

## Objectives

- Introduction to Object-Oriented Programming
- Introduction to APIs
- Problem-solving using APIs
- Broader Issue: Google Search

Jan 29, 2016

Sprengle - CSCI111

1

## Review

- What are benefits of functions?
- How do we call functions?
- What do we get access to functions that are in a module?
  - How does using the imported functions change with each type of import statement?

Jan 29, 2016

Sprengle - CSCI111

2

## Review: Benefits of Functions

- Reuse, simplify code
- Functions written by others
  - Well-written, efficiency
  - Abstraction: it works, that's all that matters!
- Greatly increase code you can write by leveraging others' code

Jan 29, 2016

Sprengle - CSCI111

3

## 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**

Jan 29, 2016

Sprengle - CSCI111

4

Super Power: Psychokinesis

## OBJECT-ORIENTED PROGRAMMING

Jan 29, 2016

Sprengle - CSCI111

5

## 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

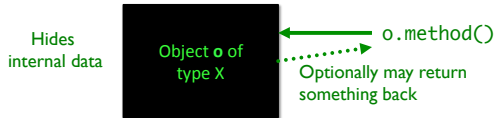
Jan 29, 2016

Sprengle - CSCI111

6

## 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



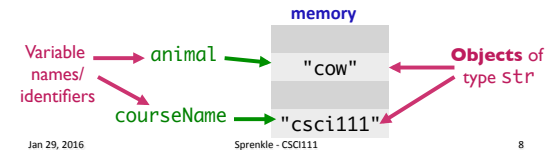
Jan 29, 2016

Sprengle - CSCI111

7

## Object-Oriented Programming

- We've been using objects
  - Just didn't call them objects
- For example: **str** is a type (or **class**)
  - We created objects of type (class) **string**
    - animal = "cow"
    - courseName = "csci111"



Jan 29, 2016

Sprengle - CSCI111

8

## 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?

Jan 29, 2016

Sprengle - CSCI111

9

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

Jan 29, 2016

Sprengle - CSCI111

10

## 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 (identified by myTV) is an object of type TV
  - You can call the above methods on any object of type TV

Jan 29, 2016

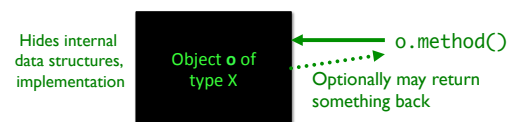
Sprengle - CSCI111

11

## 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.

Jan 29, 2016

Sprengle - CSCI111

12

## 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

Jan 29, 2016

Sprengle - CSC1111

13

## 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

Jan 29, 2016

Sprengle - CSC1111

14

## 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

Jan 29, 2016

Sprengle - CSC1111

15

## USING A GRAPHICS MODULE

Jan 29, 2016

Sprengle - CSC1111

16

## Using a Graphics Module/Library

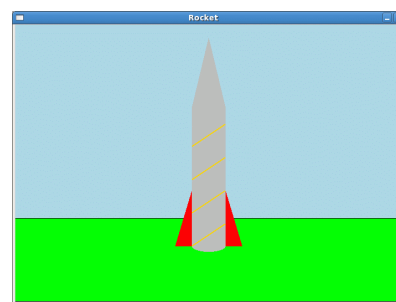
- Handout lists the various classes
  - **Constructor** is in bold
    - Creates an object of that type
  - For each class, lists *some* of their methods and parameters
  - Drawn objects have some common methods
    - Listed at end of handout
- Known as an **API**
  - **Application Programming Interface**

Jan 29, 2016

Sprengle - CSC1111

17

## Example of Output



Jan 29, 2016

Sprengle - CSC1111

18

## 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 identified by `window`

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

Jan 29, 2016

Sprengle - CSCI111

19

## 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")
```

Jan 29, 2016

Sprengle - CSCI111

20

## 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`

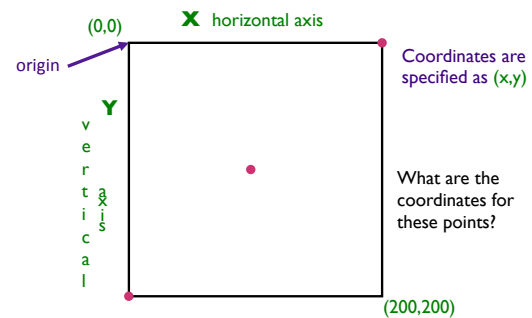
```
width = window.getWidth()
```

Jan 29, 2016

Sprengle - CSCI111

21

## A GraphWin Object's Canvas

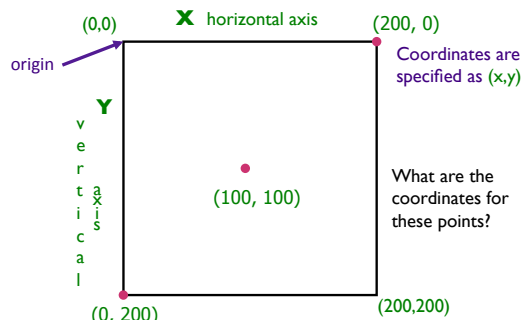


Jan 29, 2016

Sprengle - CSCI111

22

## A GraphWin Object's Canvas



Jan 29, 2016

Sprengle - CSCI111

23

## The GraphWin Class

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

Call	Meaning
<code>GraphWin()</code>	Title, width, height to defaults ("Graphics Window", 200, 200)
<code>GraphWin(&lt;title&gt;)</code>	Width, height to defaults
<code>GraphWin(&lt;title&gt;, &lt;width&gt;)</code>	Height to default
<code>GraphWin(&lt;title&gt;, &lt;width&gt;, &lt;height&gt;)</code>	

Jan 29, 2016

Sprengle - CSCI111

24

## The GraphWin API

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

Jan 29, 2016

Sprenkle - CSCI111

25

## 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 change in **win**'s state, i.e., this method is a **mutator**

Jan 29, 2016

Sprenkle - CSCI111

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()

Jan 29, 2016

Sprenkle - CSCI111

27

## 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()
```

graphics\_test.py

Jan 29, 2016

Sprenkle - CSCI111

28

## 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()
```

GraphWin object  
Also known as an instance of the GraphWin class

Constructor

Method called on GraphWin object

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

Jan 29, 2016

## 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

Jan 29, 2016

Sprenkle - CSCI111

30

## 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:
 

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

    - 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)

Jan 29, 2016

Sprengle - CSC111

31

## 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?

Jan 29, 2016

Sprengle - CSC111

`rectangle.py`

32

## OO Terminology Summary

Term	Definition	Examples
Class	A data type. Defines the data and operations for members of the class	str, TV, GraphWin
Object	An <i>instance</i> 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)

Jan 29, 2016

Sprengle - CSC111

33

## Broader Issue

Emily Honor Max Rachel R	Alicia Eric George Rachel B	Alice Allie Andrew Clark	Austin F Chris Josie Margaret Stuart	Abdur Madhav Taylor Viktor
Austin J Gunnar Holly JT	Collin Corson Ethiopia Ryan			

Jan 29, 2016

Sprengle - CSC111

34

## Google Search

- Why is Google search a “broader issue”?
- How do you measure “improved search”?
- Did Google do the right thing with the Santorum results?
- What kind of power do search engines have?
- Will you use Google differently, now that you know how it works (kind of)?

Jan 29, 2016

Sprengle - CSC111

35

## Looking Ahead

- Lab – Tuesday
- Exam - Friday

Jan 29, 2016

Sprengle - CSC111

36