

Programming and Database Fundamentals for Data Scientists

Classes and Objects

Varun Chandola

School of Engineering and Applied Sciences
State University of New York at Buffalo
Buffalo, NY, USA
chandola@buffalo.edu



Object Oriented Design

Encapsulation

Object Oriented Design

- ▶ Data-centric design instead of logic-centric
- ▶ Logic-centric design:
 - ▶ A program is organized as a logical procedure
 - ▶ Have functions as reusable logical blocks
- ▶ Data-centric design
 - ▶ A program is essentially a way to manipulate data
 - ▶ Data encapsulated as objects

How to do OOP?

- ▶ Identify objects that need to be manipulated in a program (data modeling)
- ▶ Define a class as a general description of the desired object
 - ▶ Example: Consider a banking application
 - ▶ Need to define customers
 - ▶ Each customer has a name, address, and multiple accounts
 - ▶ Each account has a type (checking or savings), current amount
 - ▶ Application: *Read data from csv files containing customer and account information and find all customers with more than \$5,000 in their bank account*
 - ▶ A class will consist of the data and the methods needed to interact with the data

Encapsulation

- ▶ A fundamental tenet of **Object Oriented Programming**
- ▶ Allows programmers to control the flow of data in a program
- ▶ Every object has some data attached to it
- ▶ Not all data is accessible to the external program
- ▶ Encapsulation controls what methods and fields are visible and how

- ▶ Define a Python class using the keyword `class`
- ▶ During the program execution, you can *instantiate* objects of a certain class
- ▶ Each class has three entities:
 - ▶ A *constructor* (using a special function called `__init__`)
 - ▶ Fields containing various data elements (mutable or immutable)
 - ▶ Methods that let you manipulate the fields or perform any task
 - ▶ Fields and methods can be defined as *public* or *private*
 - ▶ Private – only accessible within the class definition
 - ▶ Public – accessible outside (`objectname.fieldname` or `objectname.methodname()`)

Python Namespaces

- ▶ Namespaces are used to keep track of variables
 - ▶ Like a dictionary where the keys are names of variables and the values are the values of those variables.
- ▶ At a given time in a Python program, several namespaces are available:
 1. Each function has its own namespace, called the *local namespace*, which keeps track of the function's variables, including function arguments and locally defined variables.
 2. Each module has its own namespace, called the *global namespace*, which keeps track of the module's variables, including functions, classes, any other imported modules, and module-level variables and constants.
 3. Finally, a built-in namespace, accessible from any module, which holds built-in functions and exceptions.

Scope of a Namespace

- ▶ A textual region within a Python program where a namespace is directly accessible without providing the qualifying object or module name

Global and Local Scope

- ▶ One can declare a global name using the keyword `global`

Inheritance and Subclasses

- ▶ One of the most important utility of classes is the ability to define subclasses
- ▶ A *subclass* **inherits** the parent (or base) class's methods and fields
- ▶ Allows you to define new ones or modify existing ones

References