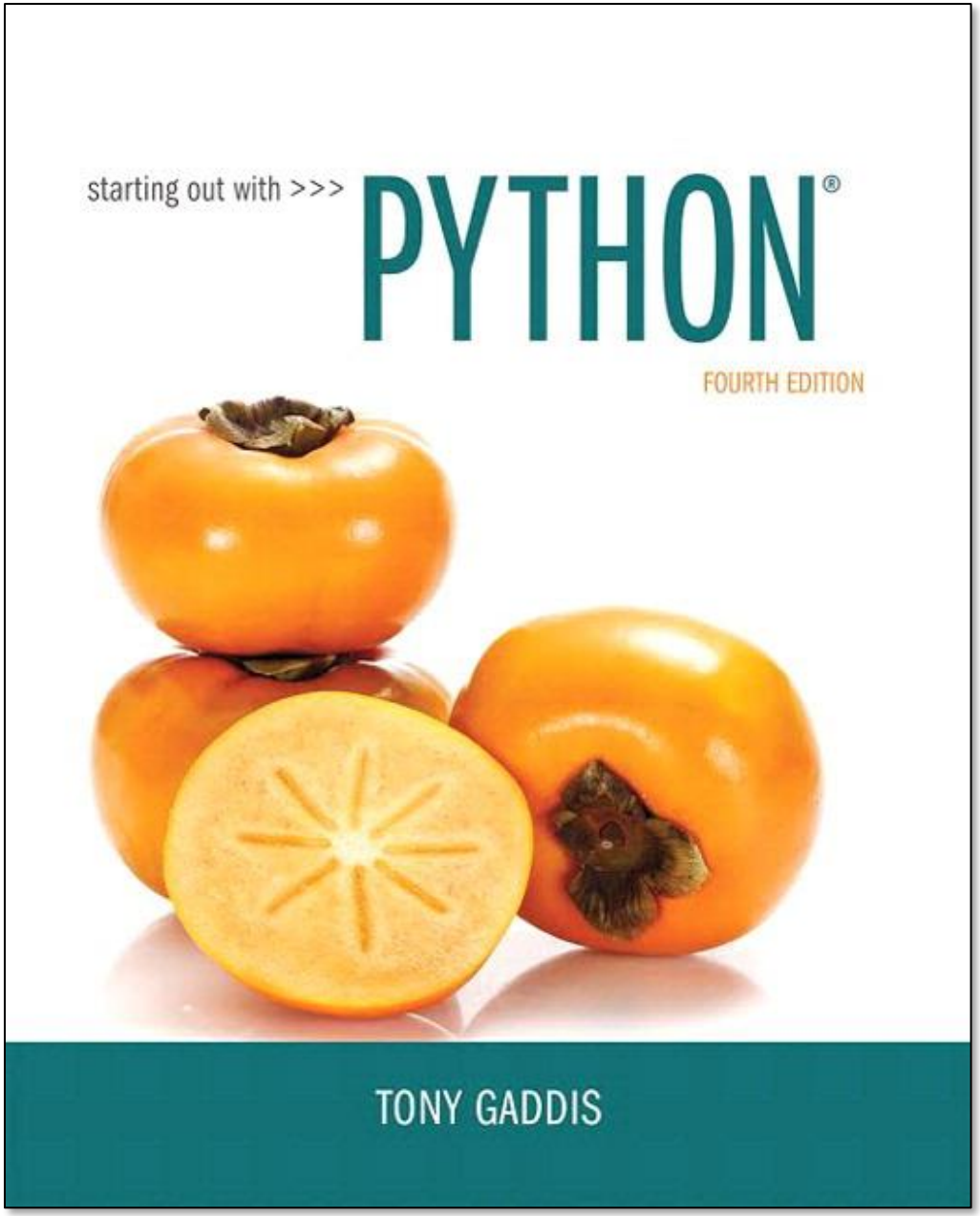
CHAPTER 10

Classes and Object-Oriented Programming

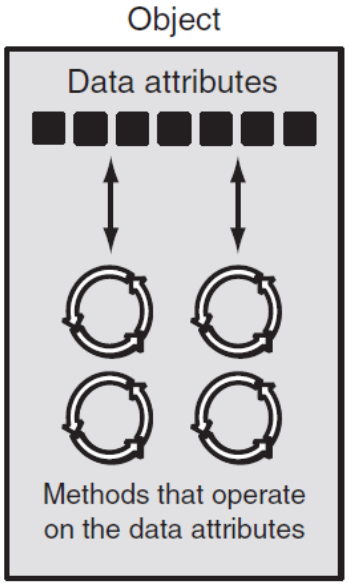


Procedural vs OOP

- **Procedural programming: writing programs made of functions that perform specific tasks**
  - Procedures typically operate on data items that are separate from the procedures
  - Data items commonly passed from one procedure to another
  - Focus: to create procedures that operate on the program's data
- **Object-oriented programming: focused on creating objects**
- **Object: entity that contains data and procedures**
  - Data is known as data attributes and procedures are known as methods
  - Methods perform operations on the data attributes
  - Encapsulation: a self contained unit combining data and code into a single object

Object-Oriented Programming

Figure 10-1 An object contains data attributes and methods

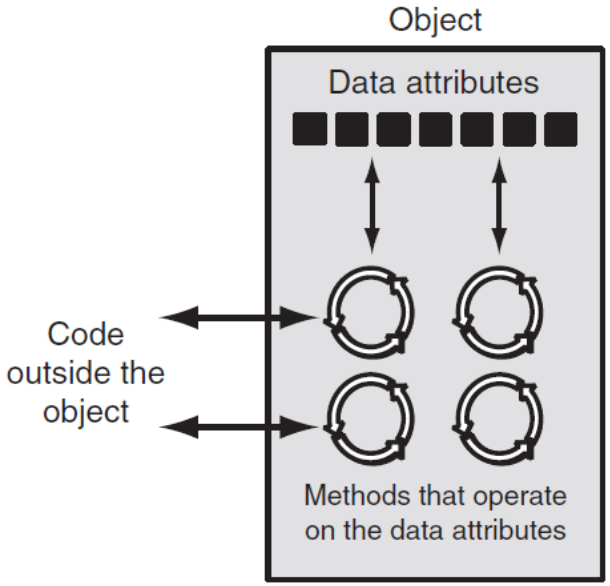


Object-Oriented Programming

- **Data hiding (making private): object's data attributes are hidden from code outside the object**
  - Access restricted to the object's methods
    - Protects from accidental corruption
    - Outside code does not need to know internal structure of the object
- **Object reusability: the same object can be used in different programs**
  - Example: 3D image object can be used for architecture and game programming

Object-Oriented Programming

Figure 10-2 Code outside the object interacts with the object's methods



An Everyday Example of an Object – Alarm Clock

- **Data Attributes (*private*)–**
  - current_second (a value in the range of 0–59)
  - current_minute (a value in the range of 0–59)
  - current_hour (a value in the range of 1–12)
  - alarm_time (a valid hour and minute)
  - alarm_is_set (True or False)

- **Methods (*public*)**
  - set_time
  - set_alarm_time
  - set_alarm_on
  - set_alarm_off

- **Methods (*private*)**
  - increment_current_second
  - increment_current_minute
  - increment_current_hour
  - sound_alarm

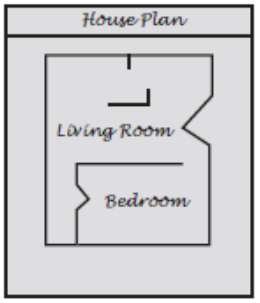
Classes

- **Class: code that specifies the data attributes and methods of a particular type of object**
  - Similar to a blueprint of a house or a cookie cutter
- **Instance: an object created from a class**
  - Similar to a specific house built according to the blueprint
  - There can be many instances of one class

Classes (cont'd.)

Figure 10-3 A blueprint and houses built from the blueprint

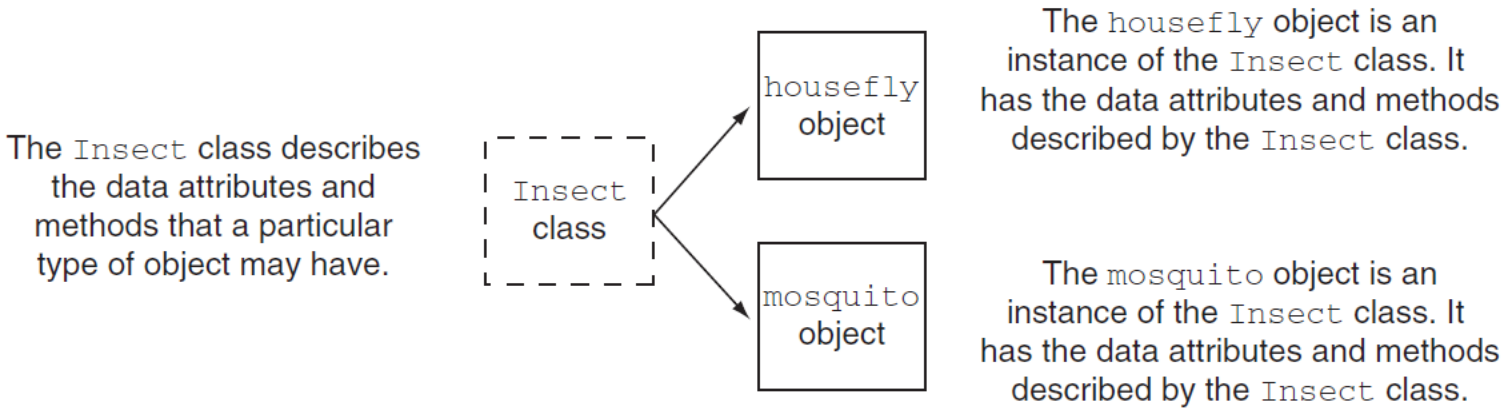
Blueprint that describes a house



Instances of the house described by the blueprint

Classes (cont'd.)

Figure 10-5 The `housefly` and `mosquito` objects are instances of the `Insect` class



Class Definitions

- **Class definition: set of statements that define a class's methods and data attributes**
  - Format: begin with `class` *`Class_name`*`:`
    - Class names often start with uppercase letter
  - Method definition like any other python function definition
    - `self` parameter: required in every method in the class – references the specific object that the method is working on

Class Definitions - Example

- **Coin Toss Example –**
  - CoinClass.py
  - CoinToss.py

Exercise 1

Create a class for an insect object. It should have 2 attributes – wings and legs. For now, the insect object has 2 wings and 4 legs. It should also have 1 method – to determine the length of flight. Length of flight should be a method that randomly assigns a number between 1 and 10 miles. Create a python program that will create an instance of the insect class and print out how many miles the insect can fly.

Data Hiding

- Users of the class should not have direct access to the attributes of the object
- Changes to the attributes of an object should ONLY be handled by methods defined in the class definition file
- Example of CoinClass.py

Class Definitions - Example

- **Bank Account Example –**
  - BankAcountClass.py
  - BankAccountProgram.py

Exercise 2

Design a class that represents a cell phone. The data that should be kept as attributes in the class are as follows:

The name of the phone's manufacturer will be assigned to the **__manufact** attribute. The phone's model number will be assigned to the **__model** attribute. The phone's retail price will be assigned to the **__retail_price** attribute. The class will also have the following methods:

An **__init__** method that accepts arguments for the manufacturer, model number, and retail price.

set_manufact, **set_model** and **set_retail_price** methods that accepts an argument for the manufacturer, model and retail_price respectively and can update it if neccessary.

get_manufact, get_model, get_retail_price method that returns the phone's manufacturer, model and price respectively.

Exercise 3

Create a student class (name the file **StudentClass.py**). The class should have 4 attributes. StudentID, Name, DOB and classification (F,S,Jr,Sr).

- Create a method that will calculate the student's current age
- Create a method that will determine when the student can register –
  - Seniors – 4/1 thru 4/3
  - Juniors – 4/4 thru 4/6
  - Sophomores – 4/7 thru 4/9
  - Freshmen – 4/10 thru 4/12
- Create a method to return the age and another method to return the registration dates.

Create a program file (name the file **StudentProgram.py**) that will create an instance of the student class and display the age of the student and when they can register.