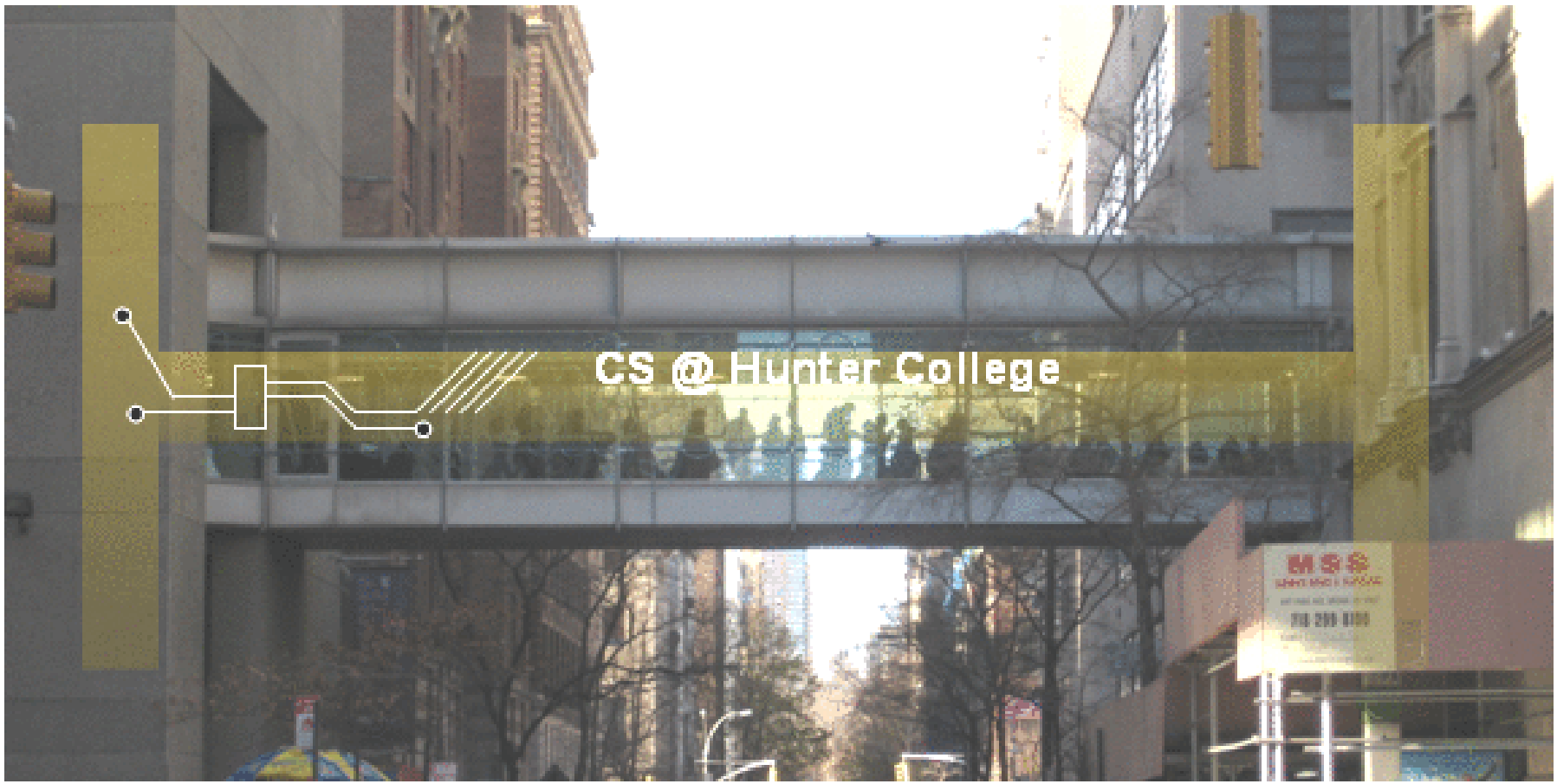
CSci 127: Introduction to Computer Science



Finished the lecture preview?

hunter.cuny.edu/csci

Today's Topics

- Recap: Modulus & Hex
- Colors
- 2D Arrays & Image Files
- Decisions

Today's Topics

- Recap: Modulus & Hex
- Colors
- 2D Arrays & Image Files
- Decisions

Challenge (Group Work)

- $2 \% 5 = ?$
- $5 \% 5 = ?$
- $10 \% 5 = ?$
- $(24 + 5) \% 26 = ?$

Challenge (Group Work)

- 2 % 5 = ?
  - 2; reason: 5 goes into 2 0 times, so the remainder is 2
  - generalize: if the number to the left of the modulus is less than the number to the right then the result is the left number
  - 2 - 5(0) = 2
- 5 % 5 = ?
  - 0; reason: 5 goes into 5 exactly once, so there is no remainder ie remainder of 0
  - generalize: when the remainder is 0, that means the left number is divisible by the right number
  - 5 - 5(1) = 0

Challege (Group Work)

- 10 % 5 = ?
  - 0; 5 goes into 10 exactly twice. There is no remainder.
  - 10 - 5(2) = 0
- (24 + 5) % 26 = ?
  - 3; reason: 24+5 is 29; 29 mod 26 is 3 because 26 goes into 29 only once. The remainder is 3 because 29 - 26(1)= 3.

From Hex to Dec

- What is hex 32 in decimal
  16*3 = 48 + 2 = 50
- What is hex 1D in decimal
  16*1 = 16 + 14 = 30 WRONG
  16*1 = 16 + 13 = 29 CORRECT
- What is hex FF in decimal
  16*15 = 240 + 15 = 255

Quizzes and Unix

- Using the command line to go through your file system instead of a graphical interface
- "Directory": another word for a folder
- How to see what's in the folder? Remember there is no graphical interface only the terminal. $ls will list the contents of the current folder.
- How to make a new folder? $mkdir newFolder will create a new folder
- How to see what folder you're in, i.e. where in the file system you are? $pwd

Challenge (Group Work)

EmpID: CSci 127 Mock Final, S19

2. (a) Fill in the boxes with the appropriate hexcode to change the color to match the comments:

```
import turtle
thomasH = turtle.Turtle()
```

i. `#Change thomasH to be the color black:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

ii. `#Change thomasH to be the color white:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iii. `#Change thomasH to be the brightest color blue:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iv. `#Change thomasH to be the color purple:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

v. `#Change thomasH to be the color gray:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

Challenge (Group Work)

EmpID: CSci 127 Mock Final, S19

2. (a) Fill in the boxes with the appropriate hexcode to change the color to match the comments:

```
import turtle
thomasH = turtle.Turtle()
```

i. `#Change thomasH to be the color black:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

ii. `#Change thomasH to be the color white:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iii. `#Change thomasH to be the brightest color blue:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iv. `#Change thomasH to be the color purple:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

v. `#Change thomasH to be the color gray:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

- Need to fill in hexcodes (always start with #):

Challenge (Group Work)

EmpID: CSci 127 Mock Final, S19

2. (a) Fill in the boxes with the appropriate hexcode to change the color to match the comments:

```
import turtle
thomasH = turtle.Turtle()
```

i. `#Change thomasH to be the color black:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

ii. `#Change thomasH to be the color white:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iii. `#Change thomasH to be the brightest color blue:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iv. `#Change thomasH to be the color purple:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

v. `#Change thomasH to be the color gray:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

- Need to fill in hexcodes (always start with #): `R R G G B B`

Challenge (Group Work)

EmpID: CSci 127 Mock Final, S19

2. (a) Fill in the boxes with the appropriate hexcode to change the color to match the comments:

```
import turtle
thomasH = turtle.Turtle()
```

i. `#Change thomasH to be the color black:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

ii. `#Change thomasH to be the color white:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iii. `#Change thomasH to be the brightest color blue:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iv. `#Change thomasH to be the color purple:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

v. `#Change thomasH to be the color gray:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

- Need to fill in hexcodes (always start with #): R R G G B B
- Black: 0 0 0 0 0 0

Challenge (Group Work)

EmpID: CSci 127 Mock Final, S19

2. (a) Fill in the boxes with the appropriate hexcode to change the color to match the comments:

```
import turtle
thomasH = turtle.Turtle()
```

i. `#Change thomasH to be the color black:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

ii. `#Change thomasH to be the color white:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iii. `#Change thomasH to be the brightest color blue:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iv. `#Change thomasH to be the color purple:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

v. `#Change thomasH to be the color gray:`

`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

- Need to fill in hexcodes (always start with #): `R R G G B B`
- Black: `0 0 0 0 0 0`
- White: `F F F F F F`

Challenge (Group Work)

EmpID: CSci 127 Mock Final, S19

2. (a) Fill in the boxes with the appropriate hexcode to change the color to match the comments:

```
import turtle
thomasH = turtle.Turtle()
```

i. `#Change thomasH to be the color black:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

ii. `#Change thomasH to be the color white:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iii. `#Change thomasH to be the brightest color blue:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iv. `#Change thomasH to be the color purple:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

v. `#Change thomasH to be the color gray:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

- Need to fill in hexcodes (always start with #): `R R G G B B`
- Black: `0 0 0 0 0 0`
- White: `F F F F F F`
- Blue: `0 0 0 0 F F`

Challenge (Group Work)

EmpID: CSci 127 Mock Final, S19

2. (a) Fill in the boxes with the appropriate hexcode to change the color to match the comments:

```
import turtle
thomasH = turtle.Turtle()
```

i. `#Change thomasH to be the color black:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

ii. `#Change thomasH to be the color white:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iii. `#Change thomasH to be the brightest color blue:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iv. `#Change thomasH to be the color purple:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

v. `#Change thomasH to be the color gray:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

- Need to fill in hexcodes (always start with #): `R R G G B B`
- Black: `0 0 0 0 0 0`
- White: `F F F F F F`
- Blue: `0 0 0 0 F F`
- Purple: `F F 0 0 F F`

Challenge (Group Work)

EmpID: CSci 127 Mock Final, S19

2. (a) Fill in the boxes with the appropriate hexcode to change the color to match the comments:

```
import turtle
thomasH = turtle.Turtle()
```

i. `#Change thomasH to be the color black:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

ii. `#Change thomasH to be the color white:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iii. `#Change thomasH to be the brightest color blue:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iv. `#Change thomasH to be the color purple:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

v. `#Change thomasH to be the color gray:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

- Need to fill in hexcodes (always start with #): `R R G G B B`
- Black: `0 0 0 0 0 0`
- White: `F F F F F F`
- Blue: `0 0 0 0 F F`
- Purple: `F F 0 0 F F`
- Gray: `4 2 4 2 4 2`

Challenge (Group Work)

EmpID: CSci 127 Mock Final, S19

2. (a) Fill in the boxes with the appropriate hexcode to change the color to match the comments:

```
import turtle
thomasH = turtle.Turtle()
```

i. `#Change thomasH to be the color black:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

ii. `#Change thomasH to be the color white:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iii. `#Change thomasH to be the brightest color blue:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

iv. `#Change thomasH to be the color purple:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

v. `#Change thomasH to be the color gray:`
`thomasH.color("#` [ ][ ][ ][ ][ ][ ] `")`

- Need to fill in hexcodes (always start with #): `R R G G B B`
- Black: `0 0 0 0 0 0`
- White: `F F F F F F`
- Blue: `0 0 0 0 F F`
- Purple: `F F 0 0 F F`
- Gray: `4 2 4 2 4 2` (any choice where RR = GG = BB).

Recap: Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by name.

Recap: Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by name.
- Can specify by numbers:

Recap: Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by name.
- Can specify by numbers:
  - Amount of Red, Green, and Blue (RGB).

Recap: Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by name.
- Can specify by numbers:
  - Amount of Red, Green, and Blue (RGB).
  - Adding light, not paint:

Recap: Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by name.
- Can specify by numbers:
  - Amount of Red, Green, and Blue (RGB).
  - Adding light, not paint:
    - ★ Black: 0% red, 0% green, 0% blue

Recap: Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by name.
- Can specify by numbers:
  - Amount of Red, Green, and Blue (RGB).
  - Adding light, not paint:
    - ★ Black: 0% red, 0% green, 0% blue
    - ★ White: 100% red, 100% green, 100% blue

Recap: Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by numbers (RGB):

Recap: Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by numbers (RGB):
  - Fractions of each:
    e.g. `(1.0, 0, 0)` is 100% red, no green, and no blue.

Recap: Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

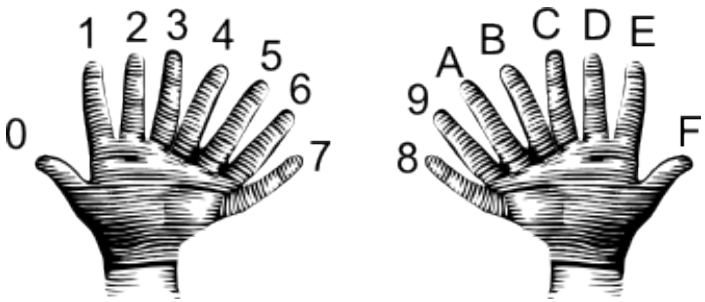
- Can specify by numbers (RGB):
  - Fractions of each:
    e.g. `(1.0, 0, 0)` is 100% red, no green, and no blue.
  - 8-bit colors: numbers from 0 to 255:
    e.g. `(0, 255, 0)` is no red, 100% green, and no blue.

Recap: Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by numbers (RGB):
  - Fractions of each:
    e.g. `(1.0, 0, 0)` is 100% red, no green, and no blue.
  - 8-bit colors: numbers from 0 to 255:
    e.g. `(0, 255, 0)` is no red, 100% green, and no blue.
  - Hexcodes (base-16 numbers)...

Recap: Hexadecimal




```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
```

Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by numbers (RGB):
  - Fractions of each:
    e.g. `(1.0, 0, 0)` is 100% red, no green, and no blue.
  - 8-bit colors: numbers from 0 to 255:
    e.g. `(0, 255, 0)` is no red, 100% green, and no blue.
  - Hexcodes (base-16 numbers):

Colors

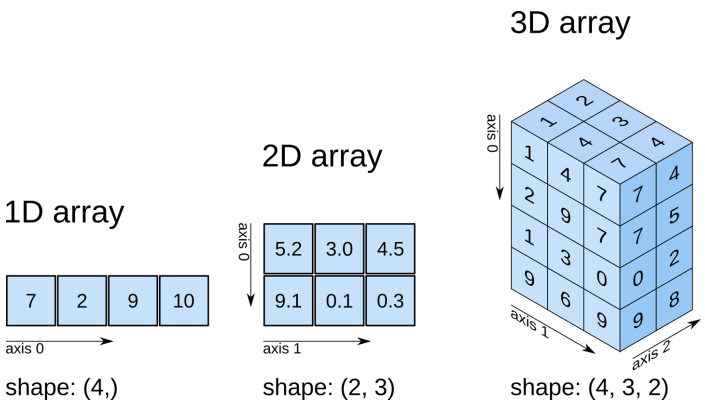
Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by numbers (RGB):
  - Fractions of each:
    e.g. `(1.0, 0, 0)` is 100% red, no green, and no blue.
  - 8-bit colors: numbers from 0 to 255:
    e.g. `(0, 255, 0)` is no red, 100% green, and no blue.
  - Hexcodes (base-16 numbers):
    e.g. `#0000FF` is no red, no green, and 100% blue.

Today's Topics

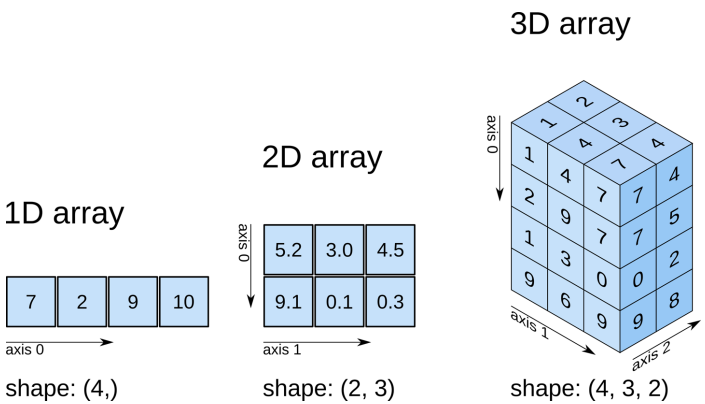
- Recap: Modulus & Hex
- Colors
- 2D Arrays & Image Files
- Decisions

Arrays



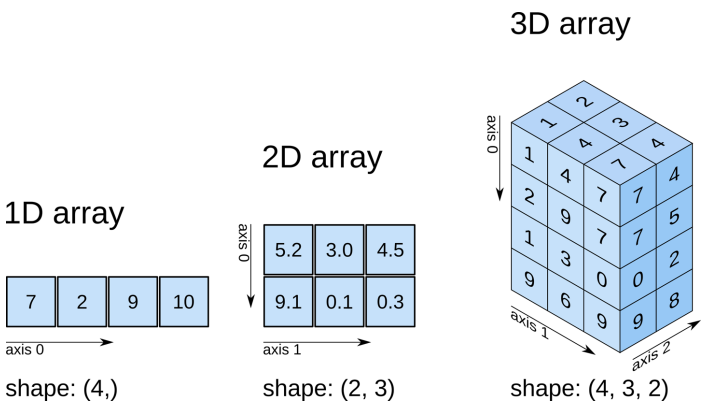
- An **array** is a sequence of elements, much like a list.

Arrays



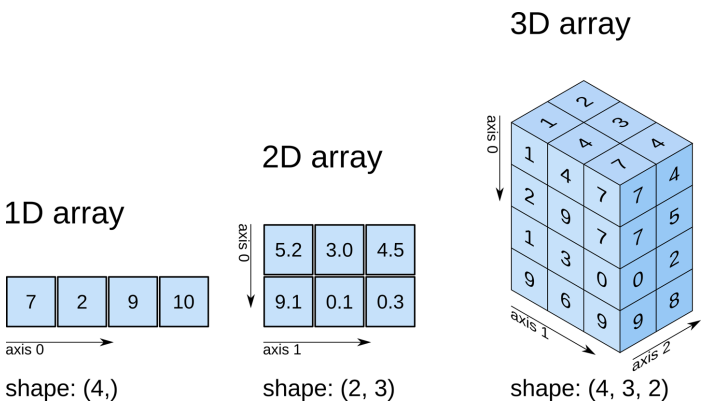
- An **array** is a sequence of elements, much like a list.
- A **2D array** is like a grid of elements, think a list of lists.

Arrays



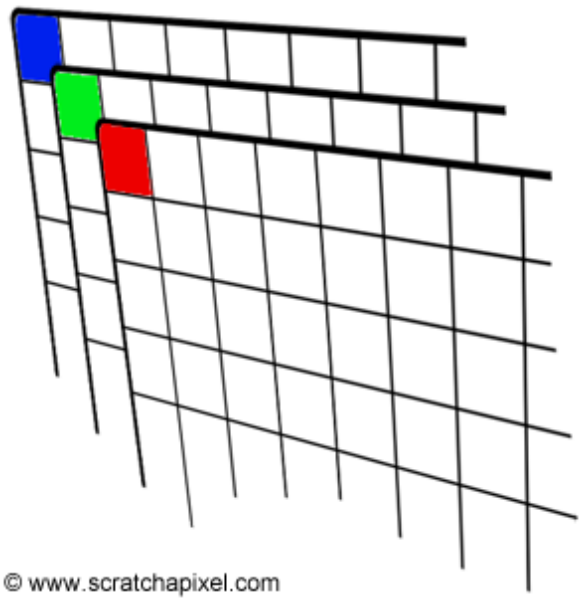
- An **array** is a sequence of elements, much like a list.
- A **2D array** is like a grid of elements, think a list of lists.
- Can keep on adding dimensions (3D, etc.)

Arrays

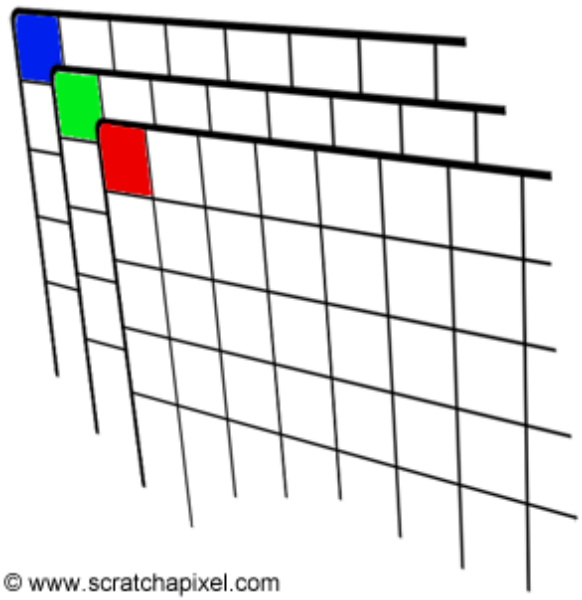


- An **array** is a sequence of elements, much like a list.
- A **2D array** is like a grid of elements, think a list of lists.
- Can keep on adding dimensions (3D, etc.)
- Can access pieces/slices as we do with strings and lists

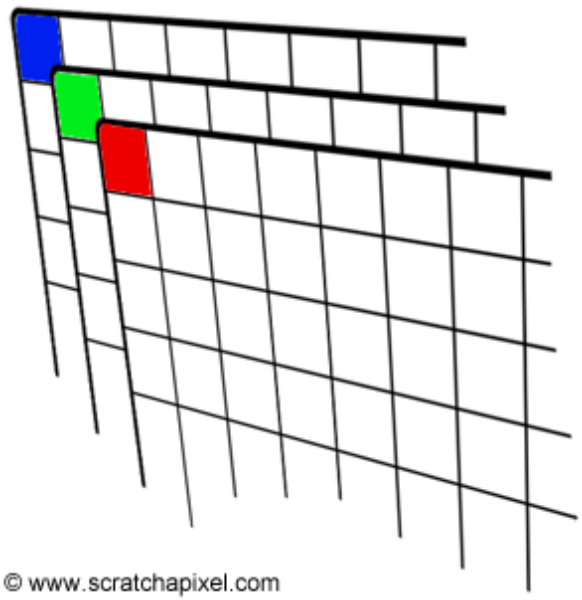
Images

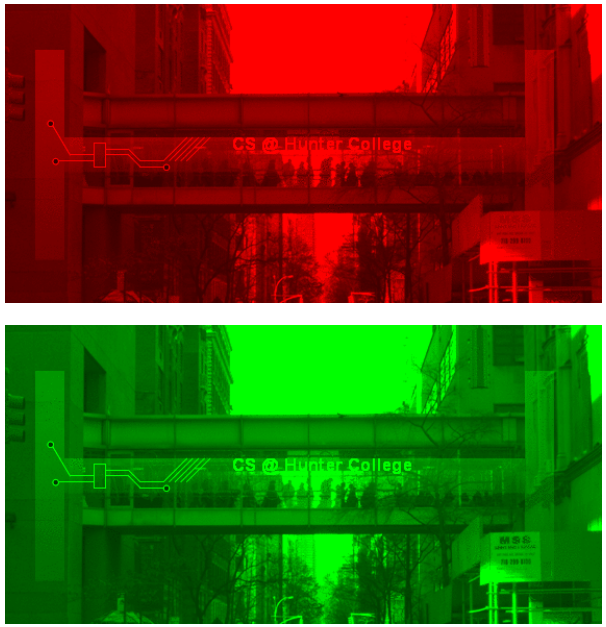


Images

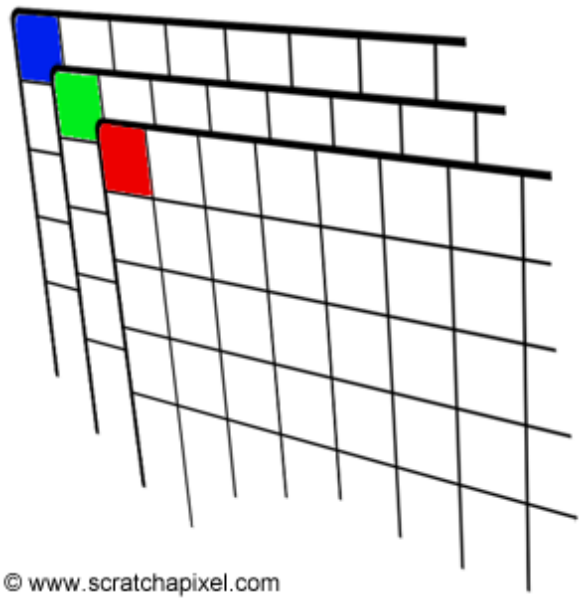


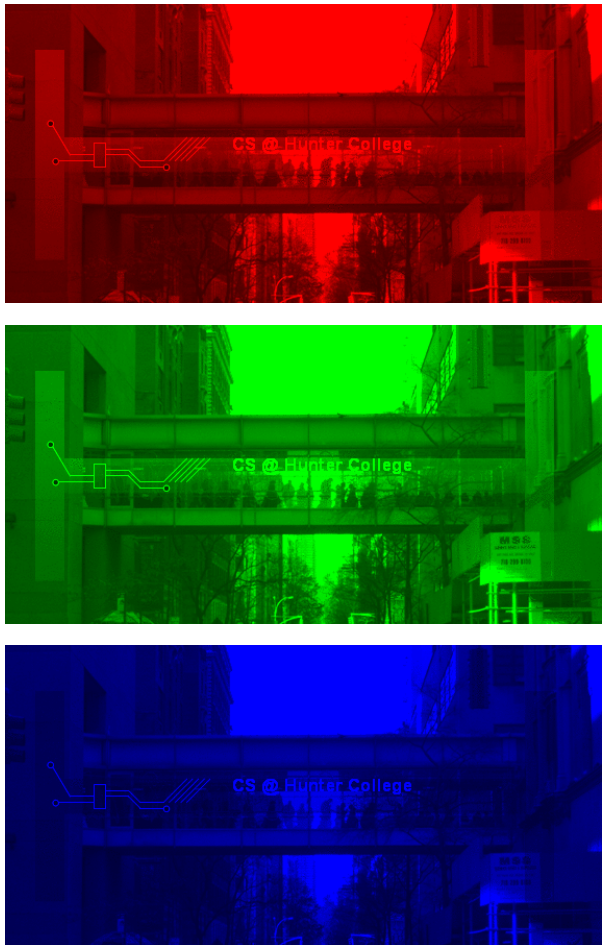
Images



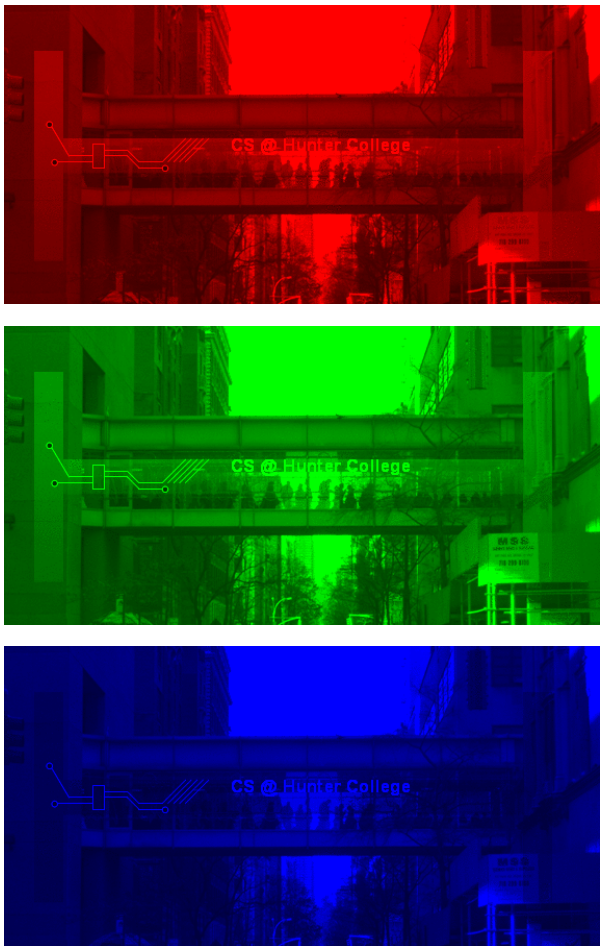


Images


© www.scratchapixel.com

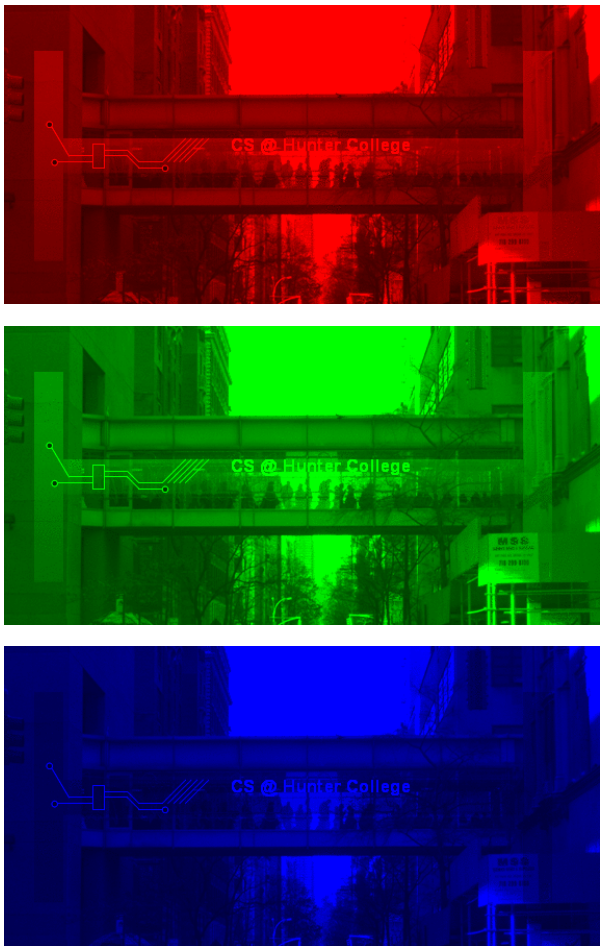

CS @ Hunter College
CS @ Hunter College
CS @ Hunter College

Useful Packages



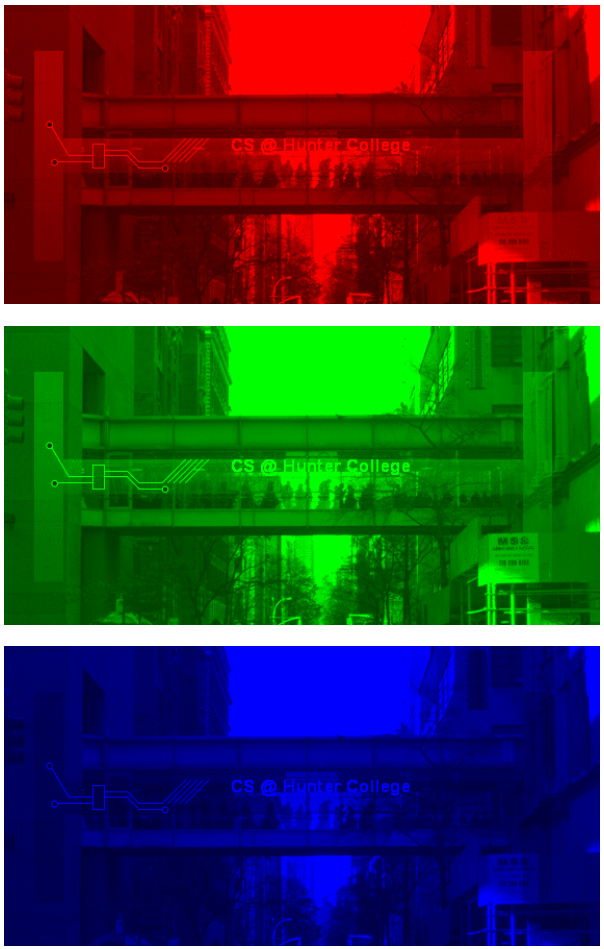
- We will use 2 useful packages for images:

Useful Packages



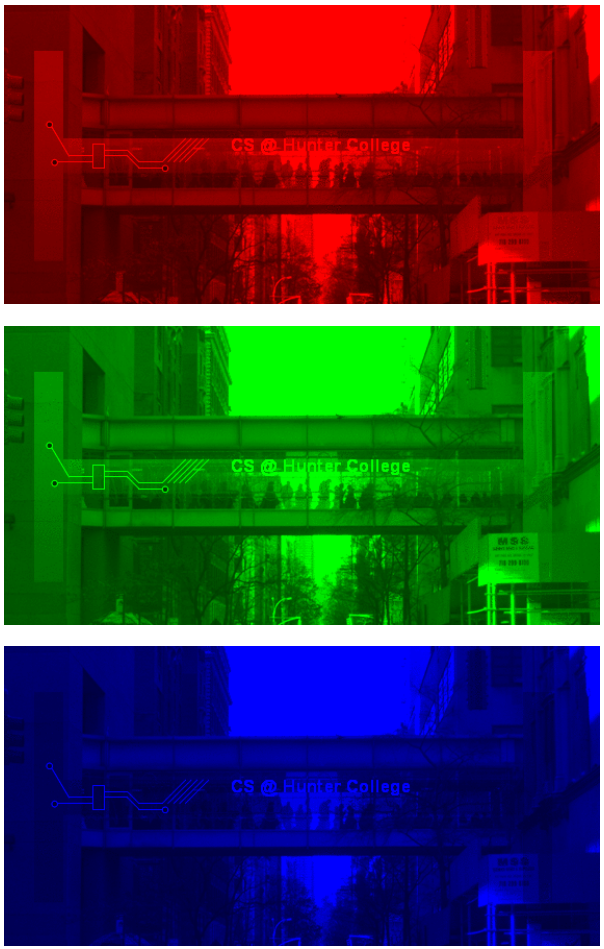
- We will use 2 useful packages for images:
  - `numpy`: numerical analysis package

Useful Packages



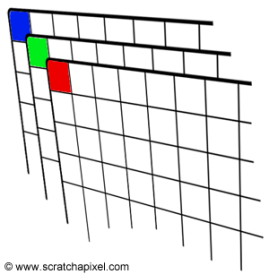
- We will use 2 useful packages for images:
  - `numpy`: numerical analysis package
  - `pyplot`: part of `matplotlib` for making graphs and plots

Useful Packages



- We will use 2 useful packages for images:
  - `numpy`: numerical analysis package
  - `pyplot`: part of `matplotlib` for making graphs and plots
- See lab notes for installing on your home machine.

Images with pyplot and numpy



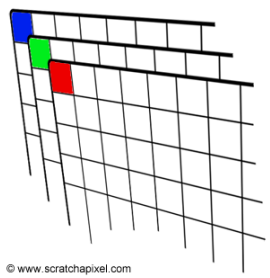

```
#Import the packages for images and arrays:
import matplotlib.pyplot as plt
import numpy as np


img = plt.imread('csBridge.png')    #Read in image from csBridge.png
plt.imshow(img)                      #Load image into pyplot
plt.show()                           #Show the image (waits until close

img2 = img.copy()          #make a copy of our image
img2[:,:,1] = 0            #Set the green channel to 0
img2[:,:,2] = 0            #Set the blue channel to 0

plt.imshow(img2)           #Load our new image into pyplot
plt.show()                 #Show the image (waits until closed to conti

plt.imsave('reds.png', img2)  #Save the image we created to the file:
```

Images with pyplot and numpy

```
#Import the packages for images and arrays:
import matplotlib.pyplot as plt
import numpy as np


img = plt.imread('csBridge.png')    #Read in image from csBridge.png
plt.imshow(img)                      #Load image into pyplot
plt.show()                           #Show the image (waits until close

img2 = img.copy()          #make a copy of our image
img2[:,:,1] = 0            #Set the green channel to 0
img2[:,:,2] = 0            #Set the blue channel to 0

plt.imshow(img2)           #Load our new image into pyplot
plt.show()                 #Show the image (waits until closed to conti

plt.imsave('reds.png', img2)  #Save the image we created to the file:
```

Creating Images

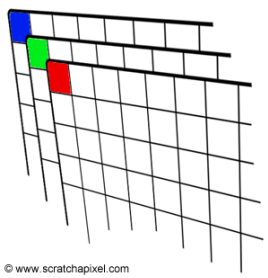
To create an image from scratch:



Creating Images

To create an image from scratch:

1. Import the libraries.

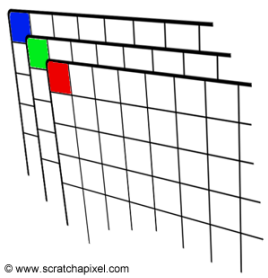


Creating Images

To create an image from scratch:

1. Import the libraries.

```
import matplotlib.pyplot as plt
import numpy as np
```



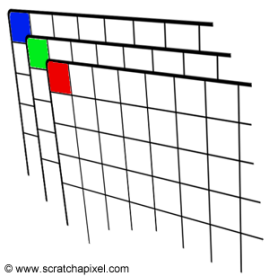
Creating Images

To create an image from scratch:

1. Import the libraries.

```
import matplotlib.pyplot as plt
import numpy as np
```

2. Create the image– easy to set all color



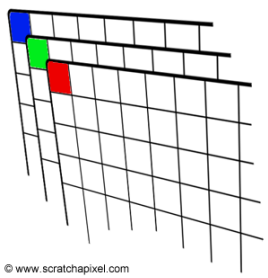
Creating Images

To create an image from scratch:

1. Import the libraries.

```
import matplotlib.pyplot as plt
import numpy as np
```

2. Create the image– easy to set all color
   1. to 0% (black):



Creating Images

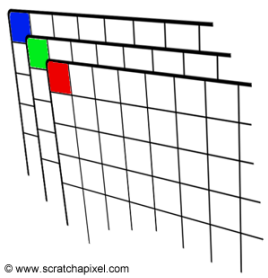
To create an image from scratch:

1. Import the libraries.

```
import matplotlib.pyplot as plt
import numpy as np
```

2. Create the image– easy to set all color
   1. to 0% (black):

```
img = np.zeros( (num,num,3) )
```



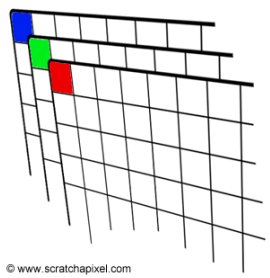
Creating Images

To create an image from scratch:

1. Import the libraries.

```
import matplotlib.pyplot as plt
import numpy as np
```

2. Create the image– easy to set all color
   1. to 0% (black):
      `img = np.zeros( (num,num,3) )`
   2. to 100% (white):



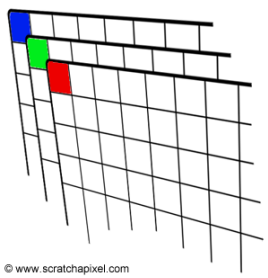
Creating Images

To create an image from scratch:

1. Import the libraries.

```
import matplotlib.pyplot as plt
import numpy as np
```

2. Create the image– easy to set all color
   1. to 0% (black):
      `img = np.zeros( (num,num,3) )`
   2. to 100% (white):
      `img = np.ones( (num,num,3) )`



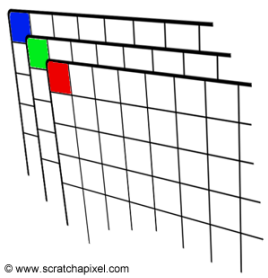
Creating Images

To create an image from scratch:

1. Import the libraries.

```
import matplotlib.pyplot as plt
import numpy as np
```

2. Create the image– easy to set all color
   1. to 0% (black):
      `img = np.zeros( (num,num,3) )`
   2. to 100% (white):
      `img = np.ones( (num,num,3) )`
3. *Do stuff to the pixels to make your image*



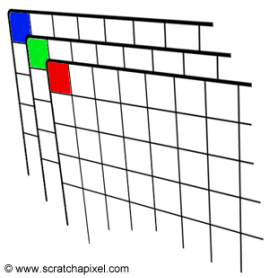
Creating Images

To create an image from scratch:

1. Import the libraries.

```
import matplotlib.pyplot as plt
import numpy as np
```

2. Create the image– easy to set all color
   1. to 0% (black):
      `img = np.zeros( (num,num,3) )`
   2. to 100% (white):
      `img = np.ones( (num,num,3) )`
3. *Do stuff to the pixels to make your image*
4. You can display your image:



Creating Images

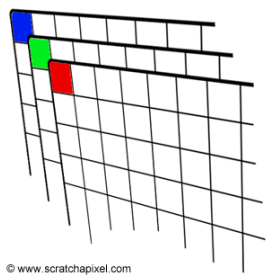
To create an image from scratch:

1. Import the libraries.

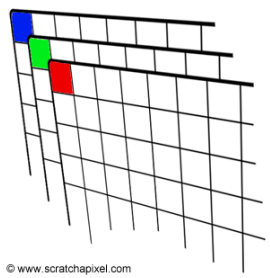
```
import matplotlib.pyplot as plt
import numpy as np
```

2. Create the image– easy to set all color
   1. to 0% (black):
      `img = np.zeros( (num,num,3) )`
   2. to 100% (white):
      `img = np.ones( (num,num,3) )`
3. *Do stuff to the pixels to make your image*
4. You can display your image:

```
plt.imshow(img)
plt.show()
```



Creating Images



To create an image from scratch:

1. Import the libraries.

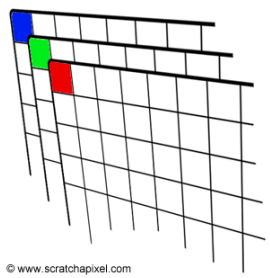
```
import matplotlib.pyplot as plt
import numpy as np
```

2. Create the image– easy to set all color
   1. to 0% (black):
      `img = np.zeros( (num,num,3) )`
   2. to 100% (white):
      `img = np.ones( (num,num,3) )`
3. *Do stuff to the pixels to make your image*
4. You can display your image:

```
plt.imshow(img)
plt.show()
```

5. And save your image:

Creating Images



To create an image from scratch:

1. Import the libraries.

```
import matplotlib.pyplot as plt
import numpy as np
```

2. Create the image– easy to set all color
   1. to 0% (black):
      `img = np.zeros( (num,num,3) )`
   2. to 100% (white):
      `img = np.ones( (num,num,3) )`
3. *Do stuff to the pixels to make your image*
4. You can display your image:

```
plt.imshow(img)
plt.show()
```

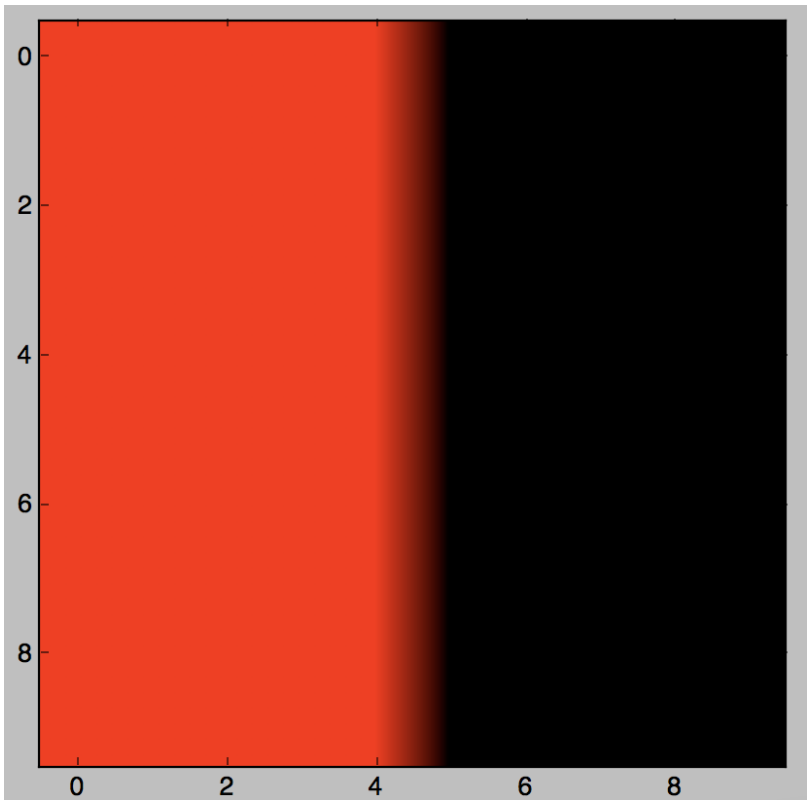
5. And save your image:

```
plt.imsave('myImage.png', img)
```

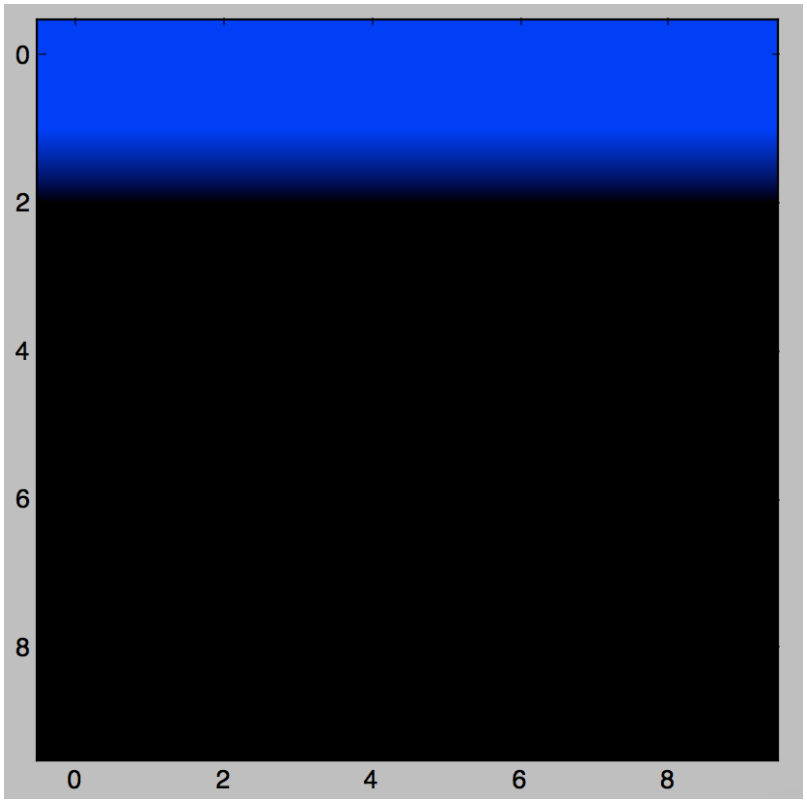
Slicing & Image Examples

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.

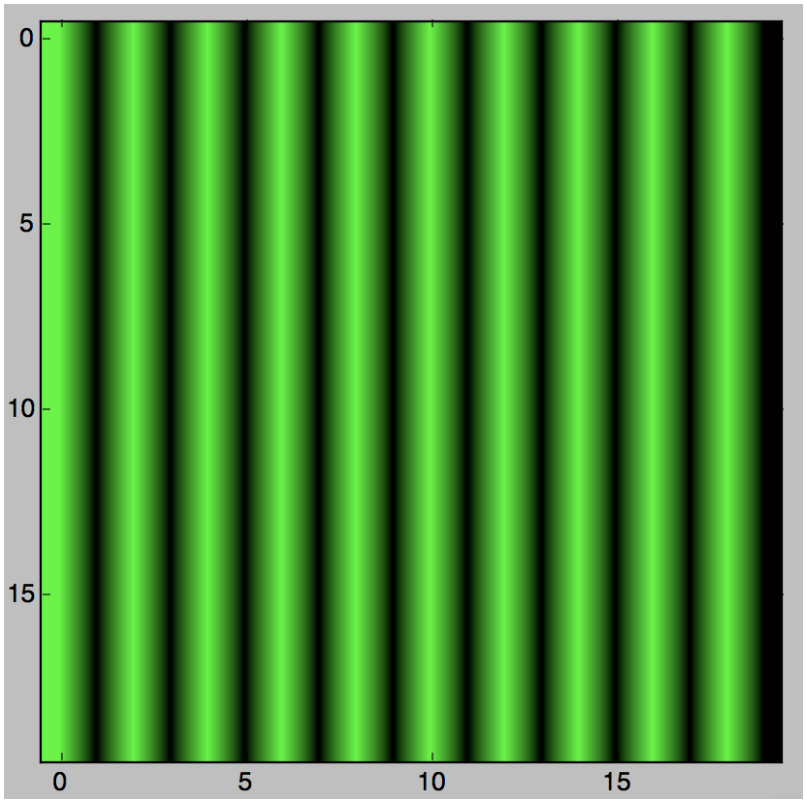
Slicing & Image Examples

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:
  - ```
    img = np.zeros( (10,10,3) )
    img[0:10,0:5,0:1] = 1
    ```
    ```

Slicing & Image Examples

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:
  - ```
    img = np.zeros( (10,10,3) )
    img[0:10,0:5,0:1] = 1
    ```


    ```

Slicing & Image Examples

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:
  - ```
    num = 10
    img = np.zeros( (num,num,3) )
    img[0:2,:,2:3] = 1.0
    ```
    ```

Slicing & Image Examples

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:
  - ```
    num = 10
    img = np.zeros( (num,num,3) )
    img[0:2,:,2:3] = 1.0
    ```


    ```

Slicing & Image Examples

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:
  - ```
    num = int(input('Enter size'))
    img = np.zeros( (num,num,3) )
    img[:,::2,1] = 1.0
    ```
    ```

Slicing & Image Examples

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:
  - ```
    num = int(input('Enter size'))
    img = np.zeros( (num,num,3) )
    img[:,::2,1] = 1.0
    ```


    ```

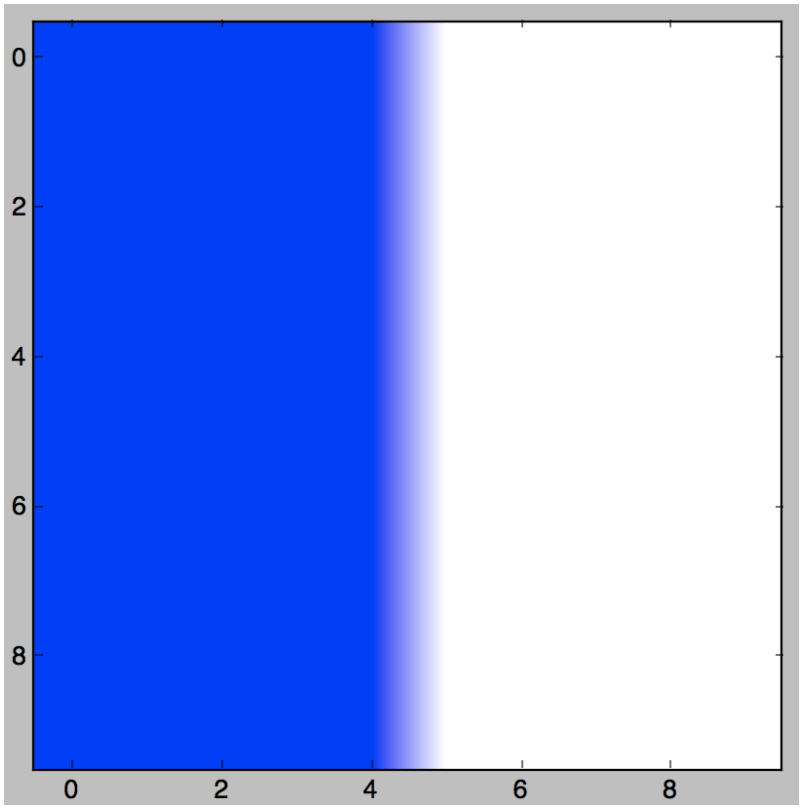
Challenge (Group Work)

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:

  - ```
    img = np.ones( (10,10,3) )
    img[0:10,0:5,0:2] = 0
    ```
    ```

Challenge (Group Work)

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:

  - ```
    img = np.ones( (10,10,3) )
    img[0:10,0:5,0:2] = 0
    ```

  - ```
    num = int(input('Enter size '))
    img = np.ones( (num,num,3) )
    img[::2,:,1:]  = 0
    ```

Challenge (Group Work)

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:

  - ```
    img = np.ones( (10,10,3) )
    img[0:10,0:5,0:2] = 0
    ```

  - ```
    num = int(input('Enter size '))
    img = np.ones( (num,num,3) )
    img[::2,:,1:]  = 0
    ```

  - ```
    img = np.zeros( (8,8,3) )
    img[::2,::2,0] = 1
    ```
    ```

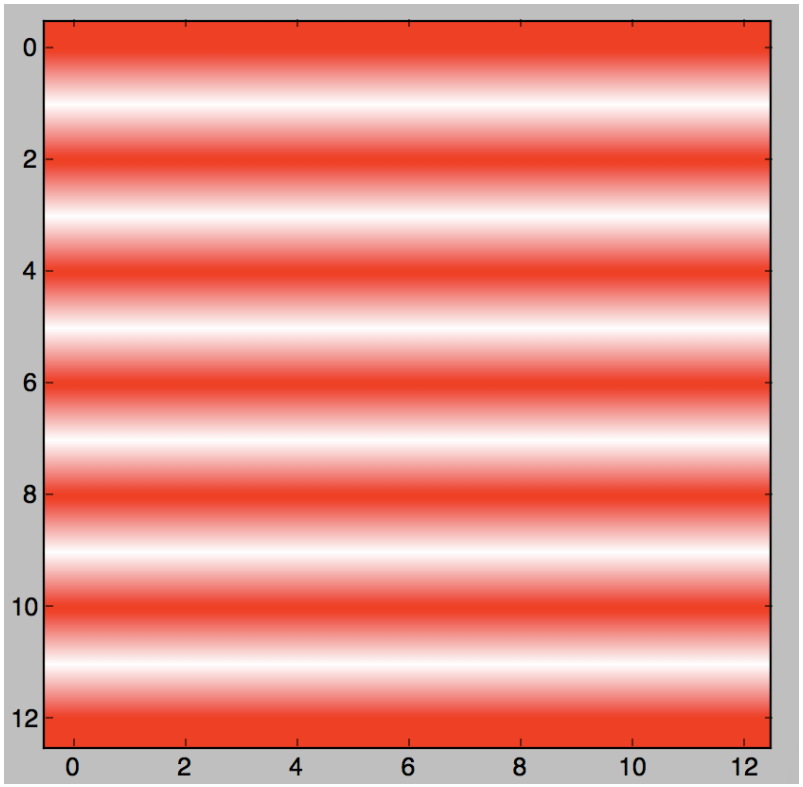
Challenge (Group Work):

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:

  - ```
    img = np.ones( (10,10,3) )
    img[0:10,0:5,0:2] = 0
    ```
    ```

Challenge (Group Work):

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:

  - ```
    img = np.ones( (10,10,3) )
    img[0:10,0:5,0:2] = 0
    ```


    ```

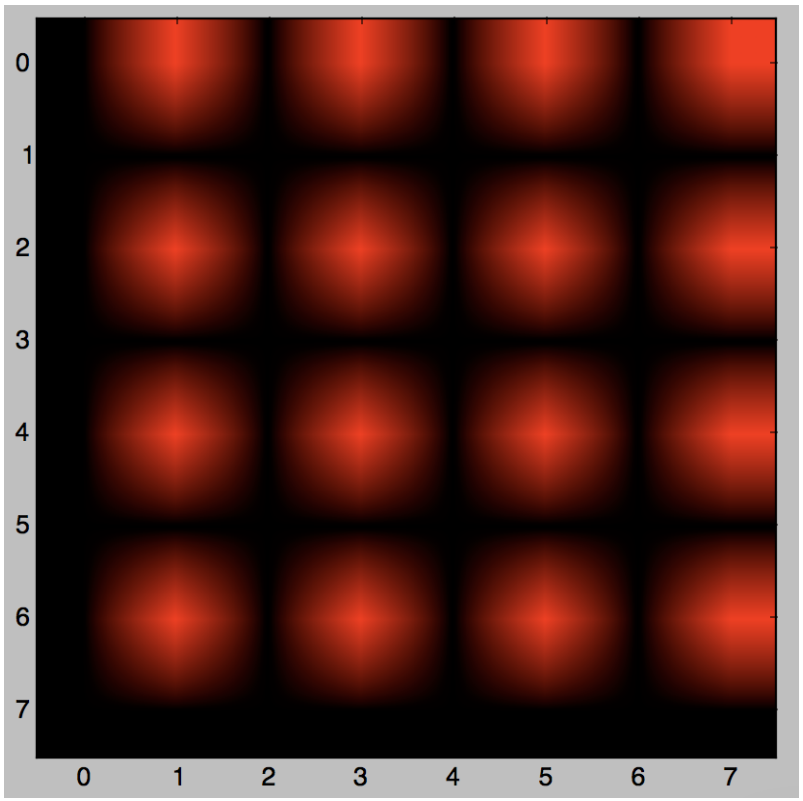
Challenge (Group Work)

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:

  - ```
    num = int(input('Enter size '))
    img = np.ones( (num,num,3) )
    img[::2,:,1:]  = 0
    ```
    ```

Challenge (Group Work)

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:

  - ```
    num = int(input('Enter size '))
    img = np.ones( (num,num,3) )
    img[::2,:,1:]  = 0
    ```


    ```

Challenge (Group Work)

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:

  - ```
    img = np.zeros( (8,8,3) )
    img[::2,1::2,0] = 1
    ```
    ```

Challenge (Group Work)

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:

  - ```
    img = np.zeros( (8,8,3) )
    img[::2,1::2,0] = 1
    ```


    ```

Today's Topics

- Recap: Modulus & Hex
- Colors
- 2D Arrays & Image Files
- Decisions

Today's Topics

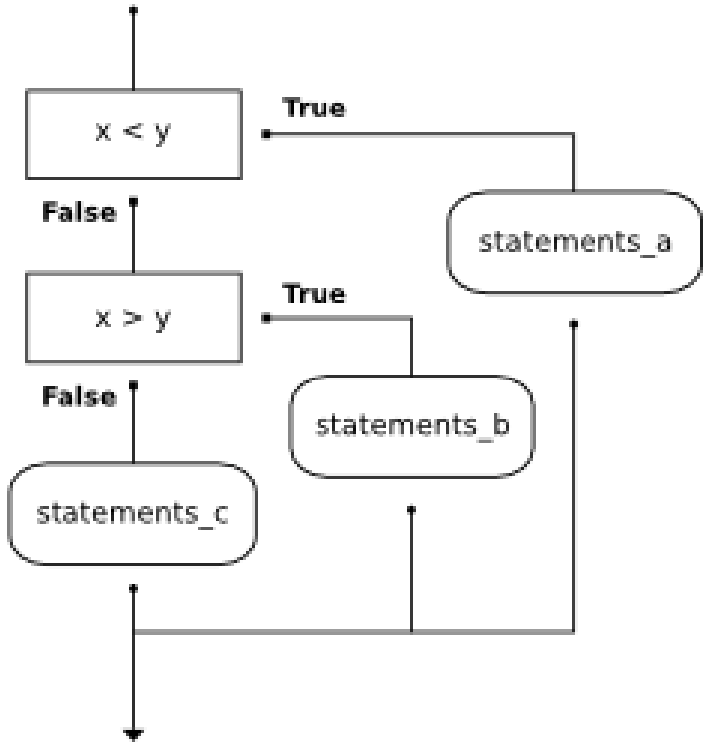
- Recap: Modulus & Hex
- Colors
- 2D Arrays & Image Files
- Decisions

Decisions

```
if x < y:
    print("x is less than y")
elif x > y:
    print("x is greater than y")
else:
    print("x and y must be equal")
```

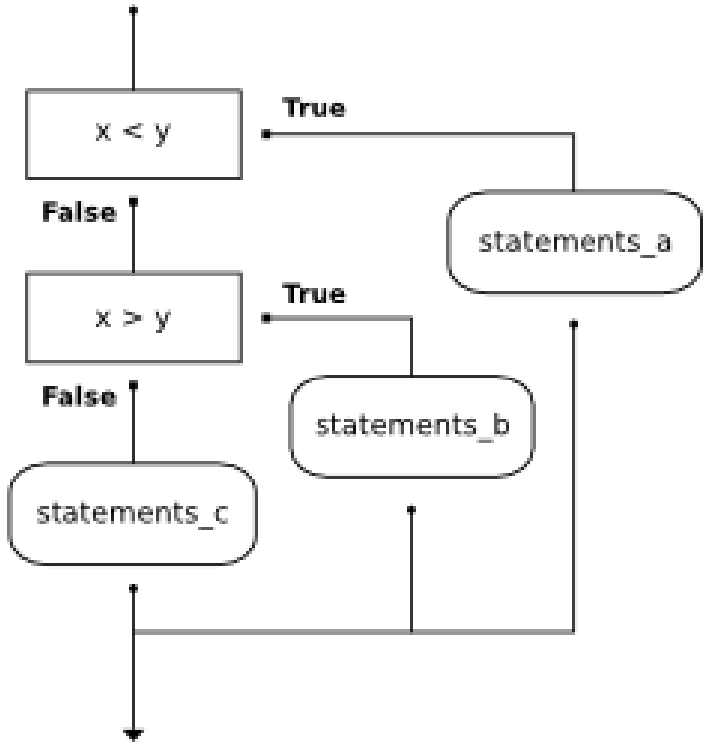
Decisions

```
if x < y:
    print("x is less than y")
elif x > y:
    print("x is greater than y")
else:
    print("x and y must be equal")
```



Decisions

```
if x < y:
    print("x is less than y")
elif x > y:
    print("x is greater than y")
else:
    print("x and y must be equal")
```



(This was just a first glance, will do much more on decisions over the next several weeks.)

Recap

- In Python, we introduced:

Recap

- In Python, we introduced:
  - Recap: Colors
  - 2D Array & Image Files
  - Decisions