CSC148 winter 2018

Introduction to computer science

week 1

Danny Heap

heap@cs.toronto.edu

BA4270 (behind elevators)

http://www.teach.cs.toronto.edu/~csc148h/winter/

416-978-5899

face-to-face best

next best

occasionally works...

January 8, 2018

Outline

Introduction

object-oriented design

What's CSC148 **not** about?

- well first, CSC108 was about `if` statements, loops, function definitions and calls, lists, dictionaries, searching, sorting, classes, documentation style. So you've got all that down...

slides on web-page under "Lectures"

otherwise... ramp-up The sessions will be in BA1130 (BA room 1130) Saturday (10 4) and Sunday (11 5) January 6th and 7th. There is space for about 150 per day, and you need to register

But what's CSC148 about?

- how to understand and write a solution for a real-world problem English problem specification → Python solution
- abstract data types (ADTs) to represent and manipulate information → hide how
  Show what with public interface (docstring)
- recursion: clever functions that call themselves
  def f(n):
  .... f(...) !
- exceptions: how to deal with unexpected situations
  you seen errors, now we can use Exceptions
- design: how to structure a program
  80%+ of time is maintaining code...
- efficiency: how much resource (time/space) does a program use? How much time/space for a problem solution?

How's this course run?

– 4 instructors
~1000 students

draft for ~2 weeks
– then can only change by majority vote

All answers in course information sheet. Spoiler alert: meaning of life is 42...

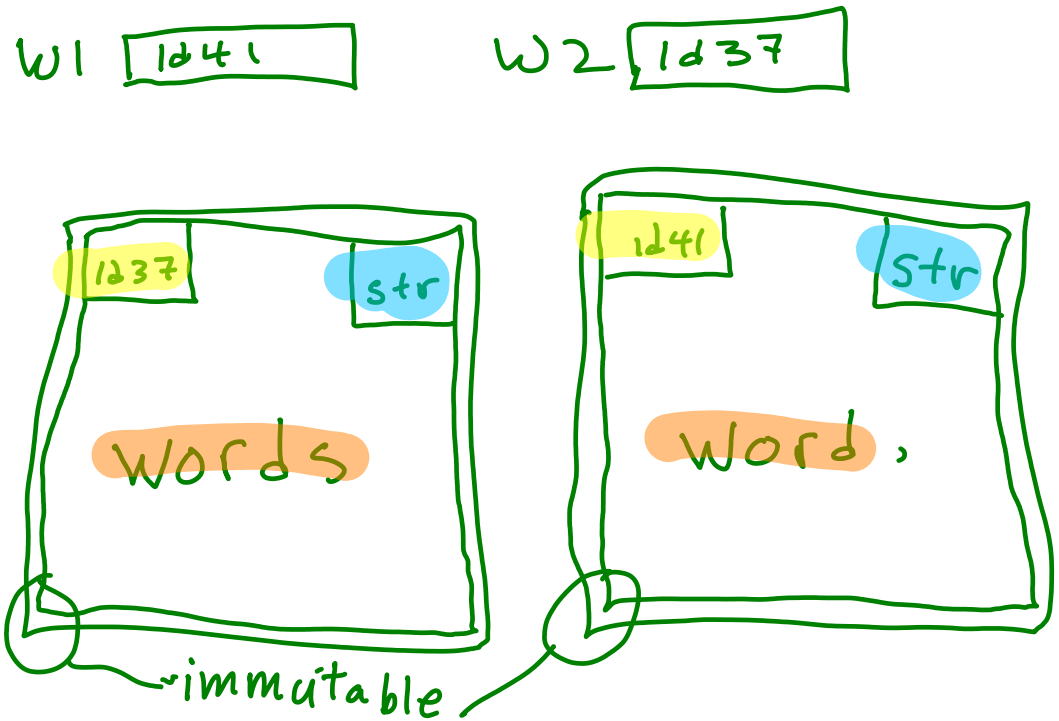
fix test conflicts now!

python infested by objects

try drawing them: **id** in upper left, **type** in upper right, **value** may include references to other objects. Compare notes with Python visualizer

Here are some built-in objects to fool around with:

```
>>> w1 = "words"
>>> w2 = "swords"[1:]
>>> w1 is w2
>>> w1 == w2
>>> w1 * w2
>>> import turtle
>>> t = turtle.Turtle()
>>> t.pos()
(0.00,0.00)
>>> t.forward(100)
```



review function design recipe

Examples
Header
Description
Body
Test

Dream up a function. Now use the function design recipe to build it, step-by-step... Now with PyCharm

nags about code defects

vandalizing existing classes

this is deeply wrong, except for teaching purposes...

```
>>> from turtle import Turtle
>>> t1 = Turtle()
>>> t1.pos()
(0.00,0.00)
>>> t1.forward(100)
>>> t1.pos()
(100.00,0.00)
>>> t1.neck
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Turtle' object has no attribute 'neck'
>>> Turtle.neck = "very reptilian"
>>> t1.neck
'very reptilian'
```

a Turtle knows where it is

a Turtle can move

no neck!

we added a neck at t1 was created!

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 sort of noun should support.

build class Point...

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
>>>
```

no attributes or methods to start out!

a module-level function attaches attributes

module-level function returns distance

make methods point to those functions

point instance here ... not here

build class Point. . . properly!

Define a class API:

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

[1] use the CSC108 function design recipe

continue building class Point... properly!

Implement the class:

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

See point.py

weird things

try these out

- 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**