Objectives

- Introduction to Object-Oriented Programming
- Introduction to APIs
- Problem-solving using APIs

Labs to Linux

- Log in to your **local** account
- Go to your labs directory, where you can see your lab3 directory
- Run the following command, changing as appropriate

```
scp -r lab3 username@hostname:labs/
```

- Example:

```
scp -r lab3 sprenkle@perlman:labs/
```

- Run `hostname` to figure out the name of your machine

Review

- What is the syntax for a `while` loop?
- When should you use a `while` loop?
- Which is more powerful: a `while` loop or a `for` loop?

Programming Paradigm: Imperative

- Most modern programming languages are imperative
- Have data (numbers and strings in variables)
- Perform operations on data using operations, such as + (addition and concatenation)
- Data and operations are separate

- Add to imperative: object-oriented programming

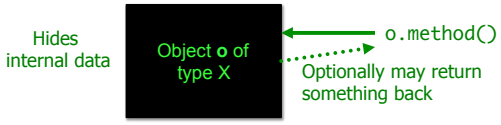
OBJECT-ORIENTED PROGRAMMING

Object-Oriented Programming

- Program is a collection of **objects**
- Objects **combine** data and methods together
- Objects interact by invoking **methods** on other objects
  - Methods perform some operation on object

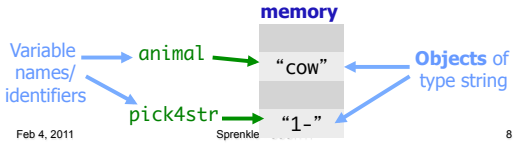
Object-Oriented Programming

- Program is a collection of **objects**
- Objects **combine** data and methods together
- Objects interact by invoking **methods** on other objects
  - Methods perform some operation on object



Object-Oriented Programming

- We've been using objects
  - Just didn't call them objects
- For example: `str` is a data type (or **class**)
  - We created objects of type (*class*) `string`
    - `animal = "cow"`
    - `pick4Str = str(randnum) + "-"`



Example of OO Programming Abstraction

- Think of a TV – It's an ***object***
- What can you do to your TV using one of two ***interfaces***: the remote or the buttons on the TV?

Example of OO Programming Abstraction

- Think of a TV – it's an ***object***
- What can you do to your TV using one of two ***interfaces***: the remote or the buttons on the TV?
  - Turn on/off
  - Change channel
  - Change volume
  - …

  } **methods**
- You don't know ***how*** that operation is being done (i.e., implemented)
  - Just know ***what it does*** and that it ***works***

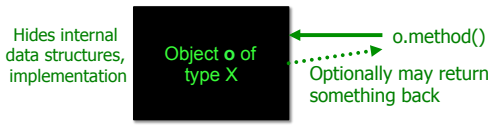
Example of OO Programming Abstraction

- Your TV is an ***object***
- ***Methods*** you can call on your TV:
  - Turn on/off
  - Change channel
  - Change volume
  - …
- TV is a ***class***, a.k.a., a data ***type***
  - Your TV (named "myTV") is an object of type TV
  - You can call the above methods on any object of type TV

Object-Oriented Programming

- Objects combine data *and* methods together

Provides **interface** (*methods*) that users interact with



Use an Application Programming Interface (**API**) to interact with a set of classes.

Class Libraries

- Python provides libraries of classes
  - Defines methods that you can call on objects from those classes
  - `str` class provides a bunch of useful methods
    - More on that later
- Third-party libraries
  - Written by non-Python people
  - Can write programs using these libraries too

Benefits of Object-Oriented Programming

- **Abstraction**
  - Hides details of underlying implementation
  - Easier to change implementation
- Easy reuse of code
- Collects related data/methods together
  - Easier to reason about data
- Less code in main program

Using a Graphics Module/Library

- Allows us to handle graphical input and output
  - Example output: Pictures
  - Example input: Mouse clicks
- Defines a collection of related graphics **classes**
- Not part of a standard Python distribution
  - Need to import from `graphics.py`
- Use the library to help us learn OO programming

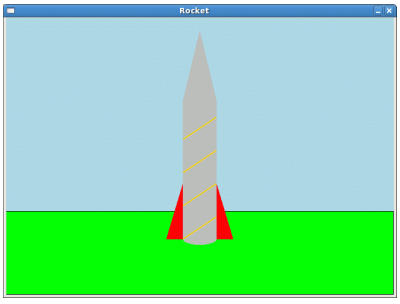
USING A GRAPHICS MODULE

Using a Graphics Module/Library

- Handout lists the various classes
  - **Constructor** is in bold
    - Creates an object of that type
  - For each class, lists *some* of their methods and parameters
  - Drawn objects have some common methods
    - Listed at end of handout
- Known as an **API**
  - **Application Programming Interface**

Example of Output

- From Fall07 class



Using the API: **Constructors**

- To create an object of a certain type/class, use the **constructor** for that type/class
  - Syntax:

```
objName = ClassName([parameters])
```

  - Note:
    - Class names typically begin with capital letter
    - Object names begin with lowercase letter
  - **objname** is known as an **instance** of the class
- Example: To create a `GraphWin` object that's named "`window`"

```
window = GraphWin("My Window",200,200)
```


Using the API: Methods

- To call a **method** on an object,
  - Syntax:

```
objName.methodName([parameters])
```

  - Method names typically begin with lowercase letter
  - Similar to calling *functions*
- Example: To change the background color of a GraphWin object named "window"

```
window.setBackground("blue")
```


Using the API: Methods

- A method sometimes **returns** output, which you may want to save in a variable
  - Class's API should say if method returns output
- Example: if you want to know the width of a `GraphWin` object named `window`

```
width = window.getWidth()
```


What Does This Code Do?

- Use OO terminology previously defined

```
from graphics import *

win = GraphWin("My Circle", 100, 100)
point = Point(50,50)
c = Circle(point, 10)
c.draw(win)
win.getMouse()
```


What Does This Code Do?

- Use OO terminology previously defined

```
from graphics import *

win = GraphWin("My Circle", 100, 100)
point = Point(50,50)
c = Circle(point, 10)
c.draw(win)
win.getMouse()
```

Constructor

`GraphWin` object
Also known as an **instance of** the **`GraphWin` class**

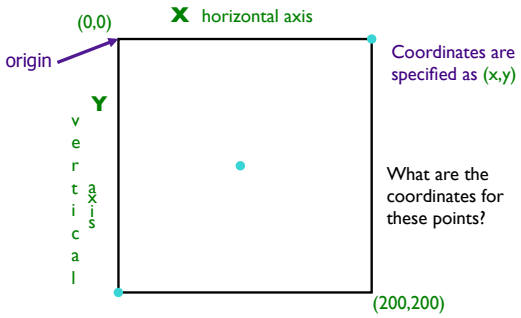
Method called on `GraphWin` object

Note: Class names start with capital letters, Method names start with lowercase letters

Using the Graphics Library

- In general, graphics are drawn on a canvas
  - A canvas is a 2-dimensional grid of pixels
- For our Graphics library, our canvas is a *window*
  - Specifically an **instance of** the **`GraphWin`** class
  - By default, a `GraphWin` object is 200x200 pixels

A GraphWin Object's Canvas



Reading Code

- After this program executes, what does the window look like?

```
from graphics import *

win = GraphWin("My Circle", 100, 100)
c = Circle(Point(50,50), 10)
c.draw(win)
win.getMouse()
```

graphics_test.py

The GraphWin Class

- All parameters to the constructor are optional
- Could call constructor as

Call	Meaning
GraphWin()	Title, width, height to defaults ("Graphics Window", 200, 200)
GraphWin(<title>)	Width, height to defaults
GraphWin(<title>,<width>)	Height to default
GraphWin(<title>, <width>, <height>)	

The GraphWin API

- **Accessor** methods for GraphWin
  - Return some information about the GraphWin
- Example methods:
  - <GraphWinObj>.getWidth()
  - <GraphWinObj>.getHeight()

The GraphWin API

- <GraphWinObj>.setBackground(<color>)
  - Colors are strings, such as "red" or "purple"
    - Can add numbers to end of string for darker colors, e.g., "red2", "red3", "red4"

```
win = GraphWin()
win.setBackground("purple")
```

  - Does *not return* anything to shell
  - Called for its change in **win**'s state, i.e., this method is a **mutator**

Colors

- Strings, such as "blue4"
- Can also create colors using the *function* color_rgb(<red>,<green>,<blue>)
  - Parameters in the range [0,255]
  - Example use:

```
win.setBackground(color_rgb(10,100,100))
```

    - Background is a dark blue/green color
  - Example color codes:
    - http://en.wikipedia.org/wiki/List_of_colors

General Categories of Methods

- Accessor
  - Returns information about the object
  - Example: `getWidth()`
- Mutator
  - Changes the state of the object
    - i.e., changes something about the object
  - Example: `setBackground()`

Using the Graphics Library

- How do we create an instance of a Rectangle?
- Draw the rectangle?
- Shift the instance of the Rectangle class to the **right** 10 pixels
- What are the x- and y- coordinates of the upper-left corner of the Rectangle now?

OO Terminology Summary

Term	Definition	Examples
Class	A data type. Defines the data and operations for members of the class	string, TV, GraphWin
Object	An *instance* of a specific class	animal, myTV, window
Method	Operations you can call on an object	setBackground(<color>), getWidth()
Constructor	Special method to create an object of a certain type/ class	GraphWin(), str(1234)

Problem: Draw a Full-Canvas Tic-Tac-Toe Board

- Using the Graphics API
- Make lines purple and line width 3
  - Keep it general, regardless of GraphWin width, height

Looking Ahead

- Broader Issue
  - Read DARPA Urban Challenge for Monday
  - Email me with your interest score
    - Or add to your Sakai summary
- Midterm next Friday
  - Exam prep document available on course web site