Smith College
Computer Science

Week 9

CSC111 - Fall 2018

Dominique Thiébaut
dthiebaut@smith.edu

- Dealing with Exceptions (**Chapter 7.4**)
- Defining Classes (**Chapter 10**)

```
# getInput: returns an integer larger
# than 0.  Expected to be robust…

def getInput():
    while  True:
        x = eval( input( "Enter a positive int:  " ) )
        if  x  >= 0:
            return x
        print( "Invalid number: Please try again: " )


def main():
    num = getInput()
    print( "you entered", num )

main()
```

getInputReadyForException.py - /Users/thiebaut/Desktop/Dropbox/111/getInputReadyForException.py (3.5.4)

```
# getInput: returns an integer larger
# than 0.
def getInput():
    while True:
        x = eval( input( "Enter a positive int: " ) )
        if x >= 0:
            return x
        print( "You entered a negative integer." )

def main():
    num = getInput()
    print( "num =", num )

main()
```

Python 3.5.4 Shell

```
eadyForException.py =
Enter a positive int: as
Traceback (most recent call last):
  File "/Users/thiebaut/Desktop/Dropbox/111/getInputRead
yForException.py", line 14, in <module>
    main()
  File "/Users/thiebaut/Desktop/Dropbox/111/getInputRead
yForException.py", line 11, in main
    num = getInput()
  File "/Users/thiebaut/Desktop/Dropbox/111/getInputRead
yForException.py", line 5, in getInput
    x = eval( input( "Enter a positive int: " ) )
  File "<string>", line 1
    as
     ^
SyntaxError: unexpected EOF while parsing
>>>
```

Ln: 106 Col: 0

getInputReadyForException.py - /Users/thiebaut/Desktop/Dropbox/111/getInputReadyForException.py (3.5.4)

```
# getInput: returns an integer larger
# than 0.
def getInput():
    while True:
        x = eval( input( "Enter a positive int: " ) )
        if x >= 0:
            return x
        print( "You entered a negative integer." )

def main():
    num = getInput()
    print( "num =", num )

main()
```

Python 3.5.4 Shell

```
eadyForException.py =
Enter a positive int: as
Traceback (most recent call last):
  File "/Users/thiebaut/Desktop/Dropbox/111/getInputRead
yForException.py", line 14, in <module>
    main()
  File "/Users/thiebaut/Desktop/Dropbox/111/getInputRead
yForException.py", line 11, in main
    num = getInput()
  File "/Users/thiebaut/Desktop/Dropbox/111/getInputRead
yForException.py", line 5, in getInput
    x = eval( input( "Enter a positive int: " ) )
  File "<string>", line 1
    as
     ^
SyntaxError: unexpected EOF while parsing
>>>
```

Ln: 106 Col: 0

Exception

RAM

Python Program

Processor

Keyboard

Disk

Network

RAM

Operating System

Python Program

Processor

Keyboard

Disk

Network

RAM

Operating System

Python Program

Processor

Keyboard

Disk

Events

Network

RAM

Operating System

Python Program

Processor

Keyboard

Disk

Network

Exception

exception1.py - /Users/thiebaut/Desktop/111b/exception1.py

```
# getInput: returns an integer larger
# than 0.  Expected to be robust…

def getInput():

   while True:
       x = int( input( "En
       if  x >= 0:
           return x
       print( "invalid inp


def main():
    num = getInput()
    print( "you entered",

main()
```

Python Shell

```
    x = int( input( "Enter a positive int:  " ) )
ValueError: invalid literal for int() with base 10: '-300.3'
>>> ================================ RESTART ================================
=========
>>>
Enter a positive int:  -3
invalid input.  Please reenter
Enter a positive int:  100
you entered 100
>>> ================================ RESTART ================================
=========
>>>
Enter a positive int:  -
Traceback (most recent call last):
  File "/Users/thiebaut/Desktop/111b/exception1.py", line 17, in <mo
dule>
    main()
  File "/Users/thiebaut/Desktop/111b/exception1.py", line 14, in mai
n
    num = getInput()
  File "/Users/thiebaut/Desktop/111b/exception1.py", line 7, in getI
nput
    x = int( input( "Enter a positive int:  " ) )
ValueError: invalid literal for int() with base 10: '-'
>>>
```

Exception

Some exceptions are generated by the Python interpreter

RAM

Operating System

Python Program

Processor

Keyboard

Disk

Network

Exception

RAM

Operating System

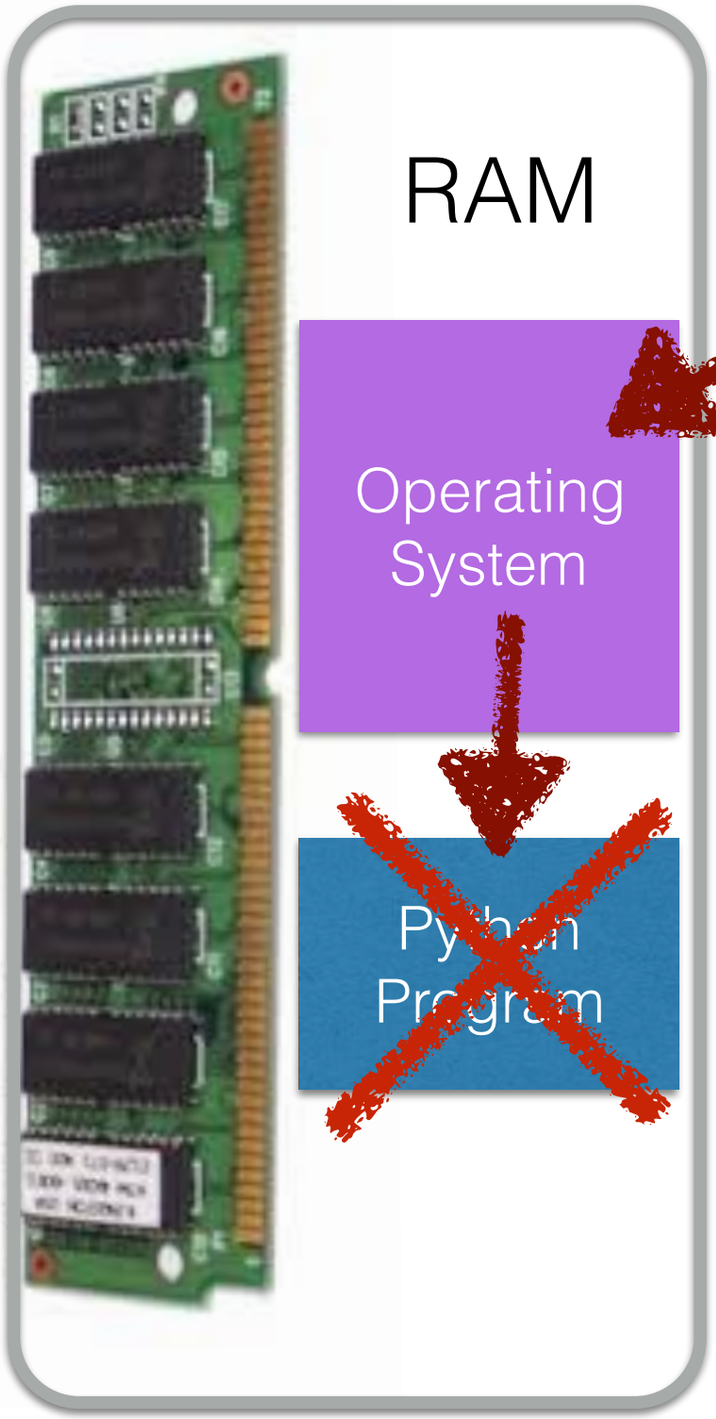
Python Program

Processor

Keyboard

Disk

Network



If Exception generated by Python Interpreter, OS "kills" the Python Program

RAM

Operating System

Python Program

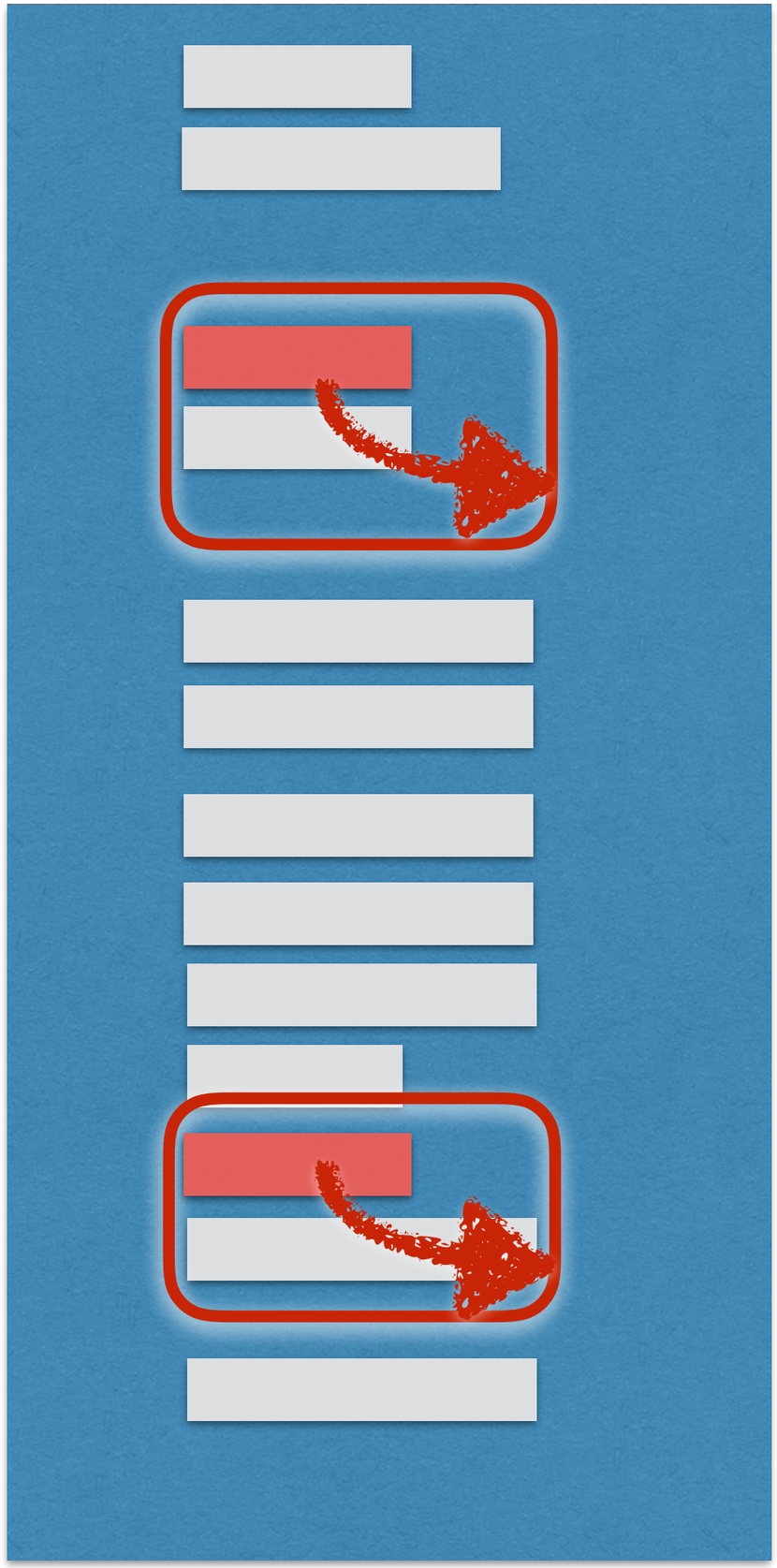
Processor

Keyboard

Disk

Network

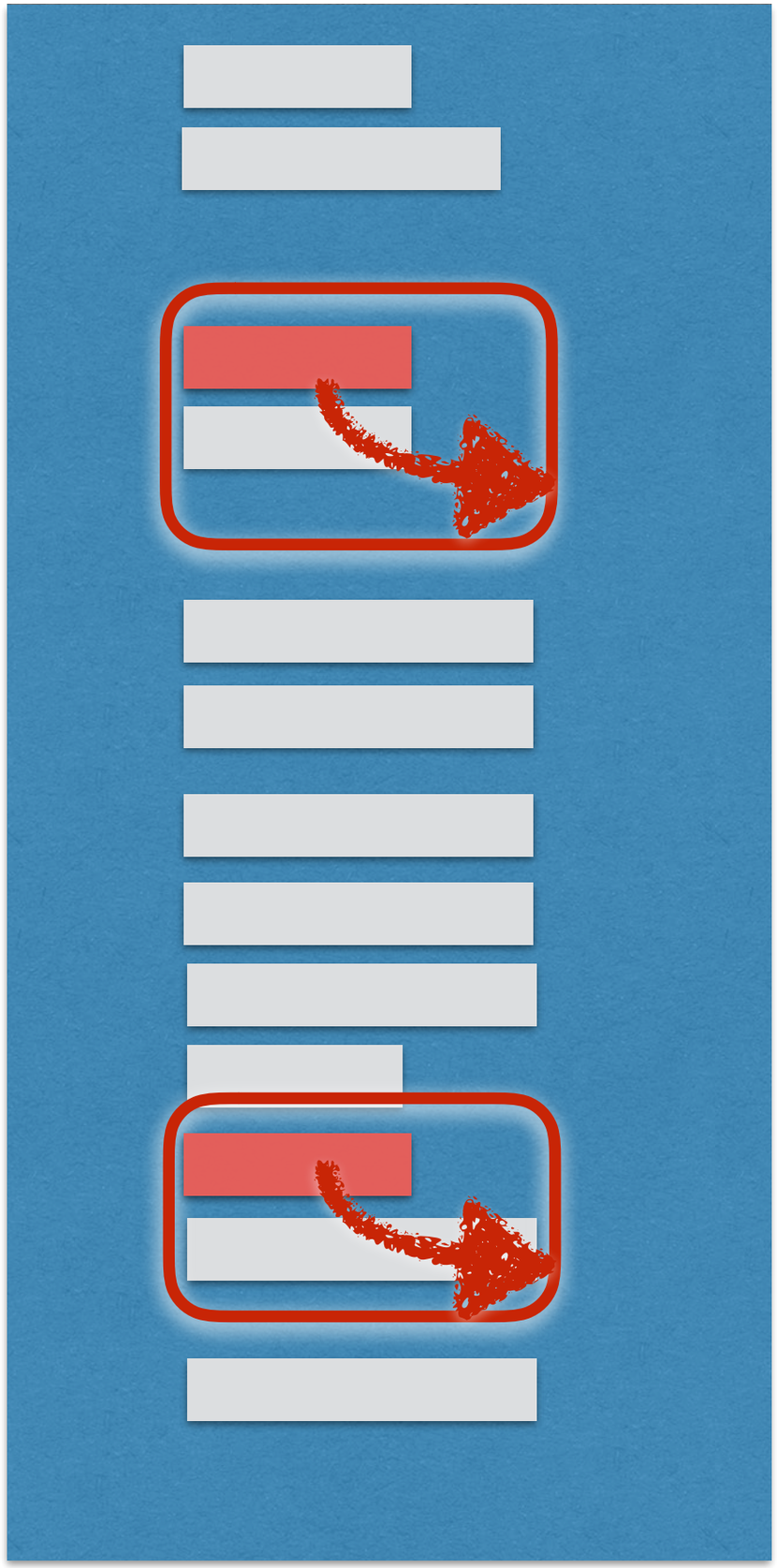
We want
to prevent exceptions
from going all the way
up to the operating system…



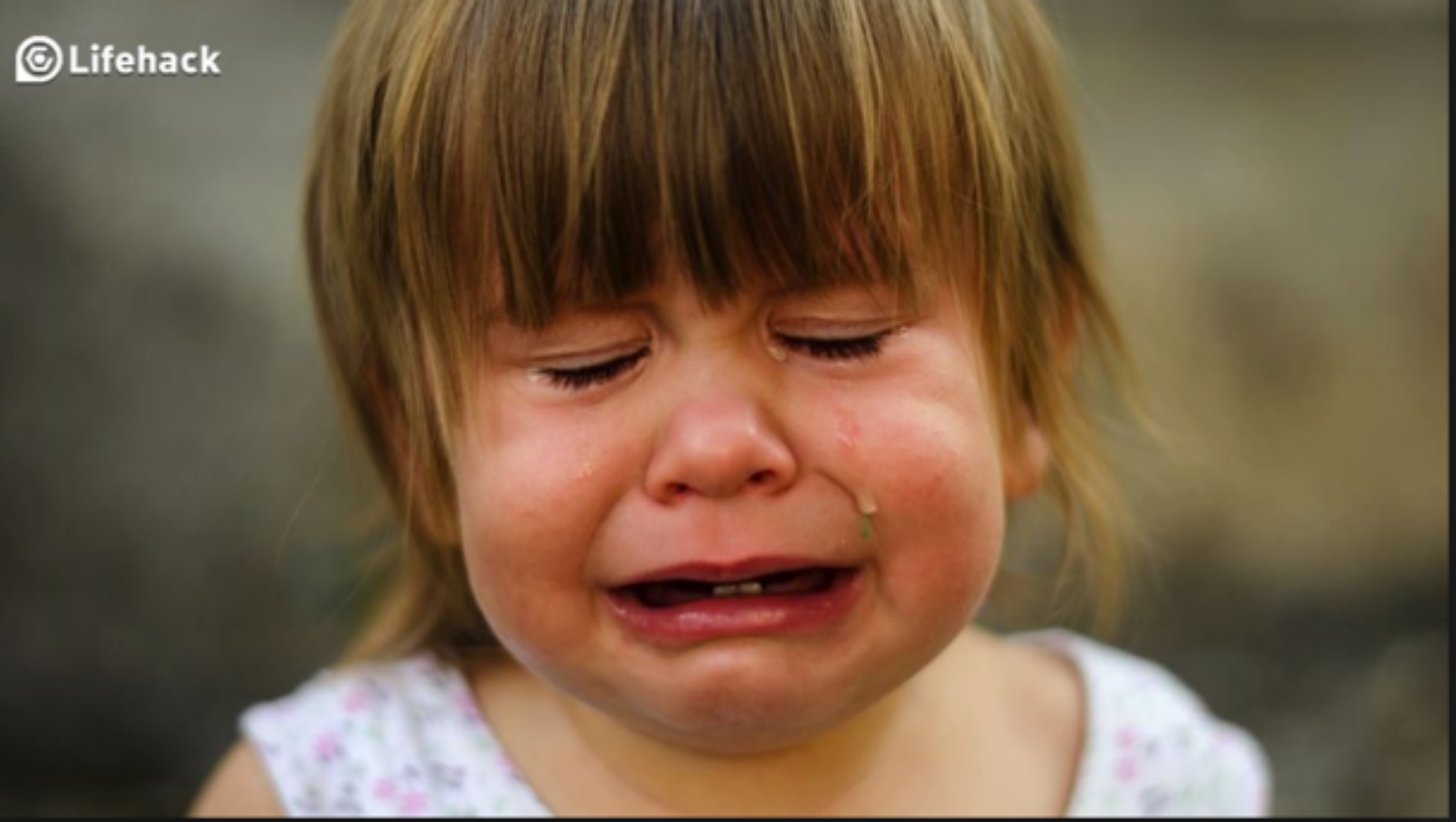
Python Program

- We want to "***isolate***" dangerous code areas

- We want a ***tight*** isolation blocks around the potentially dangerous code sections



Python Program



Coding exceptions is a pain in the neck but exceptions are an integral part of programming

Try/Except Statement

try:

python code that **might** generate
an exception

Except *exceptionXYZ* **:**

python code to run **in case** there's
an exception

exception1.py - /Users/thiebaut/Desktop/Dropbox/111/exception1.py (3.5.4)

```
# getInput: returns an integer larger
# than 0.
def getInput():
    while True:
        try:
            x = eval( input( "Enter a positive int: " ) )
        except SyntaxError:
            print( "Invalid input.  Try again!" )
            continue

        if x >= 0:
            return x

        print( "You entered a negative integer." )

def main():
    num = getInput()
    print( "num =", num )

main()
```

Ln: 7 Col: 26

Python 3.5.4 Shell

```
Python 3.5.4 (v3.5.4:3f56838976, Aug  7 2017, 12:
56:33)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on da
rwin
Type "copyright", "credits" or "license()" for mo
re information.
>>>
========== RESTART: /Users/thiebaut/Desktop/Dropbo
x/111/exception1.py ==========
Enter a positive int: -3
You entered a negative integer.
Enter a positive int: as
Invalid input.  Try again!
Enter a positive int:
```

Ln: 10 Col: 22

exception1.py - /Users/thiebaut/Desktop/Dropbox/111/exception1.py (3.5.4)

```
# getInput: returns an integer larger
# than 0.
def getInput():
    while True:
        try:
            x = eval( input( "Enter a positive int: " ) )
        except SyntaxError:
            print( "Invalid input.  Try again!" )
            continue

        if x >= 0:
            return x

        print( "You entered a negative integer." )

def main():
    num = getInput()
    print( "num =", num )

main()
```

Ln: 7 Col: 26

Python 3.5.4 Shell

```
Enter a positive int: hello
Traceback (most recent call last):
  File "/Users/thiebaut/Desktop/Dropbox/111/exception1.py", line 20, in <module>
    main()
  File "/Users/thiebaut/Desktop/Dropbox/111/exception1.py", line 17, in main
    num = getInput()
  File "/Users/thiebaut/Desktop/Dropbox/111/exception1.py", line 6, in getInput
    x = eval( input( "Enter a positive int: " ) )
  File "<string>", line 1, in <module>
NameError: name 'hello' is not defined
```

Ln: 20 Col: 4

Solution

exception2.py - /Users/thiebaut/Desktop/Dropbox/111/exception2.py (3.5.4)

```
# getInput(): returns an integer larger
# than 0.
def getInput():
    while True:
        try:
            x = eval( input( "Enter a positive int: " ) )
        except SyntaxError:
            print( "Invalid input.  Try again!" )
            continue
        except NameError:
            print( "Invalid input.  Not an integer" )
            continue

        if x >= 0:
            return x

        print( "You entered a

def main():
    num = getInput()
    print( "num =", num )

main()
```

Python 3.5.4 Shell

```
========= RESTART: /Users/thiebaut/Desktop/Dropbox/111/exception2.py
=========
Enter a positive int: as
Invalid input.  Try again!
Enter a positive int: hello
Invalid input.  Not an integer
Enter a positive int: 3
num = 3
>>>
========= RESTART: /Users/thiebaut/Desktop/Dropbox/111/exception2.py
=========
Enter a positive int: -3
You entered a negative integer.
Enter a positive int: 4
num = 4
```

Ln: 18 Col: 4

Approach to Handling Exceptions

1. Run code **without try/except** statements
2. Test thoroughly
3. Fix whatever can be fixed with "regular" python code
4. Record all exceptions that cannot be fixed otherwise, and add **try/except** to catch them.
5. Make the **try** section as *small* as possible.

Multiple Exceptions (taken from Zelle)

Solving Equation of Degree 2

Solving 2nd Degree Equations $ax^2 + bx + c = 0$

Use the quadratic formula (QF)

The roots for the equation $ax^2 + bx + c = 0$ are

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$b^2 - 4ac$ is called the *discriminant* because its value indicates what type of roots there are. Specifically,
if $b^2 - 4ac$ is a perfect square, we have fractional roots,
if $b^2 - 4ac < 0$ there is no real roots.

Solving Equation of Degree 2

Untitled

```
def ZelleExample():
    import math
    print( "solution for quadratic equation" )
    try:
        a, b, c = eval( input( "enter 3 coefficients ( a, b, c ) " ) )
        disc = math.sqrt( b*b - 4*a*c )
        root1 = (-b + disc) / (2*a)
        root2 = (-b - disc) / (2*a)
        print( "solutions: ", root1, root2 )
    except NameError:
        print( "You didn't enter 3 numbers" )
    except TypeError:
        print( "your input were not all numbers" )
    except SyntaxError:
        print( "Forgot a comma between the numbers?" )
    except ValueError:
        print( "No real roots, negative discriminant" )
    except:
        print( "Something went wrong..." )
```

Ln: 2 Col: 0

Hardening the Function

zelle.py - /Users/thiebaut/Desktop/Dropbox/111/Week9/zelle.py

```
def ZelleExample():
    import math
    print( "solution for quadratic equation" )
    try:
        a, b, c = eval( input( "enter 3 coefficients ( a, b, c ) " ) )
        disc = math.sqrt( b*b - 4*a*c )
        root1 = (-b + disc) / (2*a)
        root2 = (-b - disc) / (2*a)
        print( "solutions: ", root1, root2 )
        return True
    except NameError:
        print( "You didn't enter 3 numbers" )
    except TypeError:
        print( "your input were not all numbers" )
    except SyntaxError:
        print( "Forgot a comma between the numbers?" )
    except ValueError:
        print( "No real roots, negative discriminant" )
    except:
        print( "Something went wrong..." )
    return False
```

Ln: 22 Col: 16

Dealing with Exceptions (Chapter 7.4)

CSV Files

Defining Classes (Chapter 10)

CSV Format

- **C**omma-**S**eparated **V**alues
- Very popular way of representing information where all records have the same format
- Easy to implement
- Examples: **https://people.sc.fsu.edu/~jburkardt/data/csv/csv.html**

Example

```
"Name",      "Sex", "Age", "Height (in)", "Weight (lbs)"
"Alex",        "M",   41,          74,         170
"Bert",        "M",   42,          68,         166
"Carl",        "M",   32,          70,         155
"Dave",        "M",   39,          72,         167
"Elly",        "F",   30,          66,         124
"Fran",        "F",   33,          66,         115
"Gwen",        "F",   26,          64,         121
"Hank",        "M",   30,          71,         158
"Ivan",        "M",   53,          72,         175
"Jake",        "M",   32,          69,         143
"Kate",        "F",   47,          69,         139
"Luke",        "M",   34,          72,         163
"Myra",        "F",   23,          62,          98
"Neil",        "M",   36,          75,         160
"Omar",        "M",   38,          70,         145
"Page",        "F",   31,          67,         135
"Quin",        "M",   29,          71,         176
"Ruth",        "F",   28,          65,         131
```

Reading CSV Files

readCSVFile.py - /Users/thiebaut/Desktop/Dropbox/111/readCSVFile.py (3.5.4)

```
# readCSVFile.py
# D. Thiebaut
# Example of a program that reads a CSV file
# and displays some of its contents.

def readCSV( fileName ):
    file = open( fileName, 'r' )
    Records = [ ]
    lines = file.readlines()
    for line in lines:
        words = line.strip().split( ',' )
        Records.append( words )
    file.close()
    return Records

def main():
    fileName = input( "File name? " )
    recs = readCSV( fileName )
    for i in range( len( recs ) ):
        record = recs[i]
        # record is a tuple
        # join each word in the tuple by a tab, and print
        # resulting string
        print( "Record ", i, "=", "\t".join( record ) )

main()
```

Ln: 11 Col: 0

CSV and MS Excel

Import

What type of file do you want to import?

- (•) CSV file
  Select this file type when you want to import text files that contain comma-separated values. Most financial institutions offer this format for saving account activity.
- ( ) FileMaker Pro database
  Select this file type when you want to import data from an .fp5 or .fp7 dat
  created with FileMaker Pro. You can import all records in the database or
  specify.
- ( ) HTML file
  Select this file type when you want to import information from an HTML fil
- ( ) Text file
  Select this file type when you want to import text files. This option works
  contain values separated by tabs or spaces.

More about how to import data...

Can

Text Import Wizard - Step 1 of 3

The Text Wizard has determined that your data is Fixed Width.

If this is correct, choose Next, or choose the Data Type that best describes your data.

Original data type

Choose the file type that best describes your data:

- (•) Delimited - Characters such as commas or tabs separate each field.
- ( ) Fixed width - Fields are aligned in columns with spaces between each field.

Start import at row: 1 File origin: Macintosh

Data preview

Preview of file Macintosh HD:Users:thiebaut:Desktop:biostats.csv.

1	"Name",	"Sex",	"Age",	"Height (in)",	"Weight (lbs)"
2	"Alex",	"M",	41,	74,	170
3	"Bert",	"M",	42,	68,	166
4	"Carl",	"M",	32,	70,	155
5	"Dave",	"M",	39,	72,	167
6	"Elly",	"F",	30,	66,	124

Cancel < Back Next > Finish

CSV and MS Excel

Text Import Wizard - Step 2 of 3

This screen lets you set the delimiters your data contains. You can see how your text is affected in the preview below.

Delimiters

Tab Semicolon Comma

Space Other:

Treat consecutive delimiters as one

Text qualifier: "

Data preview

Name	"Sex"	"Age"	"Height (in)"	"Weight (lbs)"
Alex	"M"	41	74	170
Bert	"M"	42	68	166
Carl	"M"	32	70	155
Dave	"M"	39	72	167
Elly	"F"	30	66	124

Cancel < Back Next > Finis

Workbook1

New Open Save Print Import Copy Paste Format Undo Redo AutoSum

Verdana 14 B I U

Sheets Charts SmartArt Graphics WordArt

	A	B	C	D	E
1	Name	"Sex"	"Age"	"Height (in)"	"Weight (lbs)"
2	Alex	"M"	41	74	170
3	Bert	"M"	42	68	166
4	Carl	"M"	32	70	155
5	Dave	"M"	39	72	167
6	Elly	"F"	30	66	124
7	Fran	"F"	33	66	115
8	Gwen	"F"	26	64	121
9	Hank	"M"	30	71	158
10	Ivan	"M"	53	72	175
11	Jake	"M"	32	69	143
12	Kate	"F"	47	69	139
13	Luke	"M"	34	72	163
14	Myra	"F"	23	62	98
15	Neil	"M"	36	75	160
16	Omar	"M"	38	70	145
17	Page	"F"	31	67	135
18	Quin	"M"	29	71	176
19	Ruth	"F"	28	65	131
20					

Sheet1

Normal View Ready

Dealing with Exceptions (Chapter 7.4)

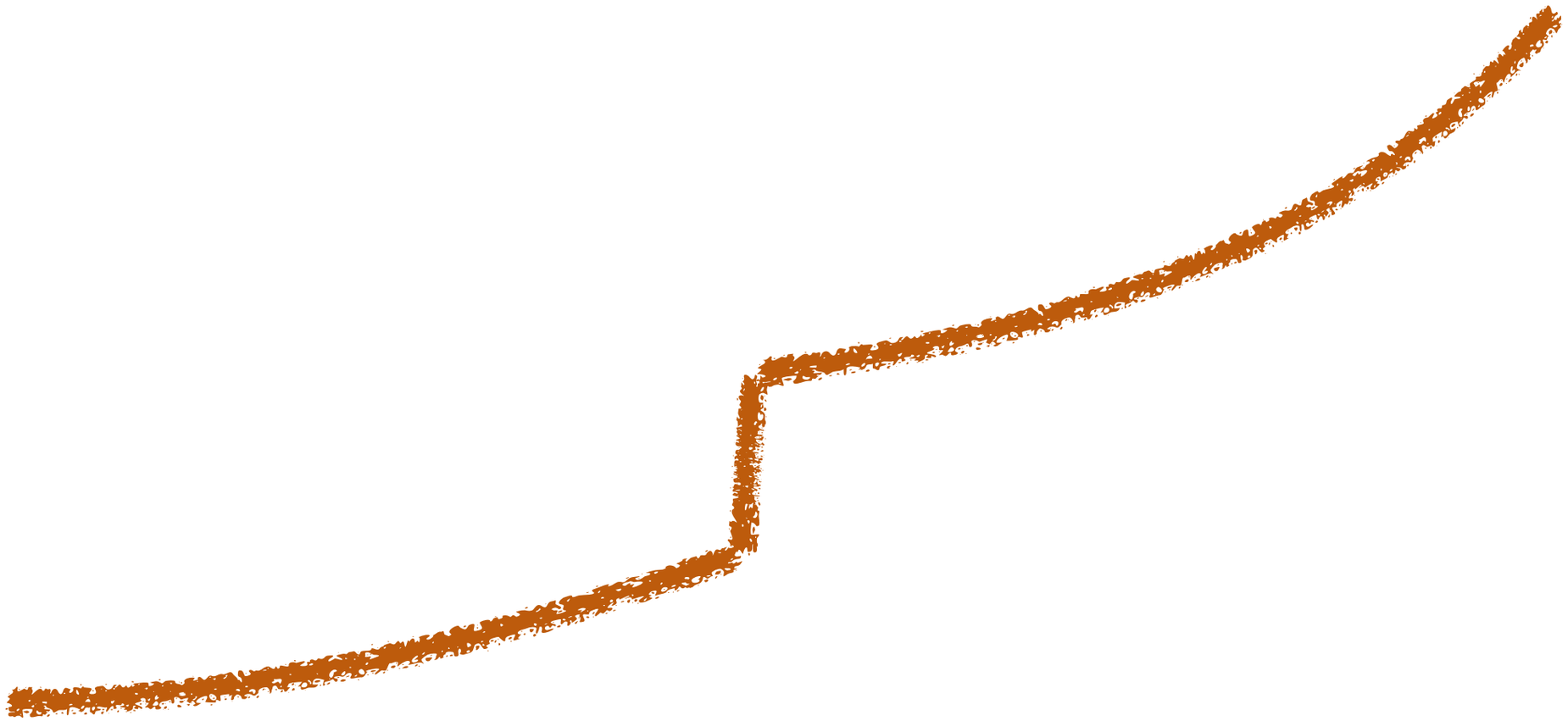
CSV Files

Defining Classes (Chapter 10)

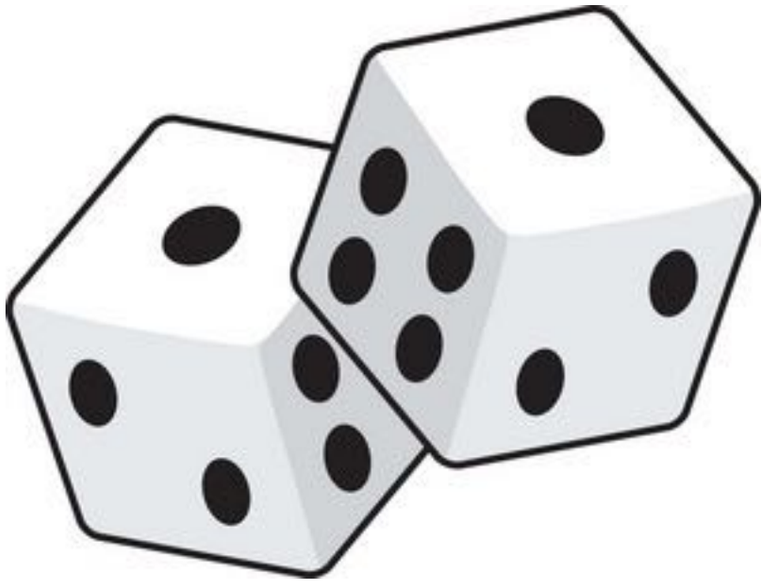
Dealing with Exceptions (Chapter 7.4)

CSV Files

Defining Classes (Chapter 10)



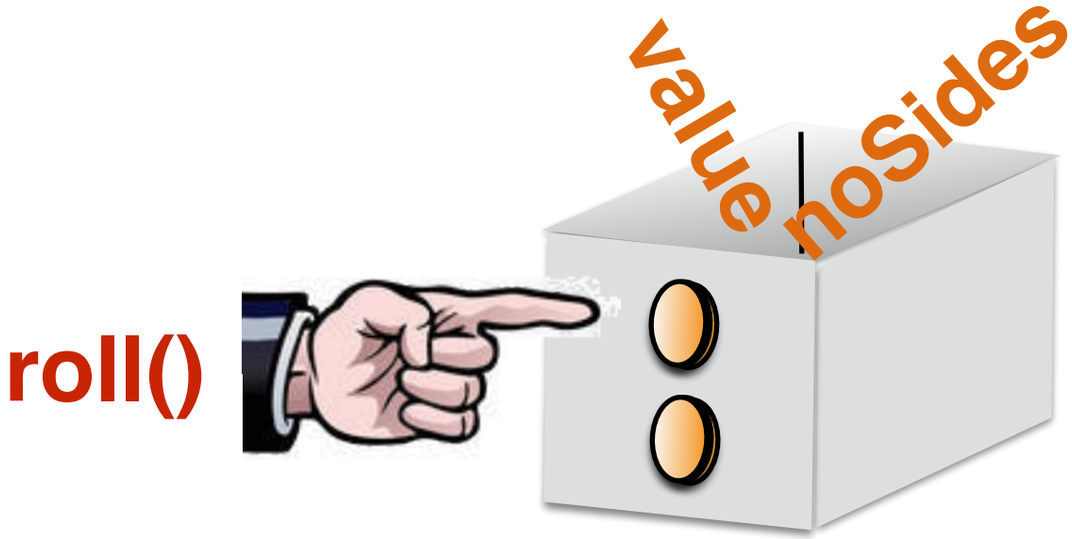
Coding Dice



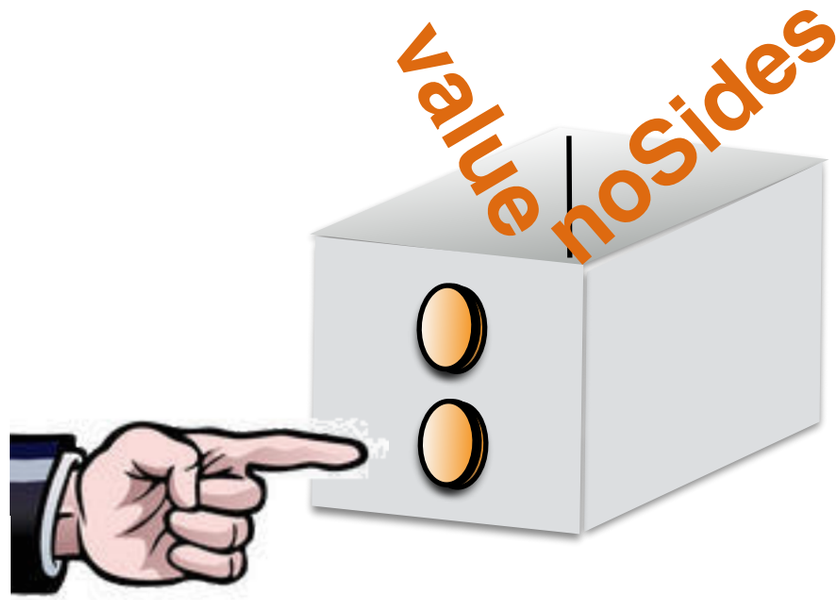
Using the Objects

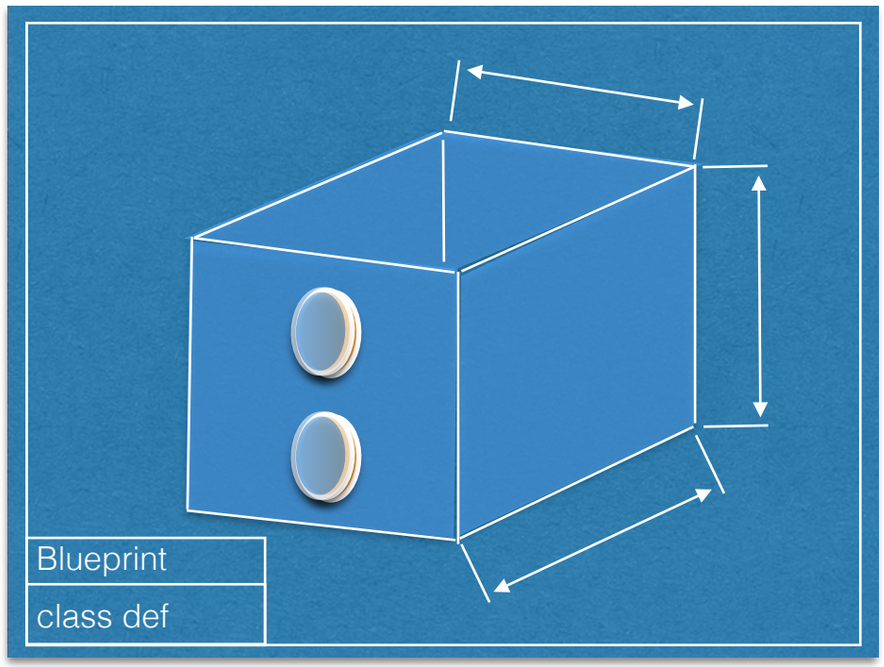
```
# Create 2 dice, one with 6 sides, one with 8
d1 = Die( 6 )
d2 = Die( 8 )

# Roll both dice
d1.roll( )
d2.roll( )

# display their value
print( "Die 1: ",  d1.getValue() )
print( "Die 2: ",  d2.getValue() )
```

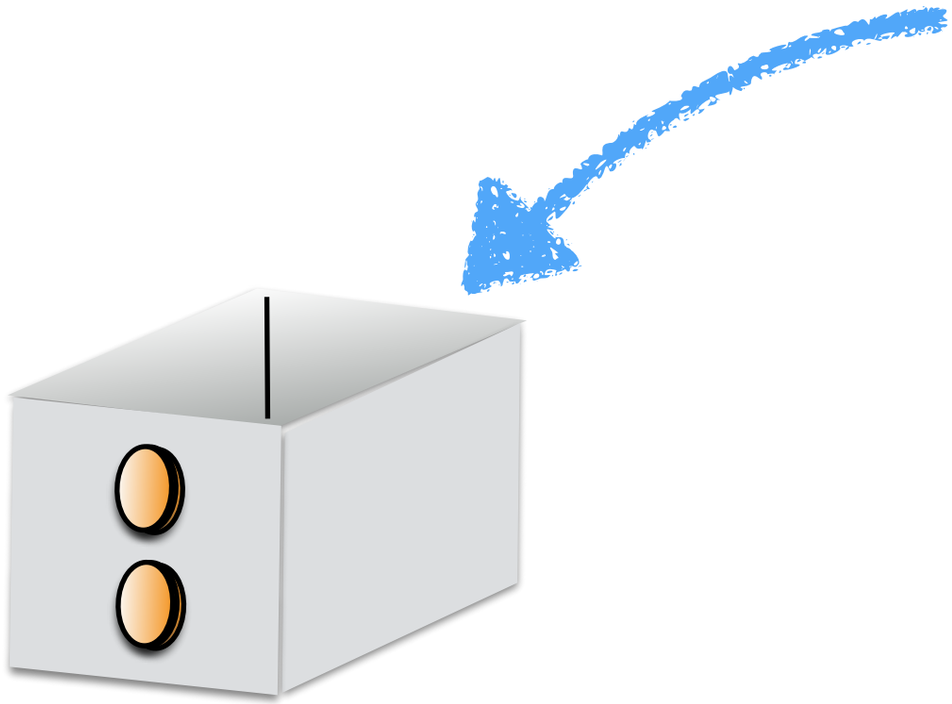
We need to create the blueprint for a box… (object)



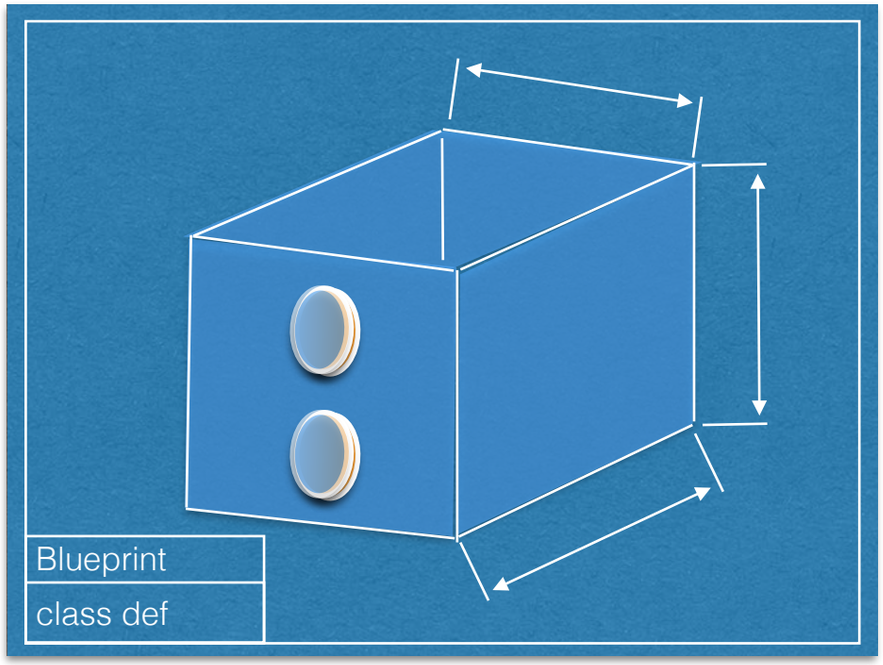
We need to create the blueprint for the box…



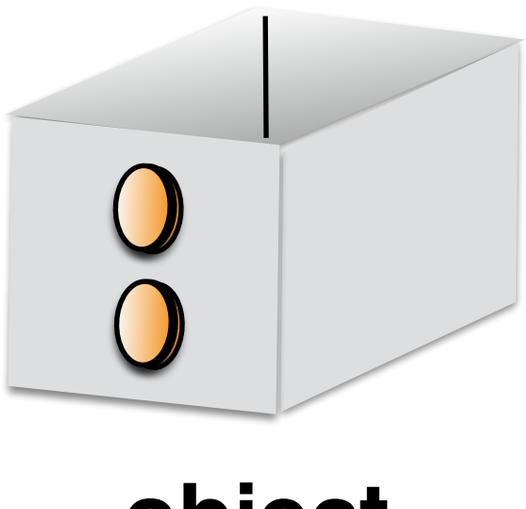

Blueprint
class def

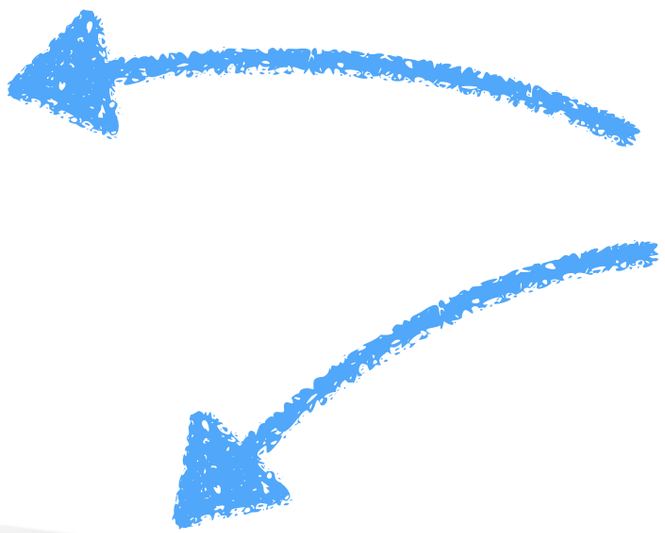


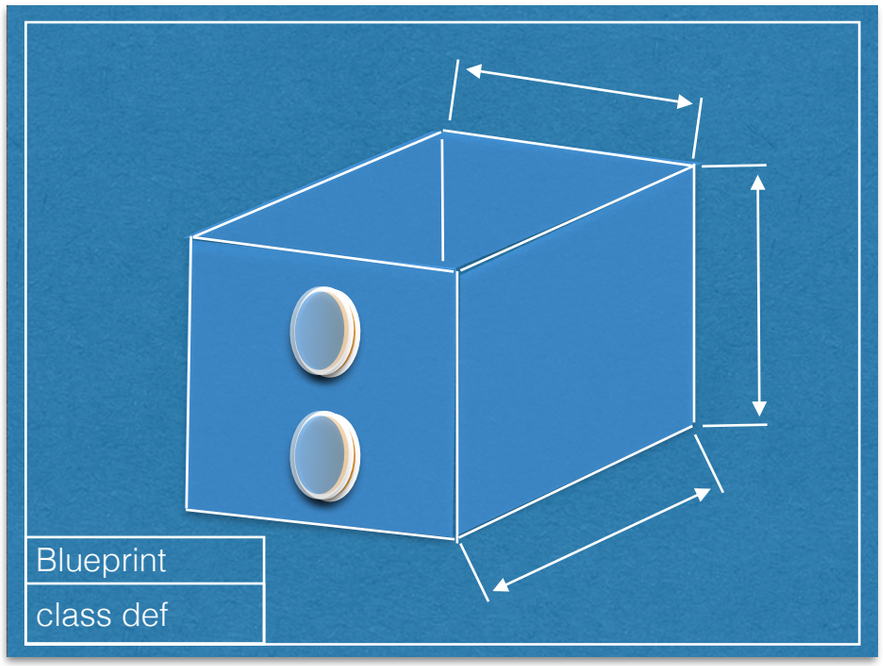
object



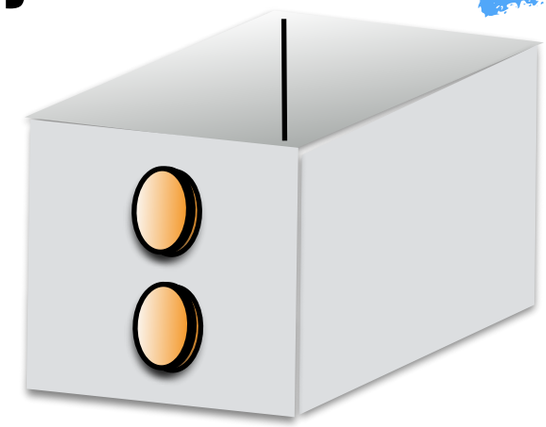
blueprint





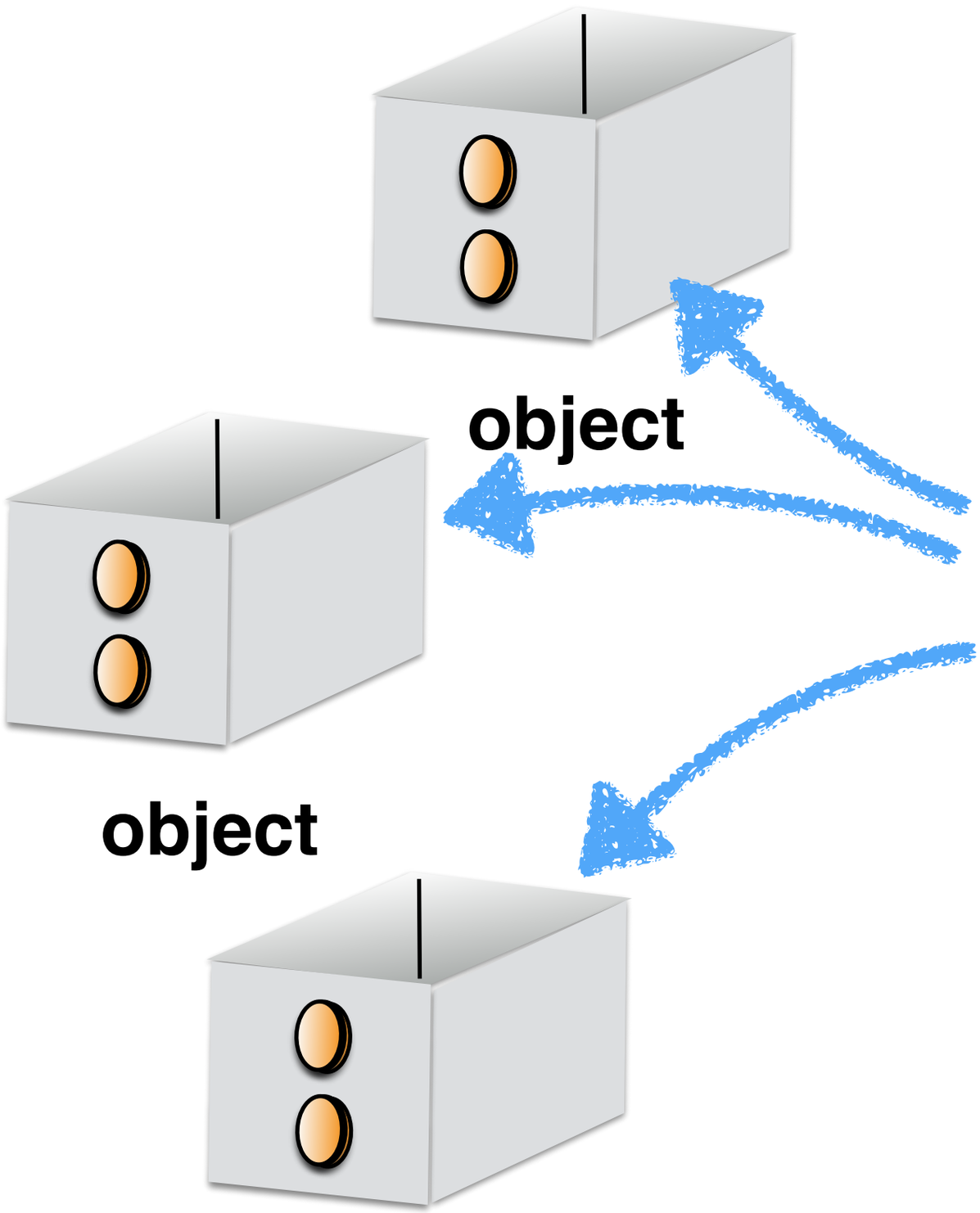


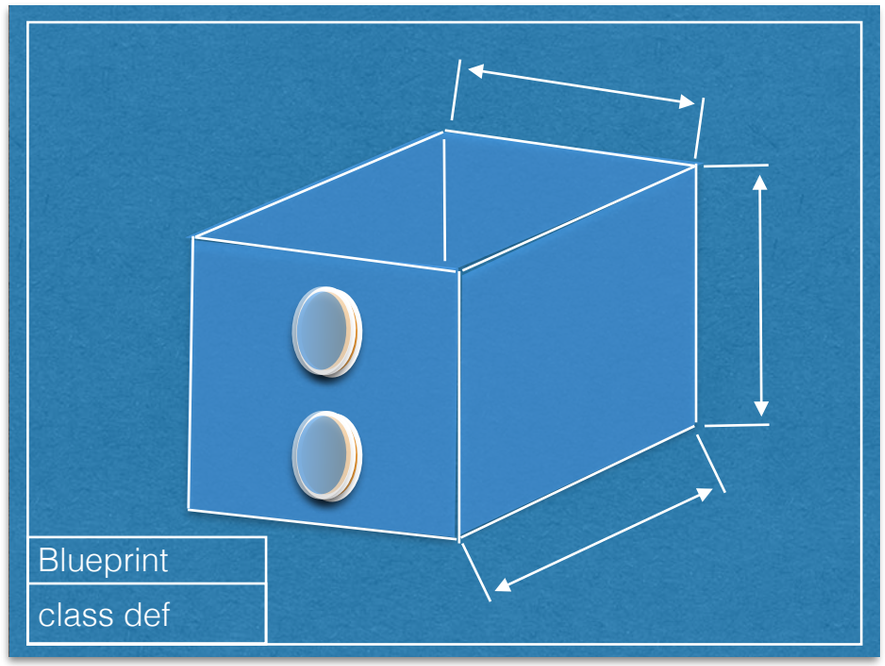
object



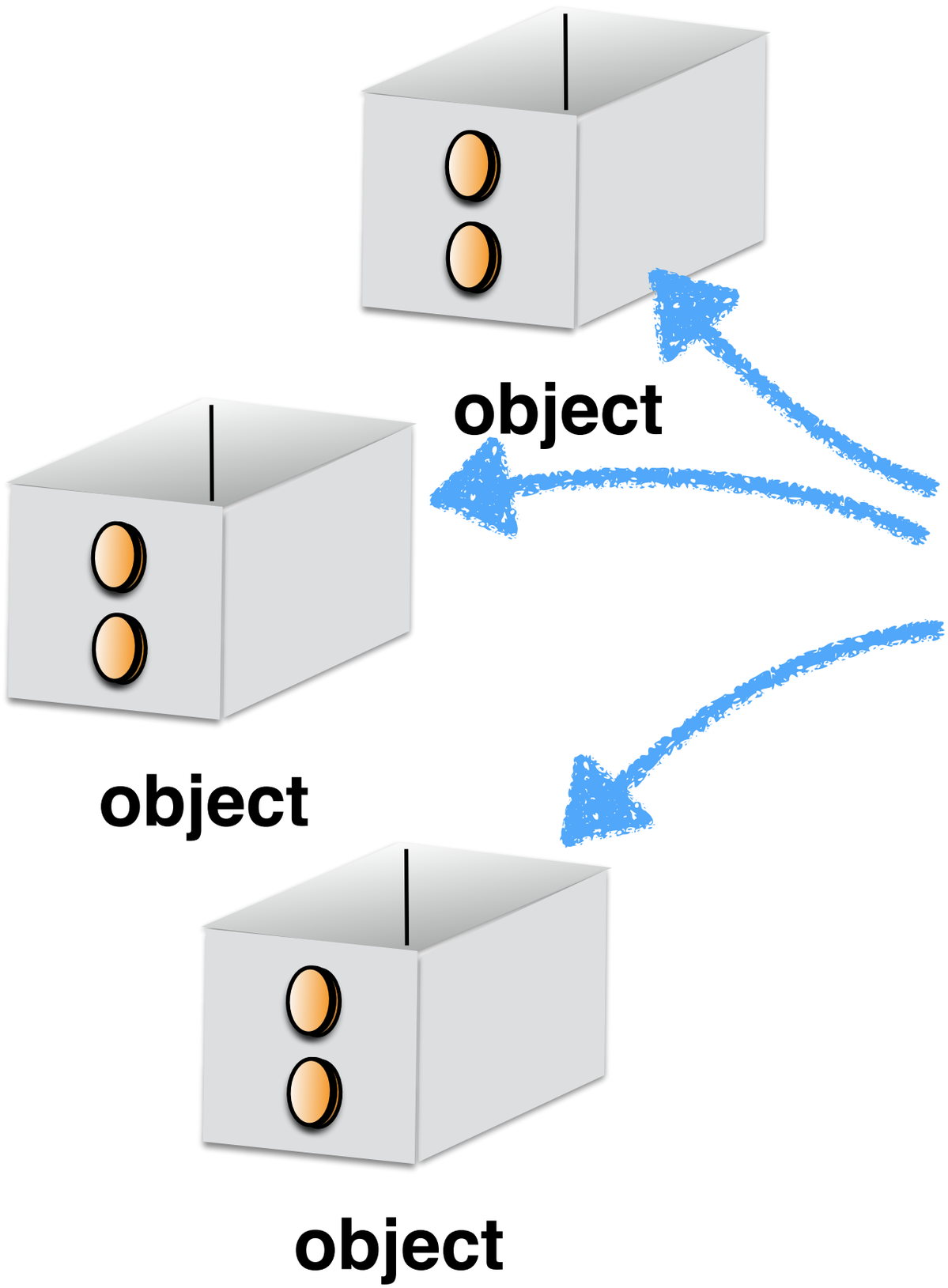
object

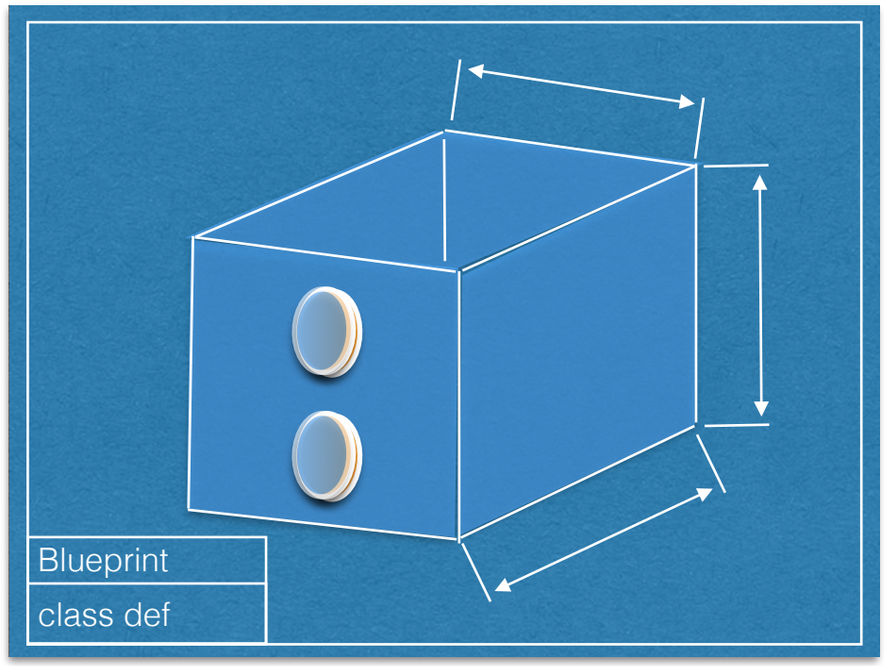
blueprint





blueprint

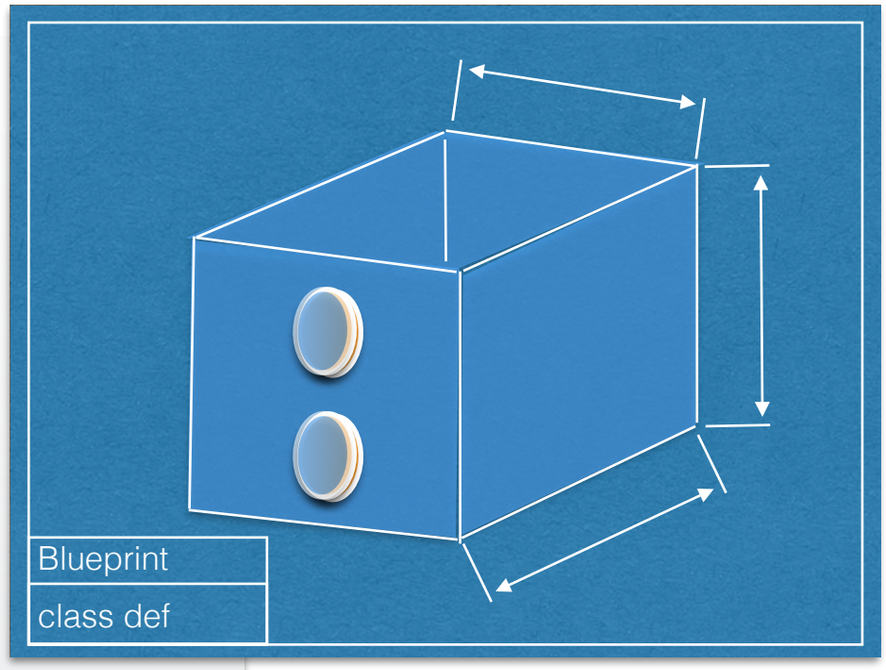




blueprint = class

A Class for a Die

```
# libraries
from random import randrange

# a class for a die
class Die:
    def __init__( self, n ):
        self.noSides = n
        self.value   = 1

def roll( self ):
    self.value = randrange( 1, self.noSides+1 )

def getValue( self ):
    return self.value
```



A Die Class

constructor

```
# libraries
import random

# a class for a die
class Die:
    def __init__( self, n ):
        self.noSides = n
        self.value   = 1

    def roll( self ):
        self.value = random.randrange( 1, self.noSides+1 )

    def getValue( self ):
        return self.value
```

A Die Class

```
# Create 2 dice, one with 6 sides
d1 = Die( 6 )
d2 = Die( 8 )

# Roll both dice
d1.roll( )
d2.roll( )

# display their value
print( "Die 1: ",  d1.getValue() )
print( "Die 2: ",  d2.getValue() )
```

```
# libraries
import random

# a class for a die
class Die:
    def __init__( self, n ):
        self.noSides = n
        self.value   = 1

    def roll( self ):
        self.value = random.randrange( 1,
                                       self.noSides+1 )

    def getValue( self ):
        return self.value
```

A Die Class

Die.__init__(6)

```
# Create 2 dice, one with 6 sides
d1 = Die( 6 )
d2 = Die( 8 )

# Roll both dice
d1.roll( )
d2.roll( )

# display their value
print( "Die 1: ",  d1.getValue() )
print( "Die 2: ",  d2.getValue() )
```

```
# libraries
import random

# a class for a die
class Die:
    def __init__( self, n ):
        self.noSides = n
        self.value   = 1

    def roll( self ):
        self.value = random.randrange( 1,
                                       self.noSides+1 )

    def getValue( self ):
        return self.value
```

A Die Class

reference to the object

```
# libraries
import random

# a class for a die
class Die:
    def __init__( self, n ):
        self.noSides = n
        self.value   = 1

    def roll( self ):
        self.value = random.randrange( 1, self.noSides+1 )

    def getValue( self ):
        return self.value
```

A Die Class

must be 1st param of all methods

```
# libraries
import random

# a class for a die
class Die:
    def __init__( self, n ):
        self.noSides = n
        self.value   = 1

    def roll( self ):
        self.value = random.randrange( 1,
                                       self.noSides+1 )

    def getValue( self ):
        return self.value
```

```
# Create 2 dice, one with 6 sides
d1 = Die( 6 )
d2 = Die( 8 )

# Roll both dice
d1.roll( )
d2.roll( )

# display their value
print( "Die 1: ",  d1.getValue() )
print( "Die 2: ",  d2.getValue() )
```

A Die Class

```
# Create 2 dice, one with 6 sides
d1 = Die( 6 )
d2 = Die( 8 )

# Roll both dice
d1.roll( )
d2.roll( )

# display their value
print( "Die 1: ",  d1.getValue() )
print( "Die 2: ",  d2.getValue() )
```

roll(d1)
roll(d2)

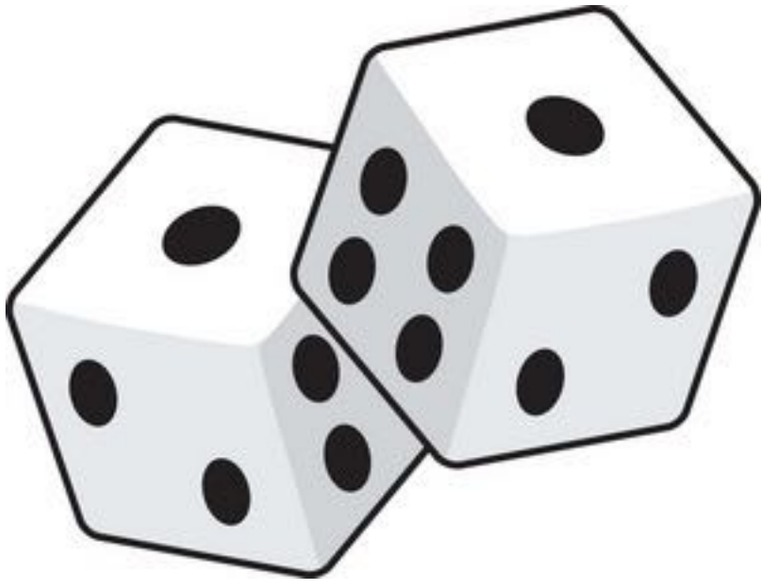
```
# libraries
import random

# a class for a die
class Die:
    def __init__( self, n ):
        self.noSides = n
        self.value   = 1

    def roll( self ):
        self.value = random.randrange( 1,
                                       self.noSides+1 )

    def getValue( self ):
        return self.value
```

A Die Class

```
# libraries
import random

# a class for a die
class Die:
    def __init__( self, n ):
        self.noSides = n
        self.value   = 1

    def roll( self ):
        self.value = random.randrange( 1, self.noSides+1 )

    def getValue( self ):
        return self.value
```

makes the variable a "member" of the object

Playing dice…



We stopped here last time…

Homework Programs

```
# hw6_5.py
# Naomi Jahan
# Homework 6, Problem 5
# 25x25 squares on 600x600 window
# box with my name
from graphics import *
from random import *

def main():
    # open the graphics window
    win = GraphWin( "hw6_5", 600, 600 )

    for i in range( 24 ):
        for j in range( 24 ):
            # define the small rectangles
            r = Rectangle( Point( i*25, j*25 ), Point( (i+1)*25, (j+1)*25 ) )

            # create a random color
            red = randint( 0, 255 )
            green = randint( 0, 255 )
            blue = randint( 0, 255 )
            color = color_rgb( red, green, blue )

            # set the rectangle's color
            r.setFill( color )

            # draw the rectangle
            r.draw( win )

    # draw white rectangle with label
    rLabel = Rectangle( Point( 100, 250 ), Point( 500, 350 ) )
    rLabel.setFill( "white" )
    rLabel.draw( win )

    # draw label
    label = Text( Point( 300, 300 ), "NAOMI JAHAN" )
    label.draw( win )

    # close the graphics window
    win.getMouse()
    win.close()

main()
```

Ln: 41 Col: 16

Great Organization!

Homework Programs

```
#hw6_5
#Mickey Mouse

from graphics import *
from random import*

win = GraphWin("PB5", 600, 600)


for x in range( 0, 600, 25 ):
    for y in range( 0, 600, 25):
        rect = Rectangle(Point(0,0), Point(25, 25))
        rect2 = Rectangle(Point(x, y), (Point(x+25, y+25)))
        red   = randint( 0, 255 )
        green = randint( 0, 255 )
        blue  = randint( 0, 255 )
        color = color_rgb( red, green, blue )
        rect2.setFill( color )
        rect.draw(win)
        rect2.draw(win)

rect3 = Rectangle(Point(100, 250), Point(500, 350))
words = Text(Point(300, 300), "Mickey Mouse")

rect3.setFill("white")
rect3.draw(win)
words.draw(win)


win.getMouse()
win.close()
```

Ln: 23 Col: 43

No main() function. Not Documented

Homework Programs

```
#hw6_5.py
#Tasaday Green
#March 29, 2018
#This program fills a 600x600 pixel window with 25x25 pixel squares of random
#colors then adds a large white rectangle with my name inside

from graphics import*
from random import*

#draw 25x25 squares to fill the window
def squares(i,j,win):
    sq=Rectangle(Point(i,j),Point(i+25,j+25))
    color=randColor()
    sq.setFill(color)
    sq.draw(win)

# create a random color from 3 different RGB values
def randColor():
    red   = randint( 0, 255 )
    green = randint( 0, 255 )
    blue  = randint( 0, 255 )
    color = color_rgb( red, green, blue )
    return color

#draw a white rectangle with name inside
def nameRect(win):
    rect=Rectangle(Point(100,250),Point(500,350))
    rect.setFill("white")
    rect.draw(win)

    name=Text(Point(300,300), "TASADAY GREEN")
    name.draw(win)

#Main function
def main():
    win=GraphWin("Homework 6 Problem 5",600,600)
    for i in range(0,600,25):
        for j in range(0,600,25):
            squares(i,j,win)
    nameRect(win)
main()
```

Ln: 11 Col: 0

Nice Job!

From Now On…

All Programs

Submitted…

Must

Be Documented…

Or Else!

Up to 1 Letter Grade
***Down*,**
For the whole assignment
if Documentation Missing

Min/Max Revisited…

Alex, 3
Max, 4
Sophia, 10
Lujun, 2
Maggie, 5

Photo credit: https://www.dreamstime.com/royalty-free-stock-photos-d-confused-person-question-mark-illustration-rendering-thinking-frastrated-man-red-white-people-man-character-image35217568

Classes and Objects

A Die Class

```
from dieClass import Die

def main():
   # Create 2 dice, one with 6 sides
   d1 = Die( 6 )
   d2 = Die( 8 )

   # Roll both dice
   d1.roll( )
   d2.roll( )

   # display their value
   print( "Die 1: ",  d1.getValue() )
   print( "Die 2: ",  d2.getValue() )

main()
```

playDice.py

```
# dieClass.py
# Definition for a die class.
import random

# a class for a die
class Die:
    def __init__( self, n ):
        self.noSides = n
        self.value   = 1

    def roll( self ):
        self.value = random.randrange( 1,
                                       self.noSides+1 )

    def getValue( self ):
        return self.value
```

dieClass.py

Why Create a Die Class? Randint Could have Sufficed

- **Modularity**
- Details are hidden (**Information hiding**)
- The Die class can easily be ***enhanced/modified*** without having to change main program
    - die with a bias
    - history of rolls
    - keeping track of statistics

Exercise

Write a program that maintains a list of *cat* objects. Cats have a *name*, a *breed*, may or may not be *vaccinated* and have an *age* expressed in years.

Image credits: nicepixy.net

Examples

Minou, 3, vac, stray
Max, 1, not-vac, Burmese
Gizmo, 2, vac, Bengal
Garfield, 4, not-vac, Orange Tabby

Using Cat Objects

```
# Example: using a cat object

# Minou, 3, vac, stray
cat1 = Cat( “Minou”, True, “stray”, 3 )

# Print if cat is vaccinated or not
if cat1.isVaccinated():
    print( cat1.getName(),
           ”is vaccinated” )
else:
    print( cat1.getName(),
           “is not vaccinated” )
```

Wanted:

A program that
- **outputs all the cats**
- **outputs only the vaccinated cats**
- **outputs the cats 2 or older**

Good Methods
To Start With When Creating a Class

- Constructor
- Inspector Methods
- Mutator Methods
- Default string representation

- **__init__**()
- **getValue**()
- **roll**()
- **__str__**()

We stopped here last time…

Converting A String to a Boolean

Untitled

```
a = "Minou, not vaccinated"
words = a.split( "," )
```

Ln: 17 Col: 23

Converting A String to a Boolean

Untitled

```
a = "Minou, not vaccinated"
words = a.split( "," )

if len( words ) == 2:
    # 1
    vaccinated = words[1].strip() == "vaccinated"
```

Ln: 17 Col: 23

Converting A String to a Boolean

Untitled

```
a = "Minou, not vaccinated"
words = a.split( "," )

if len( words ) == 2:
    # 1
    vaccinated = words[1].strip() == "vaccinated"

    # 2
    vaccinated = True
    if words[1].find( "not" ) != -1:
        vaccinated = False
```

Ln: 17 Col: 23

Converting A String to a Boolean

Untitled

```
a = "Minou, not vaccinated"
words = a.split( "," )

if len( words ) == 2:
    # 1
    vaccinated = words[1].strip() == "vaccinated"

    # 2
    vaccinated = True
    if words[1].find( "not" ) != -1:
        vaccinated = False

# 3
vaccinated = words[1].find( "not" ) != -1
```

Ln: 17 Col: 23

Converting A String to a Boolean

Untitled

```
a = "Minou, not vaccinated"
words = a.split( "," )

if len( words ) == 2:
    # 1
    vaccinated = words[1].strip() == "vaccinated"

    # 2
    vaccinated = True
    if words[1].find( "not" ) != -1:
        vaccinated = False

    # 3
    vaccinated = words[1].find( "not" ) != -1

# 4
vaccinated = a.lower().find( "not vac" ) != -1
```

Ln: 17 Col: 23

Important Concepts:

LOCAL vs. GLOBAL

locality.py - /Users/thiebaut/Desktop/locality.py (3.5.0b1)

```
def func1( x ):
    a = 3
    print( x * a )

def func2( y ):
    print( y * a )

def main():
    a = 100
    func1( 10 )
    func2( 20 )

main()
```

Ln: 9 Col: 11

What can you say about this program?
Focus on the variable **a**…

locality.py - /Users/thiebaut/Desktop/locality.py (3.5.0b1)

```
def func1( x ):
    a = 3
    print( x * a )

def func2( y ):
    print( y * a )

def main():
    a = 100
    func1( 10 )
    func2( 20 )

main()
```

Local Variable

Ln: 9 Col: 11

What can you say about this program?
Focus on the variable **a**...

locality.py - /Users/thiebaut/Desktop/locality.py (3.5.0b1)

```
def func1( x ):
    a = 3
    print( x * a )

def func2( y ):
    print( y * a )

def main():
    a = 100
    func1( 10 )
    func2( 20 )

main()
```

Local Variable

Error!

Ln: 9 Col: 11

What can you say about this program?
Focus on the variable **a**...

locality.py - /Users/thiebaut/Desktop/locality.py (3.5.0b1)

```
def func1( x ):
    a = 3
    print( x * a )

def func2( y ):
    print( y * a )

def main():
    a = 100
    func1( 10 )
    func2( 20 )

main()
```

Local Variable

Error!

Local Variable

Ln: 9 Col: 11

What can you say about this program?
Focus on the variable **a**…

locality.py - /Users/thiebaut/Desktop/locality.py (3.5.0b1)

```
a = 3
def func1( x ):
    print( x * a )

def func2( y ):
    print( y * a )

def main():
    func1( 10 )
    func2( 10 )

main()
```

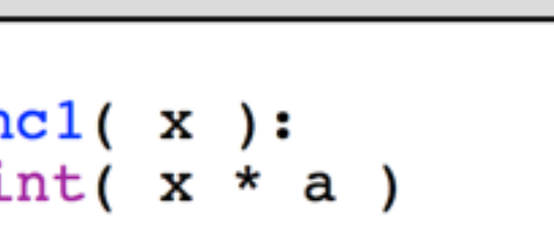
Ln: 12 Col: 6

What can you say about this **new** program?
Focus on the variable **a**...

locality.py - /Users/thiebaut/Desktop/locality.py (3.5.0b1)

```
a = 3
def func1( x ):
    print( x * a )

def func2( y ):
    print( y * a )

def main():
    func1( 10 )
    func2( 10 )

main()
```

Ln: 12 Col: 6



30
30

locality.py - /Users/thiebaut/Desktop/locality.py (3.5.0b1)

```
a = 3
def func1( x ):
    print( x * a )

def func2( y ):
    print( y * a )

def main():
    func1( 10 )
    func2( 10 )

main()
```

Ln: 12 Col: 6

Global Variable

30
30

locality.py - /Users/thiebaut/Desktop/locality.py (3.5.0b1)

```
a = 3
def func1( x ):
    print( x * a )

def func2( y ):
    a = 8
    print( y * a )

def main():
    func1( 10 )
    func2( 10 )
    print( a )

main()
```

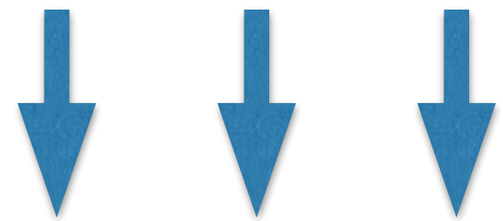
Ln: 9 Col: 2

What can you say about this **third** program?
Focus on the variable **a**…

locality.py - /Users/thiebaut/Desktop/locality.py (3.5.0b1)

```
a = 3
def func1( x ):
    print( x * a )

def func2( y ):
    a = 8
    print( y * a )

def main():
    func1( 10 )
    func2( 10 )
    print( a )

main()
```

Ln: 9 Col: 2



30

80

3

locality.py - /Users/thiebaut/Desktop/locality.py (3.5.0b1)

```
a = 3
def func1( x ):
    print( x * a )

def func2( y ):
    a = 8
    print( y * a )

def main():
    func1( 10 )
    func2( 10 )
    print( a )

main()
```

Ln: 9 Col: 2

Global Variable

Local Variable

30

80

3

locality.py - /Users/thiebaut/Desktop/locality.py (3.5.0b1)

```
a = 3
def func1( x ):
    print( x * a )

def func2( y ):
    global a
    a = 8
    print( y * a )

def main():
    func1( 10 )
    func2( 10 )
    print( a )
```

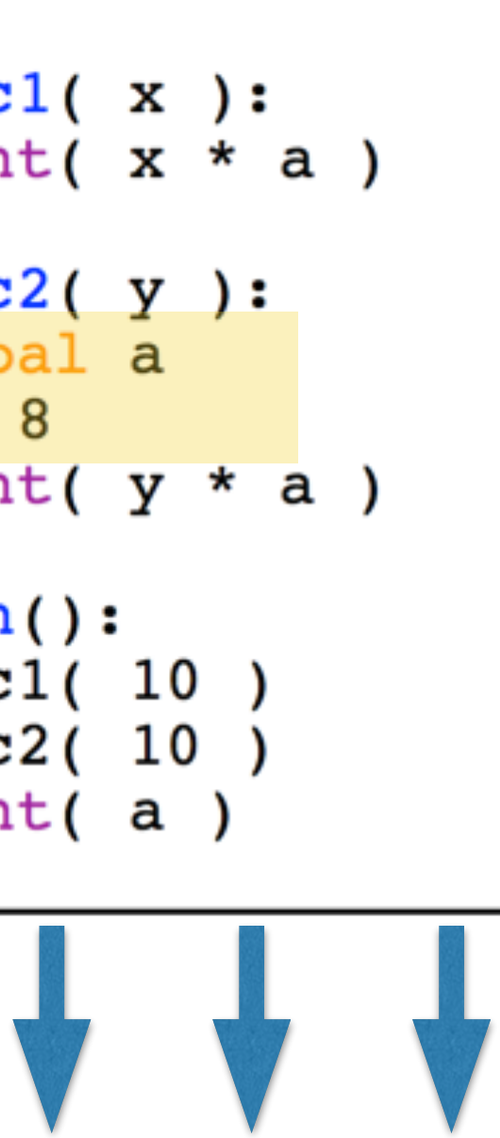
Ln: 10 Col: 10

What can you say about this **fourth** program?
Focus on the variable **a**…

locality.py - /Users/thiebaut/Desktop/locality.py (3.5.0b1)

```
a = 3
def func1( x ):
    print( x * a )

def func2( y ):
    global a
    a = 8
    print( y * a )

def main():
    func1( 10 )
    func2( 10 )
    print( a )
```

Ln: 10 Col: 10



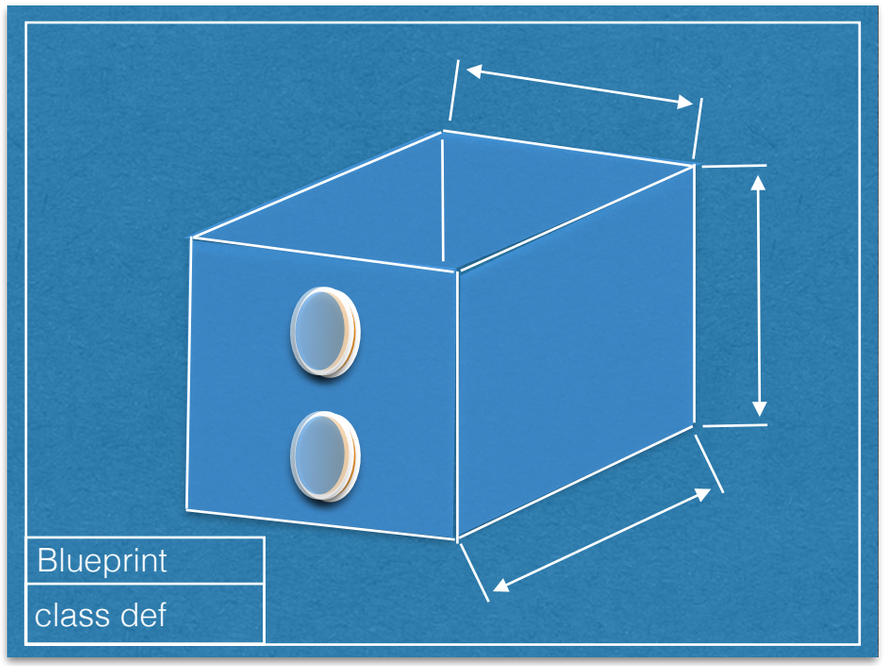
30

80

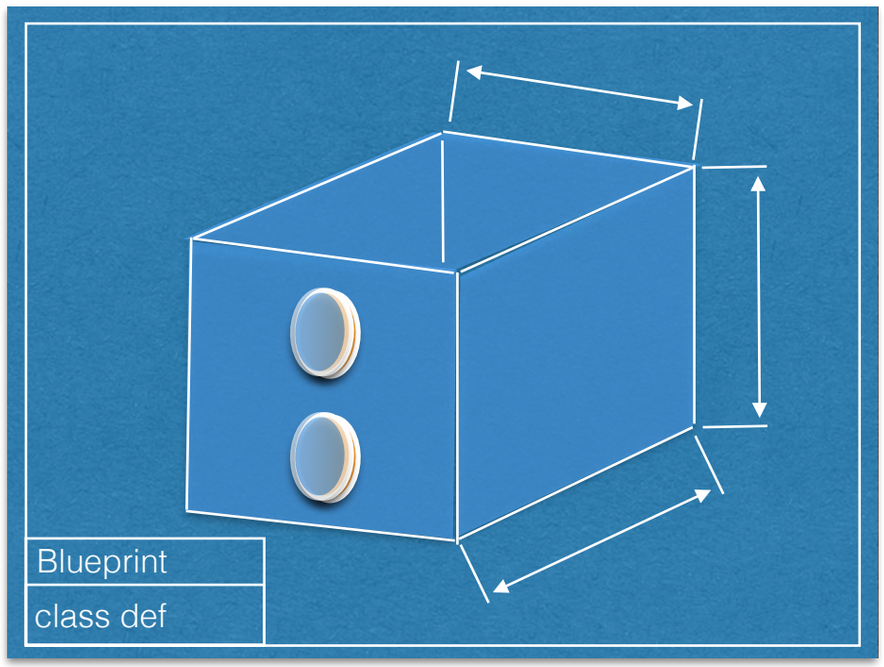
8

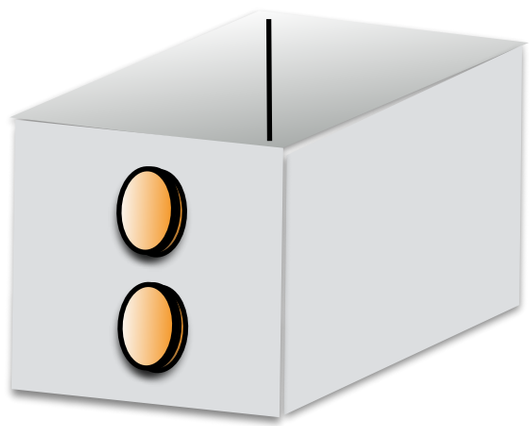
Review

Review

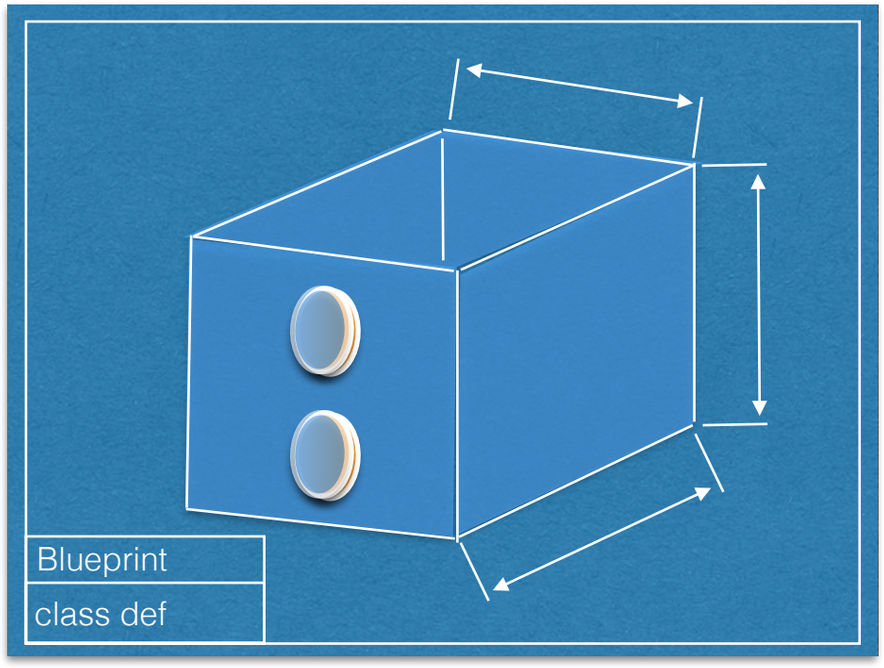


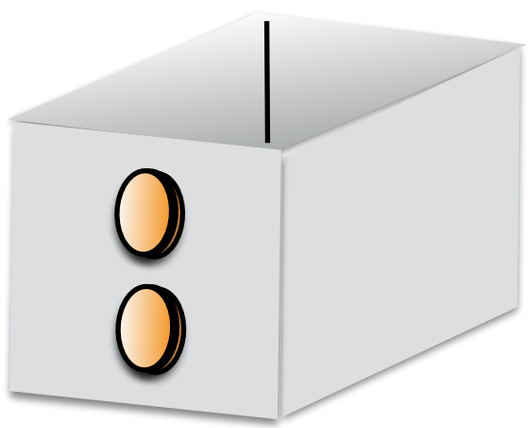
Review



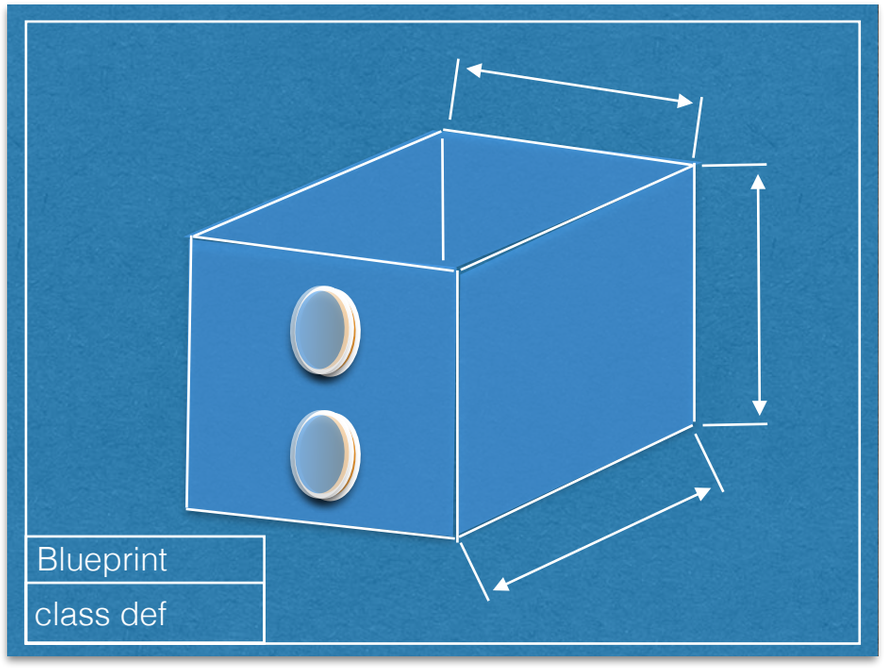


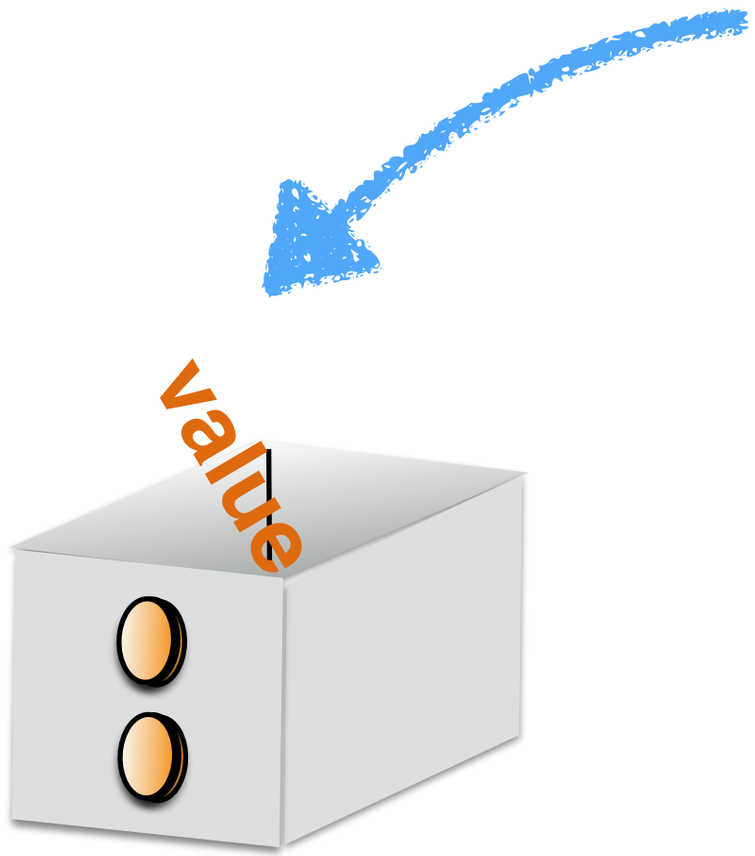
Review



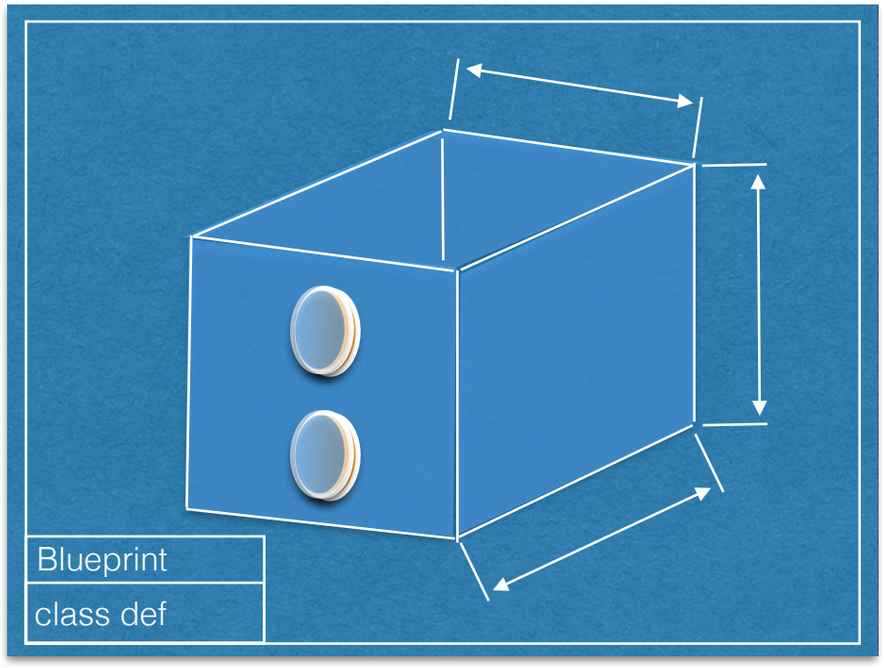


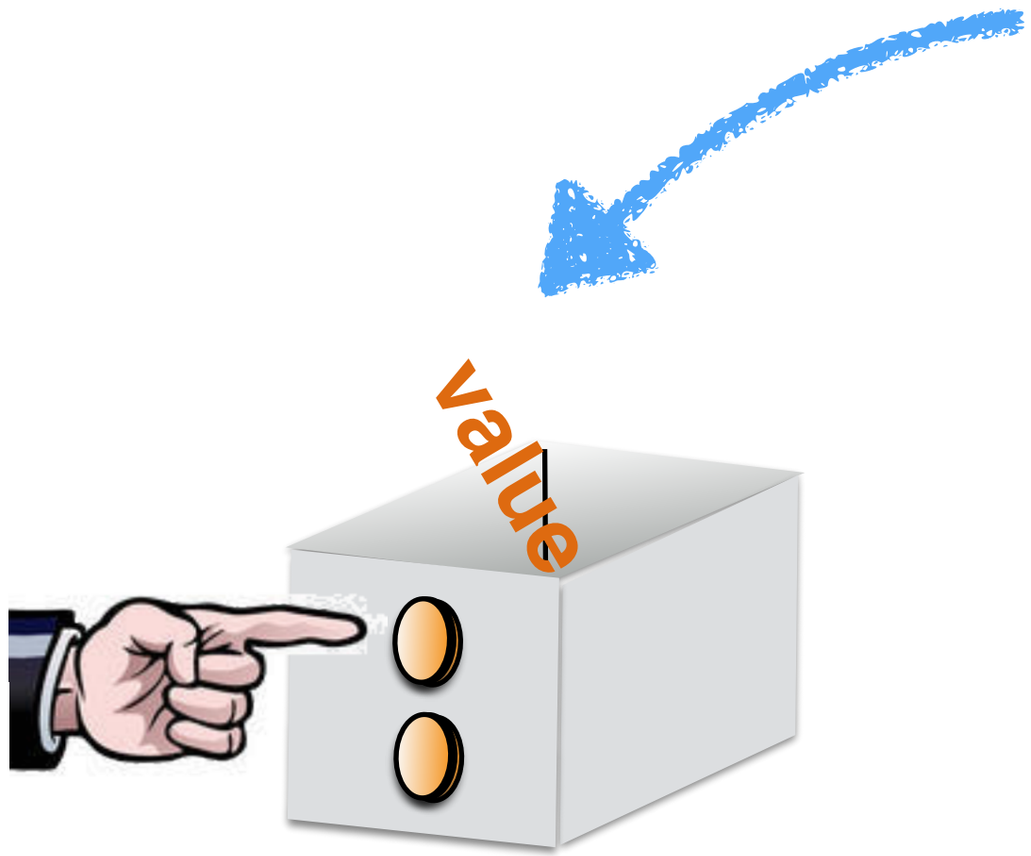
Review





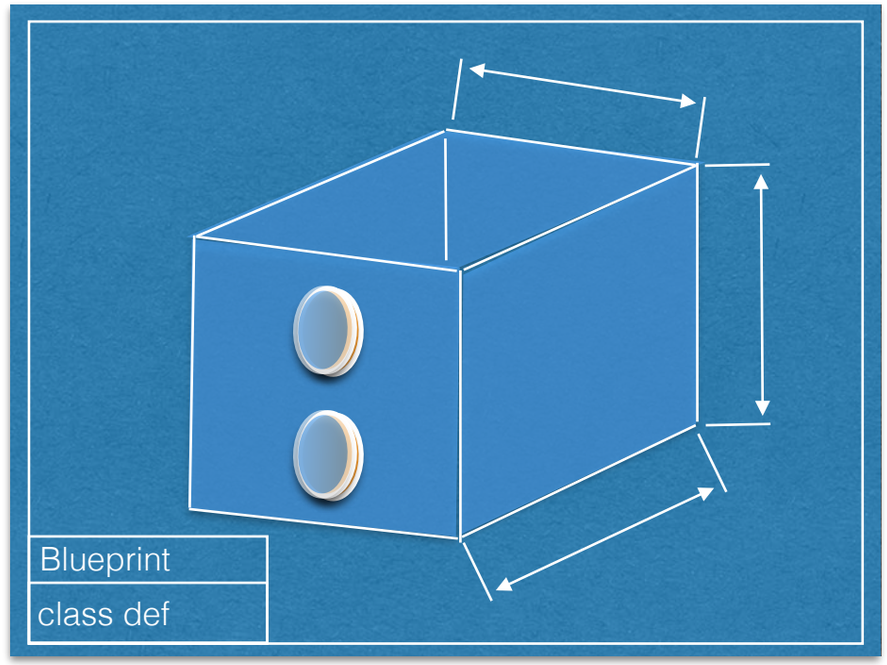
Review





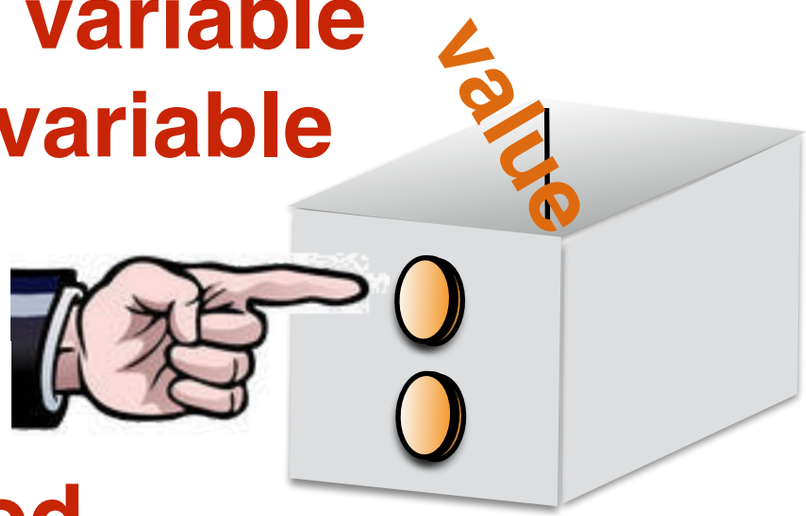
Review

Instantiation:
Object is instance of a Class



Class

member variable
instance variable



method

Review: A Die Class

```
# Create 2 dice, one with 6 sides
d1 = Die( 6 )
d2 = Die( 8 )

# Roll both dice
d1.roll( )
d2.roll( )

# display their value
print( "Die 1: ",  d1.getValue() )
print( "Die 2: ",  d2.getValue() )
```

```
# libraries
import random

# a class for a die
class Die:
    def __init__( self, n ):
        self.noSides = n
        self.value   = 1

    def roll( self ):
        self.value = random.randrange( 1,
                                       self.noSides+1 )

    def getValue( self ):
        return self.value
```

Review: A Die Class

WHY self. ?

```
# Create 2 dice, one with 6 sides
d1 = Die( 6 )
d2 = Die( 8 )

# Roll both dice
d1.roll( )
d2.roll( )

# display their value
print( "Die 1: ",  d1.getValue() )
print( "Die 2: ",  d2.getValue() )
```

```
# libraries
import random

# a class for a die
class Die:
    def __init__( self, n ):
        self.noSides = n
        self.value   = 1

    def roll( self ):
        self.value = random.randrange( 1,
                                       self.noSides+1 )

    def getValue( self ):
        return self.value
```

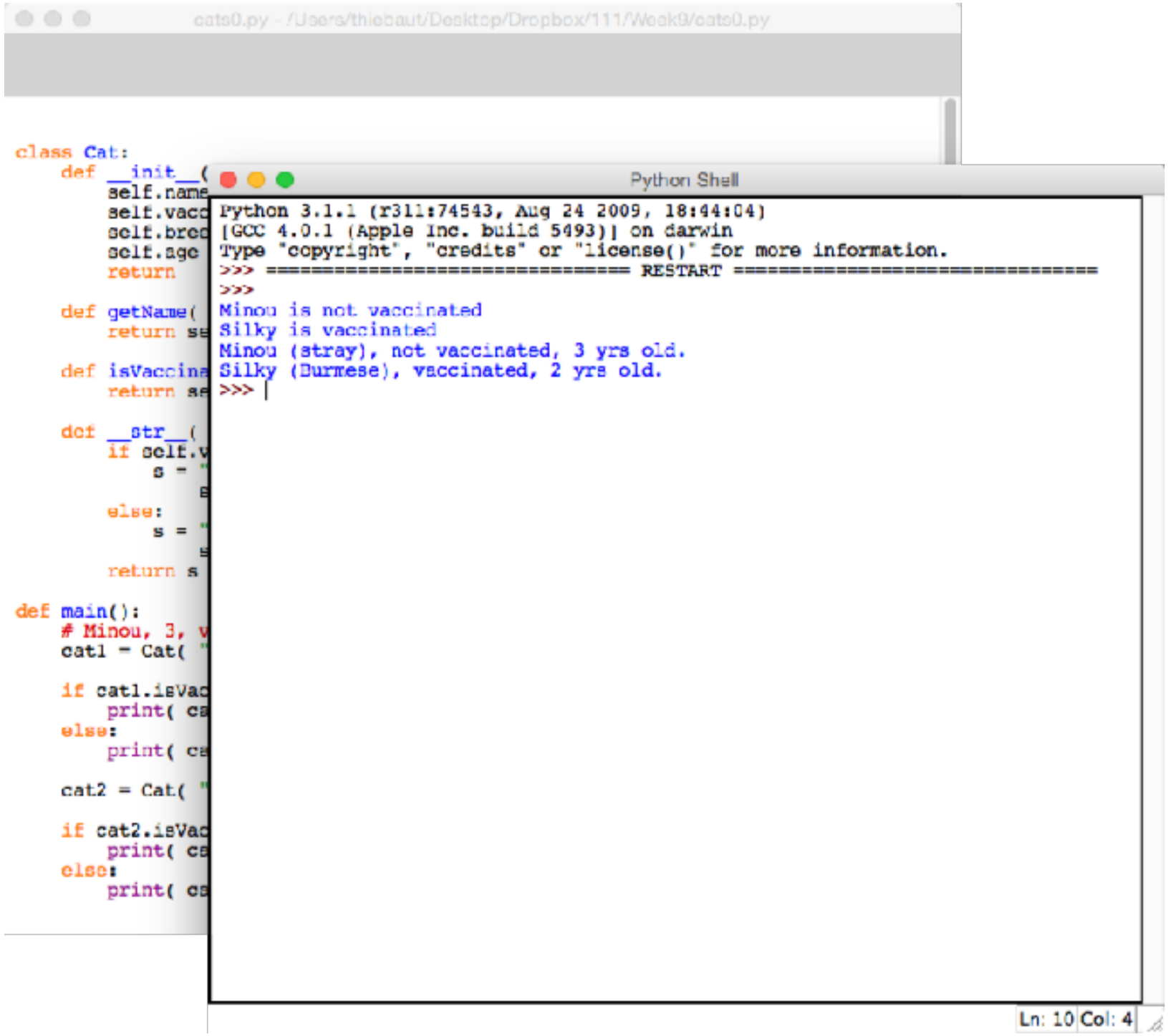
- Pair Programming in Lab 9
- Review of Classes and Objects
  - **Cats, Cats, Cats…**
    - **Default string representation**
    - **List of Cats**
    - **Reading CSV Files of Cats**
    - **Searching for a Cat in a List**

Back to Cats

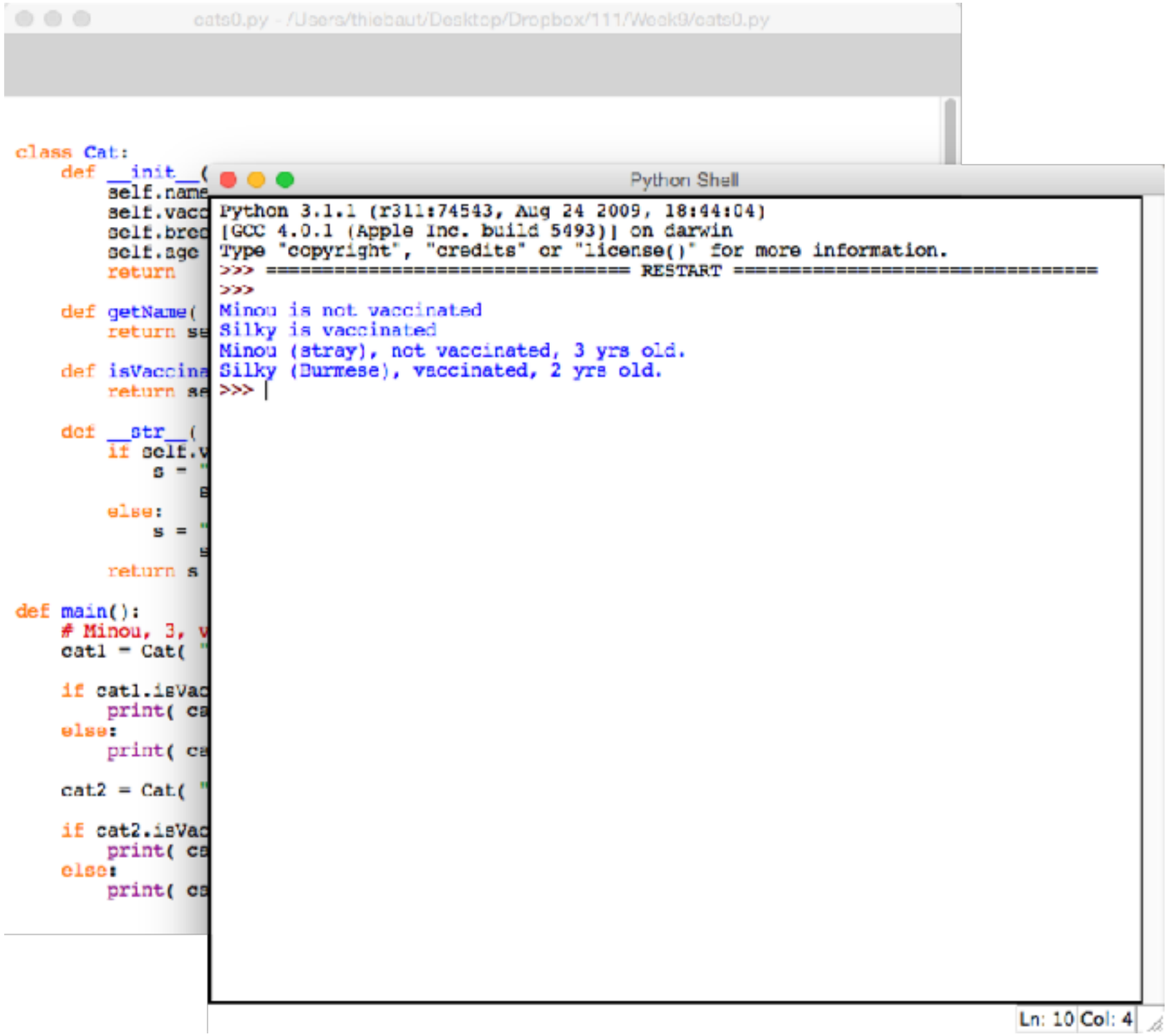
Using Cat Objects

```
# Minou, 3, vaccinated, stray
cat1 = Cat( “Minou”, True, “stray”, 3 )

if cat1.isVaccinated():
    print( cat1.getName(),
           ”is vaccinated” )
else:
    print( cat1.getName(),
           “is not vaccinated” )
```

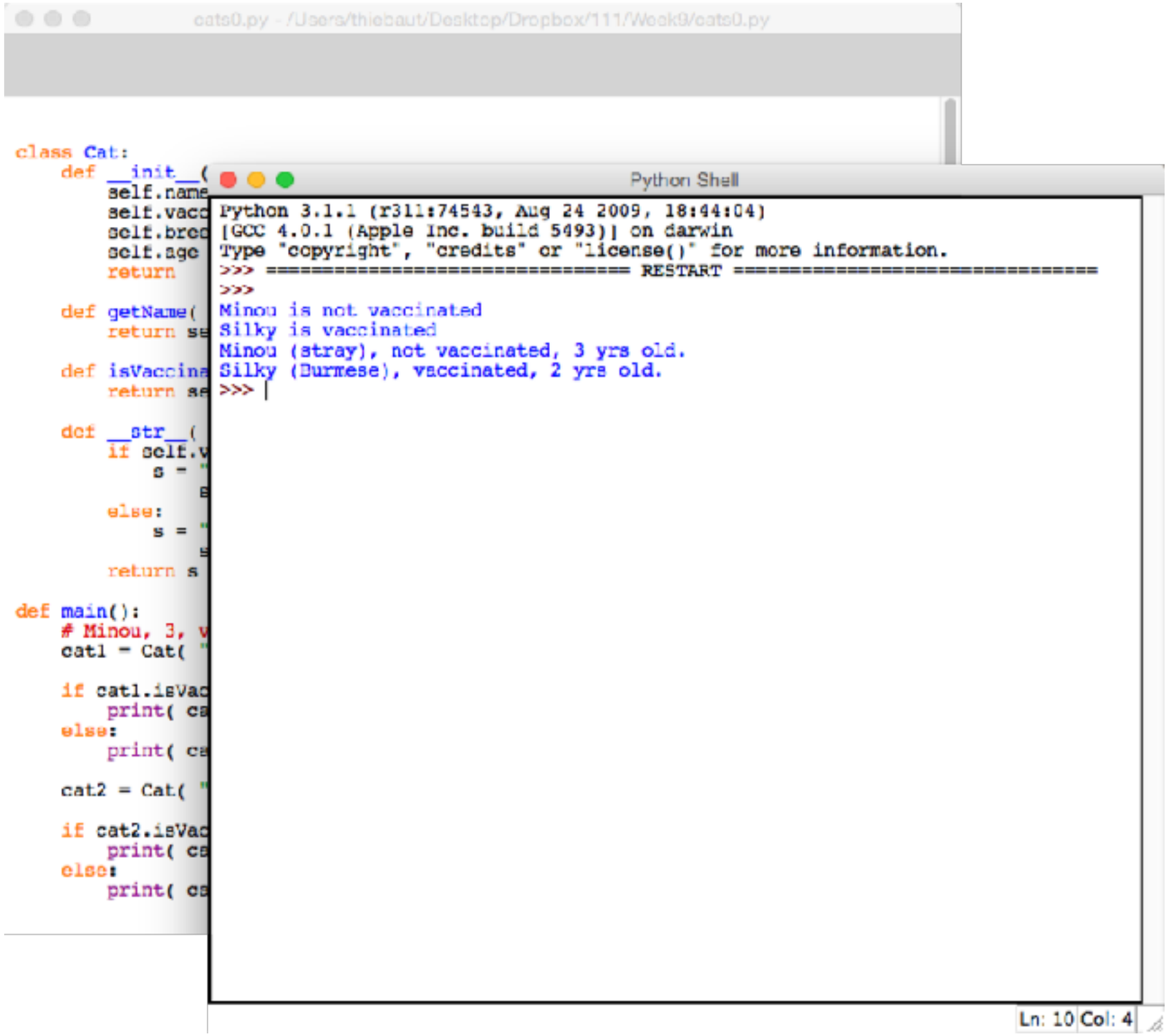
Step 1: Implement the Class



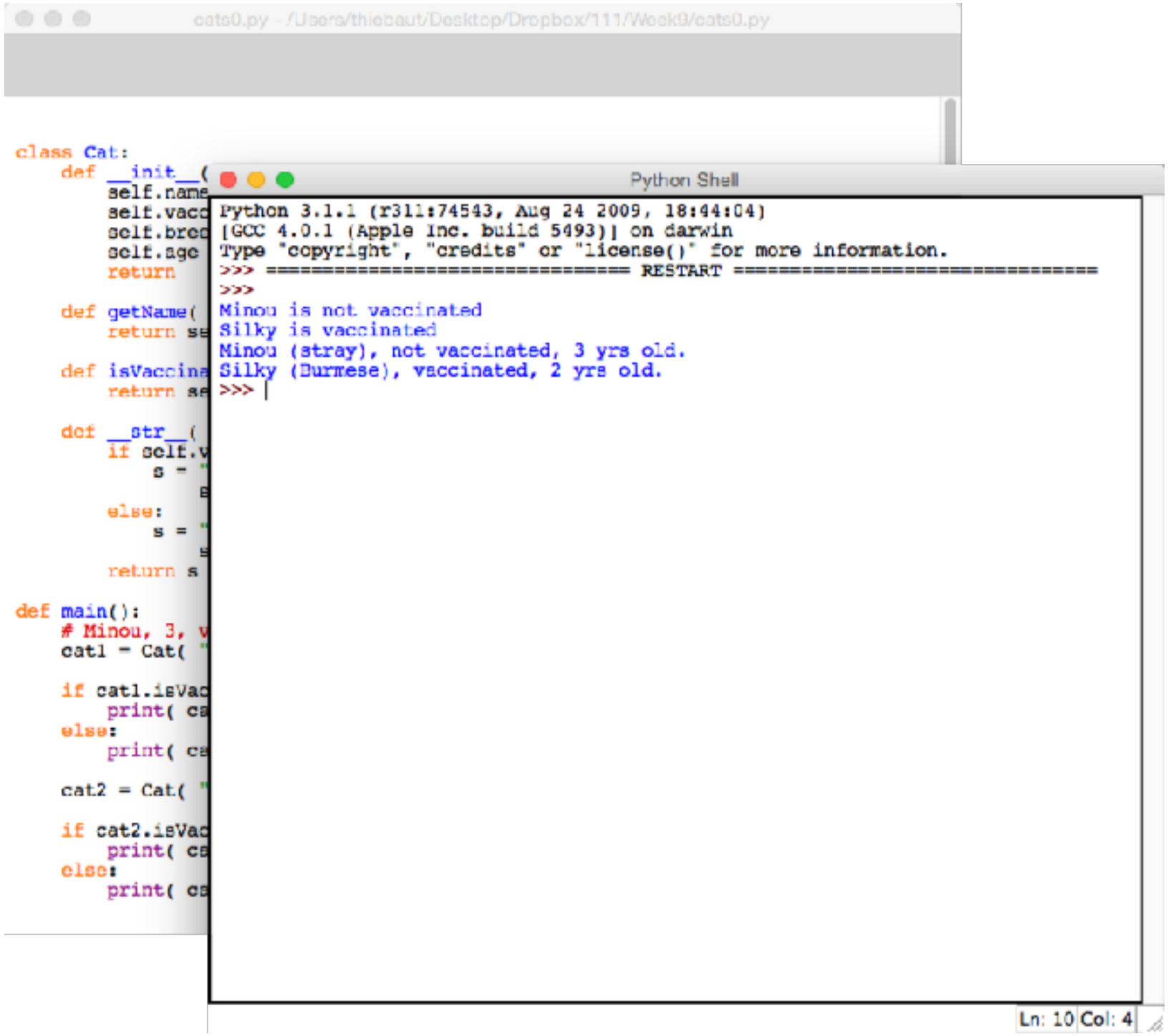
Step 2: Create a List of Cats



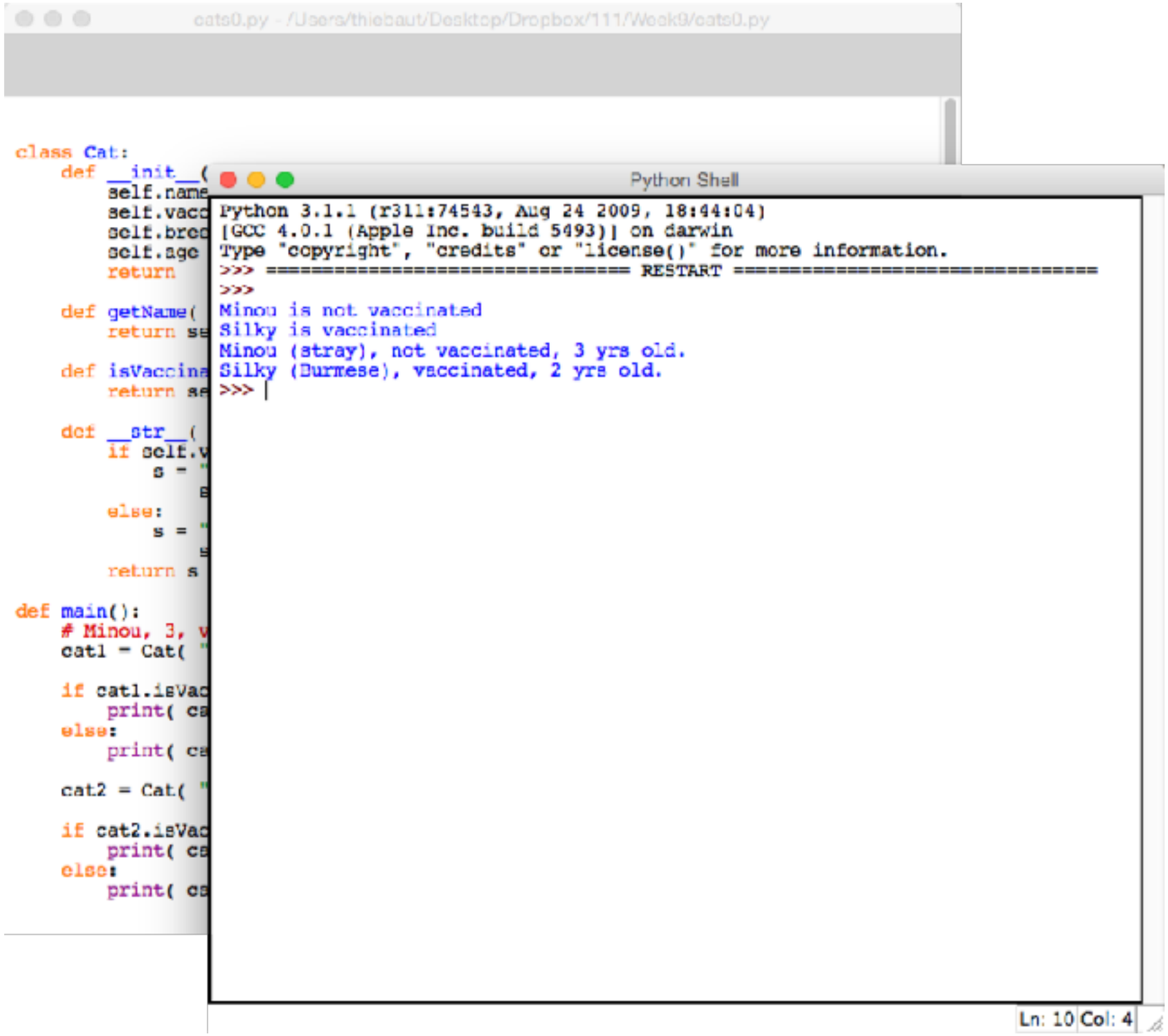
Step 3: Read a CSV File of Cats



Step 4: Display Only Vaccinated Cats

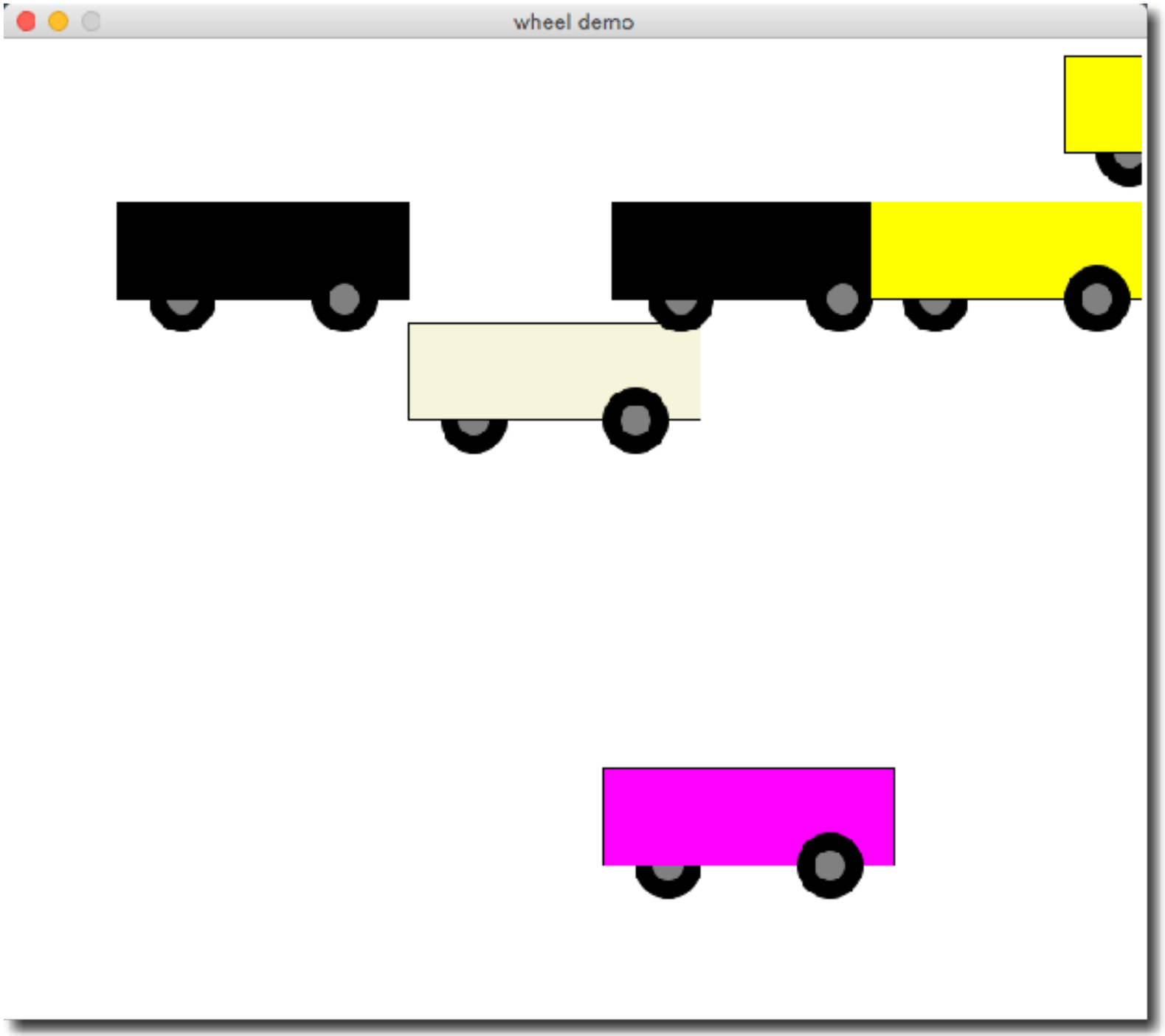


Step 5: Search for the Youngest Cat

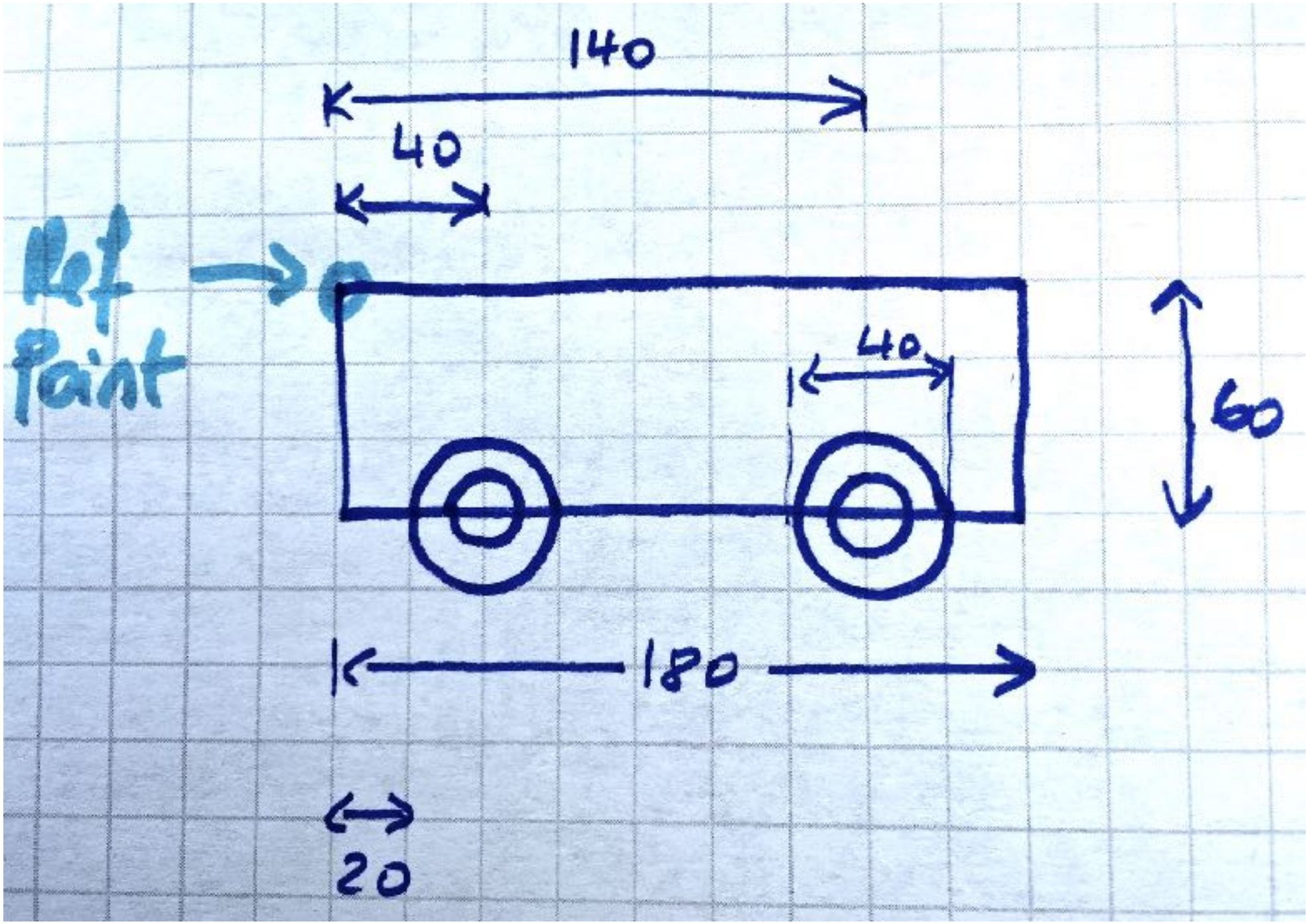


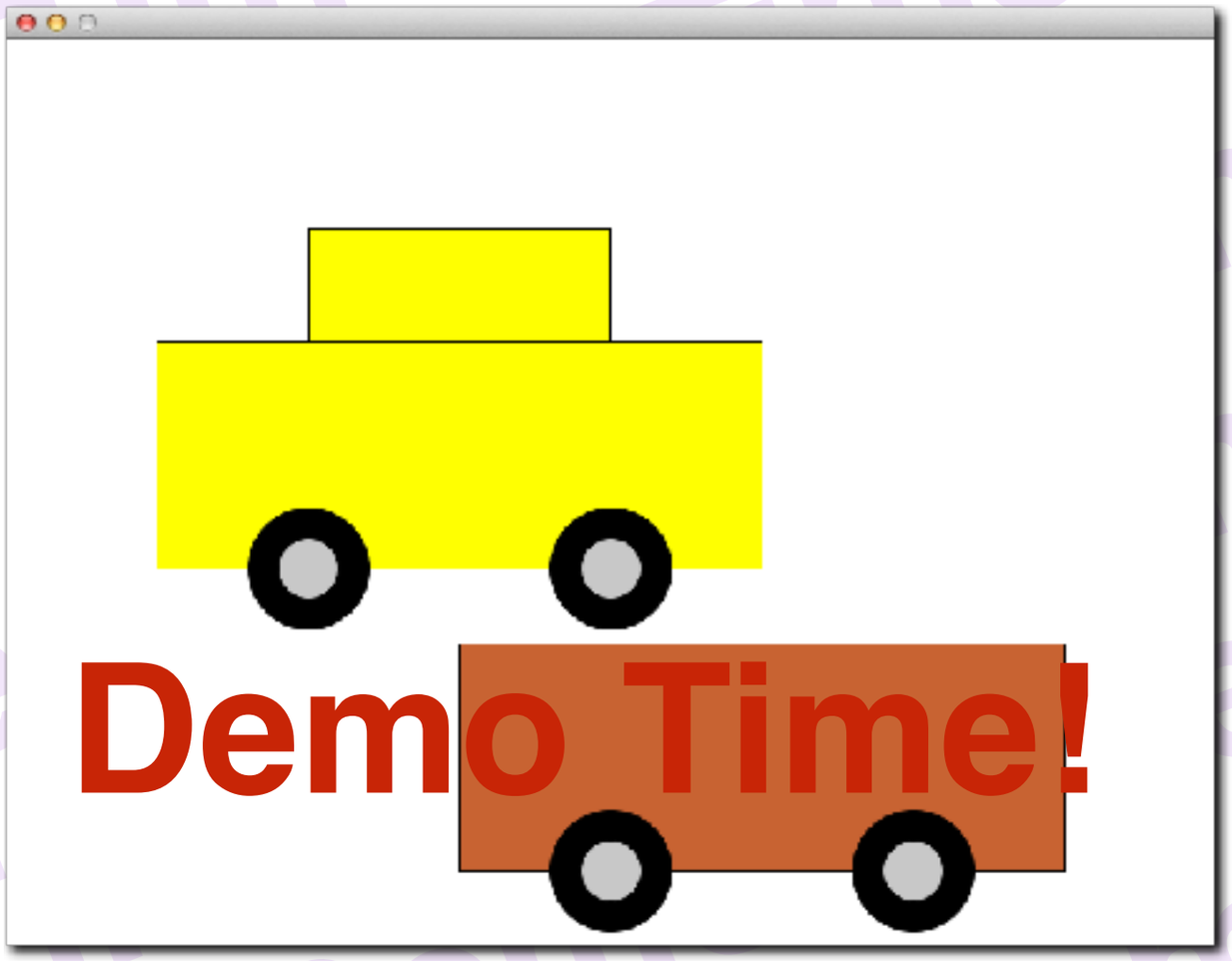
- Looping through a list of objects
- **Object-Oriented Graphics**

Graphic Cars Moving Around

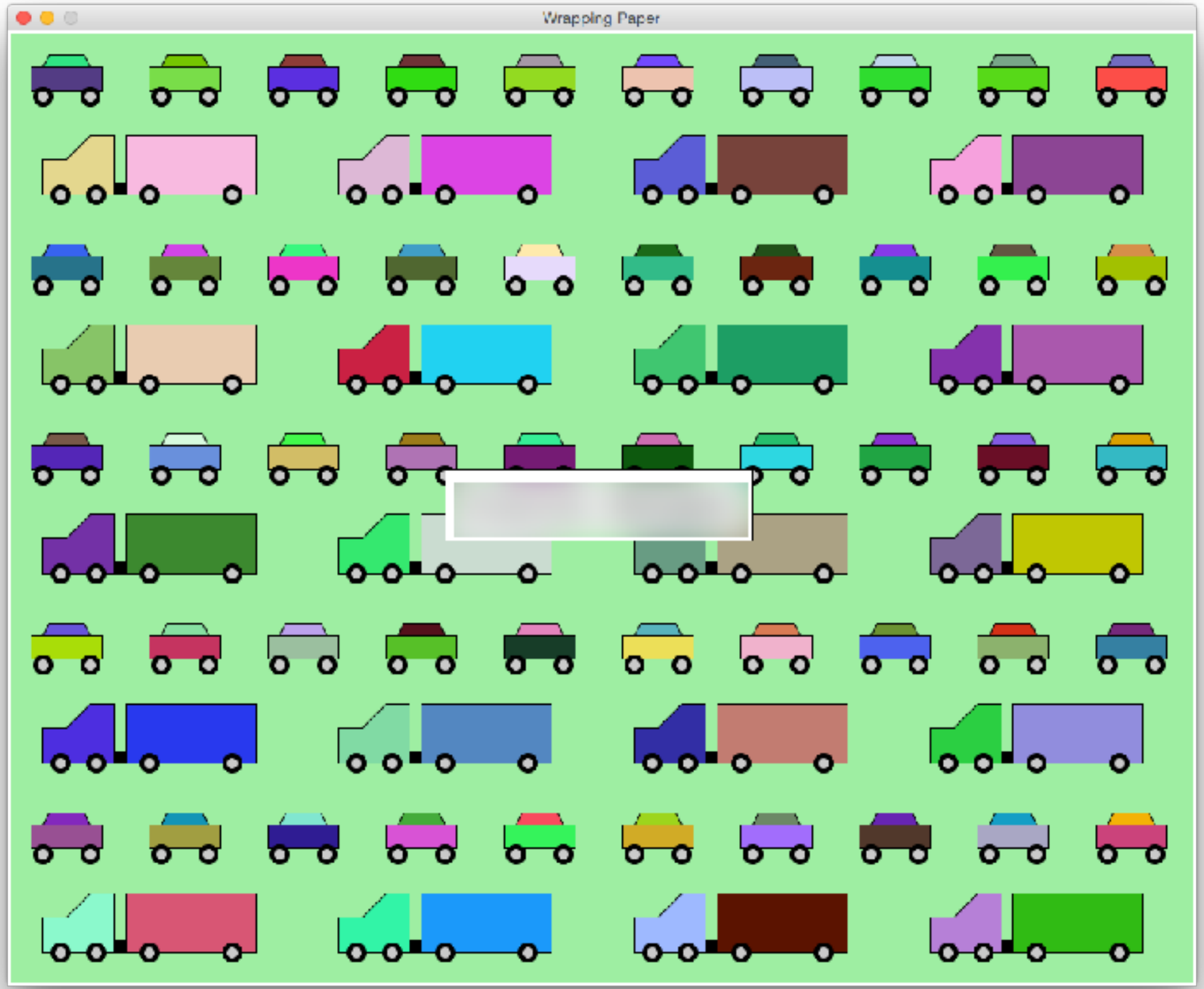


Car Geometry

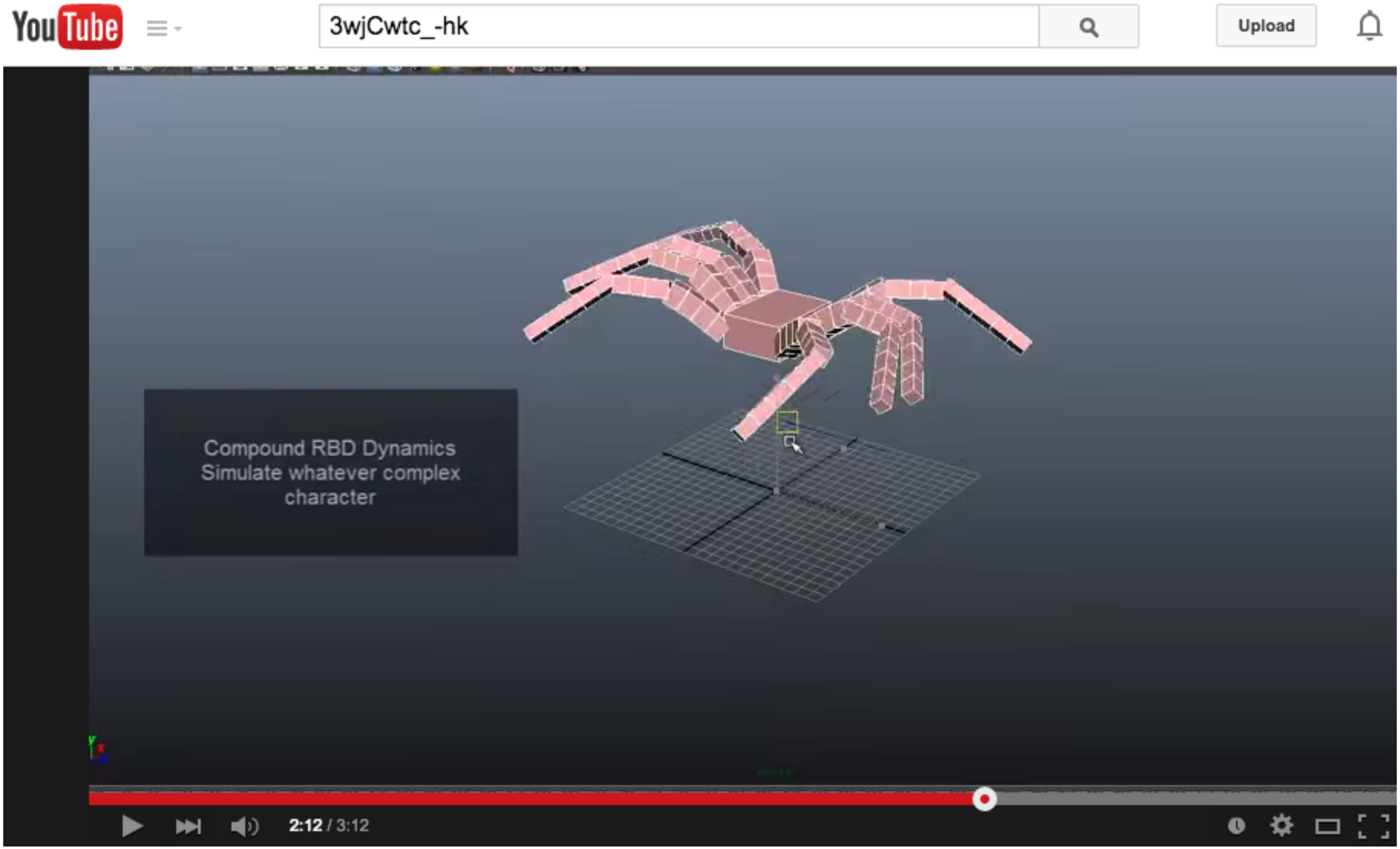



Demo Time!

Inspiration…



Video 1



https://www.youtube.com/watch?v=3wjCwtc_-hk

Video 2

https://www.youtube.com/watch?v=pqBSNAOsMDc

Video 3

https://www.youtube.com/watch?v=4GxPrESfdnM