Lecture 25
Graphics

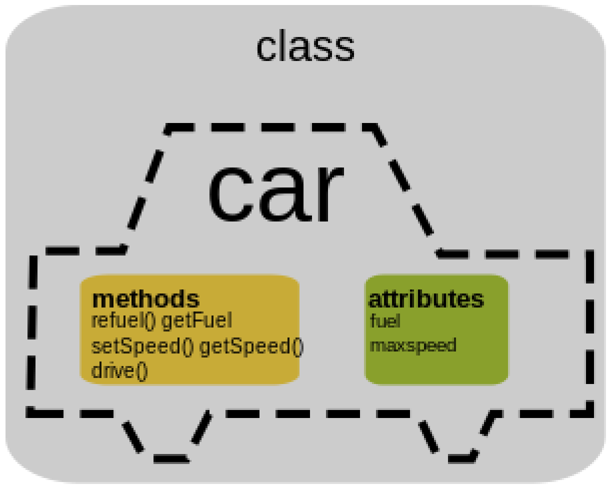
Objectives of this Lecture

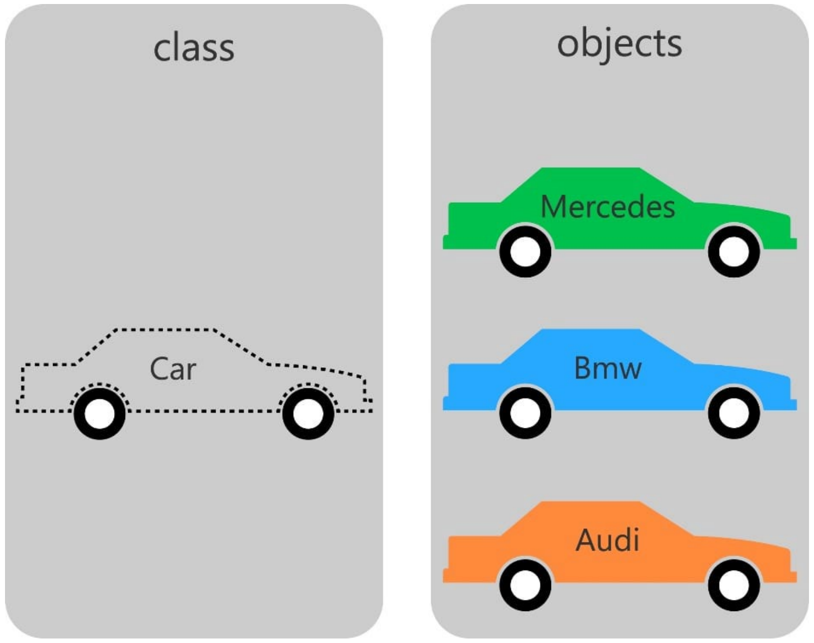
- To get familiar with the various objects available in the graphics library.
- To be able to create objects in programs.
  - *call appropriate methods to perform graphical computations*

Revision: Object Oriented Programming

- Basic idea – view a complex system as the interaction of simpler objects.
- An object is a kind of active data type that combines data and operations.
  - *Objects know stuff (contain data) and they can do stuff (have operations).*
- Objects interact by sending each other messages *(requests to do stuff)*.

Revision: OOP concept





Objects for Graphics Programming

- Most applications you’re familiar with have Graphical User Interfaces (GUI)
- GUI provides windows, icons, buttons and menus (these are also known as objects).
- There’s a simple graphics library written specifically to go with a reference book.
- Operations using this library will be used to illustrate object-oriented programming in Python

Aside: Importing Library Functions

- Many Python programmers believe it is tedious to prepend library names in front of library functions, objects, etc,
  - `math.sqrt()`
- Python allows you to import all functions from a module
  - `from math import *`

    All the functions from this library will be imported and can be used without further qualification.
  - `sqrt(5)    # rather than math.sqrt(5)`

Importing Library Functions

- We can also import one function from a library

```
>>> from math import sqrt
>>> sqrt(5)
```

- Problem is that after the import, further down the program, when you see the name of a function you have no idea where it came from.
  - *Can make debugging harder later*
- Better to leave original module name, or create short-hand:

```
>>> import math as
>>> win = m.sqrt(5)
```

Module and function

- Module:
  - allows you to logically organize your Python code
  - grouping related code that makes the code easier to understand and use
  - Simply, a file consisting of Python code which can define functions, classes, variables and may also include runnable code
- Function:
  - a block of organized, reusable code that is used to perform a single, related action.
  - provide better modularity for your application and a high degree of code reusing

Simple Graphics Programming

- Python provides graphics capabilities through Tkinter.
- The book *"Python Programming: An Introduction to Computer Science", 3rd Edition, John Zelle, Franklin Beedle*, comes with graphics.py library
  - *https://mcsp.wartburg.edu/zelle/python/graphics.py* *Copy on LMS*
- Where to put the library
  - *In the same folder as your other Python programs for this unit*

Using the graphics.py Library

- We need to import the library first

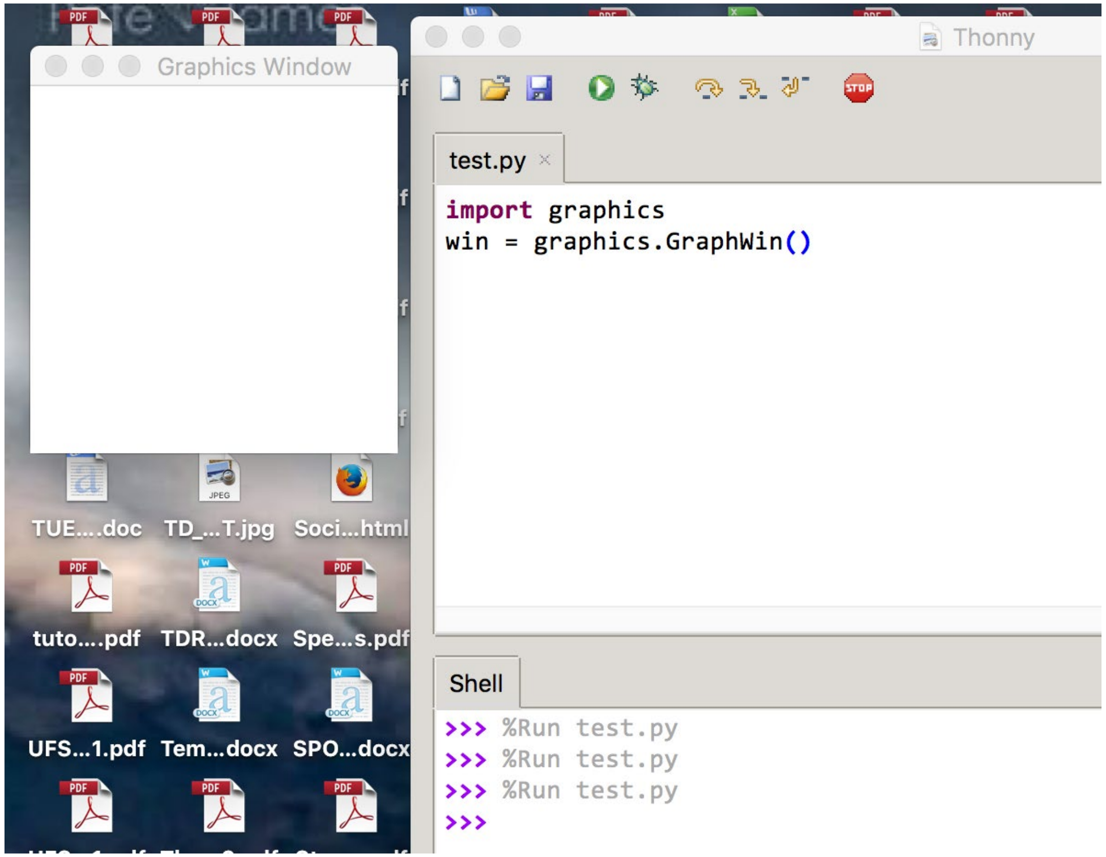
```
>>> import graphics
```

- A graphics window is a place on the screen where the graphics will appear.

```
>>> win = graphics.GraphWin()
```

- This command creates a new window object titled "Graphics Window"

Using the graphics.py



```
import graphics
win = graphics.GraphWin()
```

```
>>> %Run test.py
>>> %Run test.py
>>> %Run test.py
>>>
```

Graphics and Objects

- `GraphWin()` creates an object which is assigned to the variable *win*.
- We can manipulate the window object through this variable.
  - *Like having* `x = 6` *and then performing integer operations, e.g.,* `x *= 7`
- For example, windows can be closed/destroyed by issuing the command

```
>>> win.close()
```

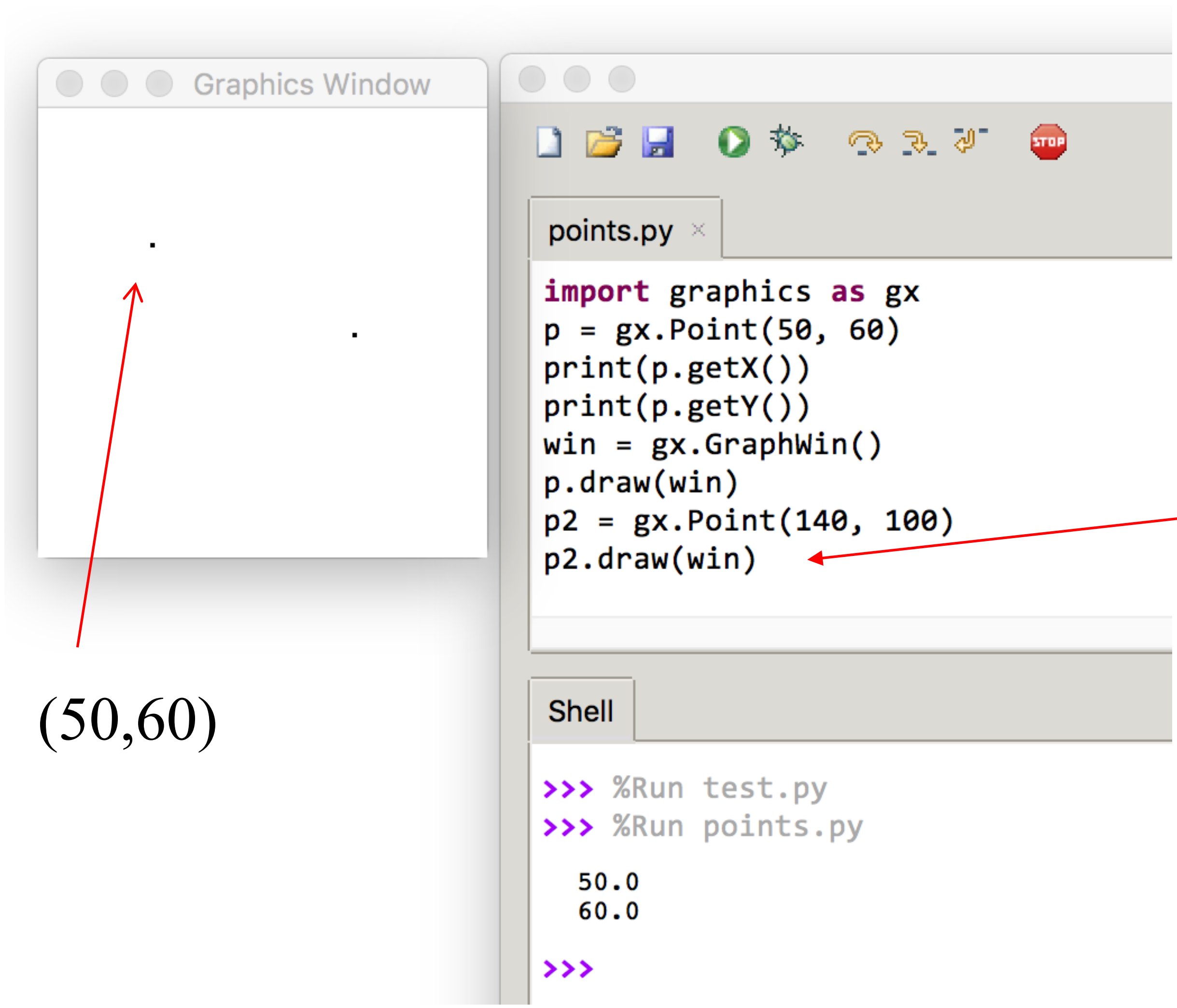
Graphics Window

- A graphics window is a collection of points called pixels (picture elements).
- The default GraphWin is 200 pixels tall by 200 pixels wide (40,000 pixels total).
- One way to get pictures into the window is one pixel at a time, which would be tedious.
- The graphics library has a number of predefined routines to draw geometric shapes.

A Point in Graphics

- The simplest object is the `Point`.
- Point locations are represented with a coordinate system (*x*, *y*), where *x* is the horizontal location of the point and *y* is the vertical location.
- The origin (0,0) in a graphics window is the upper left corner.
- X values increase from left to right, y values **from top to bottom**.
- Lower right corner is (199, 199)

Simple Graphics Commands



```
import graphics as gx
p = gx.Point(50, 60)
print(p.getX())
print(p.getY())
win = gx.GraphWin()
p.draw(win)
p2 = gx.Point(140, 100)
p2.draw(win)
```

```
>>> %Run test.py
>>> %Run points.py

  50.0
  60.0

>>>
```

(50,60)

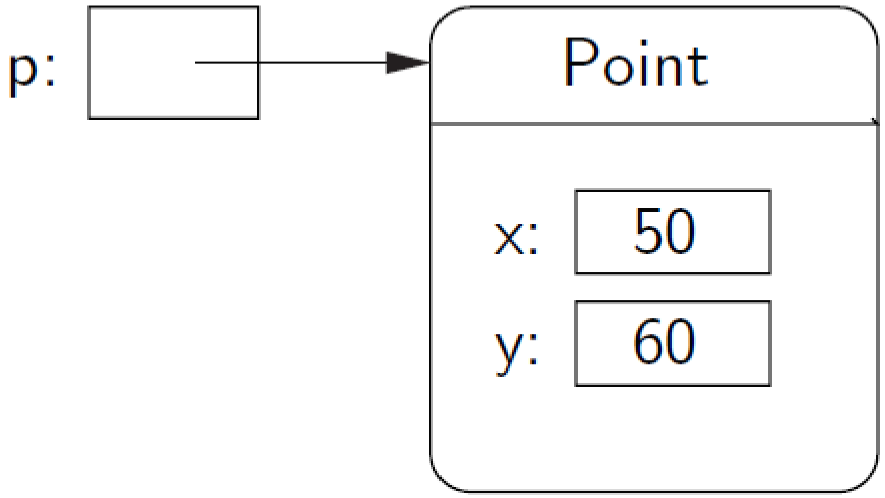
Objects only become visible when the object is drawn in the window

Creating a New object

- To create a new object of a class, we use a special operation called a *constructor*.
  <class-name>(<param1>, <param2>, …)
- A <class-name> is the name of the class we want to create a new object of, e.g., `Circle` or `Point`.
- *The parameters are required to initialize the object. For example,* `Point` *requires two numeric values;* `GraphWin` *can, optionally, take a name for the window.*
  - `Point(50, 60)`

Example of Creating a New Instance

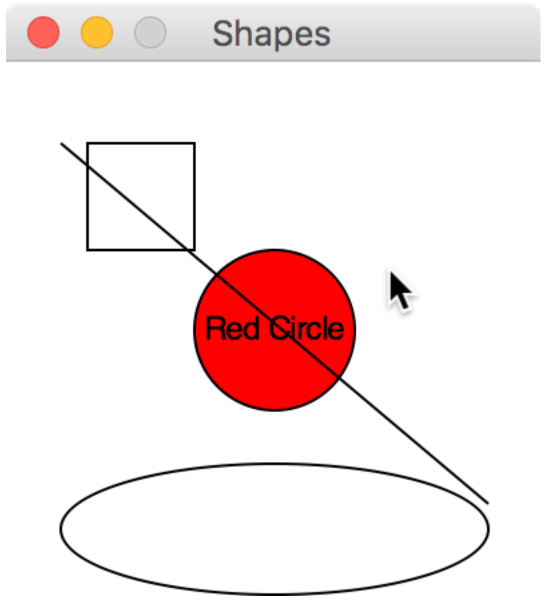
- `p = Point(50, 60)`
- The constructor for the `Point` class requires two parameters, the *x* and *y* coordinates for the point.
- These values are stored as *object variables* inside the object.



Drawing Geometric Shapes

```
import graphics as gx

### Open a graphics window
win = gx.GraphWin('Shapes')
##Draw a red circle centered at point (100,100) with radius 30
center = gx.Point(100, 100)
circ = gx.Circle(center, 30)
circ.setFill('red')
circ.draw(win)
### Put a textual label in the center of the circle
label = gx.Text(center, "Red Circle")
label.draw(win)
### Draw a square using a Rectangle object
rect = gx.Rectangle(gx.Point(30, 30), gx.Point(70, 70))
rect.draw(win)
### Draw a line segment using a Line object
line = gx.Line(gx.Point(20, 30), gx.Point(180, 165))
line.draw(win)
### Draw an oval using the Oval object
oval = gx.Oval(gx.Point(20, 150), gx.Point(180, 199))
oval.draw(win)
```



Using Graphics Objects

- Computation is preformed by asking an object to carry out one of its operations; “message”.
- In the previous example we manipulated GraphWin, Point, Circle, Oval, Line, Text and Rectangle. These are examples of *classes*.
- Each object is related to some class and the class describes the properties of the object.
  - int, float, str, None *are classes*
- If we say Snoopy is a dog, we mean Snoopy is a specific individual of the class of dogs. Snoopy is an object of the dog class.

Class – Object

Class: Think of it as a "template" or a "blueprint" used to create objects.

Object: An Object is a representation of a Class. It knows stuff and can do stuff.

Summary

- We learned some basics of Object Oriented programming
- We learned how to write simple graphics programs
- We haven't learned how to define our own classes yet. This will be covered in a CITS1001.