Graphics
Announcements

• Quiz 2 is on Friday
  - Review with ninjas tonight
• Lab 4 due Saturday
• Great job on Lab 3!
Today’s plan

• Briefly go over quiz topics
• Review Monday’s lecture
• More on graphics
• Example graphics programs
Quiz 2

• Topics from quiz 1, especially: arithmetic, string concatenation and repetition, assignment, range function, type conversion functions

• Data types: int, float, string, bool, list

• Comparison operators: ==, !=, <, <=, >, >=

• Logical operators: and, or, not

• The in operator

• Using % to determine parity (even or odd)
Quiz 2

• Indexing and slicing a sequence

• Different kinds of for loops
  - for ch in string:
  - for i in range(len(sequence)):
    ★ indexing or slicing inside such a for loop

• Accumulators: initialize, update, use
Review

- **Objects** are a way of associating multiple pieces of data into a single entity.

- Each object is an *instance* of a **class**. We create an object by calling the **constructor** for its class.

- We can access, modify, or otherwise use an object’s data through its **methods**.
Review

- Constructor syntax:
  - `<class-name>(({param1}, {param2}, ...)`)  
  - `p1 = Point(x_coordinate, y_coordinate)`

- Method syntax:
  - `<object>.<method-name>({param1}, {param2},...)`  
  - `p1.getX()`  
  - `p1.draw(window)`

- Functions, constructors, and methods are all callable, thus they need parentheses even if there are no parameters.
Review

• Strings and lists have methods too, even though they are not created with a constructor.

  - strings: `upper()`, `lower()`

  - lists: `append()`

• Lists, like objects, are **mutable** or changeable through the use of methods.

• Strings, like ints, floats, and bools, are **immutable**, or unchangeable, even when you use string methods.
Graphics

- We use the Zelle graphics library.

- There are multiple ways to do graphics, this is a way.

- Everything starts with the graphics window, an instance of the GraphWin class.
GraphWin constructor

- Parameters: title, width in pixels, height in pixels
- Side effects: opens the window
- Returns: a GraphWin object, with which we can call the GraphWin methods to change what’s in the window

```python
width = 600
goingheight = 400
window = GraphWin("Graphics example", width, height)
```
setBackground method

• Parameters: string containing a color

• Side effects: changes the background color of the window

• Returns: nothing

```python
window = GraphWin("Graphics example", width, height)
window.setBackground("white")
window.getMouse()
```
getMouse method

• Parameters: None

• Side effects: Pauses program until user clicks somewhere in graphics window

• Returns: Point object representing the (x,y) location of click

```python
window = GraphWin("Graphics example", width, height)
window.setBackground("white")
click = window.getMouse()
print("x: %d, y: %d" % (click.getX(), click.getY()))
```
Other GraphWin methods

- `getHeigh()`: Returns height of window in pixels
- `getWidth()`: Returns width of window in pixels
- `getKey()`: Pauses until user presses key, returns string representing key pressed
- `checkMouse()`, `checkKey()`: Like `getMouse()` and `getKey()` except they don’t pause the program. Typically used within a `while` loop.
Objects to draw

• **Point**: constructor needs x and y coordinates

• **Line**: constructor needs **Point** objects for the endpoints

• **Rectangle**: constructor needs **Point** objects for upper left and lower right corners (in that order)

• **Circle**: constructor needs **Point** object for center and radius in pixels

• **Polygon**: constructor needs a list of **Point** objects

• **Text**: constructor needs **Point** object on which to center text and string containing the text
Methods in common

- `draw(window)`: Draws the object in the specified window
- `undraw()`: Removes a drawn object
- `clone()`: Creates an identical copy of the object
- `move(dx, dy)`: Moves object by specified distances
- `setFill(color)`: Changes background color of object
- `setOutline(color)`: Changes color of outline

★ All except `clone()` are called for their side effects
Reference

• [https://www.cs.swarthmore.edu/courses/CS21Labs/s17/docs/graphics.php](https://www.cs.swarthmore.edu/courses/CS21Labs/s17/docs/graphics.php)
Examples
Graphics: things to remember

- Increasing the y-axis moves you down
- Methods with no parameters still need parentheses
- `from graphics import *`
- Use `window.getMouse()` to pause program once everything is drawn in `window`.
- Avoid **hard coding**. All positions, lengths, etc. should be defined in terms of the window height and window width.
Four parts of a program

• Input: `raw_input()`; clicks and keypresses in a graphics window

• Computation/Algorithm: Create objects to draw in the graphics window

• Output: `print()`; draw objects to graphics window

• Repetition: Update the graphics window
Good luck on the quiz!