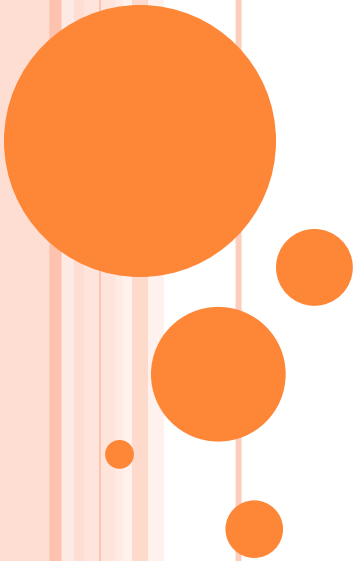


ColorWall

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
- ¾ + ¼ = ?

- Numeric types
  - type(1) int
  - type(1.0) float
- Variables
  - type(2)
  - x = 2
  - x
  - type(x)
  - x / 3

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] = ?

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?

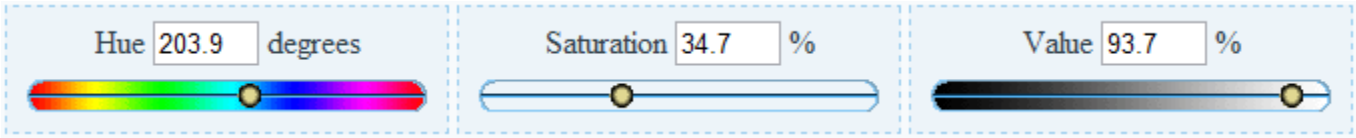
EFFECTS.PY

- 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)…}

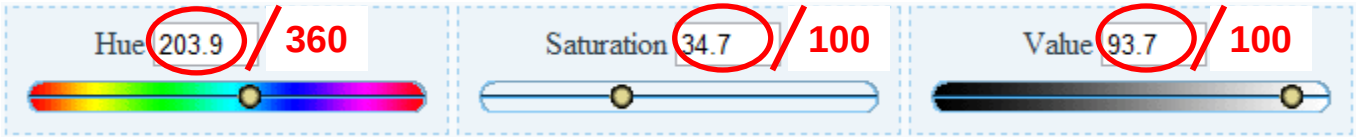
- HSV values for colors
  - Hue, Saturation, Value
  - http://www.yafla.com/yaflaColor/ColorRGBHSL.aspx



EFFECTS.PY

- colors = {‘black’ : (0, 0, 0), ‘white’ : (0, 0, 1)…}

- HSV values for colors
  - Hue, Saturation, Value
  - http://www.yafla.com/yaflaColor/ColorRGBHSL.aspx



- How to get a color from dictionary colors?
  - colors[‘white’] equivalent to (0, 0, 1)

Color the Wall

(0, 0)	(1, 0)	(2, 0)	(3, 0)	(4, 0)	(5, 0)	(6, 0)	(7, 0)
(0, 1)	(1, 1)	(2, 1)	(3, 1)	(4, 1)	(5, 1)	(6, 1)	(7, 1)
(0, 2)	(1, 2)	(2, 2)	(3, 2)	(4, 2)	(5, 2)	(6, 2)	(7, 2)
(0, 3)	(1, 3)	(2, 3)	(3, 3)	(4, 3)	(5, 3)	(6, 3)	(7, 3)
(0, 4)	(1, 4)	(2, 4)	(3, 4)	(4, 4)	(5, 4)	(6, 4)	(7, 4)
(0, 5)	(1, 5)	(2, 5)	(3, 5)	(4, 5)	(5, 5)	(6, 5)	(7, 5)
(0, 6)	(1, 6)	(2, 6)	(3, 6)	(4, 6)	(5, 6)	(6, 6)	(7, 6)
(0, 7)	(1, 7)	(2, 7)	(3, 7)	(4, 7)	(5, 7)	(6, 7)	(7, 7)

DrawWall(wall)

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

Block #

(0, 0)	(1, 0)	(2, 0)	(3, 0)
(0, 1)	(1, 1)	(2, 1)	(3, 1)
(0, 2)	(1, 2)	(2, 2)	(3, 2)
(0, 3)	(1, 3)	(2, 3)	(3, 3)

DrawWall(wall)

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

Block #

(0, 0)	**(1, 0)**	**(2, 0)**	**(3, 0)**
(0, 1)	**(1, 1)**	**(2, 1)**	**(3, 1)**
(0, 2)	**(1, 2)**	**(2, 2)**	**(3, 2)**
(0, 3)	**(1, 3)**	**(2, 3)**	**(3, 3)**

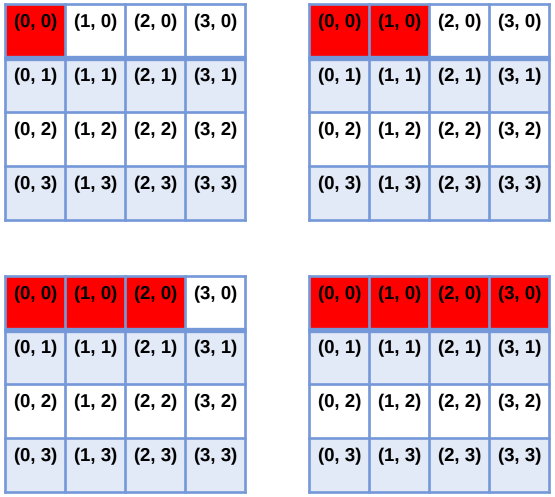
- Wait!
  - time.sleep(2)

Color a Row

(0, 0)	(1, 0)	(2, 0)	(3, 0)	(4, 0)	(5, 0)	(6, 0)	(7, 0)
(0, 1)	(1, 1)	(2, 1)	(3, 1)	(4, 1)	(5, 1)	(6, 1)	(7, 1)
(0, 2)	(1, 2)	(2, 2)	(3, 2)	(4, 2)	(5, 2)	(6, 2)	(7, 2)
(0, 3)	(1, 3)	(2, 3)	(3, 3)	(4, 3)	(5, 3)	(6, 3)	(7, 3)
(0, 4)	(1, 4)	(2, 4)	(3, 4)	(4, 4)	(5, 4)	(6, 4)	(7, 4)
(0, 5)	(1, 5)	(2, 5)	(3, 5)	(4, 5)	(5, 5)	(6, 5)	(7, 5)
(0, 6)	(1, 6)	(2, 6)	(3, 6)	(4, 6)	(5, 6)	(6, 6)	(7, 6)
(0, 7)	(1, 7)	(2, 7)	(3, 7)	(4, 7)	(5, 7)	(6, 7)	(7, 7)

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!

[0, 1, 2, 3, 4, 5, 6, 7]

```
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!

[0, 1, 2, 3, 4, 5, 6, 7]

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

(0, 0)	(1, 0)	(2, 0)	(3, 0)	(4, 0)	(5, 0)	(6, 0)	(7, 0)
(0, 1)	(1, 1)	(2, 1)	(3, 1)	(4, 1)	(5, 1)	(6, 1)	(7, 1)
(0, 2)	(1, 2)	(2, 2)	(3, 2)	(4, 2)	(5, 2)	(6, 2)	(7, 2)
(0, 3)	(1, 3)	(2, 3)	(3, 3)	(4, 3)	(5, 3)	(6, 3)	(7, 3)
(0, 4)	(1, 4)	(2, 4)	(3, 4)	(4, 4)	(5, 4)	(6, 4)	(7, 4)
(0, 5)	(1, 5)	(2, 5)	(3, 5)	(4, 5)	(5, 5)	(6, 5)	(7, 5)
(0, 6)	(1, 6)	(2, 6)	(3, 6)	(4, 6)	(5, 6)	(6, 6)	(7, 6)
(0, 7)	(1, 7)	(2, 7)	(3, 7)	(4, 7)	(5, 7)	(6, 7)	(7, 7)

For Loop!

[0, 1, 2, 3, 4, 5, 6, 7]

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

Spacing matters!

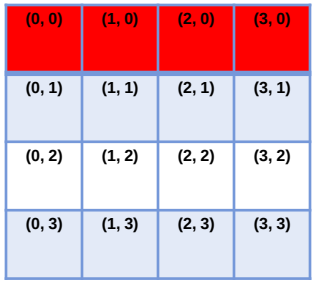
2 or 4 spaces

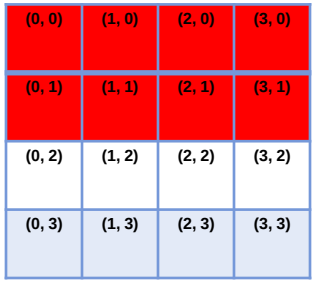
Color more Rows

(0, 0)	(1, 0)	(2, 0)	(3, 0)	(4, 0)	(5, 0)	(6, 0)	(7, 0)
(0, 1)	(1, 1)	(2, 1)	(3, 1)	(4, 1)	(5, 1)	(6, 1)	(7, 1)
(0, 2)	(1, 2)	(2, 2)	(3, 2)	(4, 2)	(5, 2)	(6, 2)	(7, 2)
(0, 3)	(1, 3)	(2, 3)	(3, 3)	(4, 3)	(5, 3)	(6, 3)	(7, 3)
(0, 4)	(1, 4)	(2, 4)	(3, 4)	(4, 4)	(5, 4)	(6, 4)	(7, 4)
(0, 5)	(1, 5)	(2, 5)	(3, 5)	(4, 5)	(5, 5)	(6, 5)	(7, 5)
(0, 6)	(1, 6)	(2, 6)	(3, 6)	(4, 6)	(5, 6)	(6, 6)	(7, 6)
(0, 7)	(1, 7)	(2, 7)	(3, 7)	(4, 7)	(5, 7)	(6, 7)	(7, 7)

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’])
  - …





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!

```
while 0 < 1:
    print ‘Right! 0 < 1’
```

INFINITE WHILE LOOPS ARE (usually) BAD!

Let’s fix it!

```
x = 0
while x < 1:                    # condition
    print ‘Right!’, x, ‘< 1’
    x = x + 0.1                 # 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!]
```

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
```

FancyRainbow(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!
```

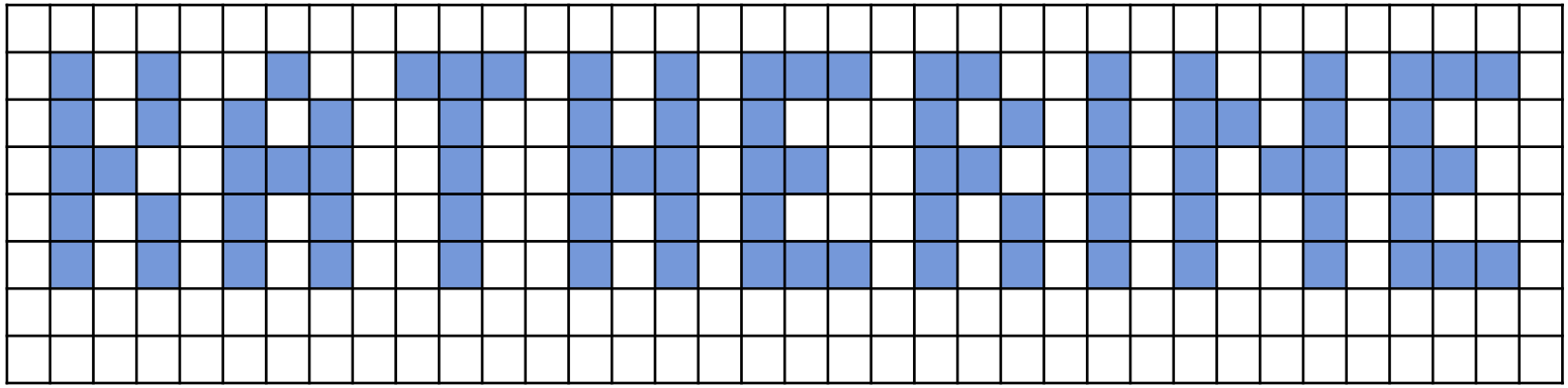
Challenge: MyEffect(wall)

Create your own effect!

Try out different things:

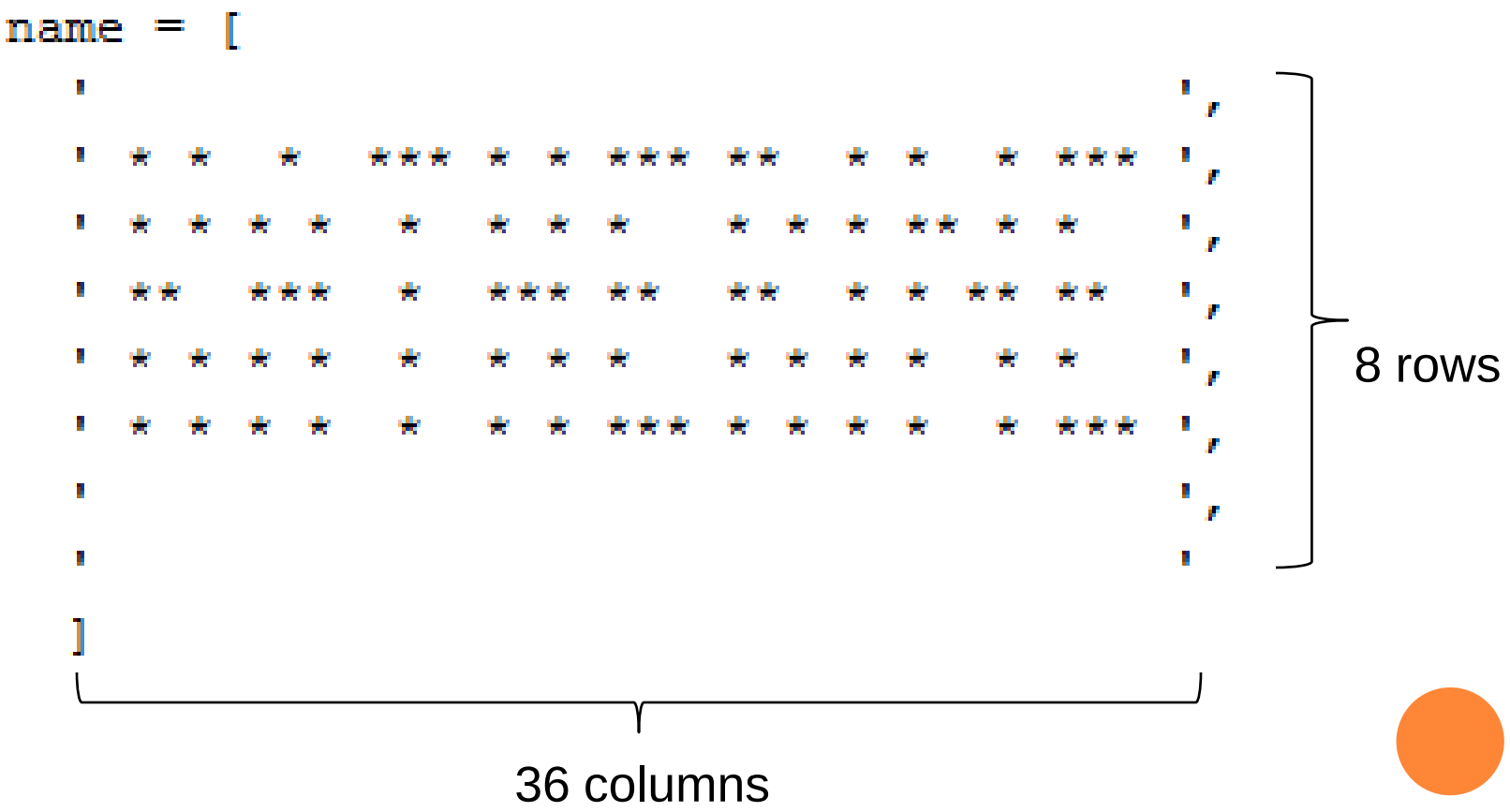
For example, what happens when you change the saturation or the value?

PrintName(wall)



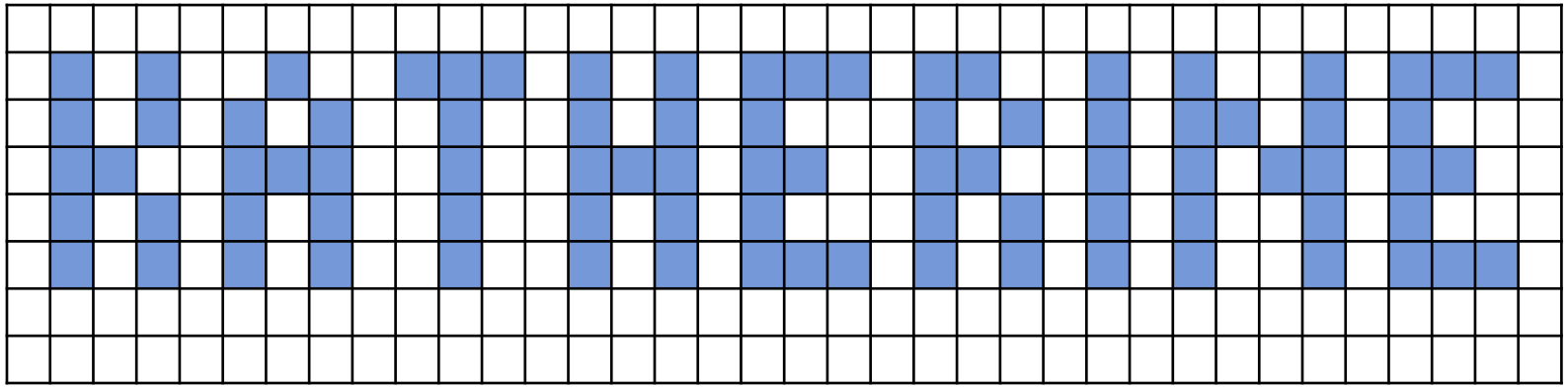
PrintName(wall)

- Create your name list



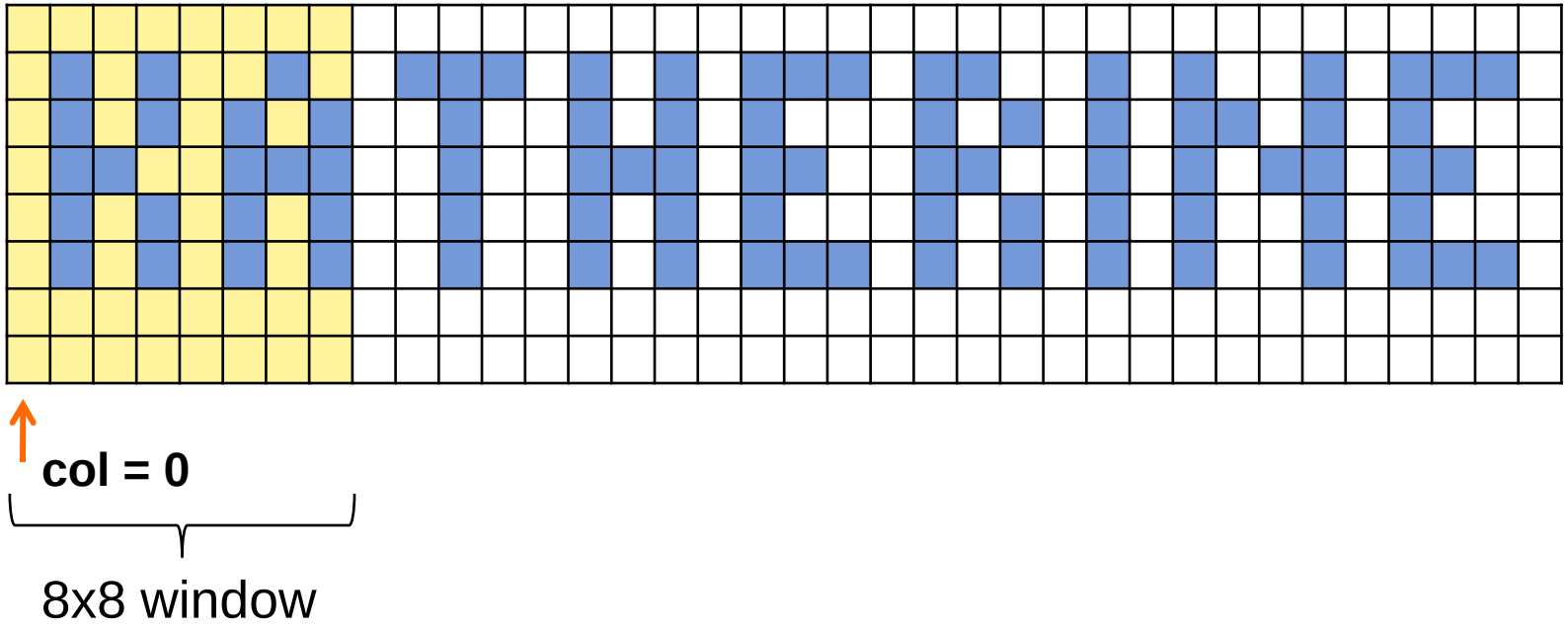
PrintName(wall)

- Let's describe the algorithm in words:

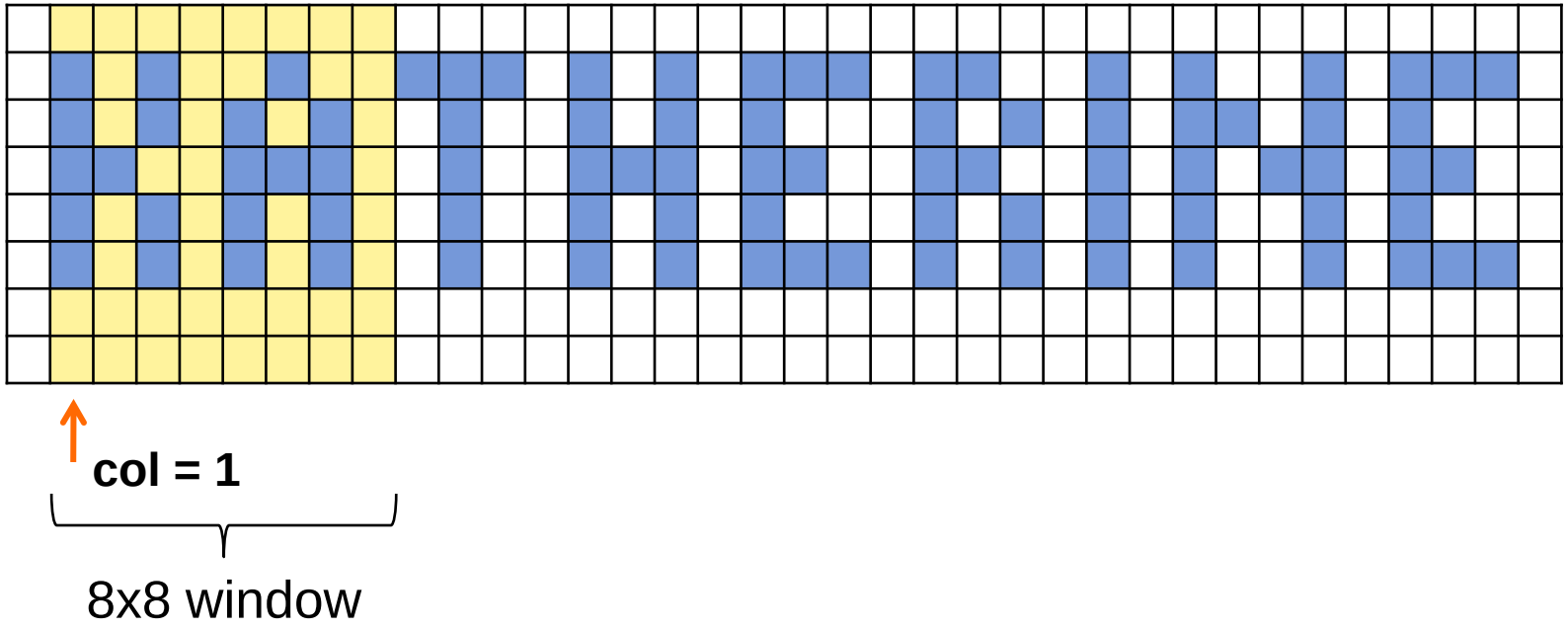


  - For each 8x8 window
    - We want to print out the dots in a different color

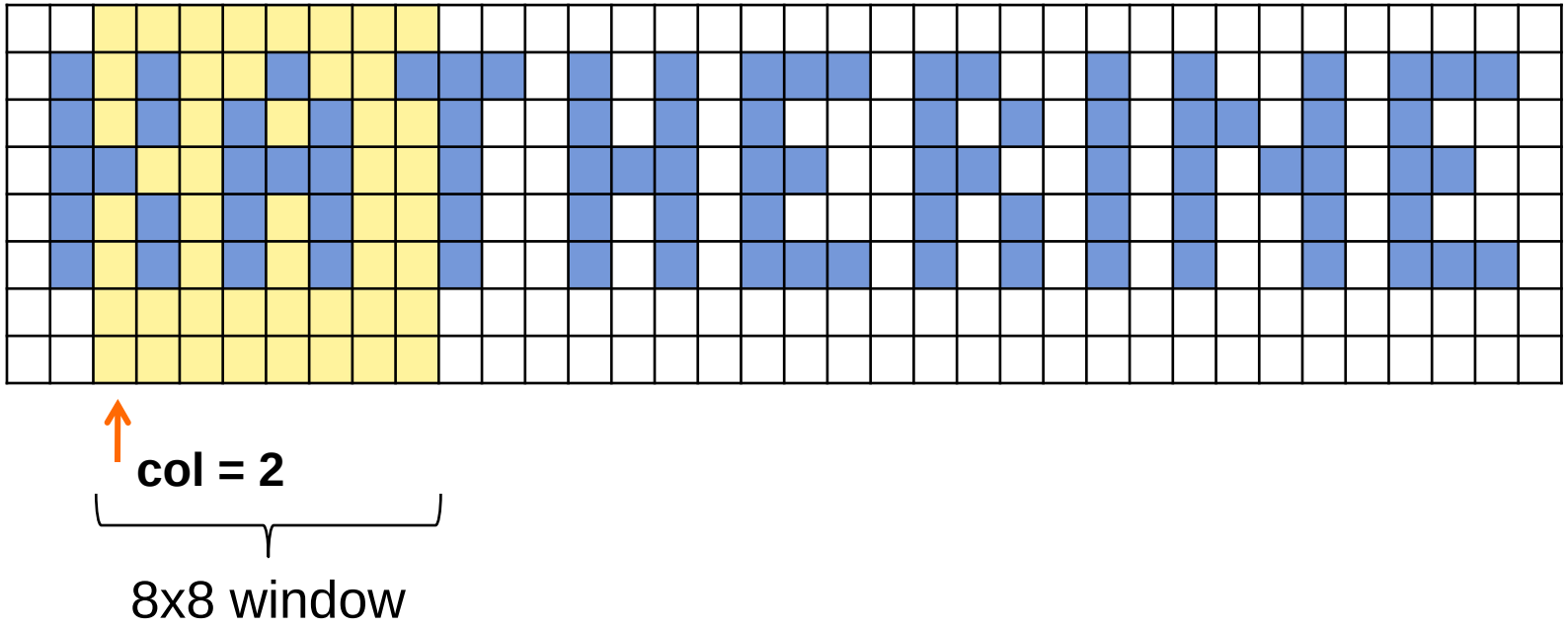
PrintName(wall)



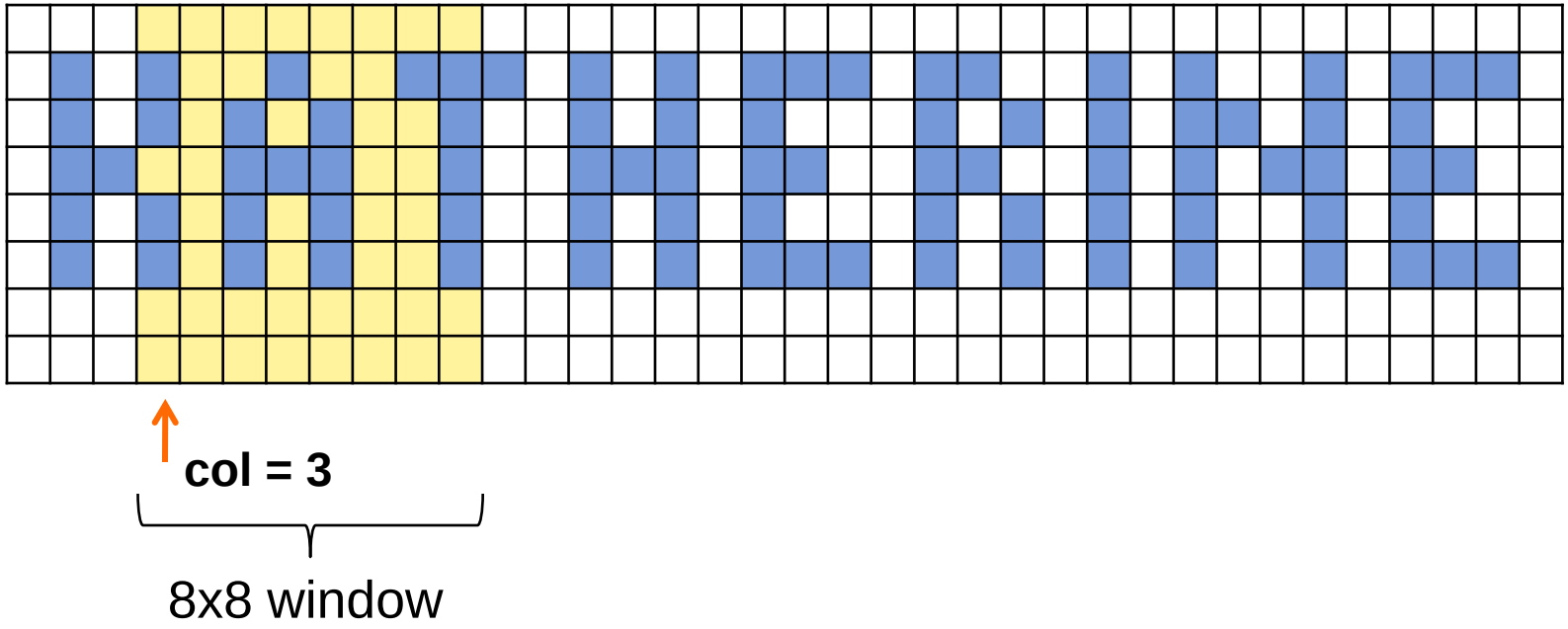
PrintName(wall)



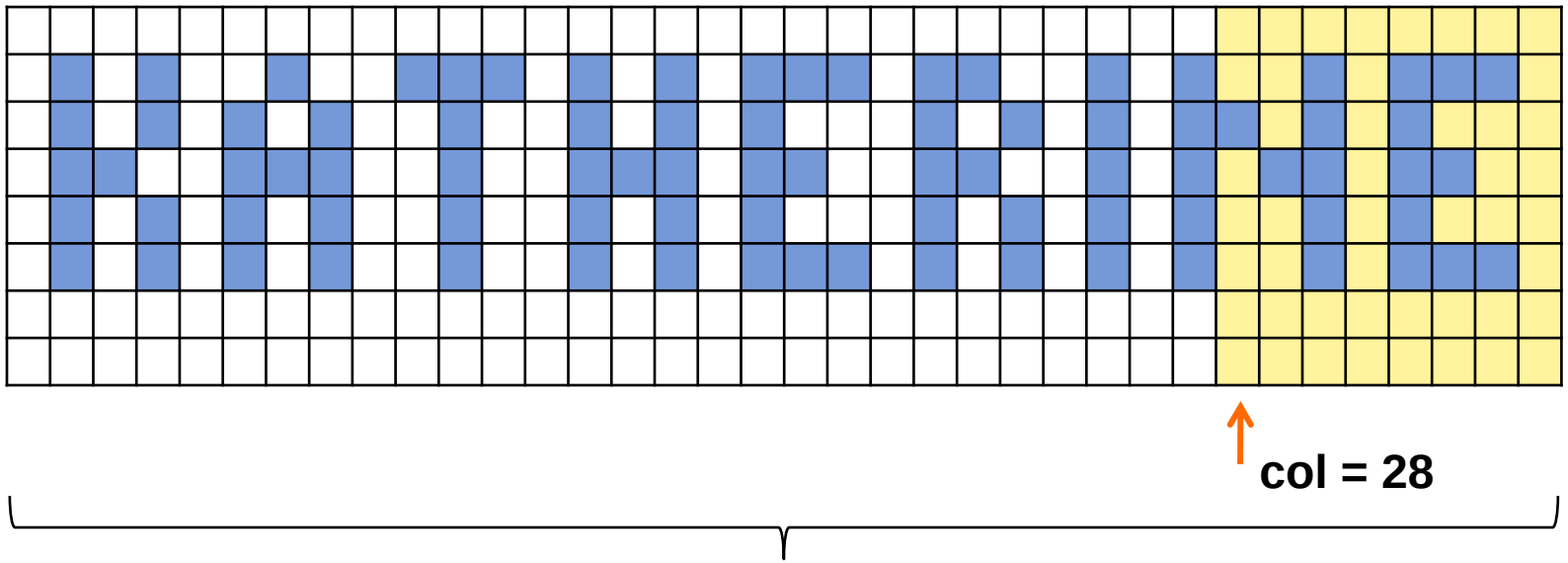
PrintName(wall)



PrintName(wall)



PRINTNAME(WALL)



What is the range of **col**?

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 ]
```

(0, 0)	(1, 0)	(2, 0)	(3, 0)
(0, 1)	(1, 1)	(2, 1)	(3, 1)
(0, 2)	(1, 2)	(2, 2)	(3, 2)
(0, 3)	(1, 3)	(2, 3)	(3, 3)

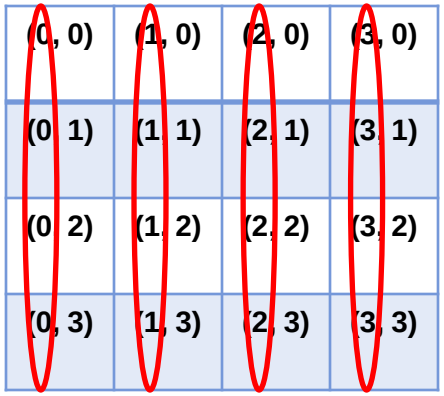
```
name = [
  '                                                ',
  ' * *  *  *** * * *** **  * *  * *** ',
  ' * * * *  *  * * *   * * * ** * *   ',
  ' **  ***  *  *** **  **  * * ** **  ',
  ' * * * *  *  * * *   * * * *  * *   ',
  ' * * * *  *  * * *** * * * *  * *** ',
  '                                                ',
  '                                                '
  ]
```

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 ]
```

x

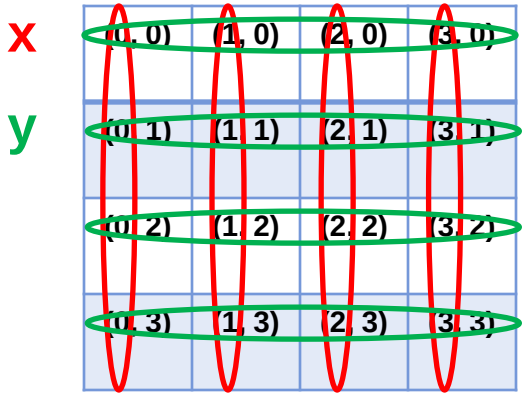



```
name = [
  '                                                ',
  ' * *  *  *** * * *** **  * *  * *** ',
  ' * * * *  *  * * *   * * * ** * *   ',
  ' **  ***  *  *** **  **  * * ** **  ',
  ' * * * *  *  * * *   * * * *  * *   ',
  ' * * * *  *  * * *** * * * *  * *** ',
  '                                                ',
  '                                                ',
  ]
```

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 ]
```




```
name = [
  '                                                ',
  ' * *  *  *** * * *** **  * *  * *** ',
  ' * * * *  *  * * *   * * * ** * *   ',
  ' **  ***  *  *** **  **  * * ** **  ',
  ' * * * *  *  * * *   * * * *  * *   ',
  ' * * * *  *  * * *** * * * *  * *** ',
  '                                                ',
  '                                                '
  ]
```

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)
```