COLOR WALL

Boston Python Workshop

Review: Lists

- Create a list

Review: Lists

- Create a list
  - `primary = ['red', 'blue', 'yellow']`

Review: Lists

- Create a list
  - `primary = ['red', 'blue', 'yellow']`
- How long is our list?

Review: Lists

- Create a list
  - `primary = ['red', 'blue', 'yellow']`
- How long is our list?
  - `len(primary)`

Review: Lists

- Create a list
  - `primary = ['red', 'blue', 'yellow']`
- How long is our list?
  - `len(primary)`
- Getting items from our list

Review: Lists

- Create a list
  - `primary = ['red', 'blue', 'yellow']`
- How long is our list?
  - `len(primary)`
- Getting items from our list
  - `primary[1] = ???`
  - `primary[-1] = ???`

Review: Lists

- Create a list of numbers

Review: Lists

- Create a list of numbers
  - `myNums = [0, 1, 2, 3]`

Review: Lists

- Create a list of numbers
  - `myNums = [0, 1, 2, 3]`
- Even easier with `range`

Review: Lists

- Create a list of numbers
  - `myNums = [0, 1, 2, 3]`
- Even easier with `range`
  - `myNums2 = range(4)`
  - `myNums2 = range(5)`
  - `myNums2 = range(1, 4)`

Review: Tuples

- Create a tuple

Review: Tuples

- Create a tuple
  - `staff= ('Jessica', 'Christine', 'Liz')`

Review: Tuples

- Create a tuple
  - `staff= ('Jessica', 'Christine', 'Liz')`
- How long is our tuple?

Review: Tuples

- Create a tuple
  - `staff= ('Jessica', 'Christine', 'Liz')`
- How long is our tuple?
  - `len(staff)`

Review: Tuples

- Create a tuple
  - `staff= ('Jessica', 'Christine', 'Liz')`
- How long is our tuple?
  - `len(staff)`
- Getting items from our tuple

Review: Tuples

- Create a tuple
  - `staff= ('Jessica', 'Christine', 'Liz')`
- How long is our tuple?
  - `len(staff)`
- Getting items from our tuple
  - `staff[1]` = ???
  - `staff[-1]` = ???

Review: Tuples

- Why are tuples different from lists?

Review: Tuples

- Why are tuples different from lists?
  - Cannot add or delete items in a tuple
  - For data we don't need to change
  - Faster! than list

Review: Dictionaries

- Contain keys and values

Review: Dictionaries

- Contain keys and values
- Create a dictionary

Review: Dictionaries

- Contain keys and values
- Create a dictionary
  - `favoriteColor= {'Jessica' : 'blue', 'Liz' : 'red'}`

Review: Dictionaries

- Contain keys and values
- Create a dictionary
  - `favoriteColor= {'Jessica' : 'blue', 'Liz' : 'red'}`
- How can we use keys to lookup values?

Review: Dictionaries

- Contain keys and values
- Create a dictionary
  - `favoriteColor= {'Jessica' : 'blue', 'Liz' : 'red'}`
- How can we use keys to lookup values?
  - `favoriteColor['Jessica']`

Review: Dictionaries

- Contain keys and values
- Create a dictionary
  - `favoriteColor= {'Jessica' : 'blue', 'Liz' : 'red'}`
- How can we use keys to lookup values?
  - `favoriteColor['Jessica']`
- Why would we want this?

Review: Dictionaries

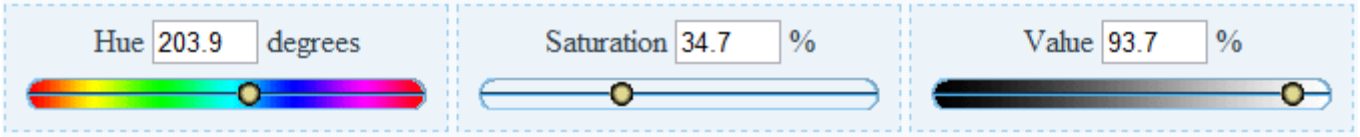
- Contain keys and values
- Create a dictionary
  - `favoriteColor= {'Jessica' : 'blue', 'Liz' : 'red'}`
- How can we use keys to lookup values?
  - `favoriteColor['Jessica']`
- Why would we want this?
  - Easy and fast way to store things that are hard to remember

ColorWall Colors

- In the HSV color space
  - Hue Saturation Value
  - http://www.yafla.com/yaflaColor/ColorRGBHSL.aspx

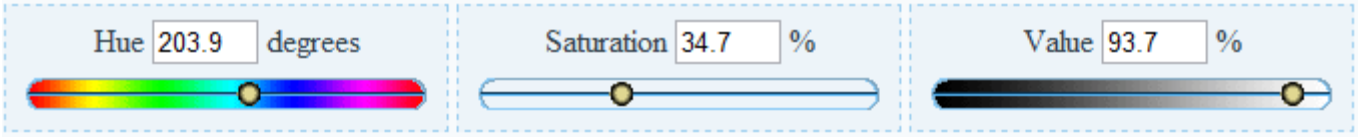
ColorWall Colors

- In the HSV color space
  - Hue Saturation Value
  - http://www.yafla.com/yaflaColor/ColorRGBHSL.aspx



ColorWall Colors

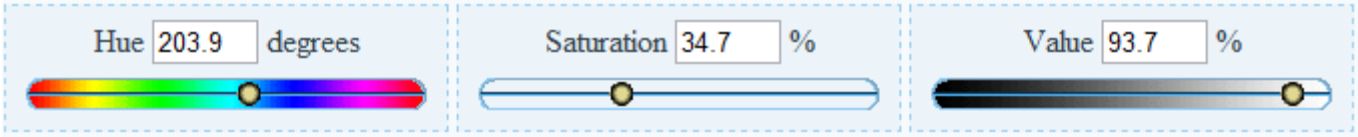
- In the HSV color space
  - Hue Saturation Value
  - http://www.yafla.com/yaflaColor/ColorRGBHSL.aspx



- But HSV colors are complicated

ColorWall Colors

- In the HSV color space
  - Hue Saturation Value
  - http://www.yafla.com/yaflaColor/ColorRGBHSL.aspx



- But HSV colors are complicated

```
# A dictionary of hsv values for some common colors.
colors = {"black":(0, 0, 0), "white":(0, 0, 1), "gray":(0, 0, 0.5),
          "red":(0, 1, 1), "blue":(0.66, 1, 1), "yellow":(0.16, 1, 1),
          "purple":(0.85, 1, 0.5), "green":(0.33, 1, 0.5), …}
```

ColorWall Effects

- Functions we write for a wall

ColorWall Effects

- Functions we write for a wall
- Describe how to color each cell in the wall

ColorWall Effects

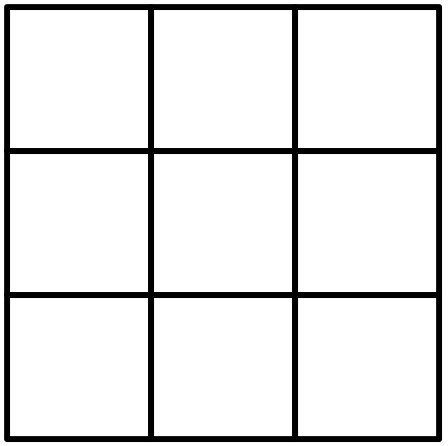
- Functions we write for a wall
- Describe how to color each cell in the wall
- Let's look at `SolidColorTest!`

ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

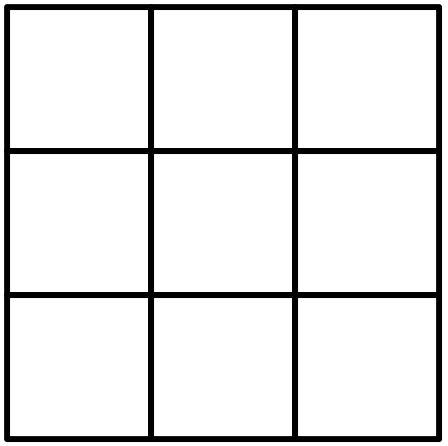


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

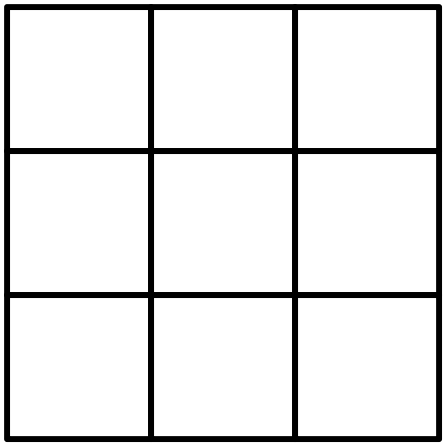


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

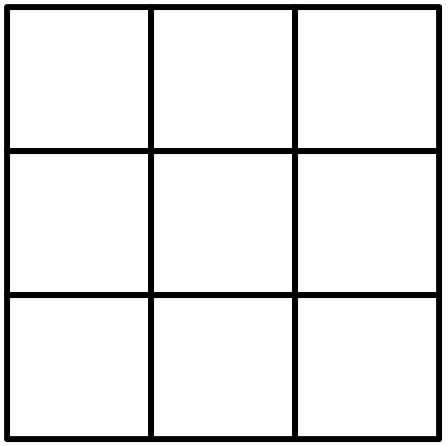


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

→   color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

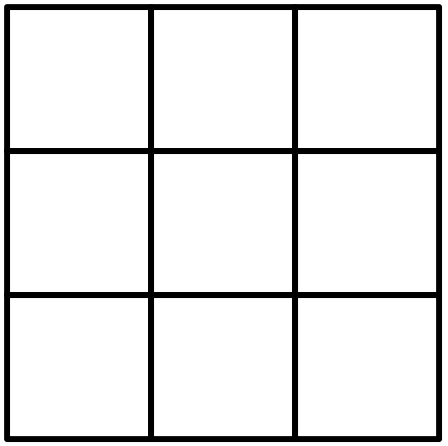


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

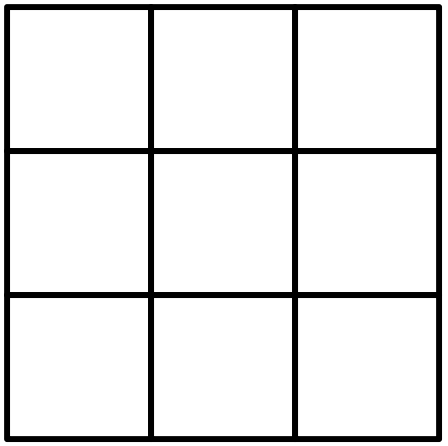


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
→   for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```



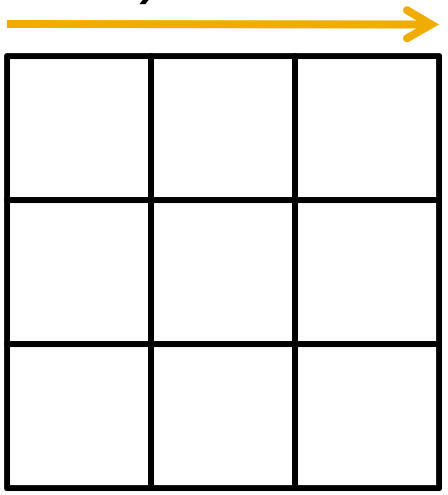
ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

x = 0



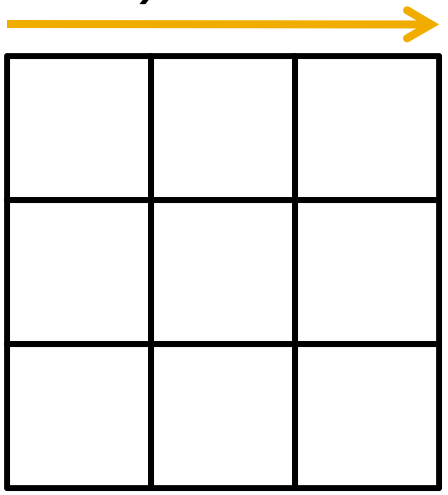
ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
→       for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

x = 0

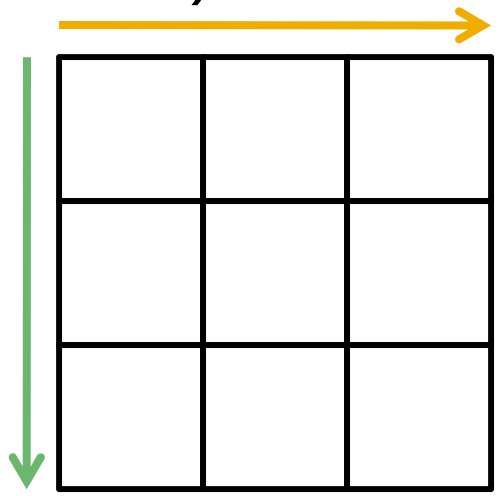


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

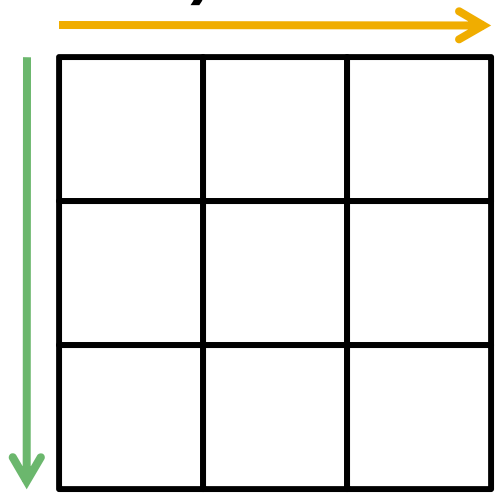


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
→           wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

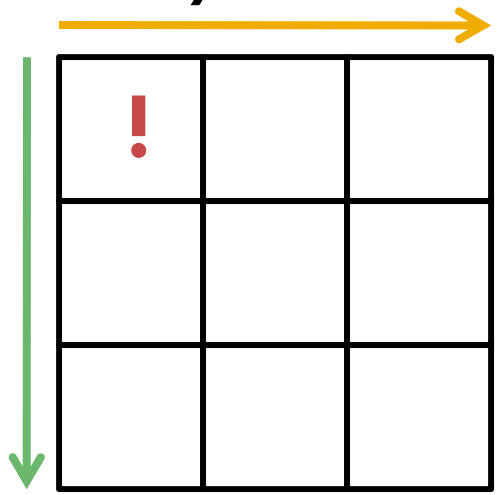


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```



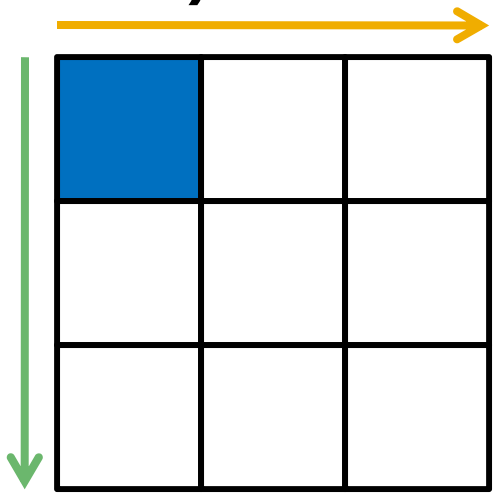
ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
→           wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

x = 0
y = 0

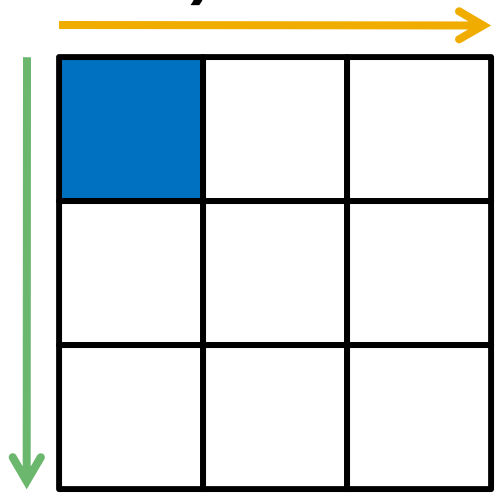


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```



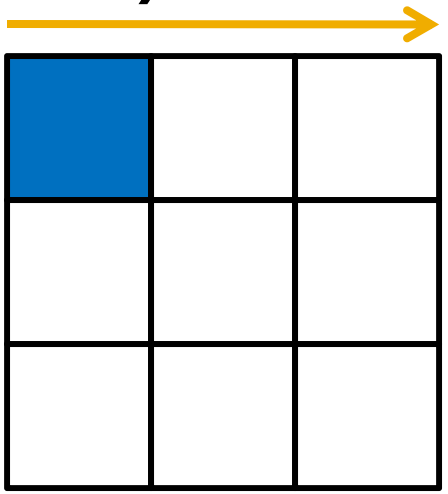
ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

x = 0
y = 1

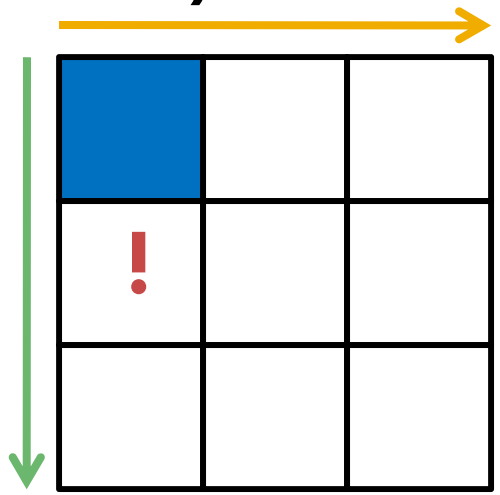


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
→           wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```



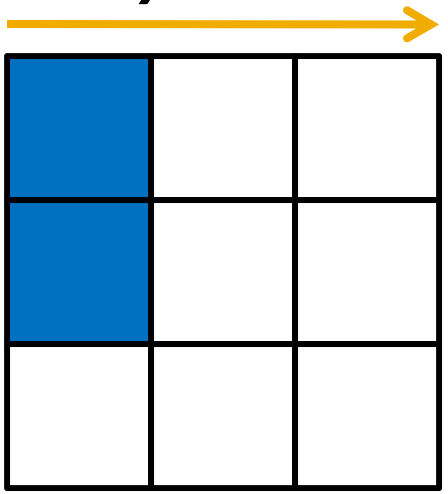
ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
→           wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

x = 0
y = 1

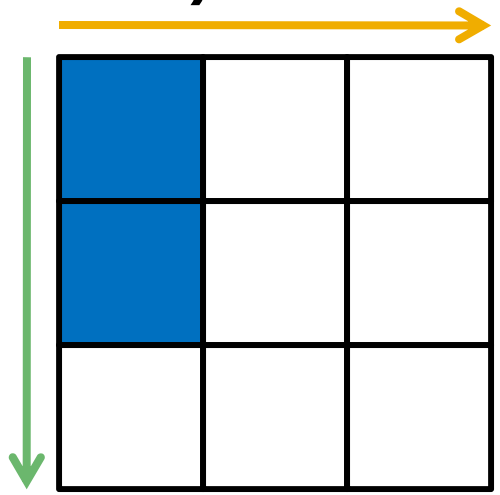


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```



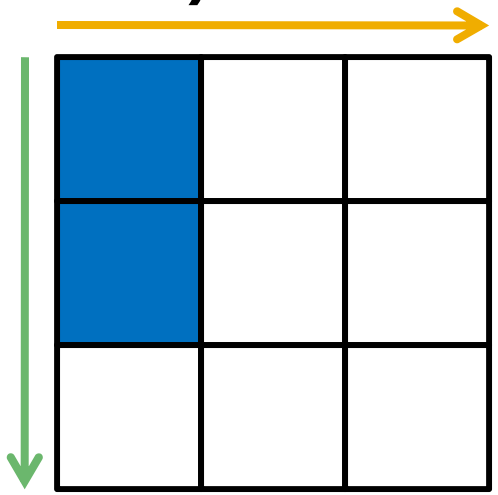
ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

x = 0
y = 2

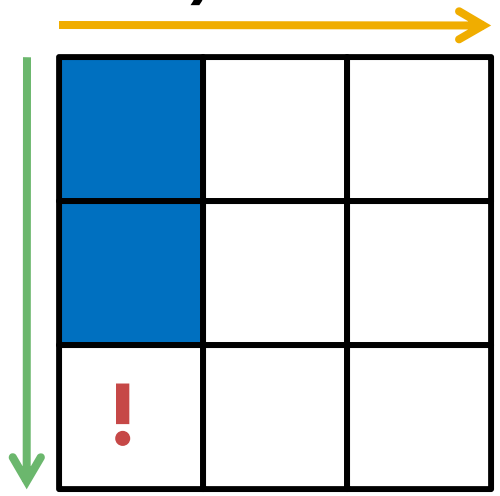


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
→           wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```



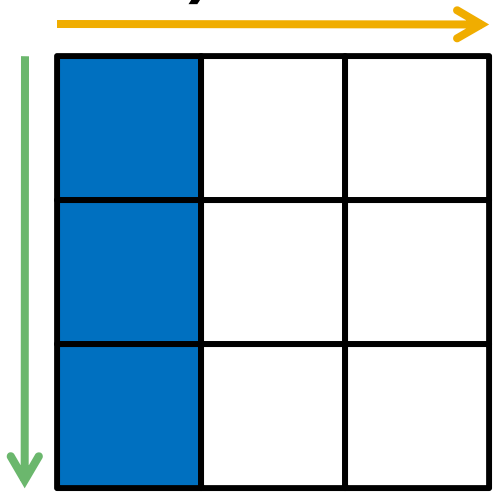
ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
→           wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

x = 0
y = 2

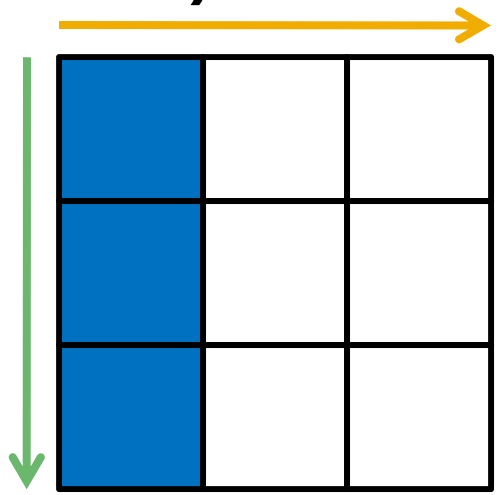


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

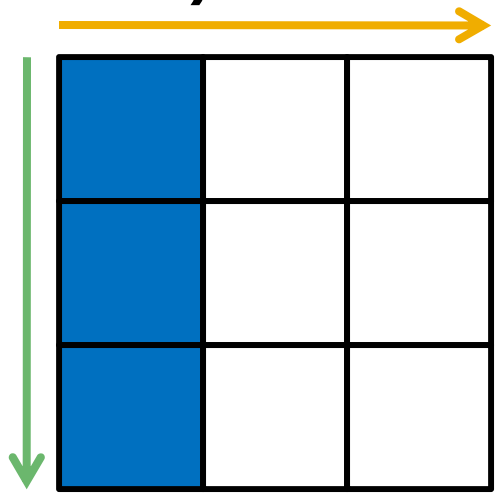


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```



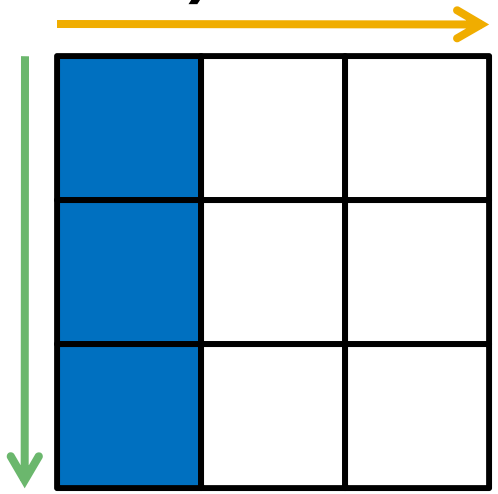
ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
→   for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

x = 1
y = ?



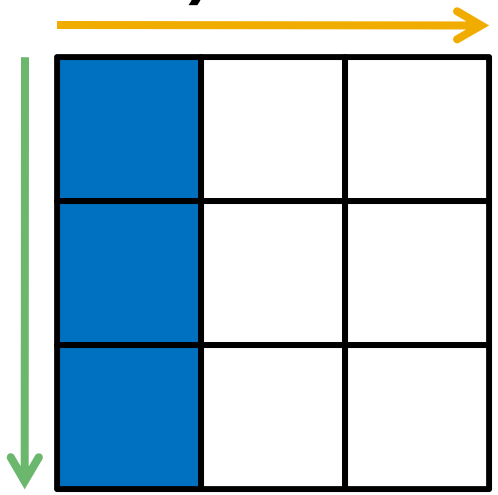
ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

x = 1
y = 0

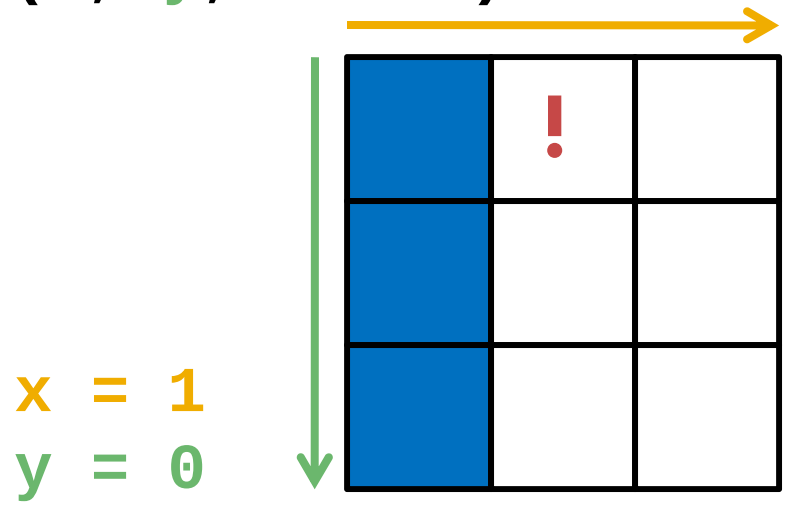


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
→           wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

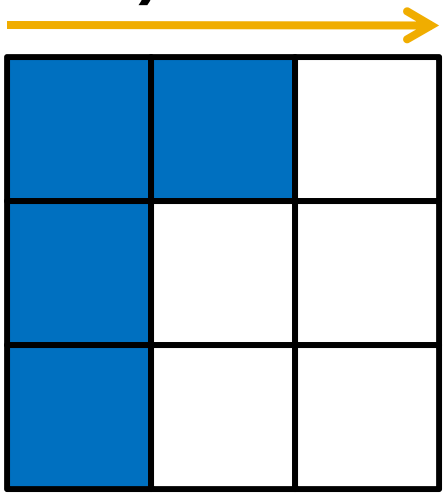


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
→           wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

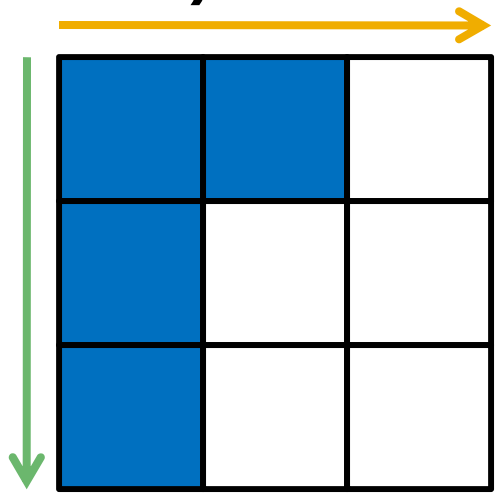


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```



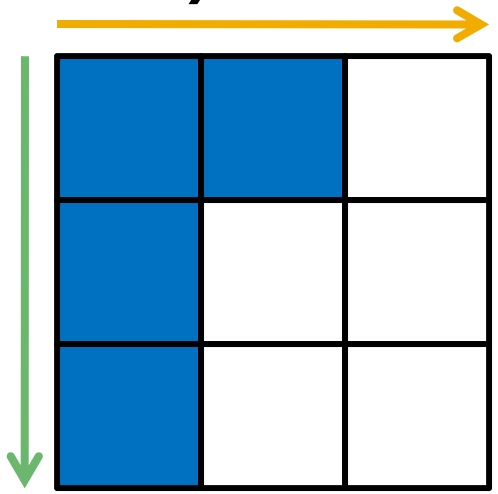
ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

x = 1
y = 1

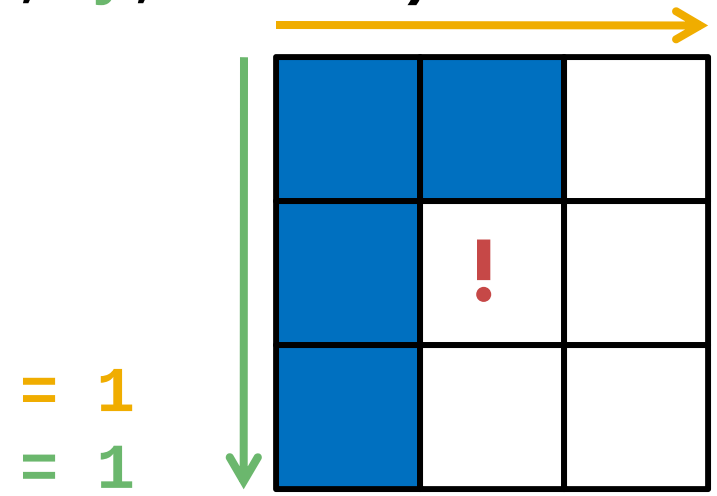


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
→           wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```



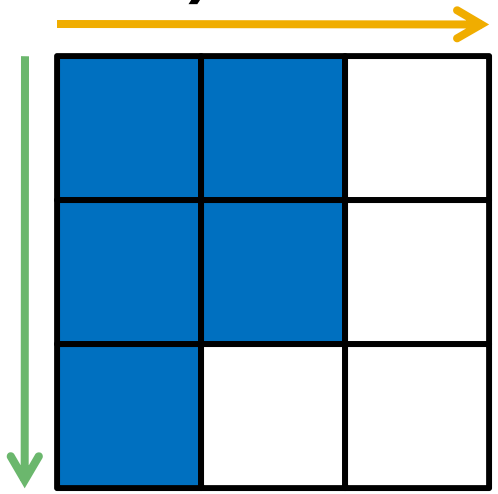
ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
→           wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

x = 1
y = 1

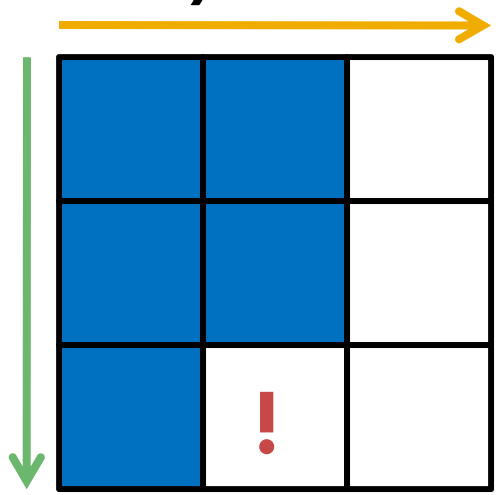


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

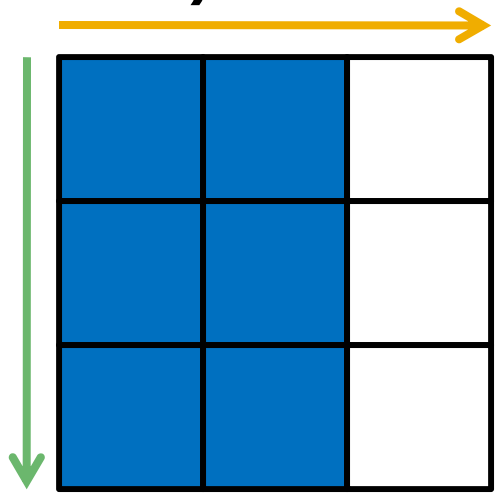


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

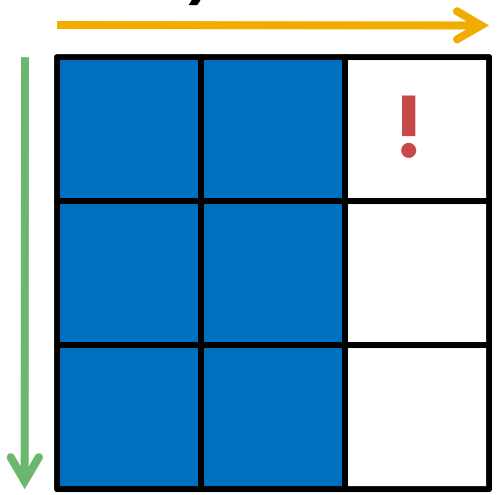


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

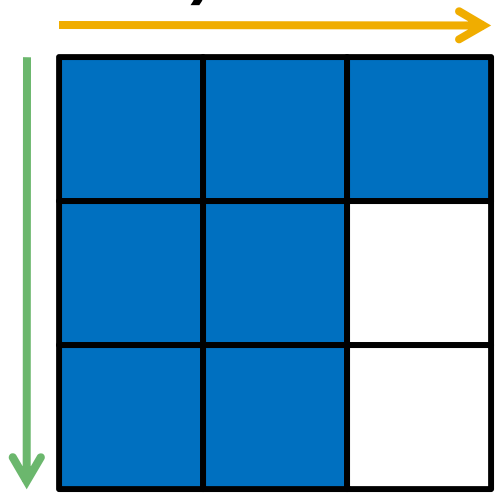


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

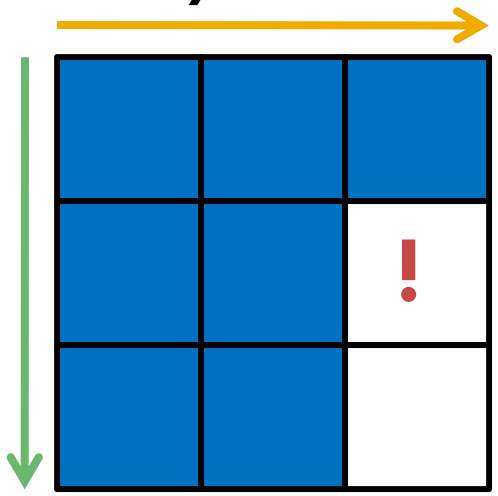


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

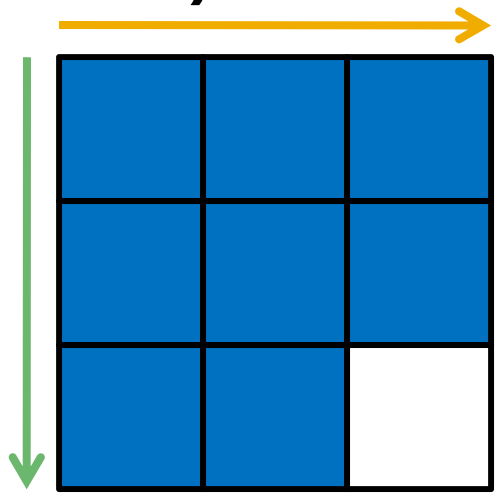


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

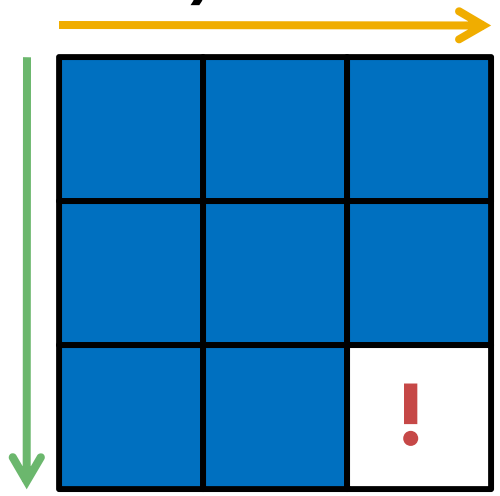


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

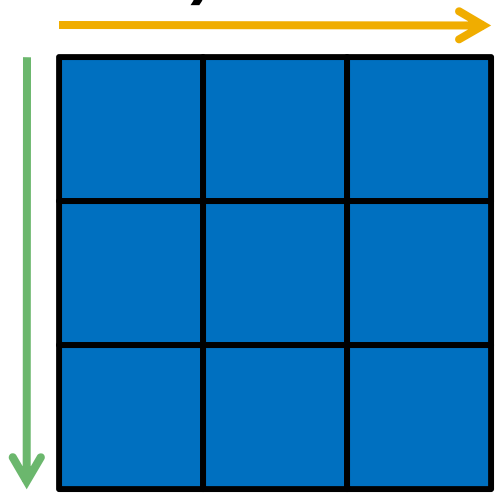


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

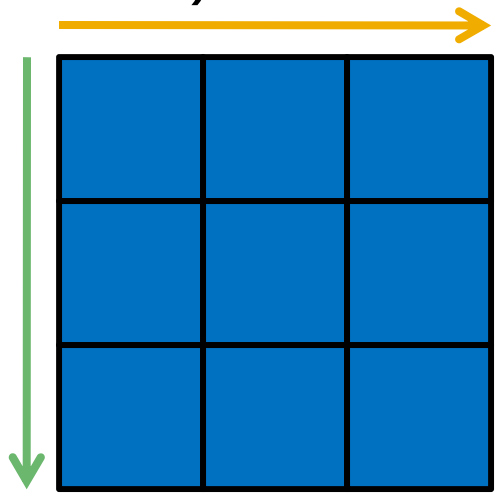


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

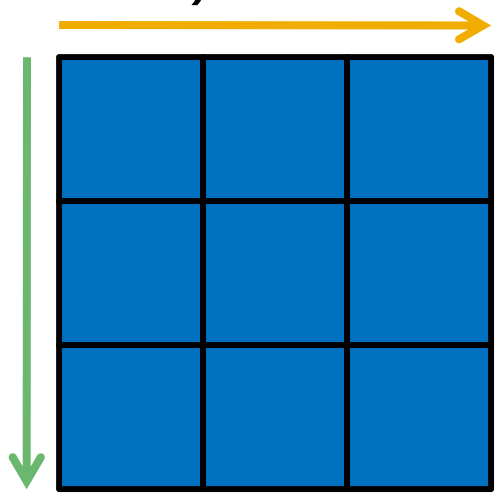


ColorWall Effects

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```



ColorWall Effects

- That was cool! But what if we want to see more than just blue?

ColorWall Effects

- That was cool! But what if we want to see more than just blue?

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

ColorWall Effects

- That was cool! But what if we want to see more than just blue?

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

ColorWall Effects

- That was cool! But what if we want to see more than just blue?

```
def SolidColorTest(wall):
    print "SolidColorTest"

    color = colors["blue"]
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(2)
```

…Let's look at `DictionaryTest`!

ColorWall Effects

```
def DictionaryTest(wall):
    print "SolidColorTest"

   for color in ["black", "white", "gray", "red", …]:
          for x in range(wall.width):
              for y in range(wall.height):
                  wall.set_pixel(x, y, colors[color])
          wall.draw()
          time.sleep(0.5)
```

ColorWall Effects

```
def DictionaryTest(wall):
    print "SolidColorTest"

   for color in ["black", "white", "gray", "red", …]:
          for x in range(wall.width):
              for y in range(wall.height):
                  wall.set_pixel(x, y, colors[color])
          wall.draw()
          time.sleep(0.5)


# A dictionary of hsv values for some common colors.
colors = {"black":(0, 0, 0), "white":(0, 0, 1), "gray":(0, 0,
          0.5), "red":(0, 1, 1), … }
```

ColorWall Effects

```
def DictionaryTest(wall):
    print "SolidColorTest"

   for color in ["black", "white", "gray", "red", …]:
          for x in range(wall.width):
              for y in range(wall.height):
                  wall.set_pixel(x, y, colors[color])
          wall.draw()
          time.sleep(0.5)


# A dictionary of hsv values for some common colors.
colors = {"black":(0, 0, 0), "white":(0, 0, 1), "gray":(0, 0,
          0.5), "red":(0, 1, 1), … }
```

ColorWall Effects

```
def DictionaryTest(wall):
    print "SolidColorTest"

   for color in ["black", "white", "gray", "red", …]:
          for x in range(wall.width):
              for y in range(wall.height):
                  wall.set_pixel(x, y, colors[color])
          wall.draw()
          time.sleep(0.5)
```

```
# A dictionary of hsv values for some common colors.
colors = {"black":(0, 0, 0), "white":(0, 0, 1), "gray":(0, 0,
          0.5), "red":(0, 1, 1), … }
```

colors.keys() = ["black", "white", "gray", "red", …]

ColorWall Effects

```
def DictionaryTest(wall):
    print "SolidColorTest"

   for color in colors.keys():
          for x in range(wall.width):
              for y in range(wall.height):
                  wall.set_pixel(x, y, colors[color])
          wall.draw()
          time.sleep(0.5)
```

```
# A dictionary of hsv values for some common colors.
colors = {"black":(0, 0, 0), "white":(0, 0, 1), "gray":(0, 0,
          0.5), "red":(0, 1, 1), … }
```

colors.keys() = ["black", "white", "gray", "red", …]

ColorWall Effects

- What if we only want some colors?

ColorWall Effects

- What if we only want some colors?
- Let's implement `RainbowTest!`

ColorWall Effects

- What if we only want some colors?
- Let's implement `RainbowTest`!
- Model it off of `DictionaryTest`, but use only some colors

ColorWall Effects

```
def RainbowTest(wall):
    print "RainbowTest"

    # Your code here!!!
```

ColorWall Effects

```
def RainbowTest(wall):
    print "RainbowTest"

    rainbow = ["red", "orange", "yellow", "green",
     "blue", "indigo", "purple"]
    for color in rainbow:
        for x in range(wall.width):
            for y in range(wall.height):
                wall.set_pixel(x, y, colors[color])
        wall.draw()
        time.sleep(0.5)
```

ColorWall Effects

- There are a lot more to explore already!

ColorWall Effects

- There are a lot more to explore already!
- Play around with them and write your own

ColorWall Effects

- There are a lot more to explore already!
- Play around with them and write your own
  - Change the color range in `Twinkle` (in `more_effects.py`)
  - Change the math that controls the spacing in `Checkerboards` (also in `more_effects.py`)
  - Change the scrolling message in `Message` (also in `more_effects.py`)
  - Or… whatever else you'd like to do!