COLORWALL
Boston Python Workshop 2011
COMMAND LINE

- On your Desktop
  - Double click on Python (command line)
THE PYTHON CALCULATOR!

- Try it out
  - 2 + 2
  - 1.5 + 2.25
  - 4 – 2
  - 100 – 0.5
  - 0 – 2
  - 2 * 3
  - 4 / 2
  - 1 / 2
  - 1.0 / 2
  - \( \frac{3}{4} + \frac{1}{4} = ? \)

- Numeric types
  - type(1) int
  - type(1.0) float

- Variables
  - type(2)
  - x = 2
  - x
  - type(x)
  - x / 3
Create a list
- `dogs = ['beagle', 'dalmatian', 'corgi', 'golden_retriever']`

How long is this list?
- `len(dogs)`

How to get an item from the list?
LIST

- Create a list
  - dogs = ['beagle', 'dalmatian', 'corgi', 'golden_retriever']

- How long is this list?
  - len(dogs)

- How to get an item from the list?
  - dogs[1] = ?
  - dogs[-1] = ?

- Create a list of numbers
  - num1 = [0, 1, 2, 3]
  - num2 = range(3)
  - num3 = range(4)
**TUPLE**

- Create a tuple
  - `bls_alum = ('Katherine', 'Kim', 'Sarah')`
- How long is this tuple?
  - `len(bls_alum)`
- How to get an item from the tuple?
  - `bls_alum[1] = ?`
TUPLE

Create a tuple

- `bls_alum = ('Katherine', 'Kim', 'Sarah')`

How long is this tuple?

- `len(bls_alum)`

How to get an item from the tuple?

- `bls_alum[1] = ?`

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
Dictionary

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

- Create a dictionary
  - `ice_cream = {'Katherine' : 'mint_choco_chip', 'Ita' : 'b&j_phish_food', 'Kim' : 'chocolate'}`

- How to access elements?
  - `ice_cream['Ita']`

- Why is this useful?
EFFECTS.PY

- **To edit your code:**
  - Go to Artemis → Week 3 → ColorWall
  - Right click on effects.py → choose Edit with Notepad++

- **Change settings on Notepad++:**
  - Click on Settings → choose Preferences…
  - Click on MISC. → uncheck Auto-indent

- **To run your code:**
  - Go back to Artemis → Week 3 → ColorWall
  - Double click on run.py
EFFECTS.PY

- colors = {'black': (0, 0, 0), 'white': (0, 0, 1)...

- HSV values for colors
  - Hue, Saturation, Value
```python
colors = {'black': (0, 0, 0), 'white': (0, 0, 1), ...}
```

**HSV values for colors**
- Hue, Saturation, Value

**How to get a color from dictionary colors?**
- `colors['white']` equivalent to `(0, 0, 1)`
**COLOR THE WALL**

<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>
**DRAW** **WALL**(wall)

- Clear the wall!
  - wall.clear()
- Set the color!
  - wall.set_pixel(0, 0, colors['red'])
- Draw the wall!
  - wall.draw()

<table>
<thead>
<tr>
<th>Block #</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0)</td>
<td>Red</td>
</tr>
<tr>
<td>(0, 1)</td>
<td></td>
</tr>
<tr>
<td>(0, 2)</td>
<td></td>
</tr>
<tr>
<td>(0, 3)</td>
<td></td>
</tr>
<tr>
<td>(1, 0)</td>
<td></td>
</tr>
<tr>
<td>(1, 1)</td>
<td></td>
</tr>
<tr>
<td>(1, 2)</td>
<td></td>
</tr>
<tr>
<td>(1, 3)</td>
<td></td>
</tr>
<tr>
<td>(2, 0)</td>
<td></td>
</tr>
<tr>
<td>(2, 1)</td>
<td></td>
</tr>
<tr>
<td>(2, 2)</td>
<td></td>
</tr>
<tr>
<td>(2, 3)</td>
<td></td>
</tr>
<tr>
<td>(3, 0)</td>
<td></td>
</tr>
<tr>
<td>(3, 1)</td>
<td></td>
</tr>
<tr>
<td>(3, 2)</td>
<td></td>
</tr>
<tr>
<td>(3, 3)</td>
<td></td>
</tr>
</tbody>
</table>
**DrawWall(wall)**

- Clear the wall!
  - `wall.clear()`
- Set the color!
  - `wall.set_pixel(0, 0, colors['red'])`
- Draw the wall!
  - `wall.draw()`

- **Wait!**
  - `time.sleep(2)`
**Color a Row**

<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 ROW

One idea

- `wall.set_pixel(0, 0, colors['red'])`
- `wall.set_pixel(1, 0, colors['red'])`
- `wall.set_pixel(2, 0, colors['red'])`
- `wall.set_pixel(3, 0, colors['red'])`
- `wall.set_pixel(4, 0, colors['red'])`
- `wall.set_pixel(5, 0, colors['red'])`
- `wall.set_pixel(6, 0, colors['red'])`
- `wall.set_pixel(7, 0, colors['red'])`
**FOR LOOP!**

for x in range(8):
    wall.set_pixel(x, 0, colors['red'])

- wall.set_pixel(0, 0, colors['red'])
- wall.set_pixel(1, 0, colors['red'])
- wall.set_pixel(2, 0, colors['red'])
- wall.set_pixel(3, 0, colors['red'])
- wall.set_pixel(4, 0, colors['red'])
- wall.set_pixel(5, 0, colors['red'])
- wall.set_pixel(6, 0, colors['red'])
- wall.set_pixel(7, 0, colors['red'])
FOR LOOP!

for x in range(8):
    wall.set_pixel(x, 0, colors[‘red’])
FOR LOOP!

for x in range(8):
    wall.set_pixel(x, 0, colors['red'])

Spacing matters!
2 or 4 spaces
COLOR MORE ROWS

<table>
<thead>
<tr>
<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>
</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>
</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>
</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>
</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>
</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>
</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>
</tr>
</tbody>
</table>
COLOR MORE ROWS

One idea

- for x in range(8):
  wall.set_pixel(x, 0, colors['red'])
- for x in range(8):
  wall.set_pixel(x, 1, colors['red'])
- for x in range(8):
  wall.set_pixel(x, 2, colors['red'])
- for x in range(8):
  wall.set_pixel(x, 3, colors['red'])
- ...

```python
for x in range(8):
    wall.set_pixel(x, 0, colors['red'])
for x in range(8):
    wall.set_pixel(x, 1, colors['red'])
for x in range(8):
    wall.set_pixel(x, 2, colors['red'])
for x in range(8):
    wall.set_pixel(x, 3, colors['red'])
...`
```
CHALLENGE: RAINBOW(WALL)

- Make your ColorWall show the colors of the rainbow!
- Red
- Orange
- Yellow
- Green
- Blue
- Purple
RAINBOW

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

for color in rainbow:
    for x in range(8):
        for y in range(8):
            wall.set_pixel(x, y, color)

    wall.draw()

    time.sleep(0.5)
**BOOLEAN**

- True or False

- $0 == 0$
- $0 == 1$
- `type(0==0)`

- `if (0==0):
  print ‘Right! 0 = 0’
else:
  print ‘Wrong! 0 != 0’`
**WHILE LOOP!**

```python
while 0 < 1:
    print 'Right! 0 < 1'
```

INFINITE WHILE LOOPS ARE (usually) BAD!

Let’s fix it!

```python
x = 0
while x < 1:
    print 'Right!', x, '< 1'
    x = x + 0.1
```

# condition
# update!
FANCY RAINBOW (WALL)

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

HUE = RAINBOW!

hue = 0
while [condition]:
    color = (hue, 1, 1)
    [color in each cell using for loops]

    [update!]
FANCYRAINBOW(WALL)

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

HUE = RAINBOW!

hue = 0
while [condition]:
    color = (hue, 1, 1)
    [color in each cell using for loops]

[update!]

x = 0
while x < 1:
    print ‘Right!’
    x = x + 0.1
FANCY RAINBOW (wall)

hue = 0
while hue < 1:  # condition
    color = (hue, 1, 1)
    for x in range(8):
        for y in range(8):
            wall.set_pixel(x, y, color)

    wall.draw()
    time.sleep(0.05)
    hue = hue + 0.01  # update!
CHALLENGE: MY EFFECT (WALL)

Create your own effect!

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

KATHERINE
**PrintName(Wall)**

- Create your name list

```python
name = [
    ,
    ,
    ,
    ,
    ,
    ,
]
```

- 8 rows
- 36 columns
PRINTNAME(wall)

Let’s describe the algorithm in words:

- For each 8x8 window
  - We want to print out the dots in a different color
PRINT_NAME(WALL)

col = 0

8x8 window
PRINTNAME(WALL)

KATHERINE

\[ \text{col} = 1 \]

8x8 window
PRINTNAME(wall)

col = 2

8x8 window
PRINT_NAME(WALL)

col = 3

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

\text{range}(29) = [0, 1, 2, 3, \ldots, 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]
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))
MESSAGE(wall)

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)