CS521 O2
Information Structures with Python

Lecture 12

Guanglan Zhang

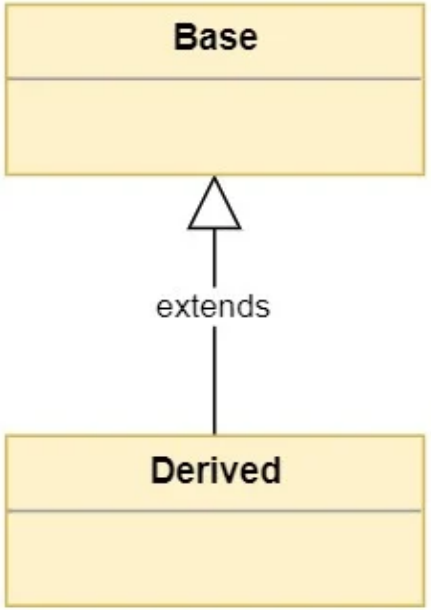
guanglan@bu.edu

Some slides adapted from Prof. Eugene Pinsky

Table of Content

Inheritance: a required feature of every OOP language

- Inheritance allows one class takes on the attributes and methods of another
- It supports code reusability
- It allows us to add more features to a class without modifying it
- Newly formed classes are called child classes, derived classes, or subclasses
- Classes from which other classes are derived are called parent classes, base classes, or super classes
- The child class can override parent methods and also can define new methods



Inheritance represented using the Unified Modeling Language (UML)

Inheritance

- Once constructed, an instance has an attribute, __class__, that indicates the class the instance was created from. This establishes the *instance-of* relationship

- Inheritance establishes an *is-a* relationship. Each class has an attribute, __bases__, that indicates the its parent classes

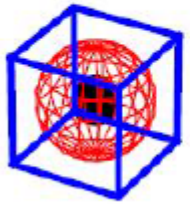
- The method resolution order (or MRO) tells Python how to search for inherited methods. Every class has an .__mro__ attribute that stores MRO information

- To get value for instance.attribute, the order of search is:
  - Look in the instance namespace, if not found, go on
  - Look in the namespace of the class of the instance (*instance-of* relationship), if not found, go on
  - Look up the class parent link (*is-a* relationship)
  - Continue until the attribute is found or there are no more *is-a* links to follow (an AttributeError is produced when not found)

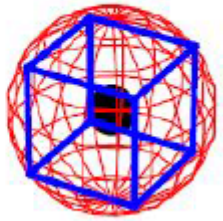
Multiple Inheritance

- Single inheritance is when a child class inherits from only one parent class
- Multiple inheritance is when a child class inherits from multiple parent classes
- Unlike Java and like C++, Python supports multiple inheritance
- In multiple inheritance, a given attribute is first searched in the current class, if it's not found, it's searched in the super classes. The super classes are searched in a depth-first, left-right fashion and each class is searched once.

sphere_in_cube



cube_in_sphere



Abstract class

- Abstract classes are classes that contain one or more abstract methods
- An abstract method is a method that is declared, but contains no implementation
- Abstract base classes provide a blueprint for concrete classes
- They provide an interface and make sure that derived concrete classes are properly implemented.
- Abstract classes cannot be instantiated, and require subclasses to provide implementations for the abstract methods
- A class that is derived from an abstract class cannot be instantiated unless all of its abstract methods are overridden

Abstract class

```
class Shape :
    def __init__ (self , r):
        self .r = r

    def volume ( self ):
        pass

class Cube (Shape):
…

class Sphere (Shape):
…

class Cylinder (Shape):
…
```

Unit Testing

- It is a software testing method to check if individual units of source code, such as functions, methods, and class, work correctly
- A unit can be viewed as the smallest testable part of an application
- Unit tests help isolate what is broken in your application and fix it faster
- An integration test checks that units in your application operate correctly together
- Python has a built-in assert() function that tests a particular condition

Structure of a test loosely using this workflow:

1. Create your inputs
2. Execute the code being tested and get the output
3. Compare the output with the expected result

Exercises

Use the Circle class we created in the last lecture as the base class, define a derived class MovingCircle that

- takes radius and (x, y) coordinates for the center with default value (0, 0)
- overrides __str__() method
- defines a new method distance() to compute its distance from (0, 0)
- Test if the distance() method works with a user-defined (x, y) coordinate

Key takeaways

- Inheritance establishes an is-a relationship.
- The MRO tells Python how to search for inherited methods
- In multiple inheritance, a given attribute is first searched in the current class, if it’s not found, it’s searched in the super classes. The super classes are searched in a depth-first, left-right fashion and each class is searched once.
- Abstract base classes separate the interface from the implementation. They define generic methods and properties that must be used in subclasses. Implementation is handled by the concrete subclasses