CSC148 winter 2014

Introduction to computer science
week 1

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cdf.toronto.edu/~heap/148/W14/
416-978-5899

January 8, 2014

Outline

Introduction

object-oriented design

What's CSC148 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. . .
- . . . otherwise, sign up for the CSC148 ramp-up session September 14th or 21st, 10–4

  `http://www.cs.toronto.edu/~buske/rampup/`

But what's CSC148 about?

- how to understand and write a solution for a real-world problem
- abstract data types (ADTs) to represent and manipulate information
- recursion: clever functions that call themselves
- exceptions: how to deal with unexpected situations
- design: how to structure a program

How's this course run?

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

python infested by objects

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

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

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’
```

Design a new class

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

"real world."

> *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.*

distance → from orig

initialize → x, y

mid point

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

class Point

attributes x, y → horiz coords & vert

build class Point…

in that deeply wrong, but informative, way

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

build class Point. . .

. . . properly!. . .

[x+7 for x in [1,2,3]]

[8, 9, 10]

```
from math import sqrt

class Point(object):
    """n-dimensional point
    """

    # notice the function annotations below
    def __init__(self: "Point", coord: [float, ]) -> None:
        """Initialize this point
        >>> p = Point([3, 4])
        """
        # list comprehensions --- [<expression> for x in iterable]
        # may be something new to you
        self.coord = [float(x) for x in coord]

# and so on
```

list comprehension

Now!

equality

What makes two points equal? Write an `__eq__` method.

problems with attributes and access

underscore says an attribute is "private"

e.g. _coords

What happens if we decide, **after** we've distributed an early version of **Point**, that we want to control access to **coord**? A bit of Python philosophy, and a trick called **property**