COLORWALL
Boston Python Workshop 2012
**List**

- Create a list
  - `dogs = ['beagle', 'dalmatian', 'corgi', 'golden retriever']`

- How to get an item from the list?
  - `dogs[1] = 'dalmatian'      dogs[-1] = 'golden retriever'`

- Create a list of numbers
  - `num_list1 = [0, 1, 2, 3]`
  - `num_list2 = range(3)`
  - `num_list3 = range(4)`
**Dictionary**

- Dictionary contains a **key** and a **value**

- Create a dictionary
  ```
  ice_cream = {'Jessica': 'green tea', 'Liz': 'peanut brittle',
              'Adam': 'mint chocolate chip'}
  ```

- How to access elements?
  ```
  ice_cream['Jessica']
  ```
# ColorWall

<table>
<thead>
<tr>
<th></th>
<th>(0, 0)</th>
<th>(1, 0)</th>
<th>(2, 0)</th>
<th>(3, 0)</th>
<th>(4, 0)</th>
<th>(5, 0)</th>
<th>(6, 0)</th>
<th>(7, 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 1)</td>
<td>(1, 1)</td>
<td>(2, 1)</td>
<td>(3, 1)</td>
<td>(4, 1)</td>
<td>(5, 1)</td>
<td>(6, 1)</td>
<td>(7, 1)</td>
<td></td>
</tr>
<tr>
<td>(0, 2)</td>
<td>(1, 2)</td>
<td>(2, 2)</td>
<td>(3, 2)</td>
<td>(4, 2)</td>
<td>(5, 2)</td>
<td>(6, 2)</td>
<td>(7, 2)</td>
<td></td>
</tr>
<tr>
<td>(0, 3)</td>
<td>(1, 3)</td>
<td>(2, 3)</td>
<td>(3, 3)</td>
<td>(4, 3)</td>
<td>(5, 3)</td>
<td>(6, 3)</td>
<td>(7, 3)</td>
<td></td>
</tr>
<tr>
<td>(0, 4)</td>
<td>(1, 4)</td>
<td>(2, 4)</td>
<td>(3, 4)</td>
<td>(4, 4)</td>
<td>(5, 4)</td>
<td>(6, 4)</td>
<td>(7, 4)</td>
<td></td>
</tr>
<tr>
<td>(0, 5)</td>
<td>(1, 5)</td>
<td>(2, 5)</td>
<td>(3, 5)</td>
<td>(4, 5)</td>
<td>(5, 5)</td>
<td>(6, 5)</td>
<td>(7, 5)</td>
<td></td>
</tr>
<tr>
<td>(0, 6)</td>
<td>(1, 6)</td>
<td>(2, 6)</td>
<td>(3, 6)</td>
<td>(4, 6)</td>
<td>(5, 6)</td>
<td>(6, 6)</td>
<td>(7, 6)</td>
<td></td>
</tr>
<tr>
<td>(0, 7)</td>
<td>(1, 7)</td>
<td>(2, 7)</td>
<td>(3, 7)</td>
<td>(4, 7)</td>
<td>(5, 7)</td>
<td>(6, 7)</td>
<td>(7, 7)</td>
<td></td>
</tr>
</tbody>
</table>
SOLIDCOLORTEST(wall)

- Pick a color
  - `color = colors["blue"]`
- Set the color
  - `wall.set_pixel(0, 0, color)`
- Draw the wall
  - `wall.draw()`
- Wait!
  - `time.sleep(2)`

<table>
<thead>
<tr>
<th>Block #</th>
<th>(0, 0)</th>
<th>(1, 0)</th>
<th>(2, 0)</th>
<th>(3, 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0, 2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0, 3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1, 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1, 2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1, 3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2, 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2, 2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2, 3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3, 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3, 2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3, 3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Color a Column

<table>
<thead>
<tr>
<th></th>
<th>(0, 0)</th>
<th>(1, 0)</th>
<th>(2, 0)</th>
<th>(3, 0)</th>
<th>(4, 0)</th>
<th>(5, 0)</th>
<th>(6, 0)</th>
<th>(7, 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 1)</td>
<td>(1, 1)</td>
<td>(2, 1)</td>
<td>(3, 1)</td>
<td>(4, 1)</td>
<td>(5, 1)</td>
<td>(6, 1)</td>
<td>(7, 1)</td>
<td></td>
</tr>
<tr>
<td>(0, 2)</td>
<td>(1, 2)</td>
<td>(2, 2)</td>
<td>(3, 2)</td>
<td>(4, 2)</td>
<td>(5, 2)</td>
<td>(6, 2)</td>
<td>(7, 2)</td>
<td></td>
</tr>
<tr>
<td>(0, 3)</td>
<td>(1, 3)</td>
<td>(2, 3)</td>
<td>(3, 3)</td>
<td>(4, 3)</td>
<td>(5, 3)</td>
<td>(6, 3)</td>
<td>(7, 3)</td>
<td></td>
</tr>
<tr>
<td>(0, 4)</td>
<td>(1, 4)</td>
<td>(2, 4)</td>
<td>(3, 4)</td>
<td>(4, 4)</td>
<td>(5, 4)</td>
<td>(6, 4)</td>
<td>(7, 4)</td>
<td></td>
</tr>
<tr>
<td>(0, 5)</td>
<td>(1, 5)</td>
<td>(2, 5)</td>
<td>(3, 5)</td>
<td>(4, 5)</td>
<td>(5, 5)</td>
<td>(6, 5)</td>
<td>(7, 5)</td>
<td></td>
</tr>
<tr>
<td>(0, 6)</td>
<td>(1, 6)</td>
<td>(2, 6)</td>
<td>(3, 6)</td>
<td>(4, 6)</td>
<td>(5, 6)</td>
<td>(6, 6)</td>
<td>(7, 6)</td>
<td></td>
</tr>
<tr>
<td>(0, 7)</td>
<td>(1, 7)</td>
<td>(2, 7)</td>
<td>(3, 7)</td>
<td>(4, 7)</td>
<td>(5, 7)</td>
<td>(6, 7)</td>
<td>(7, 7)</td>
<td></td>
</tr>
</tbody>
</table>
COLOR A COLUMN

One idea

- `wall.set_pixel(0, 0, color)`
- `wall.set_pixel(0, 1, color)`
- `wall.set_pixel(0, 2, color)`
- `wall.set_pixel(0, 3, color)`
- `wall.set_pixel(0, 4, color)`
- `wall.set_pixel(0, 5, color)`
- `wall.set_pixel(0, 6, color)`
- `wall.set_pixel(0, 7, color)`
For Loop!

for y in range(wall.height):
    wall.set_pixel(0, y, color)

- wall.set_pixel(0, 0, color)
- wall.set_pixel(0, 1, color)
- wall.set_pixel(0, 2, color)
- wall.set_pixel(0, 3, color)
- wall.set_pixel(0, 4, color)
- wall.set_pixel(0, 5, color)
- wall.set_pixel(0, 6, color)
- wall.set_pixel(0, 7, color)
**For Loop!**

```python
for y in range(wall.height):
    wall.set_pixel(0, y, color)
```

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0)</td>
<td>(1, 0)</td>
<td>(2, 0)</td>
<td>(3, 0)</td>
<td>(4, 0)</td>
<td>(5, 0)</td>
<td>(6, 0)</td>
</tr>
<tr>
<td>(0, 1)</td>
<td>(1, 1)</td>
<td>(2, 1)</td>
<td>(3, 1)</td>
<td>(4, 1)</td>
<td>(5, 1)</td>
<td>(6, 1)</td>
</tr>
<tr>
<td>(0, 2)</td>
<td>(1, 2)</td>
<td>(2, 2)</td>
<td>(3, 2)</td>
<td>(4, 2)</td>
<td>(5, 2)</td>
<td>(6, 2)</td>
</tr>
<tr>
<td>(0, 3)</td>
<td>(1, 3)</td>
<td>(2, 3)</td>
<td>(3, 3)</td>
<td>(4, 3)</td>
<td>(5, 3)</td>
<td>(6, 3)</td>
</tr>
<tr>
<td>(0, 4)</td>
<td>(1, 4)</td>
<td>(2, 4)</td>
<td>(3, 4)</td>
<td>(4, 4)</td>
<td>(5, 4)</td>
<td>(6, 4)</td>
</tr>
<tr>
<td>(0, 5)</td>
<td>(1, 5)</td>
<td>(2, 5)</td>
<td>(3, 5)</td>
<td>(4, 5)</td>
<td>(5, 5)</td>
<td>(6, 5)</td>
</tr>
<tr>
<td>(0, 6)</td>
<td>(1, 6)</td>
<td>(2, 6)</td>
<td>(3, 6)</td>
<td>(4, 6)</td>
<td>(5, 6)</td>
<td>(6, 6)</td>
</tr>
<tr>
<td>(0, 7)</td>
<td>(1, 7)</td>
<td>(2, 7)</td>
<td>(3, 7)</td>
<td>(4, 7)</td>
<td>(5, 7)</td>
<td>(6, 7)</td>
</tr>
</tbody>
</table>
NESTED LOOPS

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)
**Exercise**

- Implement RainbowTest(wall) to display the colors of the rainbow
  - Red
  - Orange
  - Yellow
  - Green
  - Blue
  - Purple
function RAINBOWTEST(wall):

    rainbow = ['red', 'orange', 'yellow', 'green', 'blue', '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)
RAINBOWTEST(wall) WITH COLUMNS

wall.clear()

rainbow = [ 'red', 'orange', 'yellow', 'lime', 'green', 'blue', 'purple', 'pink' ]

for x in range(wall.width):
    for y in range(wall.height):
        wall.set_pixel(x, y, colors[ rainbow[x] ])

wall.draw()

time.sleep(0.2)
CHECKERBOARDS\textit{(WALL)}
CHECKERBOARDS(WALL)

for i in range(10):
    for x in range(wall.width):
        for y in range(wall.height):
            if (x + y + i) % 2 == 0:
                wall.set_pixel(x, y, colors["black"])
            else:
                wall.set_pixel(x, y, colors["yellow"])
    wall.draw()
    time.sleep(0.5)
Create a tuple
- `american_flag_colors = ('red', 'white', 'blue')`

How to get an item from the tuple?
- `american_flag_colors[0] = 'red'`

Different from list?
- Cannot add or remove elements from a tuple
- Tuples are faster than lists
- Tuples are for data that does not need to be changed
EFFECTS.PY

- \[ \text{colors} = \{\text{‘black’} : (0, 0, 0), \text{‘white’} : (0, 0, 1)\ldots\} \]

- HSV values for colors
  - Hue, Saturation, Value

- How to get a color from dictionary colors?
  - \( \text{colors[‘white’]} \) equivalent to \( (0, 0, 1) \)
HueTest(wall)

http://www.yafla.com/yaflaColor/ColorRGBHSL.aspx

hue = 0
while [condition]:
    color = (hue, 1, 1)
    [color in each cell using for loops]
    [update!]
```
HueTest(wall)

hue = 0
while hue < 1:  # condition
    color = (hue, 1, 1)
    for x in range(wall.width):
        for y in range(wall.height):
            wall.set_pixel(x, y, color)
    wall.draw()
    time.sleep(0.05)
    hue = hue + 0.01  # update!
```
CREATE YOUR OWN!

Try out different things:
For example, what happens when you change the saturation or the value?
MESSAGE (WALL)

KATHERINE
Create your name list

name = [
    
    *
    *
    ***
    ****
    *
    *
    ***
    ****
    *
    *
    ***
    ****
    *
    *
    ***
    ****
    *
    *
    ***
    ****
    *
    *
    ***
    ****
    *
    *
    ***
    ****
    *
    *
    ***
    ****
    *
    *
    ***
]
MESSAGE(WALL)

• Let’s describe the algorithm in words:

For each 8x8 window
  • We want to print out the stars in a different color
MESSAGE(WALL)

KATHERINE

\( \text{col} = 0 \)

8x8 window
MESSAGE (WALL)

KATHERINE

col = 1

8x8 window
What is the range of `col`?
What is the range of \texttt{col}?

\texttt{range(29)} = [0, 1, 2, 3, ..., 28]
MESSAGE(wall)

# for each 8x8 window
for col in range(29):
    # clear the wall
    wall.clear()

    # for each block in that window
    for x in range(wall.width):
        for y in range(wall.height):
            ...

MESSAGE(wall)

# for each block in that window
for x in range(wall.width):
    for y in range(wall.height):
        # look up the dot in your name list
dot = name[y][x+col]
MESSAGE(wall)

# for each block in that window
for x in range(wall.width):
    for y in range(wall.height):
        # look up the dot in your name list
        dot = name[y][x+col]
MESSAGE(wall)

# for each block in that window
for x in range(wall.width):
    for y in range(wall.height):

        # look up the dot in your name list
        dot = name[y][x+col]

        # if the dot is a *, then color it!
        if dot == '*':
            wall.set_pixel(x, y, (0.333, 1, 1))
for col in range(29):
    wall.clear()
    for x in range(wall.width):
        for y in range(wall.height):
            dot = name[y][x+col]
            if dot == '*':
                wall.set_pixel(x, y, (0.333, 1, 1))
    wall.draw()
    time.sleep(0.07)