Objects

1. To understand the concept of objects and how they can be used to simplify programs
2. Understand that in python, everything is actually an object
3. To get familiar with the various objects available in the graphics library
4. To be able to create objects in programs

Overviews

The traditional programming view is that data is passive – it is manipulated and combined using active operations

Modern computer programs are built using an object-oriented approach.

Objects and object-oriented programming (OOP)

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 (request do to stuff)

Like

A person drives a car (the class)

Leo is driving a car (the objects)

These two objects are interacting

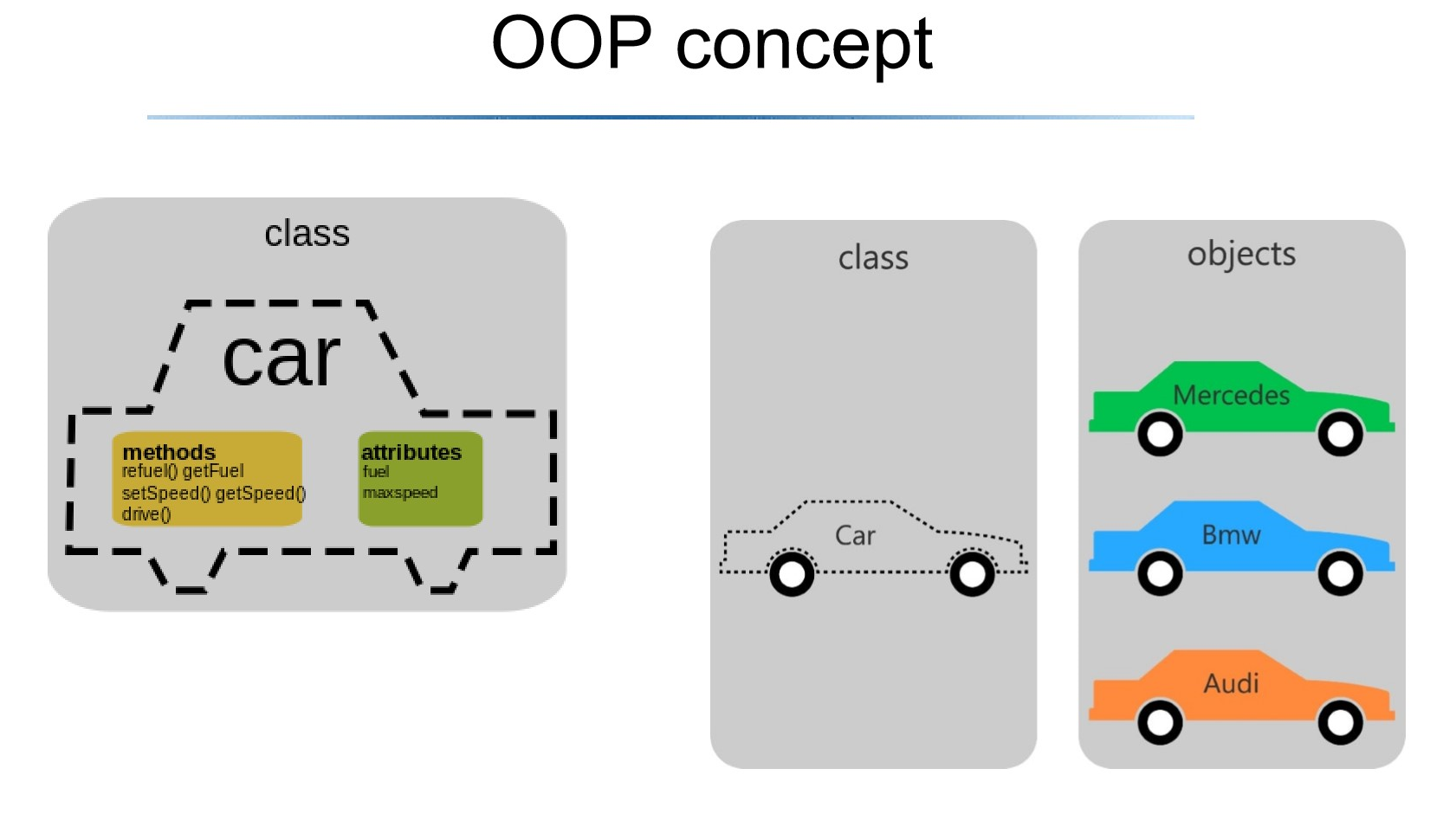
OOP concept

Class: definition of objects that share structure, properties and behaviours.

Instance: concrete object, created from a certain class.

Class 表示一类东西

Object 表示那一类东西里面的具体例子

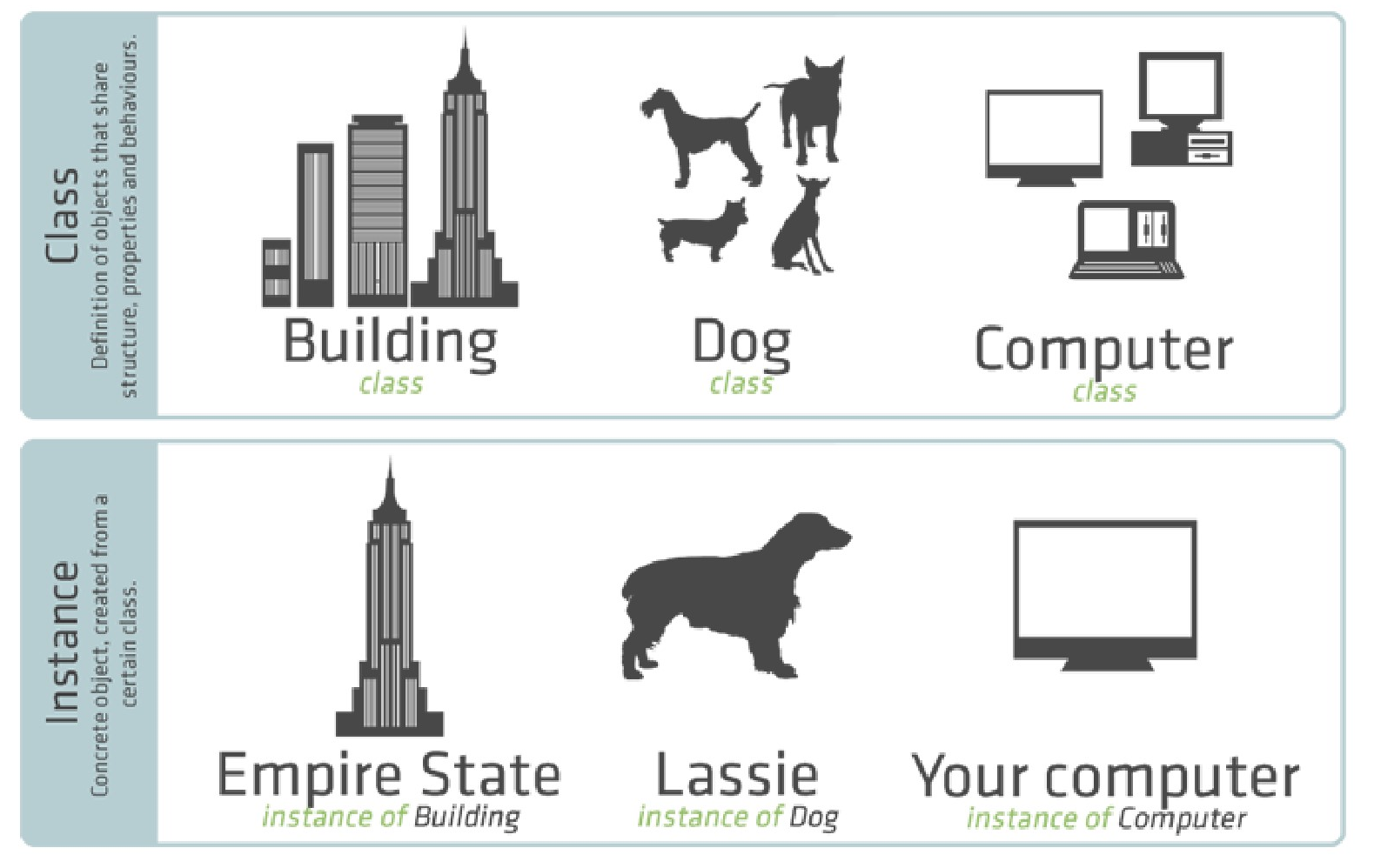


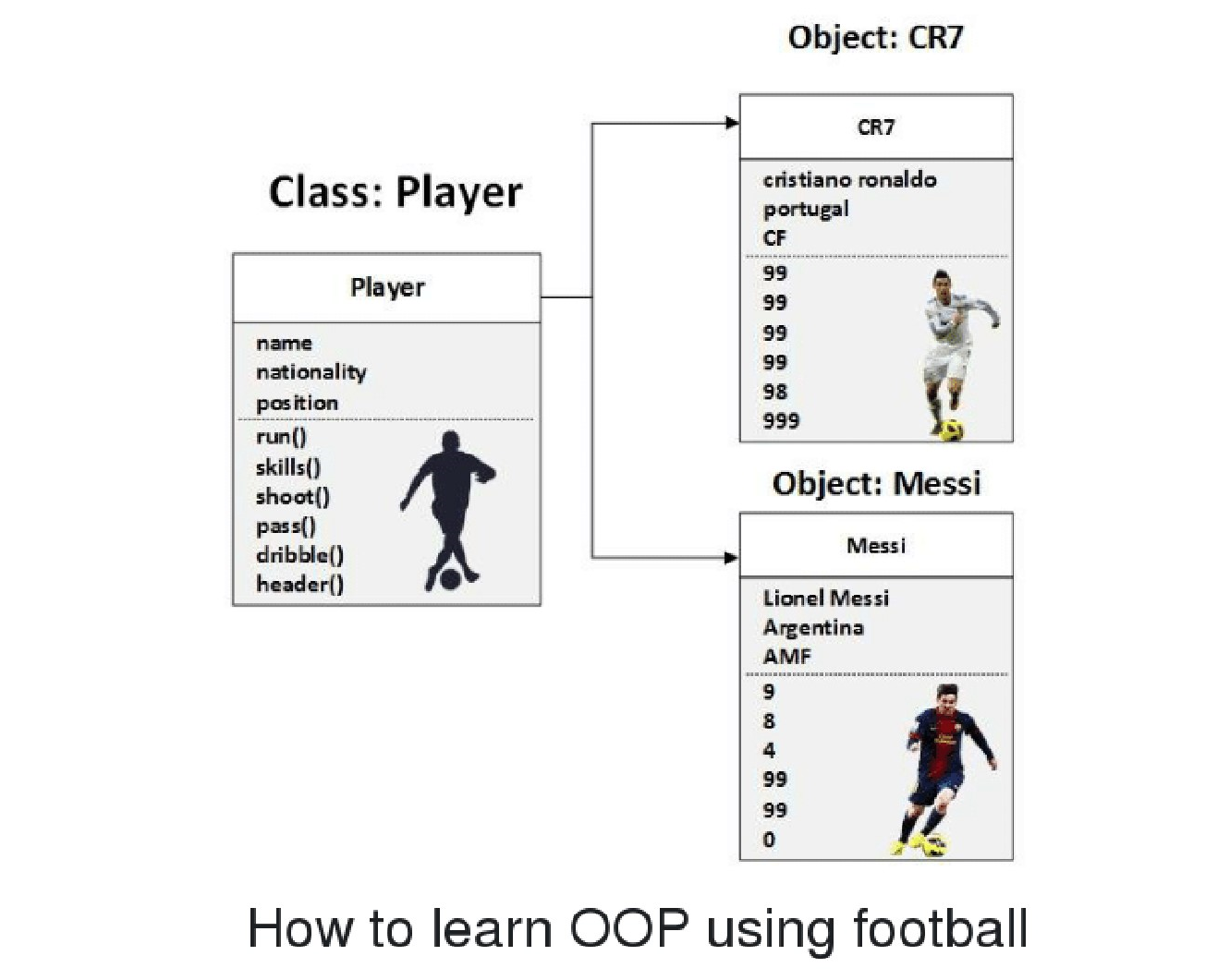
The method is all the functionality which this class can do

The attributes are the properties which we have got

Class is abstract

Object is physical and existent compared with class





Another example

Suppose we want to develop a data processing system for a university.

We must keep records on students who attend the university

Each student, each unit, etc., will be represented as different sorts of objects

University student object (all objects interact with each other)

The information in a student object

1. Name
2. Home address
3. Residential address (if away from home)
4. Units

What would the student object do?

1. The student object should respond to requests
2. We way want to send out a campus-wide mailing, so we need a campus address for each student.
3. We could send the getHomeAddress() to each student object. When the student object receives the message, it responds with the home address.

Course object

Each course might also be represented by an object:

The information:

1. Instructor
2. Students enrolled
3. Pre-requisite courses
4. When and where the class meets

Sample operations of the course-object

1. addStudent() # add student-object to course-object
2. delStudent()
3. changeroom()

Objects within objects

An object can have one or more objects inside it (for example, the course-object will have student-objects inside)

Similarly, the course-object may have an instructor-object

Class – instance – object

Class: thick of it as a template or a blueprint used to create objects

Instance: a unique copy of a class representing an object

Object: an object is an instance of a class. It knows stuff and can do stuff.

All of you, I create an instance and you would be human of our student class. But each one of you is object. And you are the instance of the class like you are some kind of representation of the class.

You create an instance of class, and them you name it, like A,

A is an object, but when you create it that is instance.

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 objects)

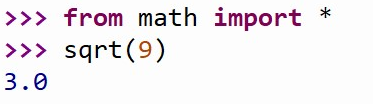
Operations using this library will be used to illustrate object-oriented programming in python

Import library functions

Many python programmers believe it is tedious to prepend library names in front of library functions, objects, etc.

Like math.sqrt ()

Python allows you to import all functions from a module



All the functions from this library will be imported and can be used without further qualification

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.

Better to have leave original module name, or create short-hand,

Like m. sqrt ()

Where to put the library

In the same folder as your other python programs for this unit

Graphics and objects

Win = Graphics. graphWin ()

graphWin () creates an object which is assigned to the variable win

we can manipulate the window object through this variable

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)

(像素)

The default graphWin is 200 pixels tall by 200 pixels wide (40,000 pixels total)

A point in graphics

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)

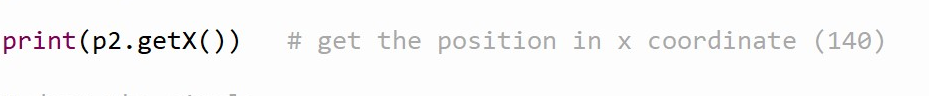
Objects only become visible when the object is drawn in the window

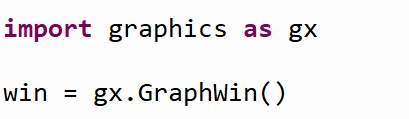
Draw the colours

Gx. setFill (“input your colour”)

Get the position of X or Y axis

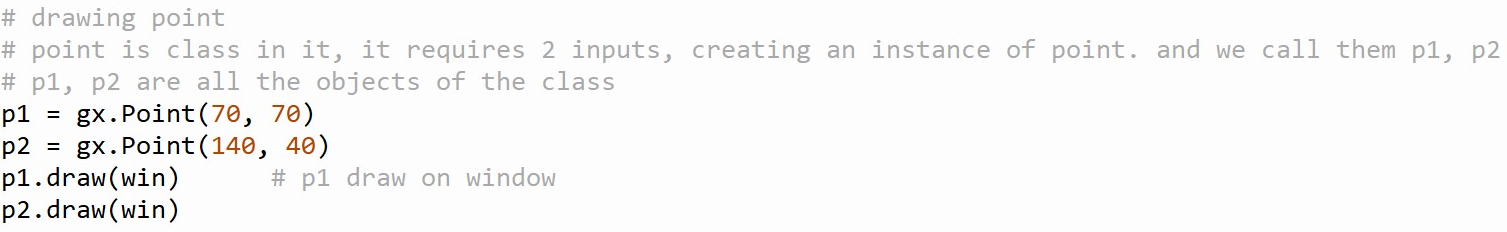
Point. getX ()



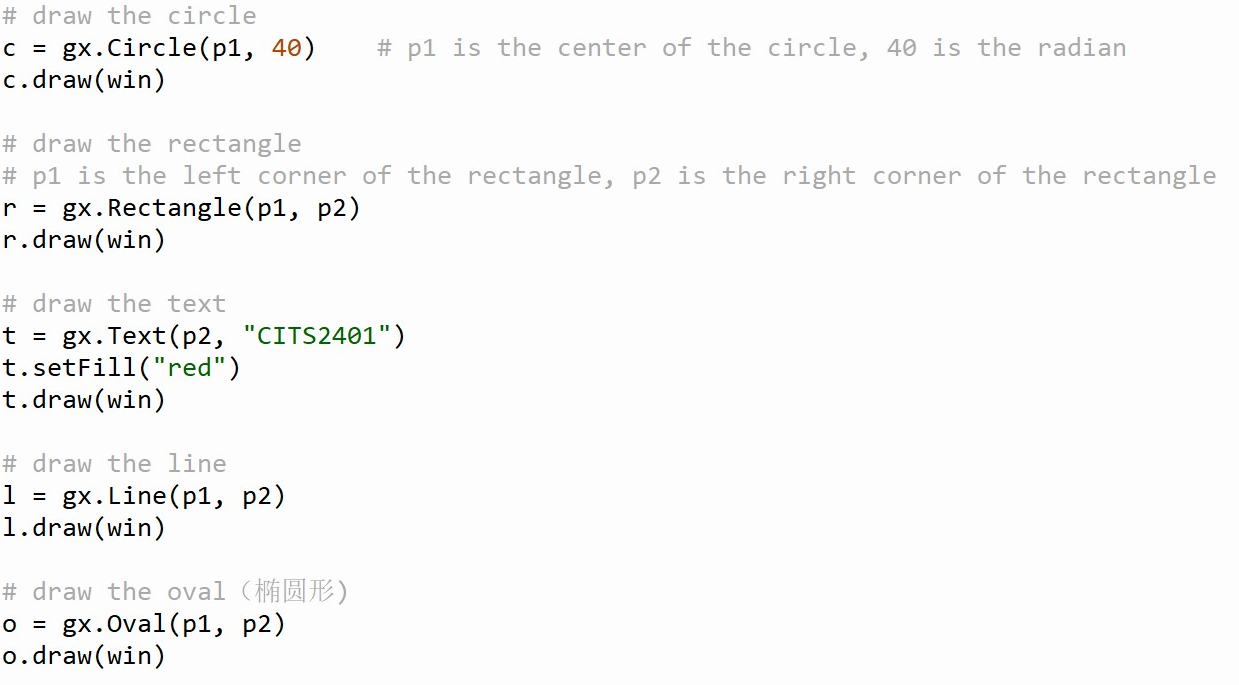


graphWin can optionally take a name for the window

Drawing points



Circle, rectangle, text, line, oval (those are class in it, and c, r, l, o are objects)



Using graphics objects

Computation is performed by asking an object to carry out one of its operations

Class: Graph, point, circle, oval, line, text and rectangle

Int, float, str, None are classes

Creating a new instance

To create a new instance of a class (object), we use a special operation called a constructor.

<class-name> (<param 1>, <param 2>, …)

Like (c. Circle (centre of the circle, radian)