Introduction to Programming

Spring 2022

Objects and Graphics

- **Overview**
- **The Object of Objects**
- **Simple Graphics Programming**
- Using Graphical Objects
- **Interactive Graphics**

Overview

- Each data type can represent a certain set of values, and each had a set of associated operations.
- The traditional programming view is that data is passive – it's manipulated and combined with active operations.

The Object of Objects

- Modern computer programs are built using an object-oriented approach.
- Basic idea – view a complex system as the interaction of simpler objects. An object is a sort 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.

The Object of Objects

- Suppose we want to develop a data processing system for a college or university.
- We must keep records on students who attend the school. Each student will be represented as an object.

The Object of Objects

- The student object would contain data like:
  - Name
  - ID number
  - Courses taken
  - Campus Address
  - Home Address
  - GPA
  - Etc.

The Object of Objects

- The student object should also respond to requests.
- We may want to send out a campus-wide mailing, so we'd need a campus address for each student.
- We could send the print campus address to each student object. When the student object receives the message, it prints its own address.

The Object of Objects

- Objects may refer to other objects.
- Each course might be represented by an object:
  - Instructor
  - Student roster
  - Prerequisite courses
  - When and where the class meets
  - Sample Operation
- add student, del student, change room, Etc.

Simple Graphics Programming

- This chapter uses the graphics.py library supplied with the supplemental materials.
- Two location choices
  - In Python's Lib directory with other libraries
  - In the same folder as your graphics program

Simple Graphics Programming

- Since this is a library, we need to import the graphics commands
- `>>> 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 titled "Graphics Window."

Simple Graphics Programming

- GraphWin is an object assigned to the variable win. We can manipulate the window object through this variable, similar to manipulating files through file variables.
- Windows can be closed/destroyed by issuing the command
- `>>> win.close()`

Simple Graphics Programming

- It's tedious to use the graphics. notation to access the graphics library routines.
- `from graphics import *`
- The "from" statement allows you to load specific functions from a library module. "*" will load all the functions, or you can list specific ones.

Simple Graphics Programming

- 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.

Simple Graphics Programming

- The simplest object is the `Point`. Like points in geometry, 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)
- Examples

Interactive Graphics

- In a GUI environment, users typically interact with their applications by clicking on buttons, choosing items from menus, and typing information into on-screen text boxes.
- Event-driven programming draws interface elements (widgets) on the screen and then waits for the user to do something.

Interactive Graphics

- An event is generated whenever a user moves the mouse, clicks the mouse, or types a key on the keyboard.
- An event is an object that encapsulates information about what just happened.
- The event object is sent to the appropriate part of the program to be processed, for example, a button event.

Interactive Graphics

- The graphics module hides the underlying, low-level window management and provides two simple ways to get user input in a `GraphWin`.

Getting Mouse Clicks

- We can get graphical information from the user via the `getMouse` method of the `GraphWin` class.
- When `getMouse` is invoked on a `GraphWin`, the program pauses and waits for the user to click the mouse somewhere in the window.
- The spot where the user clicked is returned as a `Point`.

Getting Mouse Clicks

- The following code reports the coordinates of a mouse click:

```
from graphics import *
win = GraphWin("Click Me!")
p = win.getMouse()
print("You clicked", p.getX(), p.getY())
```

- We can use the methods like `getX` and `getY` or other methods on the point returned
- Example