CSC148 WINTER IS ~~COMING~~ 2018

Here

Credit:
The lecture slides are created based on previous lecture slides by Dan Zingaro.

Larry Zhang

Office: DH-3042

Email: ylzhang@cs.toronto.edu

The teaching team

Dan Zingaro: LEC0103 (Coordinator)

Larry Zhang: LEC0101 and LEC0102

Vincent Maccio: LEC0104

and ~20 Teaching Assistants

What's in CSC148

- Object Oriented Design: principles for all programming languages
- Recursion: magical and powerful problem solving technique
    - Recursive functions
    - Recursive data structures
- Reasoning about efficiency of code
- Reasoning about sorting algorithms

Prerequisite: CSC108

Course Website

https://mcs.utm.utoronto.ca/~148/

All course materials can be found here.

Links for weekly readings.

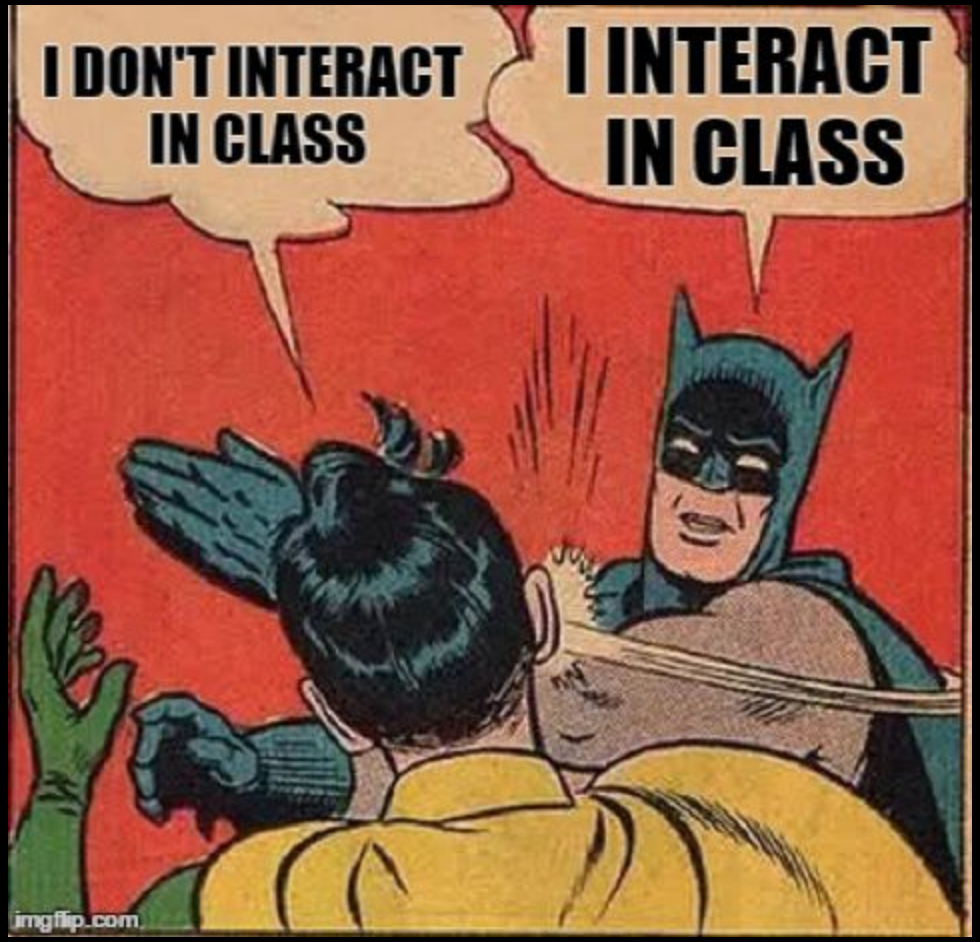
Evaluation

- Two assignments (5% each)
- Two exercises (2% each)
- Two term tests (15% each)
  - Better test counts for 20%, worse test for 10%
- Final exam (56%)
  - must get at least 40% on the exam to pass the course

Lectures

- We have three lectures per week (MWF)
- Introduce most of the new material (W and F)
- Worksheets and exercises (M)
- Get involved in classroom **interactions**
  - ➔ answering a question
  - ➔ making a guess / bet / vote
  - ➔ back-of-the envelope calculations

Emotional involvement makes the brain remember better!


I DON'T INTERACT IN CLASS
I INTERACT IN CLASS
imgflip.com

Open Labs

- This year, “labs” are replaced by open TA office hour time.
- We will post recommended lab material each week that we hope you will do for practice
- Work on this material with your peers and ask the TA if you get stuck!
- more practice ⇒ higher marks!

Exercises and Assignments

- The handouts will be on the course website
- Due at 22:00 on due date; submitted electronically using MarkUs
- Exercises are completed individually
- Assignments are completed in a group of 1-3

Assignment Late Policy

- No assignments are accepted late
- If you can’t finish the work, you can earn part marks for a good partial solution
- Illness and other disasters are another matter; contact me or Dan as soon as you can

Get Help

- Office hours
  - Attend any of the three instructor’s office hours (info on course website)
  - or make an appointment
- Open labs (led by your TA)
- Online discussion boards (link on course website)
- Anonymous feedback (link on course website)
- Form study groups!


GODIVA

Academic Integrity

In brief

- Never look at someone else's assignment work (not even a draft)
- Never show other students your assignment work (not even a draft)
- Don't post anything online (e.g. pastebin, Github)
- Discuss how to solve an assignment only with your assignment partner, the course TAs, and instructor

Academic Integrity ...

- We often handled many academic offense cases in CSC108 and CSC148
- Waste of your time, the instructors' time, Chair's time, Dean's time.
- Doesn't help you learn the course material
- Results in mark penalties and transcript annotations
- other consequences

Other awards

- For occasional in-class quizzes, or outstanding performance in tests
- Kahoot quizzes: you'll use a browser (phonen/laptop) to visit https://kahoot.it

Checklist for this week

- Read the course information sheet
- Bookmark the course website and discussion board
- Read the recommended readings for Week 1
- Log in to the online discussion board. Make sure your login works!
- If you plan on working on your own computer, install the necessary softwares. The course website has some instructions.
- Drop by office hours and introduce yourself!

This Friday

Object Oriented Programming

Object Oriented Programming

Procedural vs Object-Oriented

- Procedural programming
  - The program is mainly a description of **procedures**
  - “do this, do this, then do this”
- Object-oriented programming
  - The program is mainly a description of **objects and their relations**
  - “a student has a student number, a UTORID and can take courses, drop courses , etc; a classroom has a podium, desks and students”
  - rather than **take_course(stu, course)**, we do **stu.take_course(course)**
  - Better suited for modelling the real world.
- OO is not always better than procedural, e.g., sqrt(9) is better than 9.sqrt()

Python Objects

You have seen some already in CSC108. Examples?

DEMO

```
>>> w1 = "words"
>>> w2 = w1.lower()
>>> w1 is w2
False
>>> import turtle
>>> t = turtle.Turtle()
>>> t.pos()
(0.00,0.00)
>>> t.forward(100)
```

References to an object

Classes

- We can already create some types of objects using the built-in object types, e.g., string, list, dict…
- But sometimes, we want more **types** of objects, e.g., Student, Building, Game, Circle …
- You can add a **new type** in Python by defining a **new class**.
- then you can **instantiate an object** of that class
- Class names are **nouns**: Student, Course, Rectangle, Animal, Cat, Ship, (and str, list, dict!)

How to define a class?

A class's definition consists of **methods** and **attributes**

- **Methods**: actions that an object of this class can perform.
  - e.g., for ship
  - e.g., move(), turn(), shoot(), raise_shields(), lower_shields(), teleport()
  - like functions, but inside the class
- **Attributes**: a feature or characteristic of an object
  - implemented using an instance variable
  - it's something that an object **HAS**, not something that it **DOES** (not an action)
  - e.g. for ship: weight, length, width, direction, speed

Creating an object

After defining the class, we don’t have an object yet.

We need to call a **constructor** method to **instantiate** an object of the class

In Python, the constructor’s name is **`__init__()`**

```
class Ship:

    def __init__(self):

        self.weight = 100
        self.speed = 3
        self.heading = "north"
```

Fun fact

In Python, **EVERYTHING** is an object, even functions.

http://www.diveintopython.net/getting_to_know_python/everything_is_an_object.html

Object-Oriented Analysis

What is object-oriented analysis

- Given a description of something in real world
- identify
  - the classes
  - the methods and attributes of each class
  - the relations between the classes
- Important skill for solving real-world problems!

Design a new class

Somewhere in the real world there is a description of points in two-dimensional space:

> *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.*

Find the most important noun (good candidate for a class. . .), its most important attributes, and operations that this noun should support.

```
>>> from math import sqrt
>>> class Point:
...     pass
...
>>> def initialize(point, x, y):
...     point.x = x
...     point.y = y
...
>>> def distance_from_origin(point):
...     return sqrt(point.x**2 + point.y**2)
...
>>> Point.__init__ = initialize
>>> Point.distance_from_origin = distance_from_origin
>>> p2 = Point(12, 5)
>>> p2.distance_from_origin()
13.0
```

Bad Design!

```
class Point:

  def __init__(self, x=0, y=0):
    '''(Point, float, float)
    Construct Point with given x and y coordinates.
    '''

    self.x = x
    self.y = y

  def distance_from_origin(self):
    '''(Point) -> float

    Return distance from Point to origin.
    '''
    return sqrt(self.x ** 2 + self.y ** 2)

  def __str__(self):
    '''(Point) -> str

    Return string representation of Point.
    '''
    return '({}, {})'.format(self.x, self.y)
```

Good design

Design Another New Class

A rational number consists of a numerator and denominator; the denominator cannot be 0. Rational numbers are written like 7/8. Typical operations include determining whether the rational is positive, adding two rationals, multiplying two rationals, comparing two rationals, and converting a rational to a string.

Find the most important noun (good candidate for a class...), its most important attributes, and operations that this noun should support.

Inheritance

Inheritance

- Many times, a new class will have much in common with an existing class
  - Example: we already have the class **Ship**, now we need to define a new class for **Indestructible Ship**
- Approach #1: Copy-and-paste code from the old class, and then making a few changes.
  - What if you then discover a bug in the original class?
  - What if you find that there are many classes you'd like to create from the original one?
  - **BAD approach!**
- Approach #2: The new class to **specialize** an existing class by specifying **only** what is different between them.
  - This is called Inheritance
  - The class that inherits is called a **subclass**, and the class that is inherited from is its **superclass**

Inherit from a class

`class B(A):`

means B is a subclass of A, i.e., B inherits from A

When calling a method on an `IndestructibleShip`:

- If the method exists in `IndestructibleShip`, it is called
- Otherwise, the one in the **superclass** is called

```
class Ship:

  def __init__(self, hitpoints):
    self.hitpoints = hitpoints
    self.location = 0

  def move_left(self):
    self.location -= 1

  def move_right(self):
    self.location += 1

  def get_hit(self, amount):
    self.hitpoints -= amount

class IndestructibleShip(Ship):

  def get_hit(self, amount):
    pass
```

Relationships between classes: **IS-A** and **HAS-A**

IS-A

- Inheritance models ‘**is-a**” relationships
- The subclass “**is-a**” subset of the superclass
- The indestructible ships **is-a** subset of ships

HAS-A

- a ship **“has-a”** location, a ship **has-a** engine.
- “**has-a**” relationships is modelled by **instance variables**
- Do **NOT** use **inheritance** for **has-a** relationships!

Exercise: Identify **IS-A** / **HAS-A**

- Point and LineSegment
  - **HAS-A**: a LineSegment has two points
- Integer and Rational
  - **IS-A**: Integer is a subset of Rational
- Rational and Irrational
  - **neither**
- Rectangle and Shape
  - **IS-A**: Rectangle is a set subset of Shape
- Rectangle and Square
  - this one is interesting…, try writing code for it and see if inheritance makes sense for it
  - then discuss (office hours, discussion boards, study groups)

Next week

- Exercises
- Stacks and Queues