6.189 Day 8

Readings

How To Think Like A Computer Scientist, chapter 16

Exercise 8.1 – Inheritance

This is a paper and pencil exercise. Consider the following code:

```
class Spell:
    def __init__(self, incantation, name):
        self.name = name
        self.incantation = incantation

    def __str__(self):
        return self.name + ' ' + self.incantation + '\n' + self.get_description()

    def get_description(self):
        return 'No description'

    def execute(self):
        print self.incantation


class Accio(Spell):
    def __init__(self):
        Spell.__init__(self, 'Accio', 'Summoning Charm')

class Confundo(Spell):
    def __init__(self):
        Spell.__init__(self, 'Confundo', 'Confundus Charm')

    def get_description(self):
        return 'Causes the victim to become confused and befuddled.'

def study_spell(spell):
    print spell

spell = Accio()
spell.execute()
study_spell(spell)
study_spell(Confundo())
```

1. What are the parent and child classes here?

2. What does the code print out? (Try figuring it out without running it in Python)

3. Which `get_description` method is called when '`study_spell(Confundo())`' is executed? Why?

4. What do we need to do so that '`print Accio()`' will print the appropriate description ('`This charm summons an object to the caster, potentially over a significant distance`')? Write down the code that we need to add and/or change.

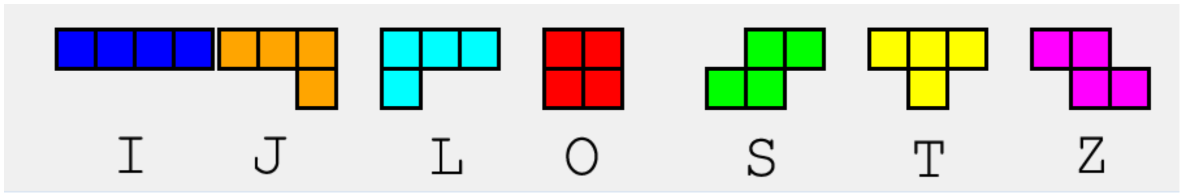
Exercise 8.2 – Tetrominoes

Tetris is deemed by some to be the most popular video game of all time. It is a puzzle game developed by Alexey Pajitnov in 1984 while he was working at the Academy of Science of the former USSR in Moscow. There have been hundreds of variants of the game developed since. The year 2009 celebrated the 25th anniversary of its creation.

We are going to create our own version of the basic Tetris game for the final project. The goal of this exercise is to get familiar with the game and to create the shapes (also called tetrominoes) used in the game. Here are several links where you can get a taste of the game if you've never played it before - http://www.tetrisfriends.com/games/Survival/game.php (fancy gui) and http://www.tetrislive.com/ (simpler gui), and
http://vadim.oversigma.com/games/gbt.html (uses the Green Building as a screen for playing

the game). Just remember you need to stop playing at some point `:D`.

Each of the tetrominoes has 4 blocks; a block is a rectangle (more specifically, a square). (Notice relationships here? 'Has' signifies containment, or an attribute. 'Is' signifies inheritance.) Here is an example of the basic tetrominoes.



8.2.1 - Blocks

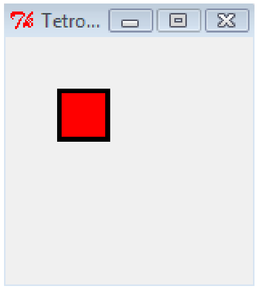
The tetris board is typically 10x20 squares, where a square has a width of 30 pixels. Each block occupies a single square at a time. For this problem, we'll think of the board not in terms of pixels but in terms of squares - so we'll pass in Points in terms of square position, and your methods should convert these into pixels for display purposes.

Create a `Block` class that inherits from the `Rectangle` class and save it in a file called `tetrominoes.py`. It should have `x` and `y` attributes that correspond to the position of the block on the tetris board, in terms of squares, *not* pixels. The position `(0,0)` is the top left corner of the board, and `(9, 19)` is the bottom right corner.

Here is how you can use the block:

```
win = GraphWin("Tetrominoes", 150, 150)

# the block is drawn at position (1, 1) on the board
block = Block(Point(1, 1), 'red')
# the draw method for your block should deal with converting
# the Point into pixels
block.draw(win)

win.mainloop()
```

and what it will look like on a 5x5 board:



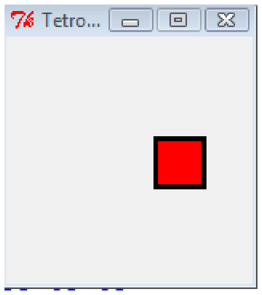
Hint: Think about how to initialize the Rectangle superclass: remember that we initialize a superclass by calling its init method, ie, `superclass.__init__(self, param1, param2, ...)`

8.2.2 - Moving Blocks

Add a `move` method to your `Block` class that will take as parameters `dx` and `dy` telling the block to move `dx` squares in the `x` direction and `dy` squares in the `y` direction. Again, our y-axis will be pointing downwards.

Here is how you can move the block:

```
win = GraphWin("Tetrominoes", 150, 150)

# the block is drawn at position (1, 1) on the board
block = Block(Point(1, 1), 'red')
# Draw the block
block.draw(win)
# And move it by 2 squares in the x direction, 1 in the y direction
block.move(2, 1)

win.mainloop()
```



Hint: Think about how to reuse the Rectangle superclass's move method.

8.2.3 - Tetromino

Create a `Shape` class that has a *list* of *blocks* as an attribute, and both `move` and `draw` methods. The `move` method will also take as parameters `dx` and `dy`, which tells the shape to move `dx` squares in the `x` direction and `dy` squares in the `y` direction.

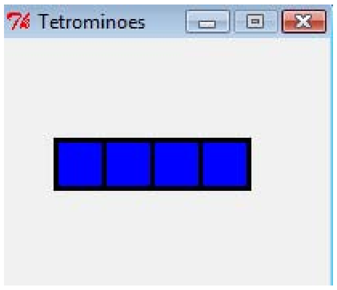
Once you have your `Shape` class made, add in this code for the `I_shape` class, a subclass of the `Shape` class.

```
class I_shape(Shape):
    def __init__(self, center):
        coords = [Point(center.x - 2, center.y),
                  Point(center.x - 1, center.y),
                  Point(center.x    , center.y),
                  Point(center.x + 1, center.y)]
        Shape.__init__(self, coords, "blue")
```

The parameter `center` is a `Point` that holds the position of the central block in the shape.

Verify that your code displays the `I_shape` correctly. Here is an example test:

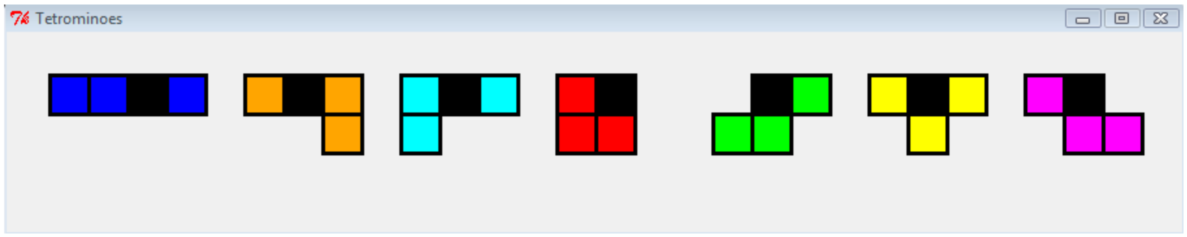
```
win = GraphWin("Tetrominoes", 200, 150)


shape = I_shape(Point(1, 1))
shape.draw(win)
shape.move(2, 1)

win.mainloop()
```



In the code above, the third block of the I shape is originally drawn at position (1, 1) and then moved to position (3, 2).

8.2.4 - Many Tetrominoes

Now create a subclass for each of the other 6 shapes. One thing you need to know is which is the center block. Here are all the shapes with their center block marked in black.



Here is example code for displaying all the shapes above (of course, all your blocks should be the same color- the black blocks are only to represent the 'center' block for each shape).

```
win = GraphWin("Tetrominoes", 900, 150)
# a list of shape classes
tetrominoes = [I_shape, J_shape, L_shape, O_shape, S_shape, T_shape, Z_shape]
dx = 0
for tetromino in tetrominoes:
    # create a shape centered at row 1, column 3
    shape = tetromino(Point(3, 1))
    shape.draw(win)
    shape.move(dx, 0)
    dx += 4

win.mainloop()
```

Note: you do not have to follow the color scheme of the tetrominoes shown in the picture.