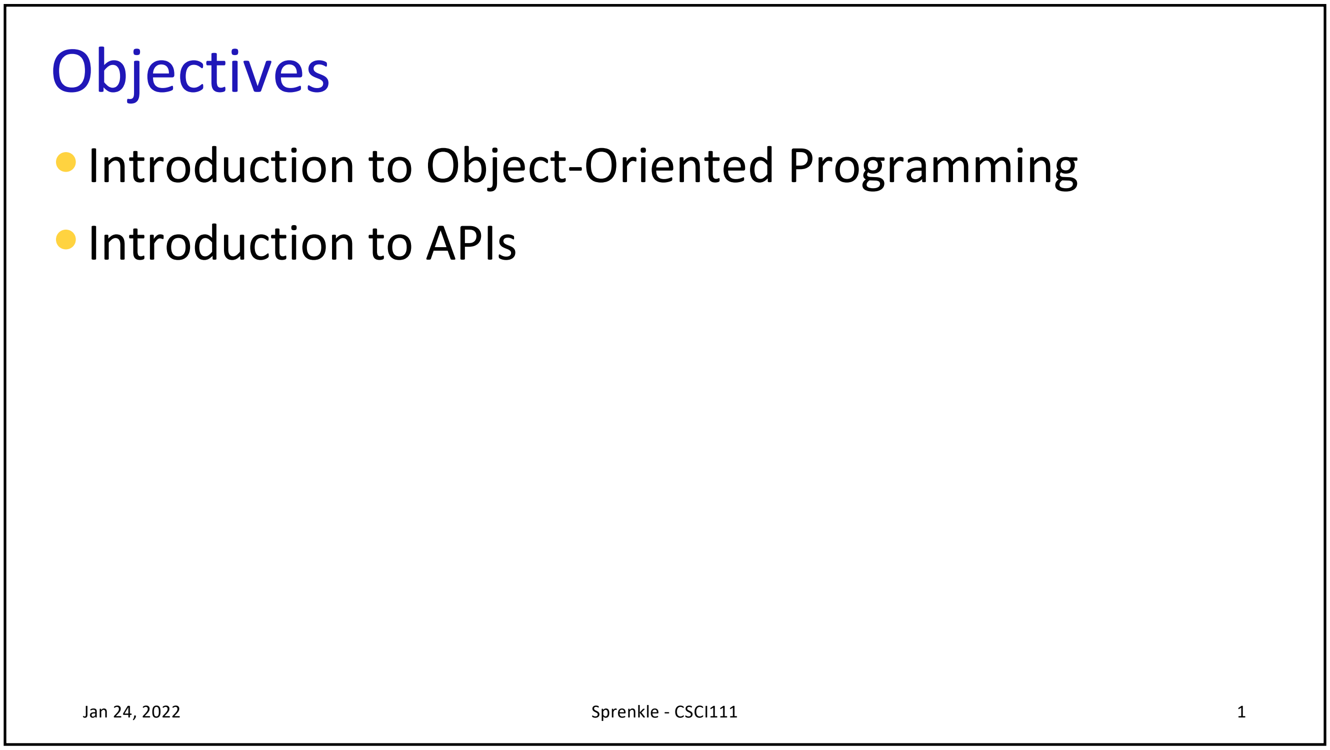
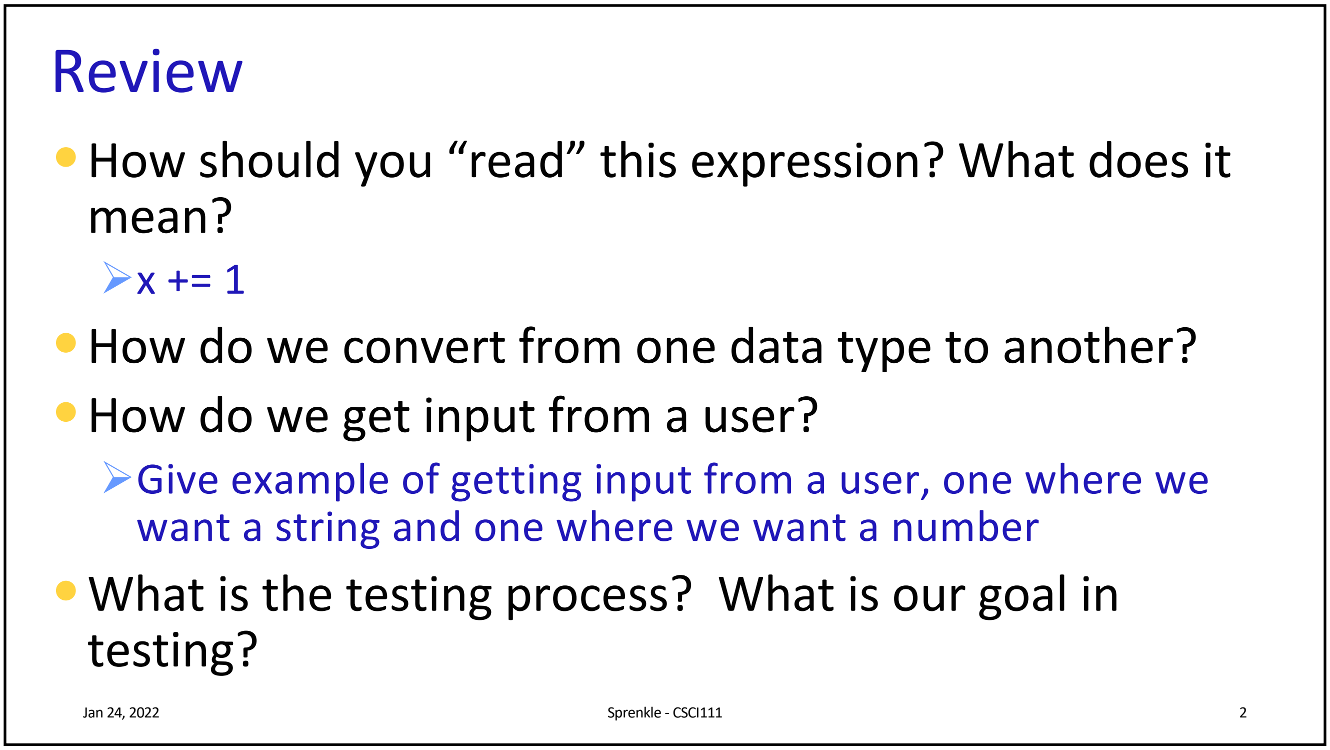


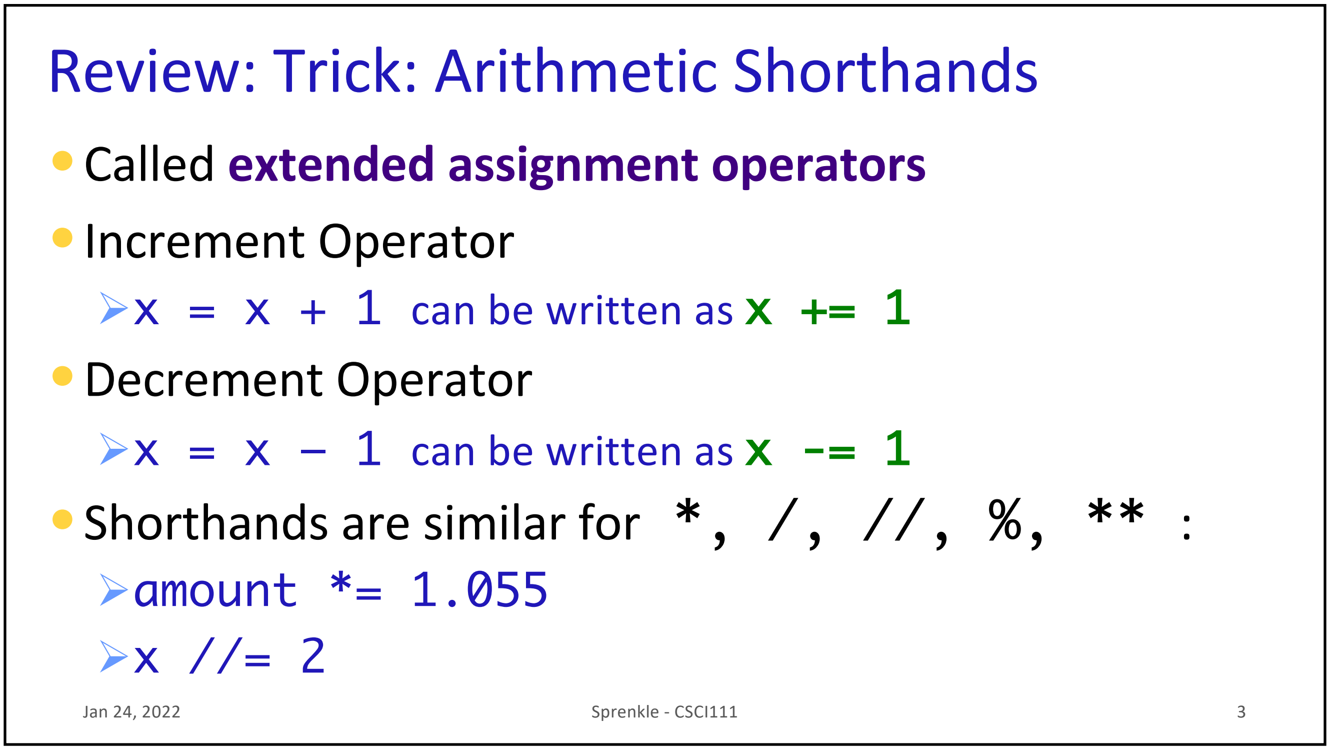
Objectives

- Introduction to Object-Oriented Programming
- Introduction to APIs



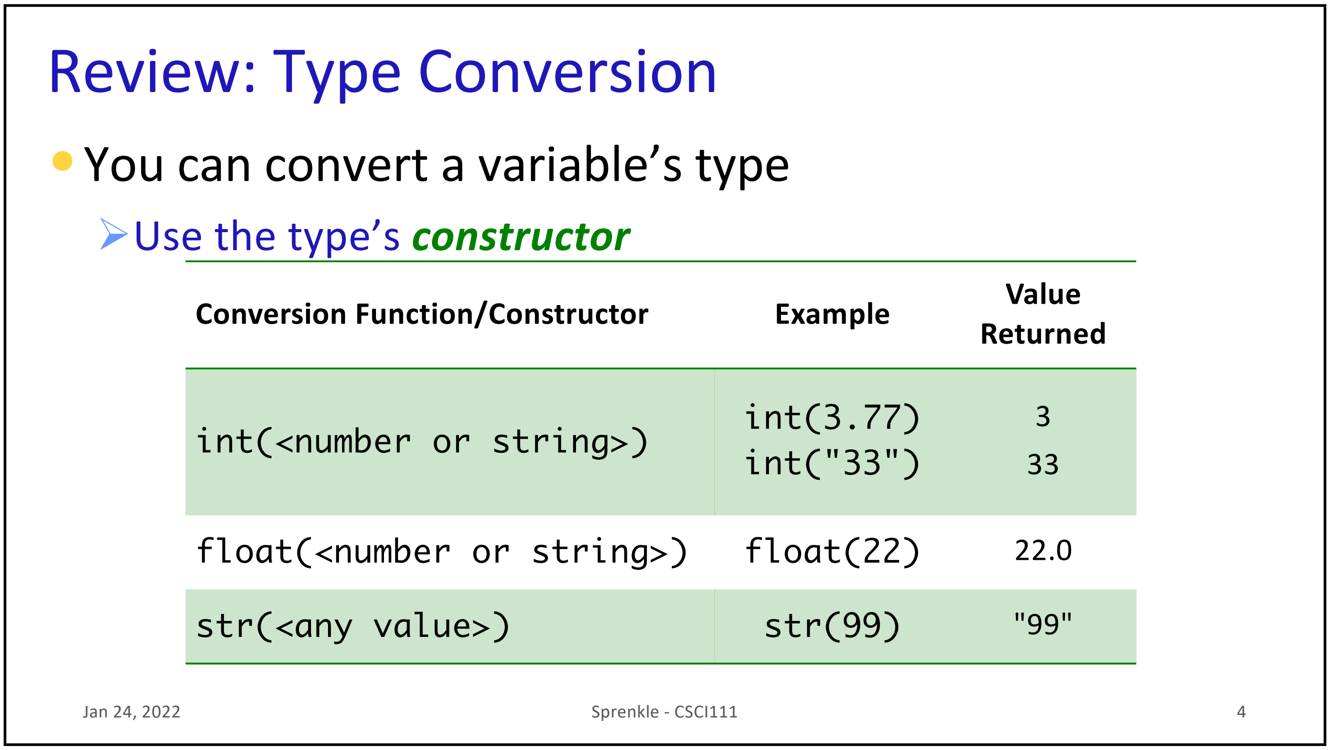
Review

- How should you “read” this expression? What does it mean?
  - x += 1
- How do we convert from one data type to another?
- How do we get input from a user?
  - Give example of getting input from a user, one where we want a string and one where we want a number
- What is the testing process? What is our goal in testing?



Review: Trick: Arithmetic Shorthands

- Called **extended assignment operators**
- Increment Operator
  - `x = x + 1` can be written as **`x += 1`**
- Decrement Operator
  - `x = x – 1` can be written as **`x -= 1`**
- Shorthands are similar for `*, /, //, %, ** :`
  - `amount *= 1.055`
  - `x //= 2`



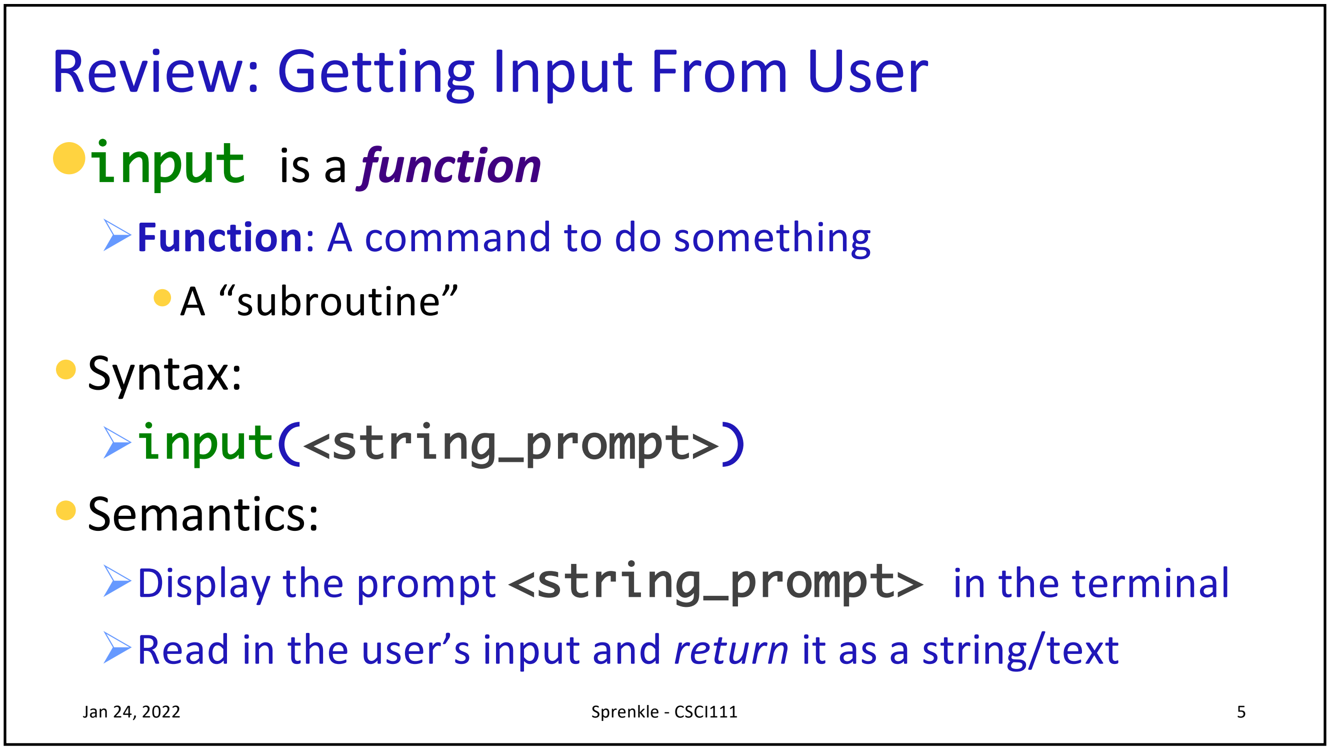
Review: Type Conversion

- You can convert a variable’s type
  - Use the type’s ***constructor***

Conversion Function/Constructor	Example	Value Returned

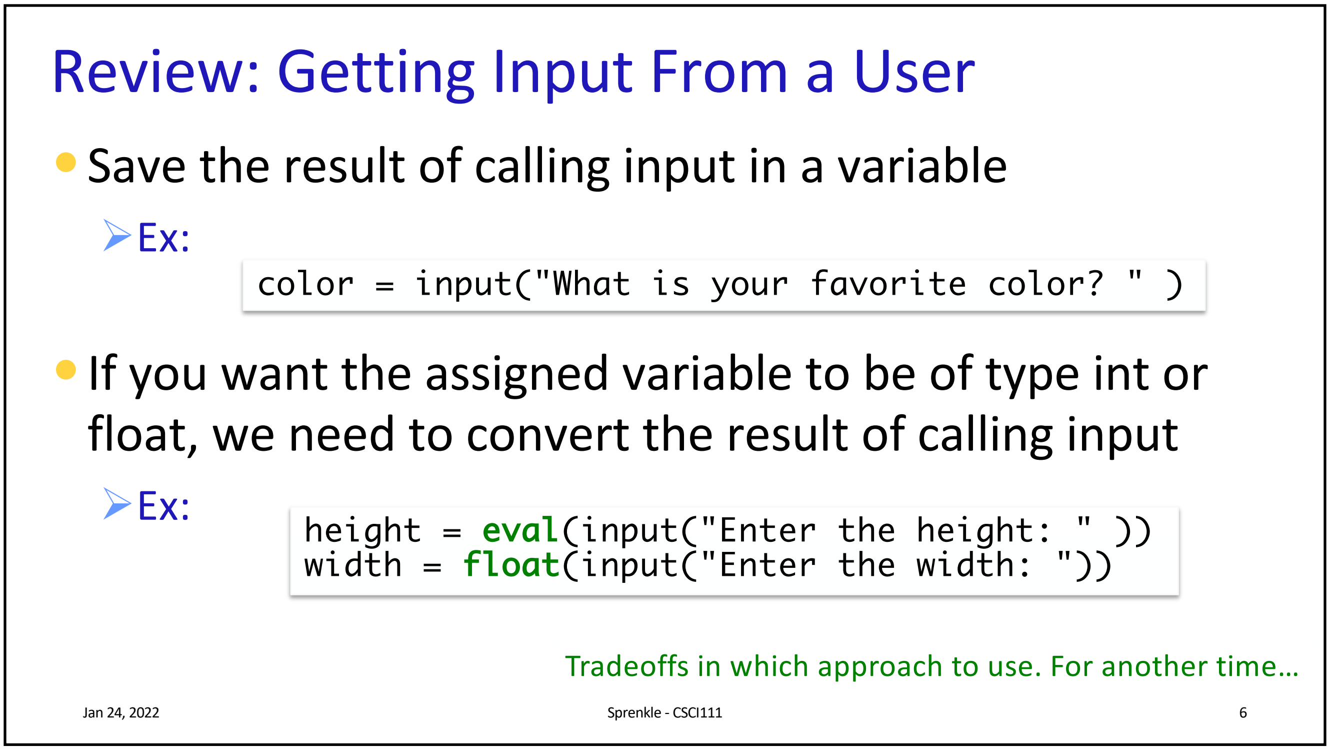
`int(<number or string>)`	`int(3.77)`
	`int("33")`
		3
		33

`float(<number or string>)`	`float(22)`	22.0
`str(<any value>)`	`str(99)`	"99"



Review: Getting Input From User

- `input` is a ***function***
  - **Function**: A command to do something
    - A “subroutine”
- Syntax:
  - `input(<string_prompt>)`
- Semantics:
  - Display the prompt `<string_prompt>` in the terminal
  - Read in the user’s input and *return* it as a string/text



Review: Getting Input From a User

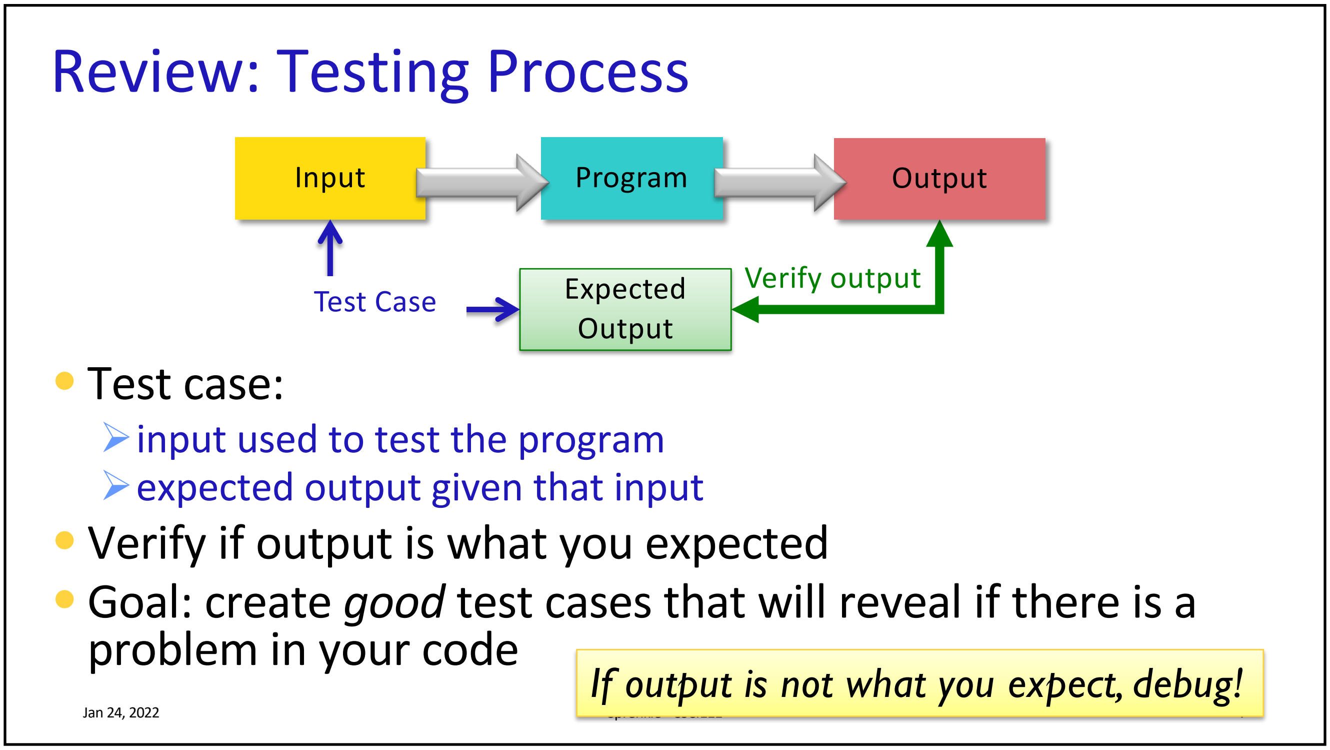
- Save the result of calling input in a variable
  - Ex:

```
color = input("What is your favorite color? " )
```

- If you want the assigned variable to be of type int or float, we need to convert the result of calling input
  - Ex:

```
height = eval(input("Enter the height: " ))
width = float(input("Enter the width: "))
```

Tradeoffs in which approach to use. For another time…



Review: Testing Process

Input

Program

Output

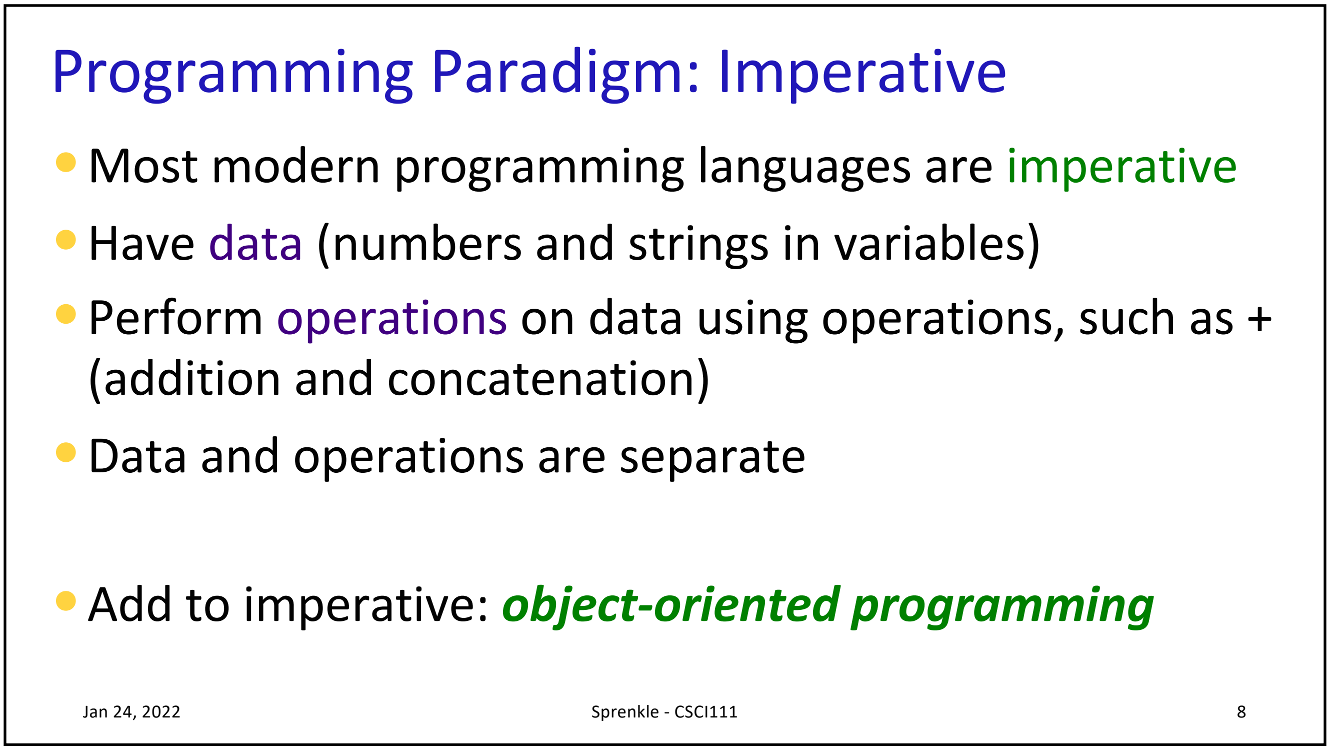
Test Case

Expected Output

Verify output

- Test case:
  - input used to test the program
  - expected output given that input
- Verify if output is what you expected
- Goal: create *good* test cases that will reveal if there is a problem in your code

If output is not what you expect, debug!



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***

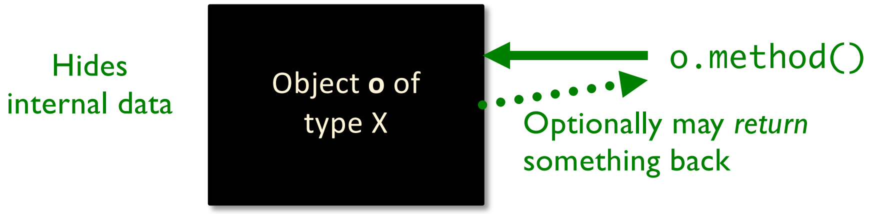
OBJECT-ORIENTED PROGRAMMING

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

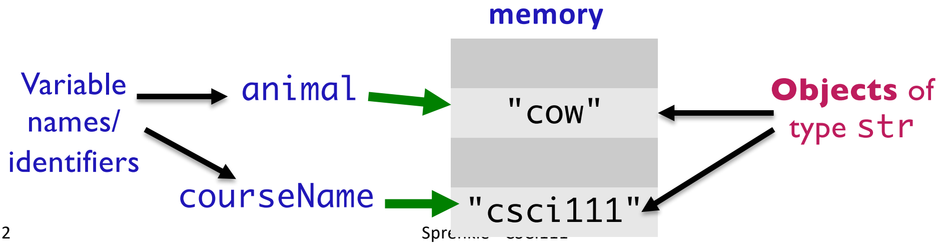
Object-Oriented Programming

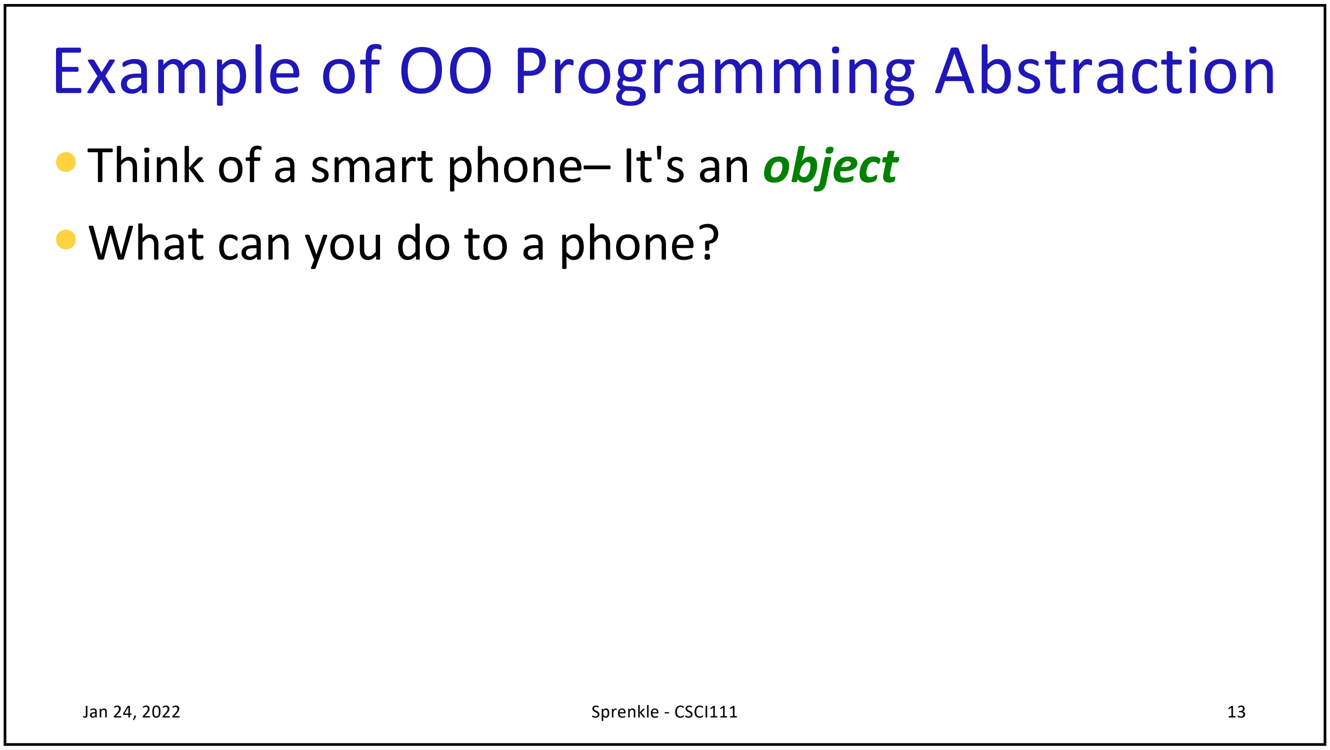
- Program is a collection of ***objects***
- Objects **combine** data and methods together
- Objects interact by invoking ***methods*** on other objects
  - Methods perform some operation on object



Object-Oriented Programming

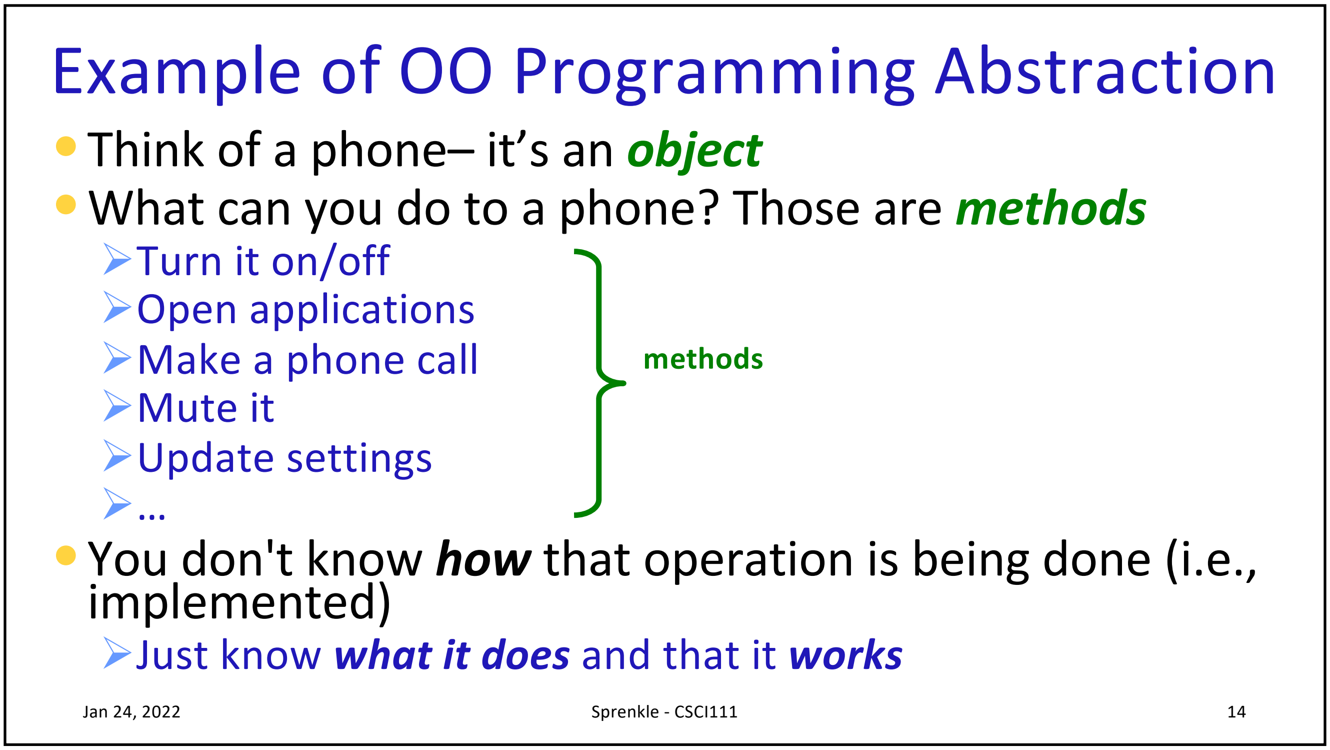
- We've been using objects--just didn't call them objects
- For example: `str` is a data type (or **class**)
  - We created objects of type (*class*) `string`
    - `animal = "cow"`
    - `coursename = "csci111"`





Example of OO Programming Abstraction

- Think of a smart phone– It's an ***object***
- What can you do to a phone?



Example of OO Programming Abstraction

- Think of a phone– it’s an ***object***
- What can you do to a phone? Those are ***methods***
  - Turn it on/off
  - Open applications
  - Make a phone call
  - Mute it
  - Update settings
  - …

methods

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

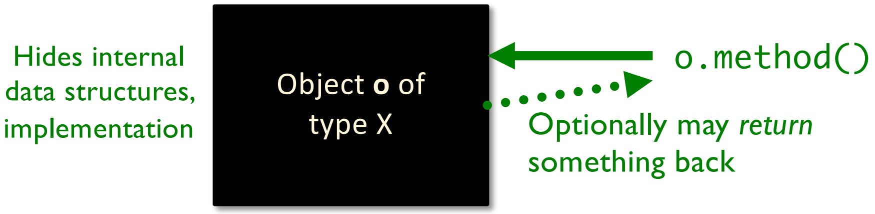
Example of OO Programming Abstraction

- A smart phone is an ***object***
- ***Methods*** you can call on your smart phone:
  - Turn it on/off
  - Open applications
  - Make a phone call
  - Mute it
  - Update settings
  - …
- `SmartPhone` is a ***class***, a.k.a., a data ***type***
  - My smart phone (identified by `myPhone`) is an object of type `SmartPhone`
  - Call the above methods on any object of type `SmartPhone`

Object-Oriented Programming

- Objects combine data *and* methods together

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



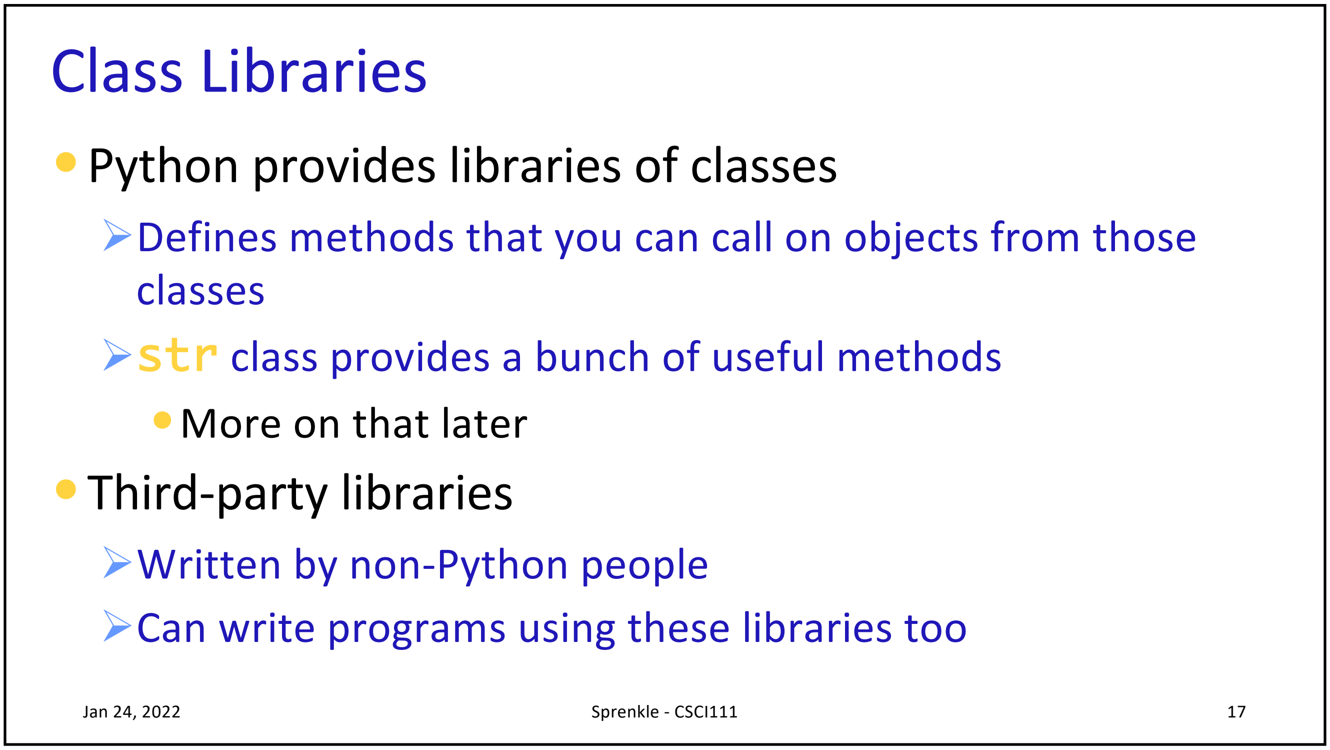
Hides internal data structures, implementation

Object **o** of type X

`o.method()`

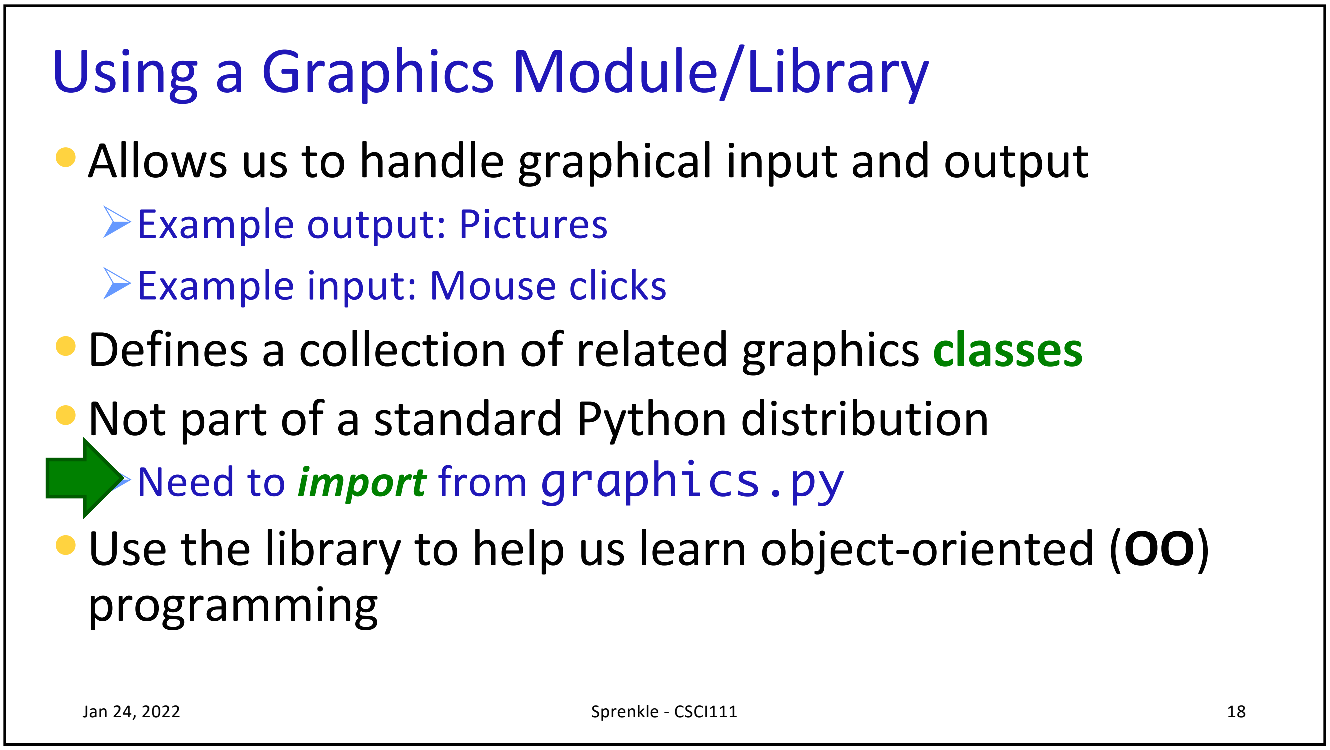
Optionally may *return* something back

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



Using a Graphics Module/Library

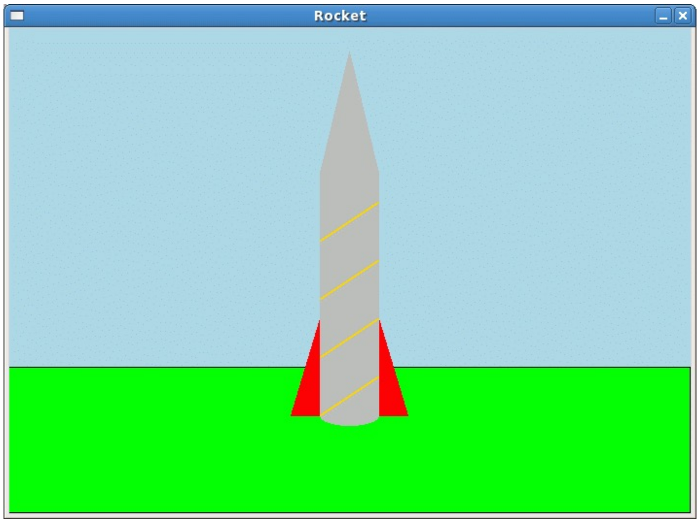
- Allows us to handle graphical input and output
  - Example output: Pictures
  - Example input: Mouse clicks
- Defines a collection of related graphics **classes**
- Not part of a standard Python distribution
  - Need to ***import*** from `graphics.py`
- Use the library to help us learn object-oriented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

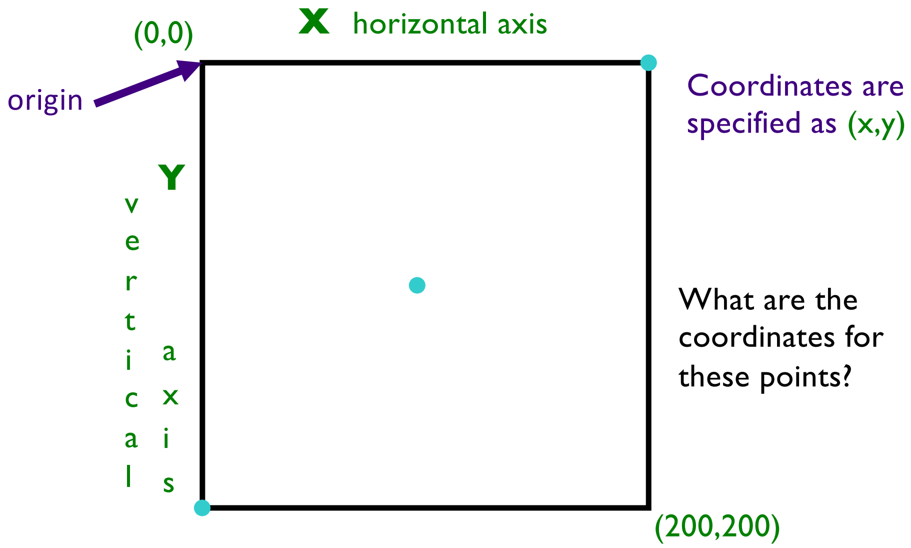
Example of Output



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



(0,0)

X horizontal axis

origin

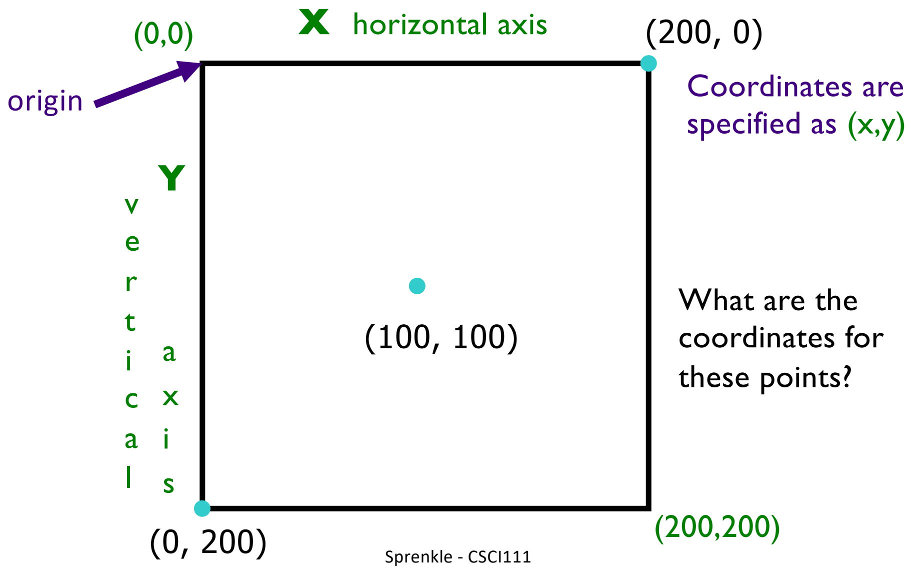
Coordinates are specified as (x,y)

Y vertical axis

What are the coordinates for these points?

(200,200)

A GraphWin Object's Canvas



(0,0)

X horizontal axis

(200, 0)

origin

Coordinates are specified as (x,y)

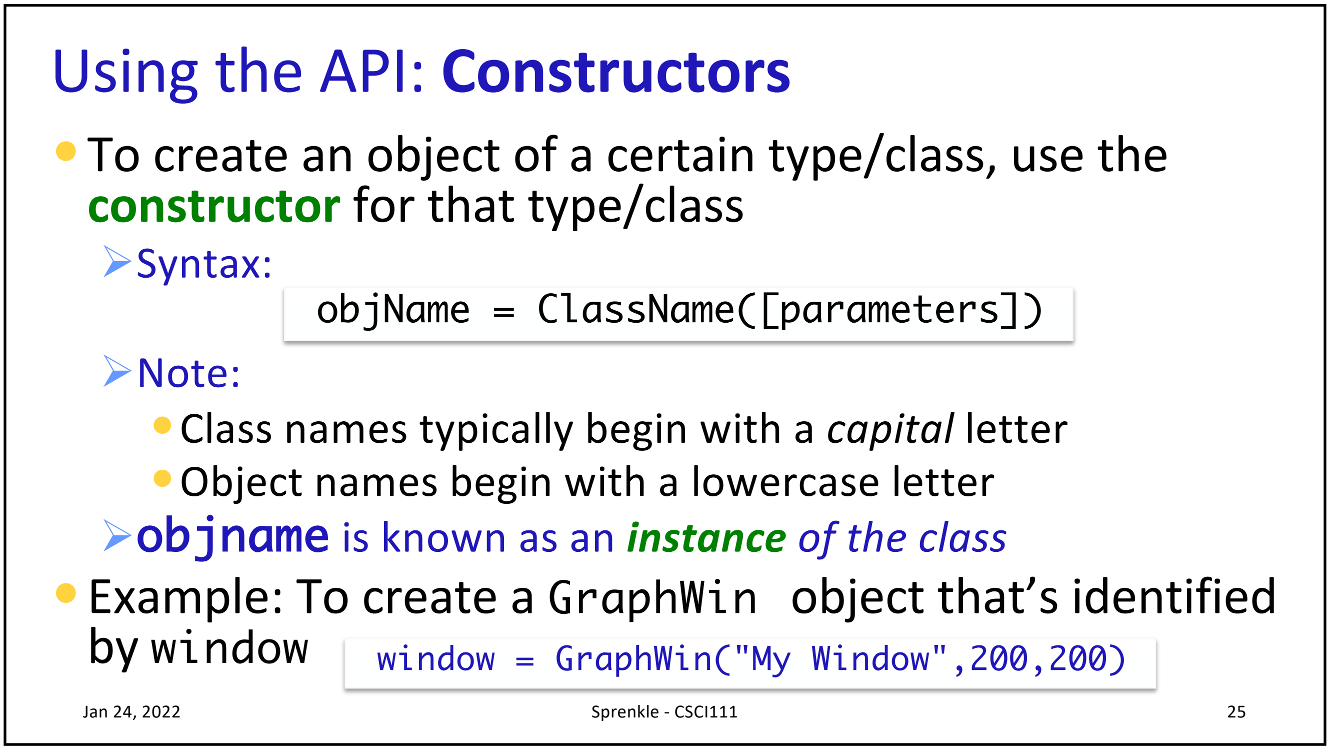
Y vertical axis

(100, 100)

What are the coordinates for these points?

(0, 200)

(200,200)



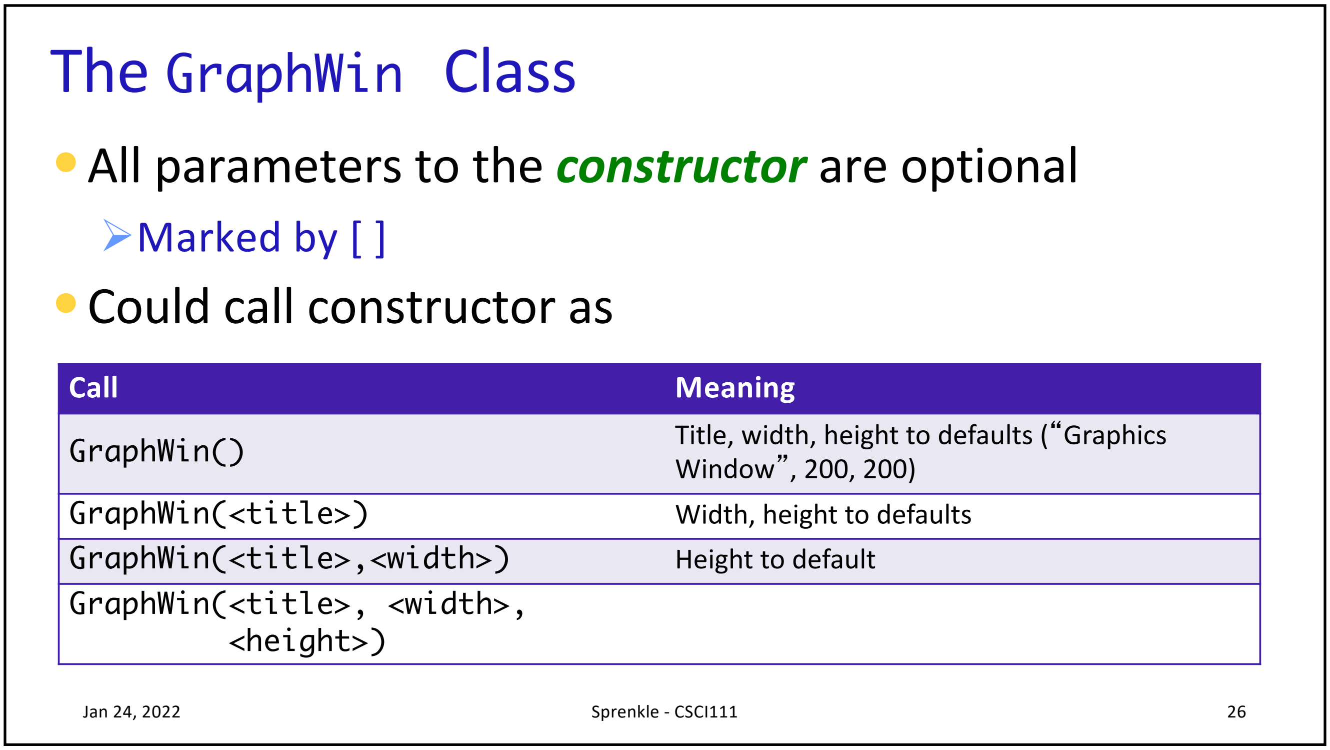
Using the API: **Constructors**

- 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 a *capital* letter
    - Object names begin with a 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)
```



The `GraphWin` Class

- All parameters to the ***constructor*** are optional
  - Marked by [ ]
- 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>)`	

Using the API: Methods

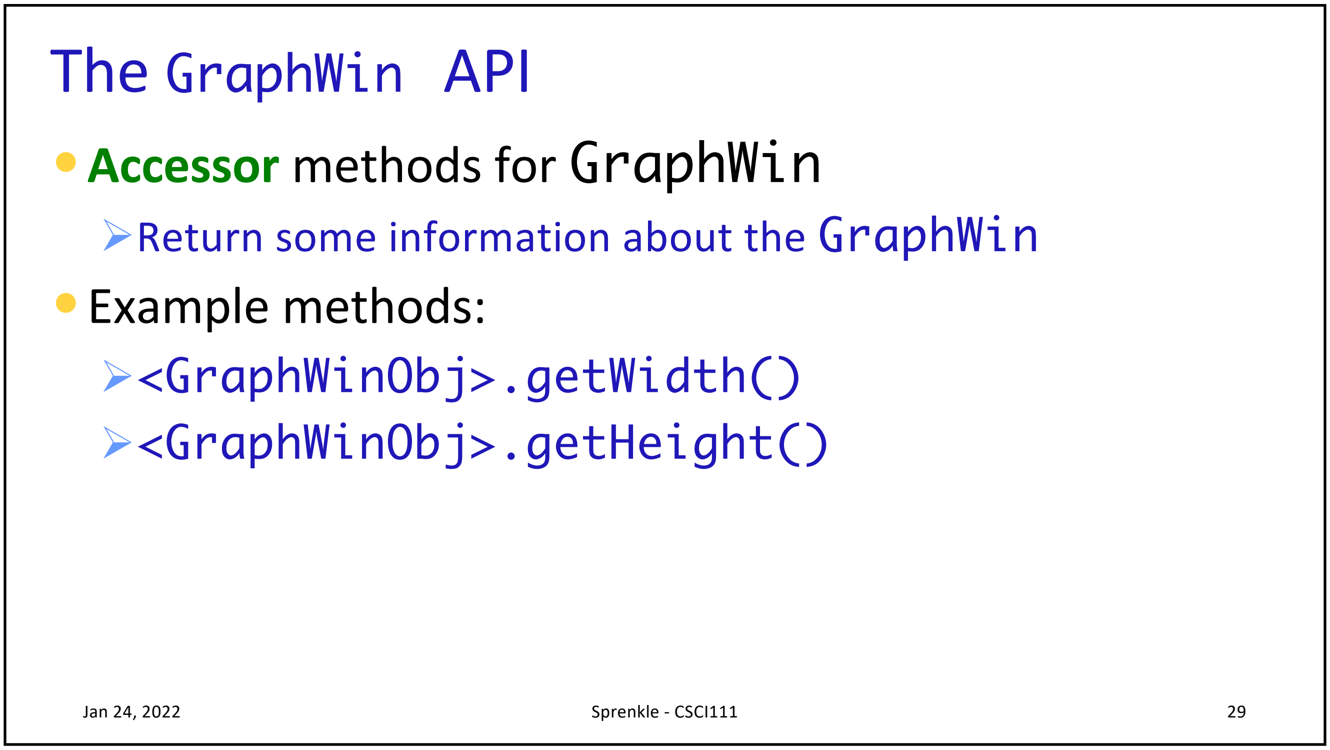
- To call a ***method*** on an object,
  - Syntax: `objName.methodName([parameters])`
  - Similar to calling *functions*
- Method names typically begin with lowercase letter
- Example: To change the background color of a `GraphWin` object named `window`

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

Using the API: Methods

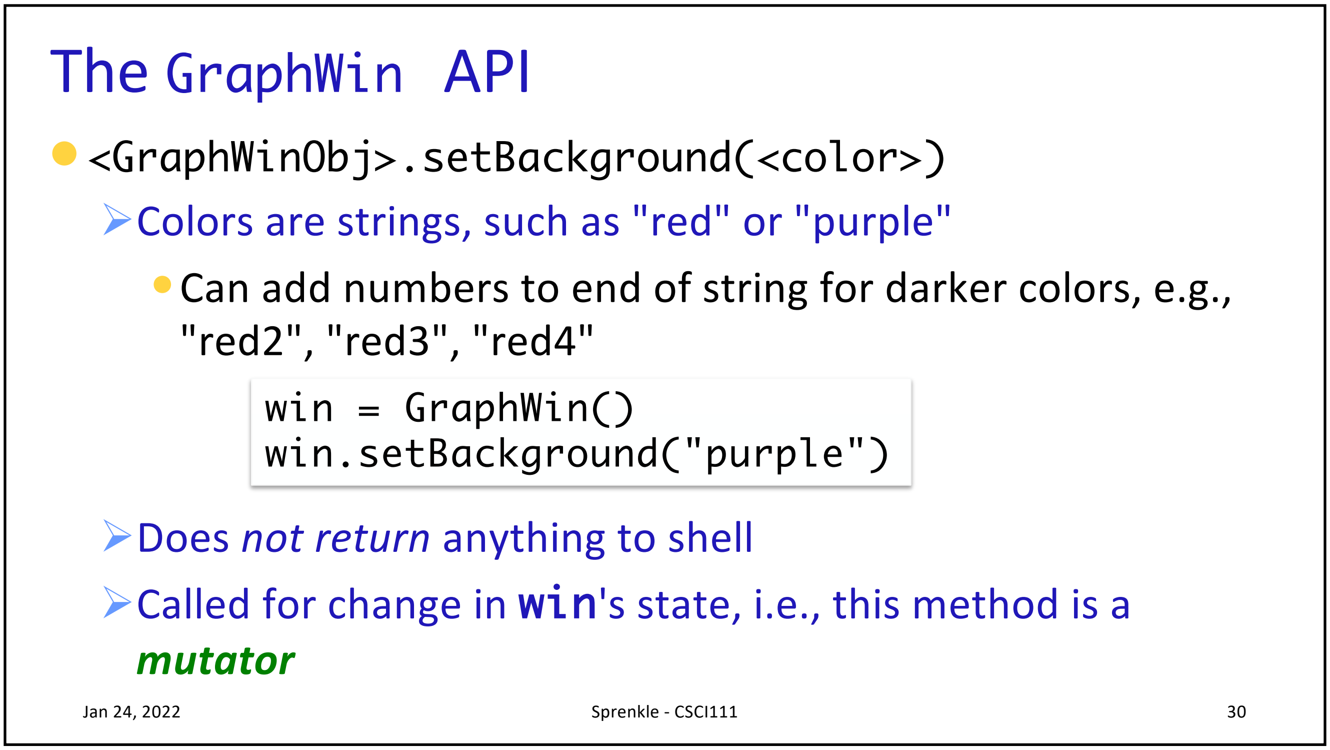
- A method sometimes ***returns*** output, which you may want to save in a variable
  - Class’s API should say if method returns output
  - Referred to as an ***accessor* method**
- Example: if you want to know the *width* of a `GraphWin` object named `window`

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



The GraphWin API

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

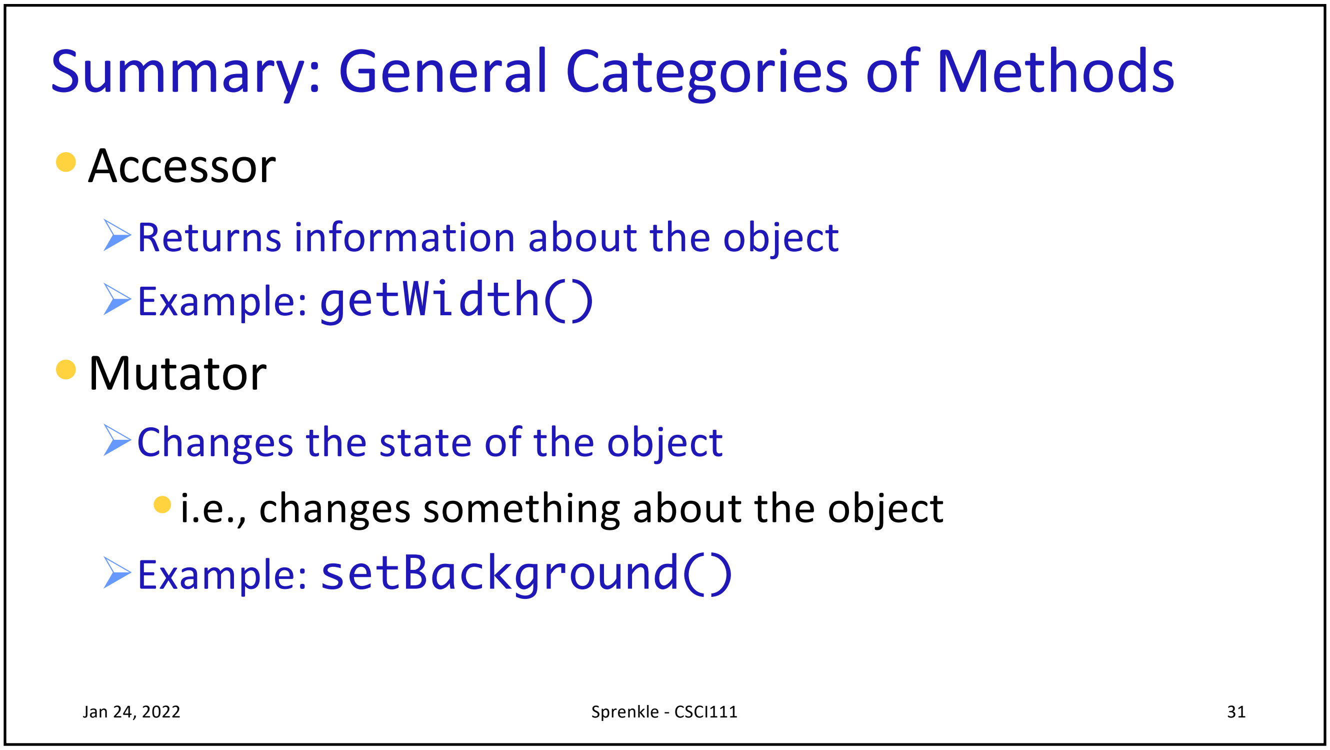


The GraphWin API

- <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"

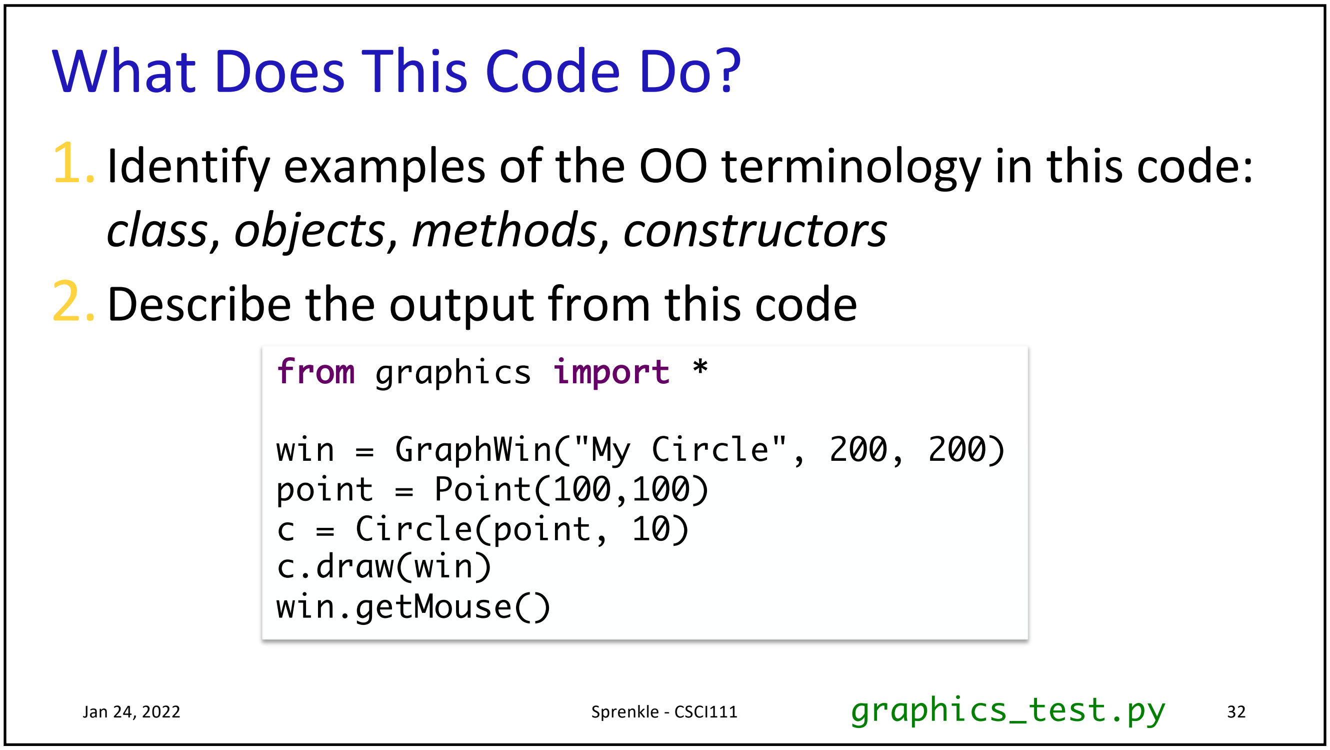
```
win = GraphWin()
win.setBackground("purple")
```

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



Summary: 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()`



What Does This Code Do?

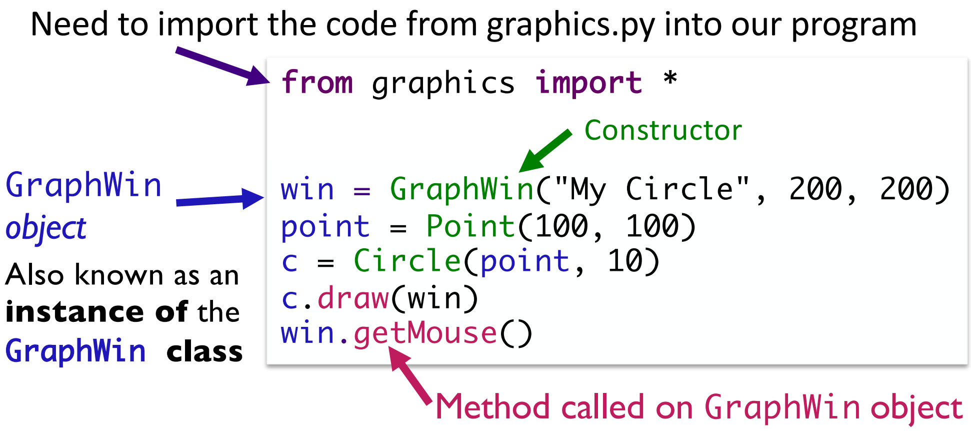
1. Identify examples of the OO terminology in this code: *class, objects, methods, constructors*
2. Describe the output from this code

```
from graphics import *

win = GraphWin("My Circle", 200, 200)
point = Point(100,100)
c = Circle(point, 10)
c.draw(win)
win.getMouse()
```

graphics_test.py

What Does This Code Do?



Need to import the code from graphics.py into our program

```
from graphics import *

win = GraphWin("My Circle", 200, 200)
point = Point(100, 100)
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

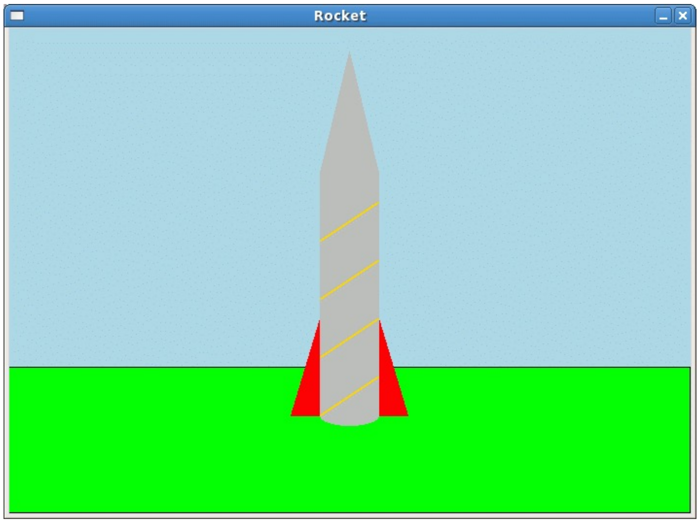
Typical OOP Programming Process:

1. Create an instance of an class
2. Call methods on that object

Benefits of Object-Oriented Programming

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

What objects make up this scene?

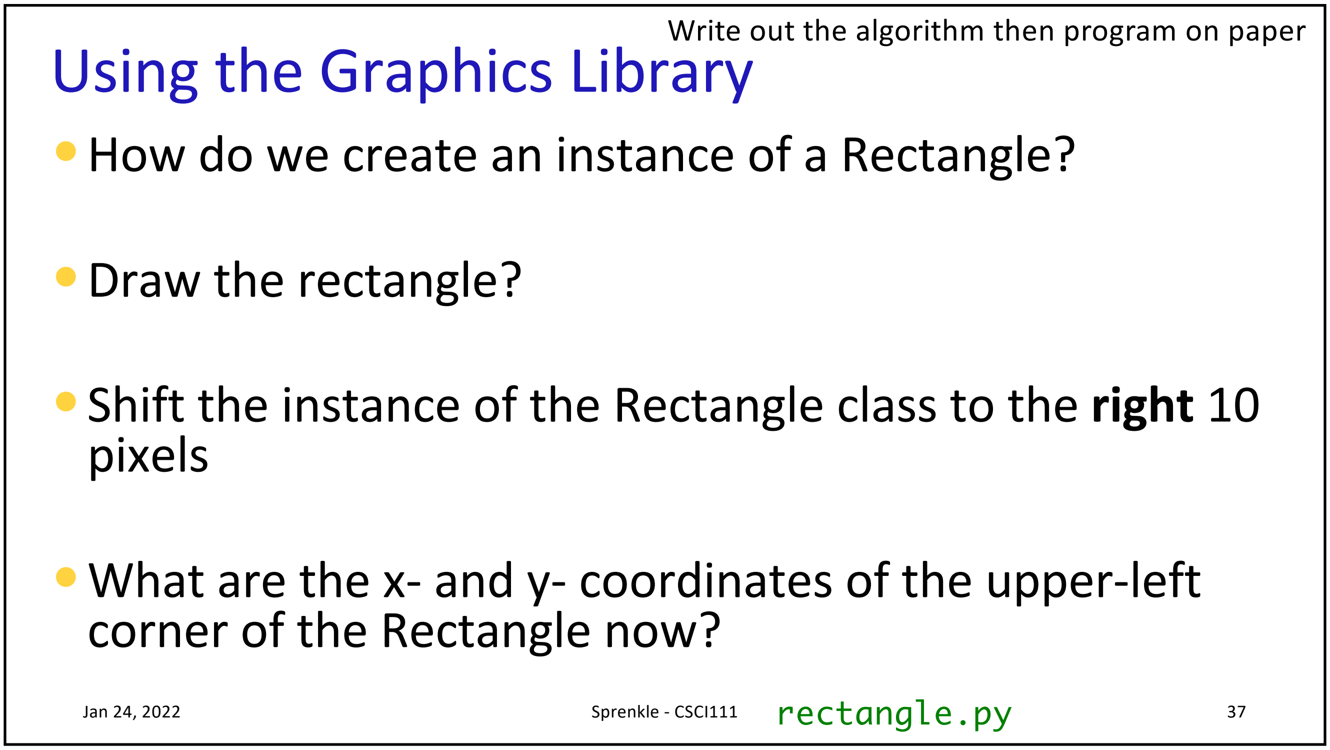


Colors

- 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`

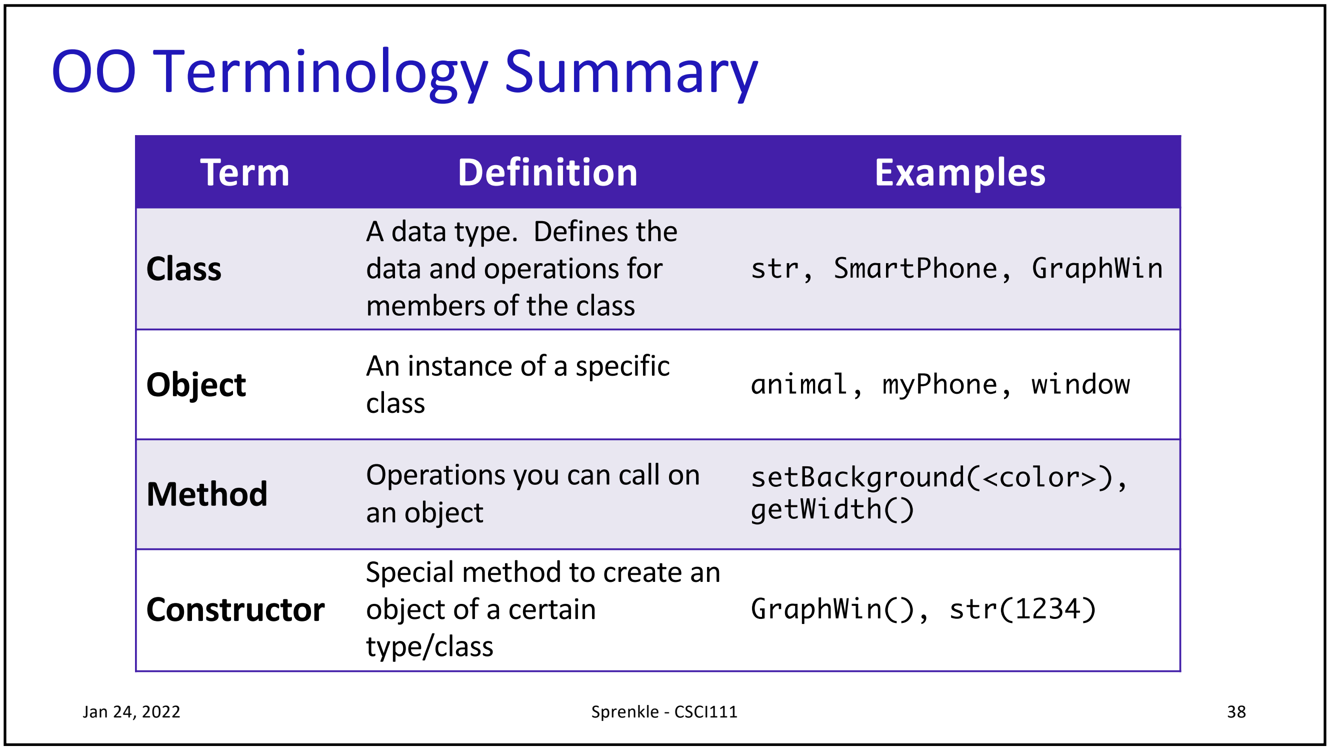


Write out the algorithm then program on paper

Using the Graphics Library

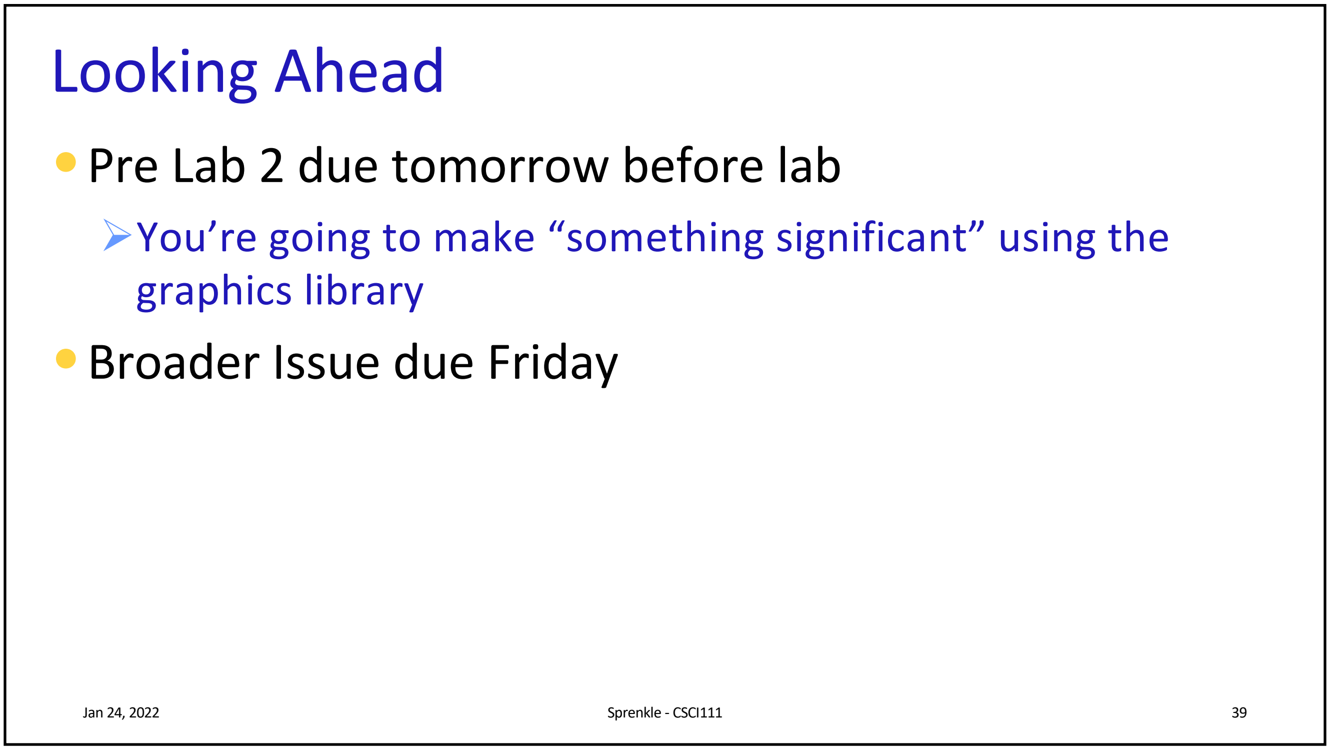
- How do we create an instance of a Rectangle?
- Draw the rectangle?
- Shift the instance of the Rectangle class to the **right** 10 pixels
- What are the x- and y- coordinates of the upper-left corner of the Rectangle now?

`rectangle.py`



OO Terminology Summary

Term	Definition	Examples
Class	A data type. Defines the data and operations for members of the class	`str, SmartPhone, GraphWin`
Object	An instance of a specific class	`animal, myPhone, 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)`



Looking Ahead

- Pre Lab 2 due tomorrow before lab
  - You’re going to make “something significant” using the graphics library
- Broader Issue due Friday