

Objectives

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

Feb 4, 2007

Sprengle - CS111

1

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

Feb 4, 2007

Sprengle - CS111

2

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

Feb 4, 2007

Sprengle - CS111

3

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



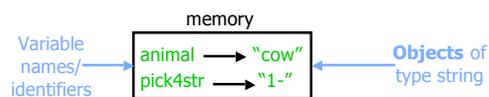
Feb 4, 2007

Sprengle - CS111

4

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) + "-"



Feb 4, 2007

Sprengle - CS111

5

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 keys on the TV?

Feb 4, 2007

Sprengle - CS111

6

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 keys 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

Feb 4, 2007

Sprenkle - CS111

7

Example of OO Programming Abstraction

- Think of a TV -- it's 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

Feb 4, 2007

Sprenkle - CS111

8

Object-Oriented Programming

- Objects **combine** data and methods together

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



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

Feb 4, 2007

Sprenkle - CS111

9

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 next week
- Third-party libraries
 - Not written by Python folks
 - Can write programs using these libraries too

Feb 4, 2007

Sprenkle - CS111

10

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

Feb 4, 2007

Sprenkle - CS111

11

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

Feb 4, 2007

Sprenkle - CS111

12

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**

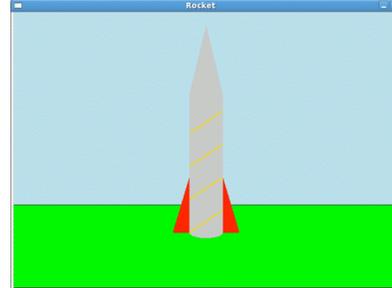
Feb 4, 2007

Sprenkle - CS111

13

Example of Output

- From last semester's class



Feb 4, 2007

Sprenkle - CS111

14

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 that class names typically begin with a capital letter and object names begin with a 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)`

Feb 4, 2007

Sprenkle - CS111

15

Using the API: Methods

- To call a method on an object,
 - Syntax:
 - `objname.methodname([parameters])`
 - Note that method names typically begin with a lowercase letter
 - Similar to calling *functions*
- Example: To change the background color of a GraphWin object named "window"
`window.setBackground("blue")`

Feb 4, 2007

Sprenkle - CS111

16

Using the API: Methods

- A method sometimes "**returns** output", which you may want to save in a variable
 - The class's API should say if the method returns output
- Example: if you want to know the width of a GraphWin object named "window"
 - `width = window.getWidth()`

Feb 4, 2007

Sprenkle - CS111

17

What Does This Code Do?

- Use OO terminology previously defined

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

Feb 4, 2007

Sprenkle - CS111

18

What Does This Code Do?

- Use OO terminology previously defined

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

GraphWin object → win = GraphWin("My Circle", 100, 100)
Also known as an **instance of the GraphWin class**

Constructor → GraphWin("My Circle", 100, 100)
Method called on GraphWin object → win.getMouse()

Feb 4, 2007

Sprengle - CS111

19

Using the Graphics Library

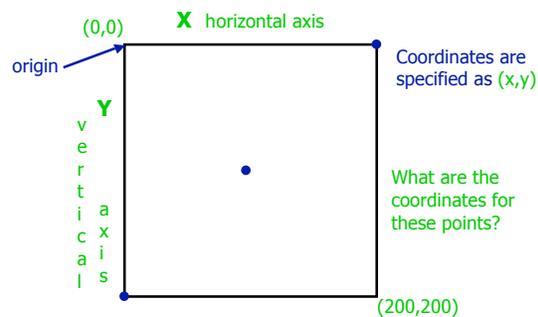
- In general, graphics are drawn on a canvas
 - A canvas is a 2-dimensional grid of pixels
- For our Graphics library, our canvas is a *window*
 - Specifically an **instance of the GraphWin class**
 - By default, a GraphWin object is 200x200 pixels

Feb 4, 2007

Sprengle - CS111

20

A GraphWin Object's Canvas



Feb 4, 2007

Sprengle - CS111

21

Snippet of 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()
```

Feb 4, 2007

Sprengle - CS111

graphics_test.py

22

The GraphWin Class

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

Feb 4, 2007

Sprengle - CS111

23

The GraphWin API

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

Feb 4, 2007

Sprengle - CS111

24

The GraphWin API

- `<GraphWin>.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**

Feb 4, 2007

Sprenkle - CS111

25

## 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](http://en.wikipedia.org/wiki/List_of_colors)

Feb 4, 2007

Sprenkle - CS111

26

## 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()`

Feb 4, 2007

Sprenkle - CS111

27

## 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?

Feb 4, 2007

Sprenkle - CS111

[rectangle.py](#)

28

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

- Using the Graphics API
- Make lines purple and line width 3

Feb 4, 2007

Sprenkle - CS111 [tictactoe.py](#)

29

## Modification to Tic-Tac-Toe

- **clone** a vertical line and horizontal line and shift appropriately
- Why clone?
  - Maintain the same properties (color, line-width, length)
  - Simplifies code

Feb 4, 2007

Sprenkle - CS111

[tictactoe2.py](#)

30

## 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)              |

Feb 4, 2007

Sprenkle - CS111

31

## Midterm Prep

- Cumulative up to today
  - We keep using the ideas from the first day of class
  - Basic Linux commands used during every lab
- Similar problems as in handouts, class discussion, labs
- Read code and explain what it does
  - What it displays as output
- Sections: Very Short Answer, Short Answer, Write Code
- Online prep document

Feb 4, 2007

Sprenkle - CS111

32

## Grading Overview

- Labs: 40%
- 2 Exams: 30%
- Final: 20%
- Broader Issues: 5%
- Participation: 5%

Feb 4, 2007

Sprenkle - CS111

33