory)
- data type
- value

```
>>> 5

>>> "Python is fun"


>>> x = 5

>>> s = "Python is fun"
```

Immutable data type

- Once stored in memory, it **cannot** change!
  - e.g. integers, strings, boolean

```
>>> x = 1
>>> y = x
>>> x = 2
```

Mutable data type

A data type that is not immutable! e.g. lists.

```
>>> x = [1, 2, 3]

>>> y = x

>>> id(x)
?
>>> x[0] = 4

>>> x
?
```

```
>>> id(x)
?
>>> y
?
>>> x = [4, 2, 3]

>>> id(x)
?
```

Extra care, when using aliasing for a mutable data type

Two types of verifying equality

- == equality of **values** in memory
- is equality of **addresses** in memory

```
>>> x = [5]
>>> z = [5]
>>> y = x
>>> x == y
?
>>> x == z
?
>>> x is z
?
```

Class

= abstract (advanced) data type = data type

revisit previous slides and change all

"data type", "abstract data type" and "advanced data type"
to

class

- A class represents objects that have
  - the same type and number of **attributes**
  - the same actions/functions (**methods**) applicable to them
  - e.g., int, str, Rectangle, Point, Rational, etc.

Rectangle class

A rectangle can be defined in many different ways. Here, assume a rectangle is defined by its top-left coordinates as well as its width and height. A rectangle is usually represented by a quadruple (x, y, w, h) where x and y represent the top-left coordinate, w represents the width, and h represents the height. For example, (10, 20, 300, 400) represents a rectangle that its top-left coordinate is located at point (10,20), its width is 300 and its height is 400. Some of the typical operations that one associates with rectangles might be translating the rectangle to the right, left, up, and down, or if two rectangles are conceptually equal, which means have same coordinate and size, or asking if a rectangle falls within another rectangle., etc.

Rectangle class

```
class Rectangle:
    """
    a rectangle defined by its top-left coordinates,
      width and height
    """
```

Constructor

```
def __init__(self, x, y, width, height):
    """Initialize this Rectangle.

    @type self: Rectangle.
    @type x: int
        The x coordinate of top-left
         corner of this rectangle
    @type y: int
        The y coordinate of top-left
        corner of this rectangle
    @type width: int
        The width of this rectangle
    @type height: int
        The height of this rectangle
```

Constructor

```
>>> r = Rectangle(100, 200, 300, 400)
>>> r.x
100
>>> r.y
200
>>> r.width
300
>>> r.height
400
"""
```

```
self.x = x
self.y = y
self.width = width
self.height = height
```

Method to translate right

```
def translate_right(self, num):
    """Moves the rectangle to right by a number of pixel

    @type self: Rectangle
    @type num: int
    @rtype: NoneType

    >>> r = Rectangle(100, 200, 300, 400)
    >>> r.translate_right(20)
    >>> r.x
    120
    """

    self.x = self.x + num
```

Explore more

- Develop a few other methods for class Rectangle.
  - e.g. translate to left, up and down
  - or, equality, is_in, etc.
- What if the input parameter (num) is negative in any of these methods, such as translate_right?
- Define a new class from scratch, e.g. class Point.

Point class

In two dimensions, a point is two numbers (coordinates) that are treated collectively as a single object. Points are often written in parentheses with a comma separating the coordinates. For example, (0, 0) represents the origin, and (x, y) represents the point x units to the right and y units up from the origin. Some of the typical operations that one associates with points might be calculating the distance of a point from the origin, or from another point, or finding a midpoint of two points, or asking if a point falls within a given rectangle or circle.

Design roadmap 1/2

- 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)
     - **refer to CSC108 function design recipe**
  4. decide which attributes your class should provide without calling a method, list them in the class docstring

Design roadmap 2/2

- Implement the class:
  1. body of special methods,
     `__init__`, `__eq__`, `__str__`
  2. body of other methods
     e.g. `distance`
  3. testing (more on this later)

Rational class

- Python has a built-in type for floating-point numbers, but not a built-in type for representing rational numbers:

  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:

  $<\qquad \leq \qquad > \qquad \geq \qquad =$

- So, create your own Rational class, following the design roadmap provided.