Lecture 23:
More on Subclassing
(Chapter 18)
CS 1110
Introduction to Computing Using Python

Revised after lecture: on slide 12, the class Shape folder’s tab should read Shape(object); the class Circle folder’s tab should read Circle(Shape)

Topics
Continuation from last lecture
• Design considerations for overriding methods
• Class attributes
• Different kinds of comparisons on objects

Announcements
• Assignment 5 due Wedn May 5th
  • Minor update—read cover page of A5 pdf posted on course website
• Prelim 2: we expect feedback to be available on Monday
• WICC (student org Women in Computing At Cornell) invites responses from CIS students on how the semester has gone:
  https://forms.gle/L72qPkYvYJDJ8cds9

Example

```python
class Shape:
    """A shape located at x,y""
    def __init__(self, x, y): ...
    def draw(self): ...

class Circle(Shape):
    """An instance is a circle.""
    def __init__(self, x, y, radius): ...
    def draw(self): ...

class Rectangle(Shape):
    """An instance is a rectangle.""
    def __init__(self, x, y, ht, len): ...
    def draw(self): ...
```

Goal: Make a drawing app
Rectangles, Stars, Circles, and Triangles have a lot in common, but they are also different in very fundamental ways....

Extending Classes
```python
class <name>(<superclass>):
    """Class specification""
    <class variables>
    <initializer>
    <methods>

    Class to extend (may need module name: <modulename>.<superclass>)
```

From last lecture:
• Parent class (superclass)
• Child class (subclass)
• Attributes, methods are inherited
• Can override parent’s method
• Function call super() to access method of parent

[Optional] wondering what’s in the object class? See https://docs.python.org/3/reference/datamodel.html#basic-customization
Design choices for method `draw`

**Who draws what?**

```python
class Shape:
    #"Moves pen to correct location"
    def draw(self):
        turtle.penup()
        turtle.setx(self.x)
        turtle.sety(self.y)
        turtle.pendown()

class Circle(Shape):
    #"Draws Circle"
    def draw(self):
        super().draw()
        turtle.circle(self.radius)
```

Note: need to import the `turtle` module which allows us to move a pen on a 2D grid and draw shapes.

No matter the shape, we want to pick up the pen, move to the location of the shape, put the pen down. But only the shape subclasses know how to do the actual drawing.

Class attributes

```python
class Shape:
    # Inherits from object by default
    object
    #"Instance is shape @ x,y"
    # Class Attribute tracks total num shapes
    NUM_SHAPES = 0

class Circle(Shape):
    #"Instance is a Circle @ x,y with radius"
    # Class Attribute tracks total num circles
    NUM_CIRCLES = 0
```

Q1: Name Resolution and Inheritance

```python
class A:
    x = 3 # Class Variable
    y = 5 # Class Variable

    def f(self):
        return self.x

    def g(self):
        return 10

class B(A):
    y = 4 # Class Variable
    z = 42 # Class Variable

    def g(self):
        return 14

    def h(self):
        return 18
```

Execute the following:

```python
>>> a = A()
>>> a.y = 20
```

What is the value of `b.x`?

A: 4  B: 3  C: 42  D: ERROR  E: I don't know
Q2: Name Resolution and Inheritance

```python
class A:
    x = 3 # Class Variable
    y = 5 # Class Variable
    def f(self):
        return self.g()
    def g(self):
        return 10

class B(A):
    y = 4 # Class Variable
    z = 42 # Class Variable
    def g(self):
        return 14
    def h(self):
        return 18
```

• Execute the following:
  ```python
  >>> a = A()
  >>> b = B()
  ```

• What is value of \( a.z \)?
  - A: 4
  - B: 3
  - C: 42
  - D: ERROR
  - E: I don't know

Q3: \texttt{eq} vs. \texttt{is}

\begin{itemize}
  
  \item \texttt{=} compares equality
  \item \texttt{is} compares identity
\end{itemize}

\begin{itemize}
  
  \item \texttt{c1} = \texttt{Circle(1, 1, 25)}
  \item \texttt{c2} = \texttt{Circle(1, 1, 25)}
  \item \texttt{c3} = \texttt{c2}
\end{itemize}

\begin{itemize}
  
  \item \texttt{c1} \texttt{==} \texttt{c2} \rightarrow ?
  \item \texttt{c1} \texttt{is} \texttt{c2} \rightarrow ?
  \item \texttt{c2} \texttt{==} \texttt{c3} \rightarrow ?
  \item \texttt{c2} \texttt{is} \texttt{c3} \rightarrow ?
\end{itemize}

Q4: \texttt{isinstance} and Subclasses

```python
>>> s1 = Rectangle(0,0,10,10)
>>> isinstance(s1, Square)
```