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]	[0, 1, 2, 3]
num_list2 = range(3)	[0, 1, 2]
num_list3 = range(4)	[0, 1, 2, 3]

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

(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)

SolidColorTest(wall)

- Pick a color
  - color = colors[“blue”]
- Set the color
  - wall.set_pixel(0, 0, color)

Block #

- Draw the wall
  - wall.draw()
- Wait!
  - time.sleep(2)

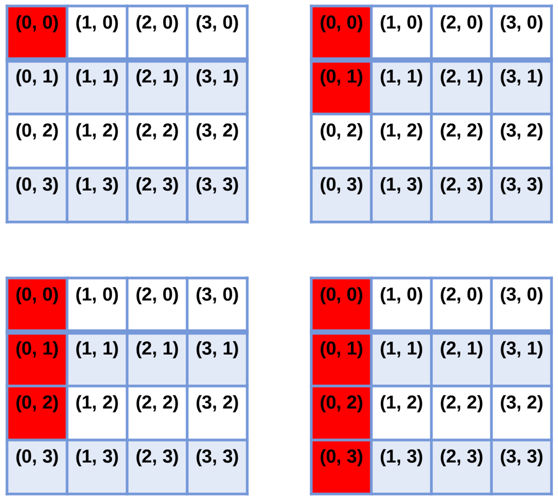
(0, 0)	(1, 0)	(2, 0)	(3, 0)
(0, 1)	(1, 1)	(2, 1)	(3, 1)
(0, 2)	(1, 2)	(2, 2)	(3, 2)
(0, 3)	(1, 3)	(2, 3)	(3, 3)

Color a Column

🟥 (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 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!

$\leftarrow$ [0, 1, 2, 3, 4, 5, 6, 7]

```
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!

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

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

(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)

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

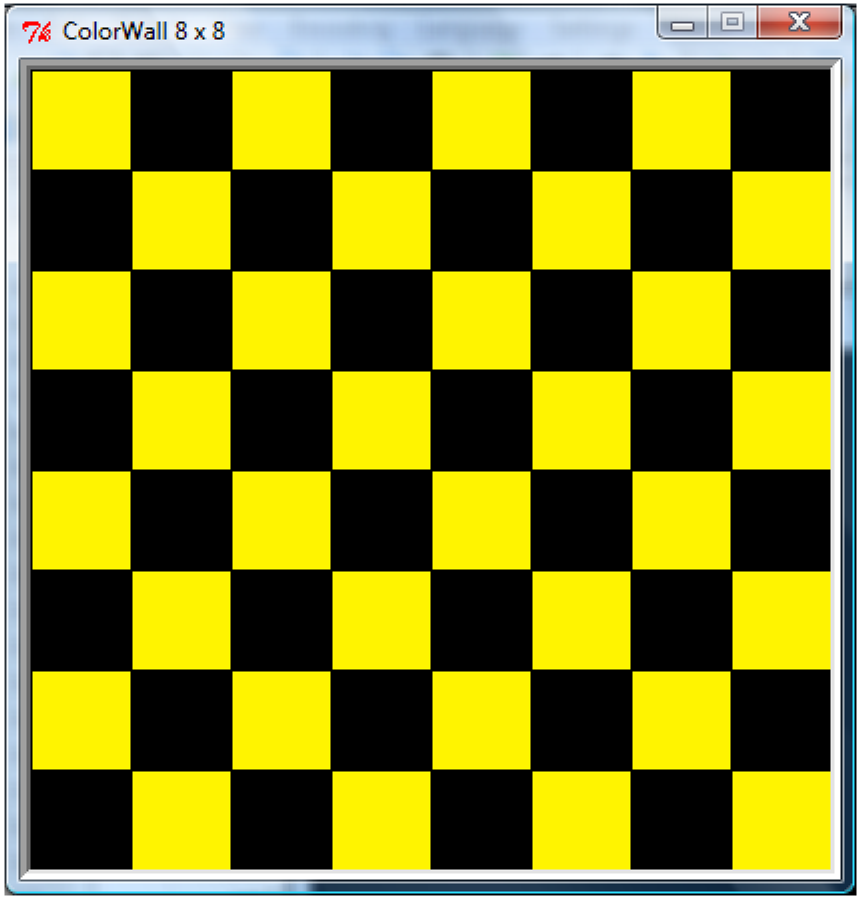
RainbowTest(wall)

```
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(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)
```

TUPLE

- Create a tuple
  - american_flag_colors = (‘red’, ‘white’, ‘blue’)

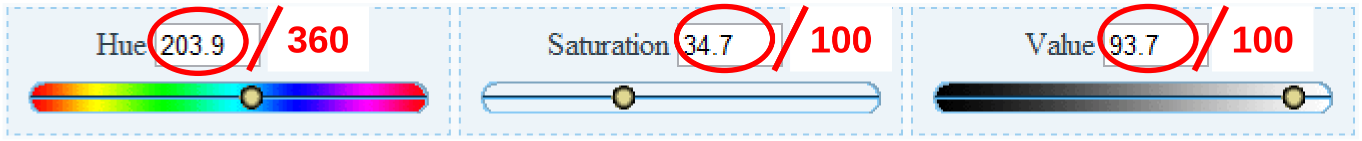
- How to get an item from the tuple?
  - american_flag_colors[0] = ‘red’

- 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

- 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)

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)

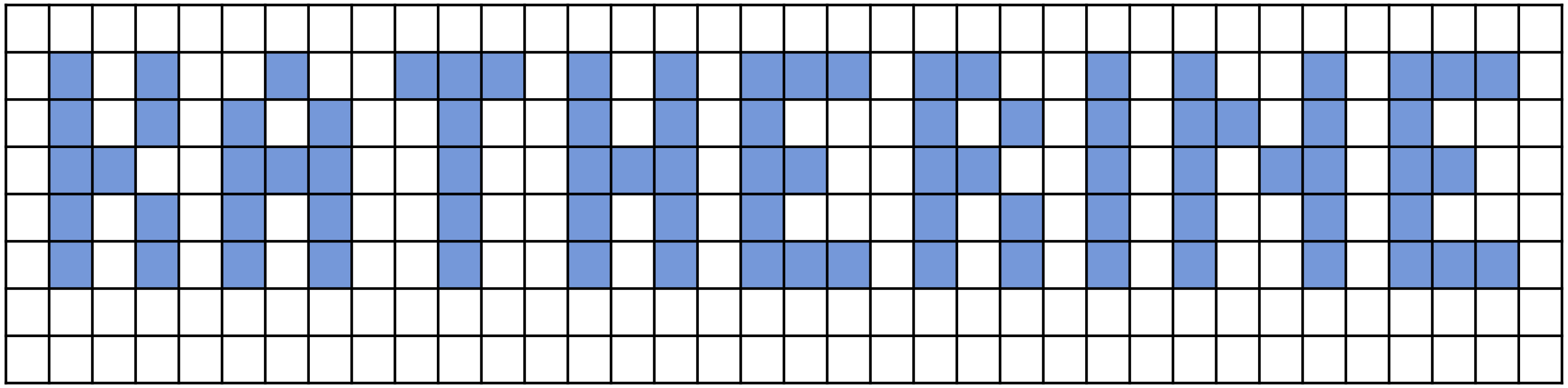
```
hue = 0
while hue < 1:                              # condition
    color = (hue, 1, 1)

    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:

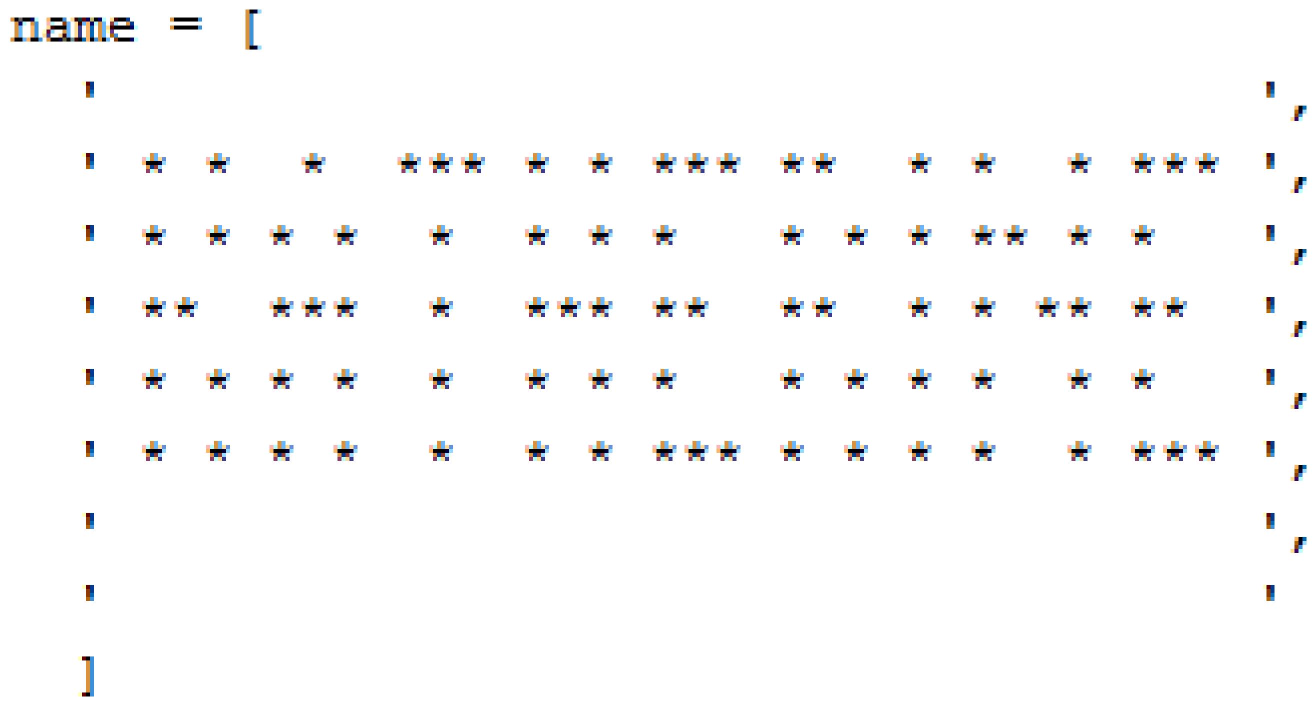
For example, what happens when you change the saturation or the value?

MESSAGE(WALL)



MESSAGE(WALL)

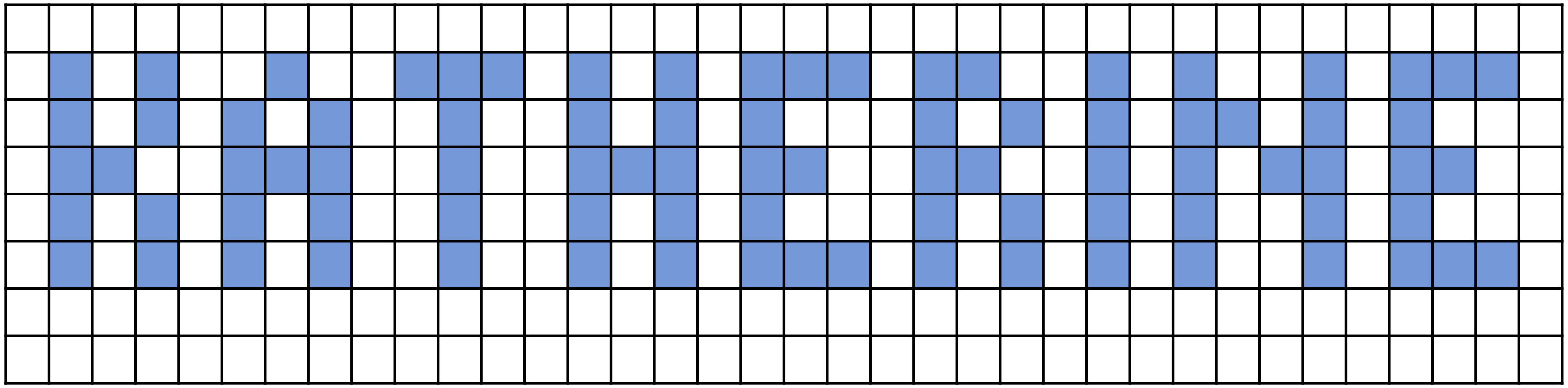
- Create your name list



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

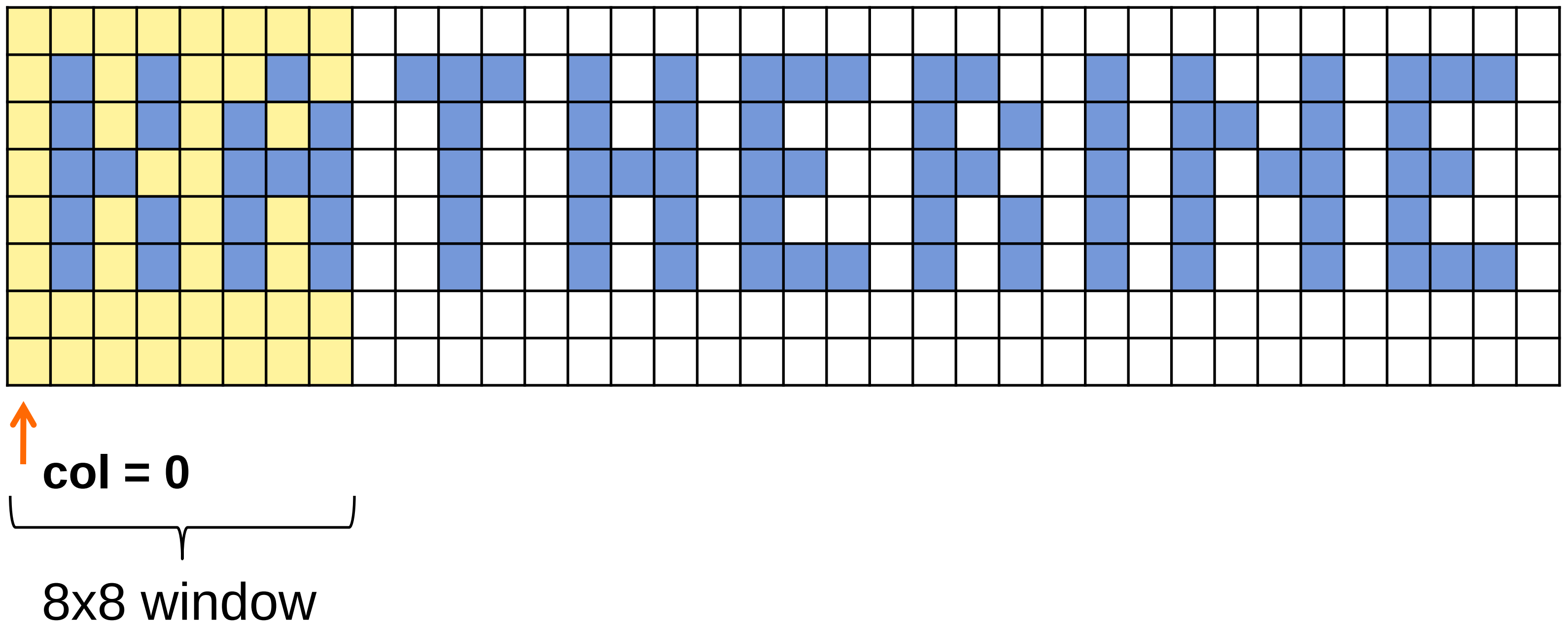
Message(wall)

- Let’s describe the algorithm in words:

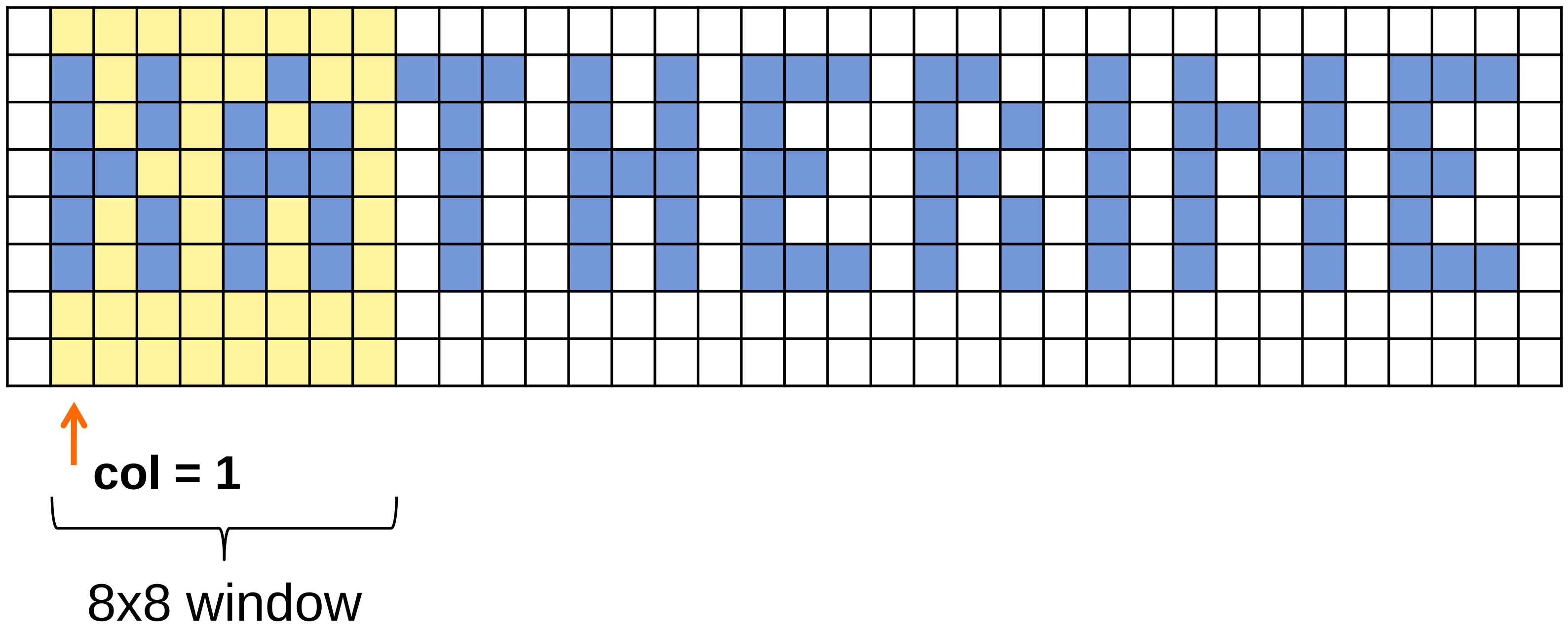


- For each 8x8 window
  - We want to print out the stars in a different color

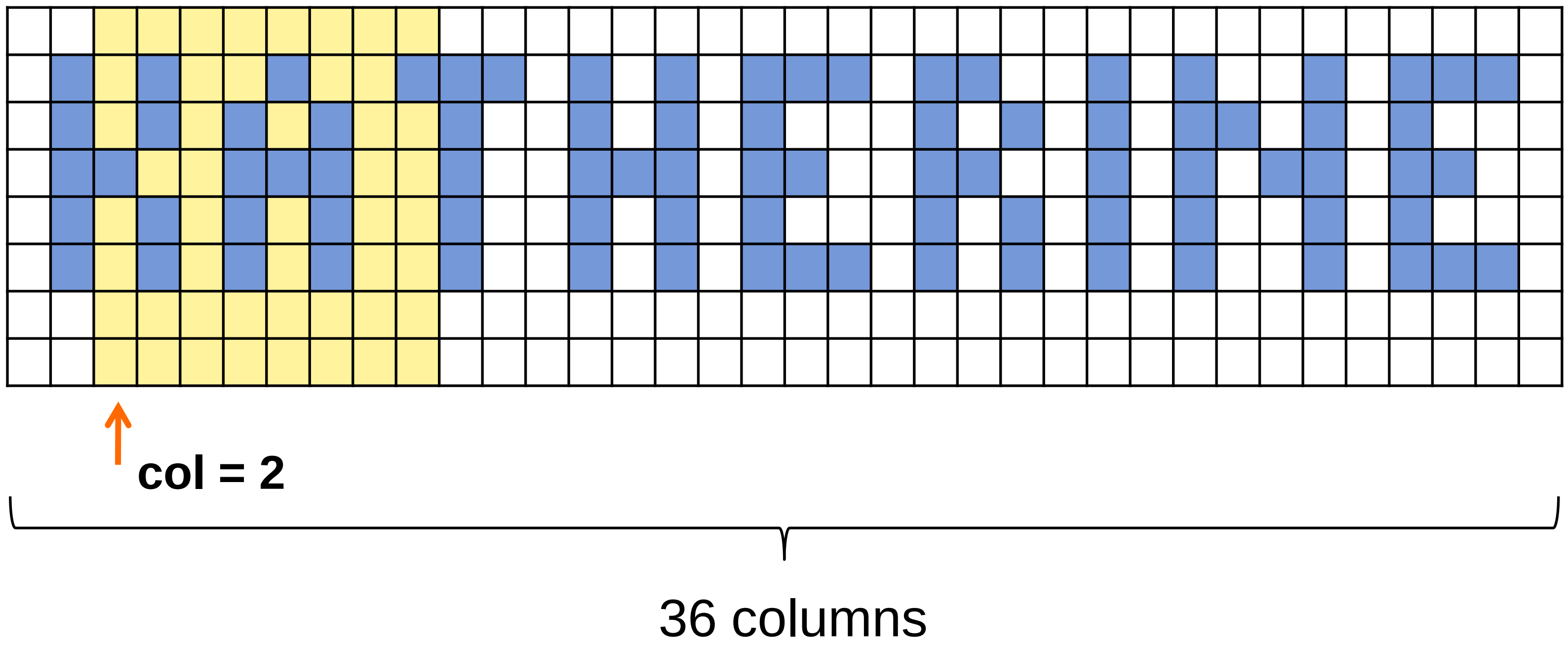
MESSAGE(WALL)



MESSAGE(WALL)

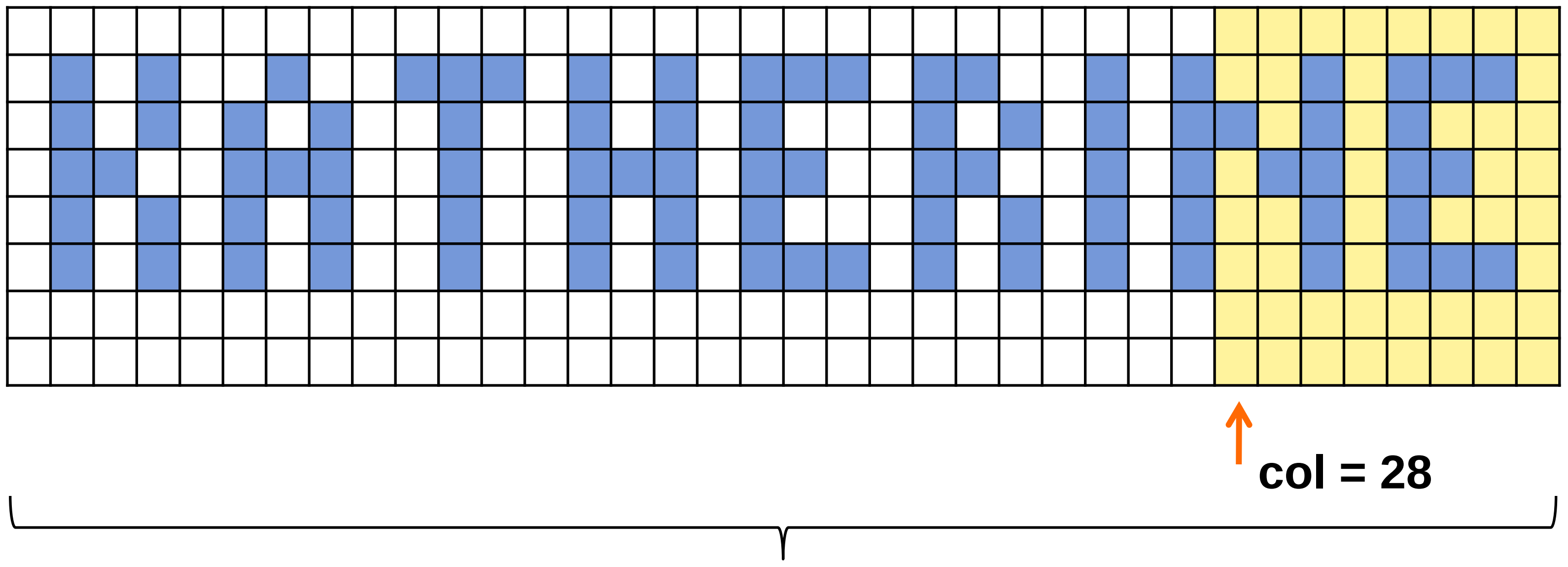


MESSAGE(WALL)



What is the range of **col**?

MESSAGE(WALL)



36 columns

What is the range of **col**?

range(29) = [0, 1, 2, 3,…, 28]

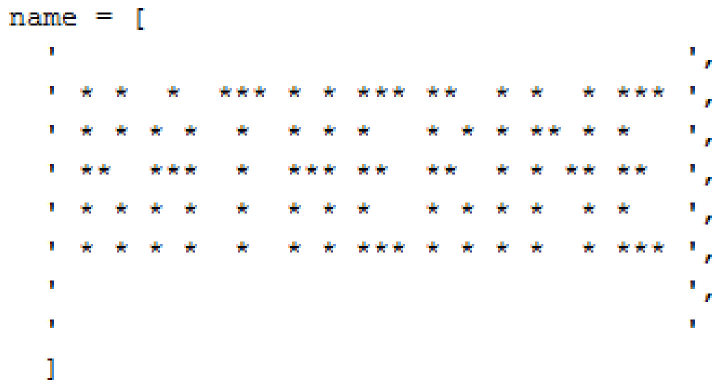
MESSAGE(WALL)

```
# 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 ]
```

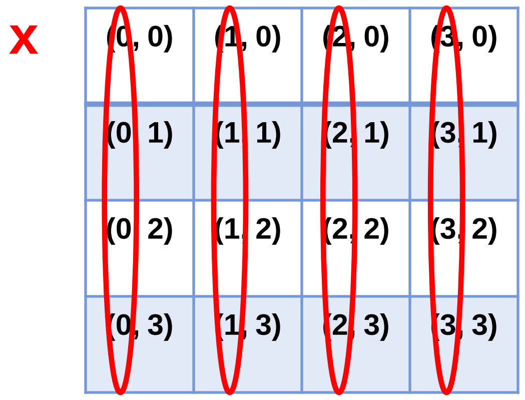
(0, 0)	(1, 0)	(2, 0)	(3, 0)
(0, 1)	(1, 1)	(2, 1)	(3, 1)
(0, 2)	(1, 2)	(2, 2)	(3, 2)
(0, 3)	(1, 3)	(2, 3)	(3, 3)

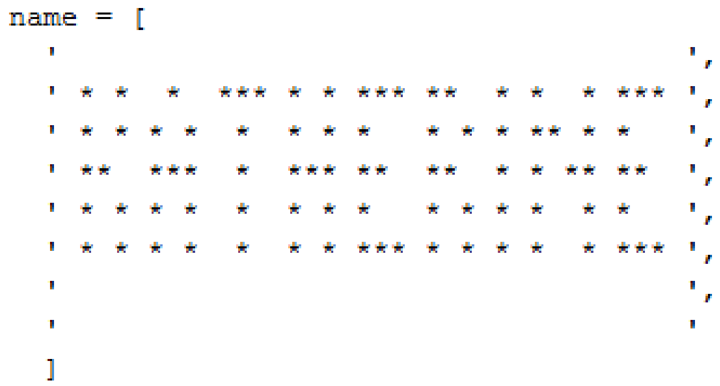


```
name = [
    '                                    ',
    ' * *  *  *** * * *** **  * *  * *** ',
    ' * * * *  *  * * *   * * * ** * *   ',
    ' **  ***  *  *** **  **  * * ** **  ',
    ' * * * *  *  * * *   * * * *  * *   ',
    ' * * * *  *  * * *** * * * *  * *** ',
    '                  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

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 ]
```





```
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)
```