CS21: INTRODUCTION TO COMPUTER SCIENCE

Prof. Mathieson
Fall 2017
Swarthmore College
Amaechi    owl
Sajal      parrot
Matt       pelican
Youssef    pigeon
David      puffin
Ian        quail
Brandon    raven
Andrew     robin
Allan      seagull
Sagnik     sparrow
Nick       lark
Rutger     kestrel
Adi        ibis
Austin     hawk
Maddie     dodo
Mikey      cuckoo
Peem       crane
Shani      cardinal
Patrick    duck
Ari        heron
Yusa       buzzard
Sid        bluejay
Miryam     eagle
Talia      egret
Sophia     falcon
Skylar     finch
Anar       flamingo
Kyle       grouse
Bayliss    ostrich
Shirline   osprey
Claudia    magpie
Chris      loon
Abby       swan
Outline Oct 2:

- Lab 2 examples
- Introduction to object-oriented programming
- Start graphics
- Random circle program (circles.py)
- Cat face program (cat_face.py)

Notes

- Quiz 2 this Friday (study guide online)
- Lab 4 due Saturday night
- Office Hours 3-5pm on Friday (or by appointment)
Lab 2 Examples

(not posted online)
Graphics and Object Oriented Programming (OOP)
Graphics example: Swimming Fish
Goals for this week

• Understand the idea of OOP

• Be able to create objects and call methods

• Become comfortable with the vocabulary of OOP

• Be able to use the graphics library documentation to learn new types and methods
Idea of Object Oriented Programming

Objects have:

* Data
* Methods
* Type
Idea of Object Oriented Programming

Objects have:

* Data
* Methods
* Type

```python
>>> p = Point(100, 200)
>>> p.setFill("red")
>>> type(p)
<class 'graphics.Point'>
```
Idea of Object Oriented Programming

Objects have:

* Data
* Methods
* Type

Constructor for the Point class

```python
>>> p = Point(100, 200)
>>> p.setFill("red")
>>> type(p)
<class 'graphics.Point'>
```
Idea of Object Oriented Programming

Objects have:

* Data
* Methods
* Type

Constructor for the `Point` class

The x and y coordinates form the data for p

```python
>>> p = Point(100, 200)
>>> p.setFill("red")
>>> type(p)
<class 'graphics.Point'>
```
Idea of Object Oriented Programming

Objects have:
* Data
* Methods
* Type

setFill(..) is a method, not a function

Constructor for the Point class

The x and y coordinates form the data for p

```python
>>> p = Point(100, 200)
>>> p.setFill("red")
>>> type(p)
<class 'graphics.Point'>
```
Idea of Object Oriented Programming

Objects have:
* Data
* Methods
* Type

Constructor for the Point class

The x and y coordinates form the data for p

```
>>> p = Point(100,200)
>>> p.setFill("red")
>>> type(p)
<class 'graphics.Point'>
```

The type of p is Point, p is an instance of the Point class

setFill(..) is a method, not a function
Graphics Setup

- **Origin**: $(0, 0)$
- **Width**: 800
- **Height**: 600
- **GraphWin**
- **Line**
- **Circle** $r = 100$ at $(100, 400)$
- **Rectangle**
- **Point** $p = (100, 200)$
- **Point** $q = (500, 300)$
- **Axes**
  - **$x$-axis**
  - **$y$-axis**

Coordinates:
- $(0, 0)$
- $(800, 0)$
- $(800, 600)$
- $(0, 600)$
Random circles (circles.py)
Websites to bookmark

• Graphics library documentation:
  http://mcsp.wartburg.edu/zelle/python/graphics/graphics.pdf

• Colors we can use: http://wiki.tcl.tk/37701
GraphWin class

• **GraphWin(title, width, height)** – constructs a new graphics window (default width and height are both 200)
• **setBackground(color)** – set the background color
• **close()** – closes the window
• **getMouse()** – waits for the user to click, returns the click position as a **Point**
• **checkMouse()** – does not wait for the user to click, returns the click position as a **Point**, or None if no position clicked
Methods for all Graphics Objects

• `setFill(color)` – sets the interior color of an object
• `setOutline(color)` – sets the outline color of an object
• `setWidth(pixels)` – sets the outline width (doesn’t work for `Point`)
• `draw(window)` – draws the object on the given window
• `undraw()` – removes the object from a graphics window
• `move(dx,dy)` – moves the object dx in the x direction and dy in the y direction
• `clone()` – returns a duplicate (new copy) of the object
Point class

• `Point(x,y)` – constructs a new point at the given position
• `getX()` – returns the current x coordinate
• `getY()` – returns the current y coordinate
Line class

- **Line(point1, point2)** – constructs a line from point1 to point2
- **setArrow(string)** – sets the arrowhead of a line ("first", "last", "both", "none")
- **getCenter()** – returns the midpoint of the line
- **getP1(), getP2()** – returns a clone of the corresponding endpoint
Circle class

- **Circle(center, radius)** — constructs a circle at the given position and with the given radius
- **getCenter()** — returns a clone of the center point
- **getRadius()** — returns the radius
- **getP1(), getP2()** — returns a clone of the corresponding corner of the circle’s bounding box (upper left, lower right)
Rectangle class

- **Rectangle(point1, point2)** – constructs a rectangle with opposite corners at the given points (upper left, lower right)
- **getCenter()** – returns the center point
- **getP1(), getP2()** – returns a clone of the corner point
Polygon class

- **Polygon(point1, point2, point3, ...)** – constructs a polygon with the given points as vertices (also accepts a list of points)
- **getPoints()** – returns a list of the points in the polygon
Cat Face Exercise
Step 1: (optional) create a grid

- Window 600 x 600
- Grid lines every 100
- Line example:

```python
# first vertical line
p1 = Point(100, 0)
p2 = Point(100, height)
l = Line(p1, p2)
l.draw(win)
```
Step 2: create a face and eyes

• Create a left eye using a circle
• Clone (copy) the left eye to make the right eye
• Move the right eye over

```python
right_eye = left_eye.clone()
right_eye.move(dx, dy)
right_eye.draw(win)
```
Step 3: create nose, ears, mouth

- Create mouth as a rectangle
- Create nose as a polygon
- Create ears as polygons
- Remove background grid
- Change colors!

http://wiki.tcl.tk/37701