CSci 127: Introduction to Computer Science
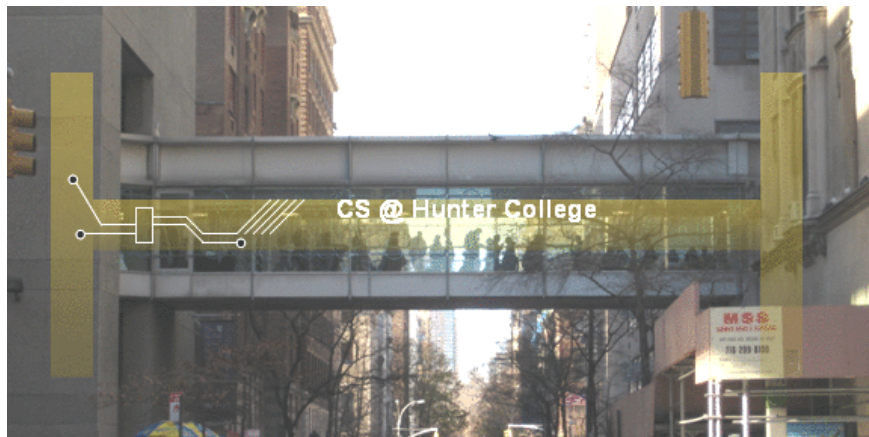

Finished the lecture preview?

- This lecture will be recorded

Reminder

- Students often pose as expert even when they are not

Reminder

- Students often pose as expert even when they are not

- Immediate, *easy* help is NOT REAL HELP

Reminder

- Students often pose as expert even when they are not

- Immediate, *easy* help is NOT REAL HELP

- a.k.a. that student will not be there on your next interview

Reminder

- Students often pose as expert even when they are not

- Immediate, *easy* help is NOT REAL HELP

- a.k.a. that student will not be there on your next interview

- **Our UTAs are the experts** for this course

Reminder

- Students often pose as expert even when they are not
- Immediate, *easy* help is NOT REAL HELP
- a.k.a. that student will not be there on your next interview
- **Our UTAs are the experts** for this course
- Trained to help you with coursework while you LEARN important **skills**

Tutoring is Fundamental

Career Readiness

- National Association of Colleges and Employers (NACE)

Career Readiness

- National Association of Colleges and Employers (NACE)

- 8 Competencies, here you will start developing 4:

Career Readiness

- National Association of Colleges and Employers (NACE)

- 8 Competencies, here you will start developing 4:
    - Critical Thinking/Problem Solving

Career Readiness

- National Association of Colleges and Employers (NACE)

- 8 Competencies, here you will start developing 4:

  - Critical Thinking/Problem Solving
  - Oral Communications

Career Readiness

- National Association of Colleges and Employers (NACE)

- 8 Competencies, here you will start developing 4:
    - Critical Thinking/Problem Solving
    - Oral Communications
    - Teamwork/Collaboration

Career Readiness

- National Association of Colleges and Employers (NACE)

- 8 Competencies, here you will start developing 4:
    - Critical Thinking/Problem Solving
    - Oral Communications
    - Teamwork/Collaboration
    - Digital Technology

Today's Topics

- Recap: Colors
- 2D Arrays & Image Files
- Decisions
- Design Challenge: Airplanes

Today's Topics

- **Recap: Colors**
- 2D Arrays & Image Files
- Decisions
- Design Challenge: Airplanes

Challenge

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

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

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

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

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

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

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

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

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
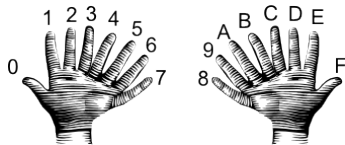

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 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: Colors
- **2D Arrays & Image Files**
- Decisions
- Design Challenge: Airplanes

Arrays
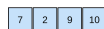
3D array

2D array

1D array

shape: (4,)

shape: (2, 3)

shape: (4, 3, 2)

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

Arrays

3D array

2D array

1D array

| 7 | 2 | 9 | 10 |

axis 0

shape: (4,)

| 5.2 | 3.0 | 4.5 |
| 9.1 | 0.1 | 0.3 |

axis 1
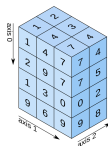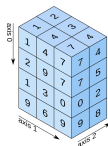
shape: (2, 3)

shape: (4, 3, 2)

- 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
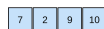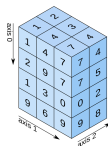
3D array

2D array

1D array

shape: (4,)

shape: (2, 3)

shape: (4, 3, 2)

- 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
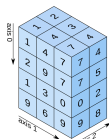
3D array

2D array

1D array

| 7 | 2 | 9 | 10 |

shape: (4,)

axis 0

| 5.2 | 3.0 | 4.5 |
| 9.1 | 0.1 | 0.3 |

axis 1

axis 0

shape: (2, 3)

axis 2

axis 1

axis 0

shape: (4, 3, 2)

- 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
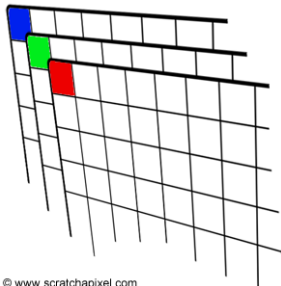
© www.scratchapixel.com
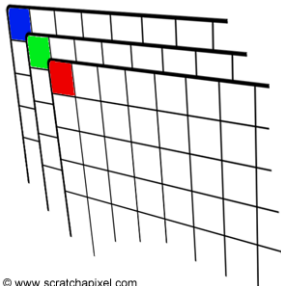
Images

© www.scratchapixel.com
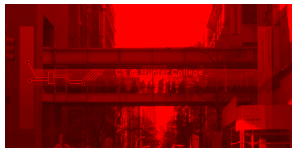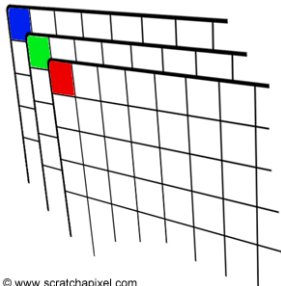
Images

© www.scratchapixel.com
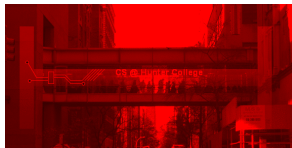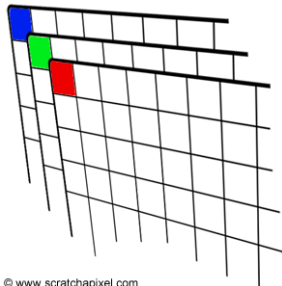
Images

© www.scratchapixel.com
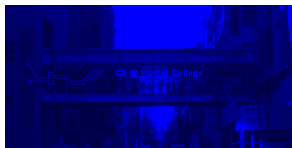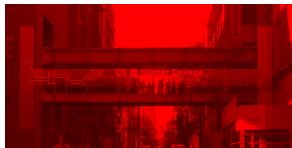
Useful Packages
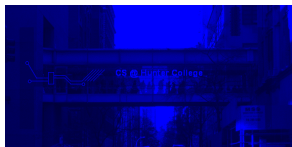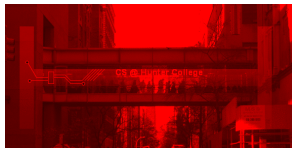
- We will use 2 useful packages for images:

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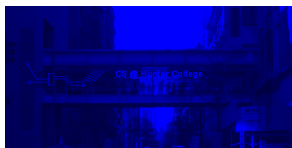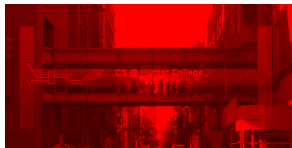
- We will use 2 useful packages for images:
  - `numpy`: numerical analysis package

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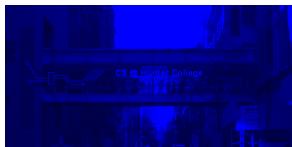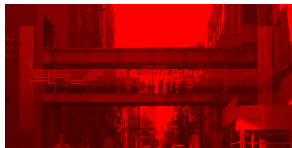
- We will use 2 useful packages for images:
  - ▶ `numpy`: numerical analysis package
  - ▶ `pyplot`: part of `matplotlib` for making graphs and plots

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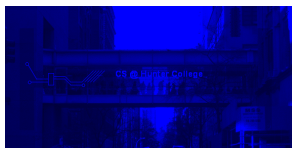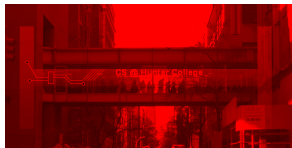
- We will use 2 useful packages for images:
  - ▶ `numpy`: numerical analysis package
  - ▶ `pyplot`: part of `matplotlib` for making graphs and plots
- See lab notes for installing on your home machine.

Images with `pyplot` and `numpy`
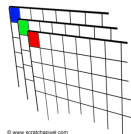

```python
#Import the packages for images and arrays:
import matplotlib.pyplot as plt
import numpy as np


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`

```python
#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
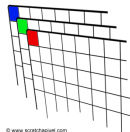
To create an image from scratch:

© www.scratchapixel.com

Creating Images

To create an image from scratch:
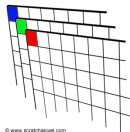
1. Import the libraries.

Creating Images

To create an image from scratch:

1. Import the libraries.

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


© www.scratchapixel.com

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

© www.scratchapixel.com

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) )
      ```

© www.scratchapixel.com

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):

© www.scratchapixel.com

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) )
      ```

© www.scratchapixel.com

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:

© www.scratchapixel.com

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()
   ```
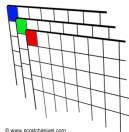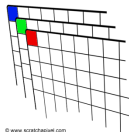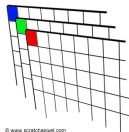
© www.scratchapixel.com

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:

© www.scratchapixel.com

Creating Images
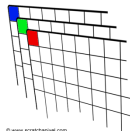
To create an image from scratch:

1. Import the libraries.

   ```
   import matplotlib.pyplot as plt
   import numpy as np
   ```
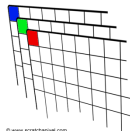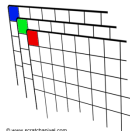
2. Create the image– easy to set all color

   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)
   ```

© www.scratchapixel.com
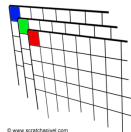
More on `numpy` arrays

```
>>> a[0,3:5]
array([3,4])


>>> a[4:,4:]
array([[44, 45],
       [54, 55]])


>>> a[:,2]
array([2,12,22,32,42,52])


>>> a[2::2,::2]
array([[20,22,24]
       [40,42,44]])
```


numpy tutorial

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

- 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

- 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

- 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) )
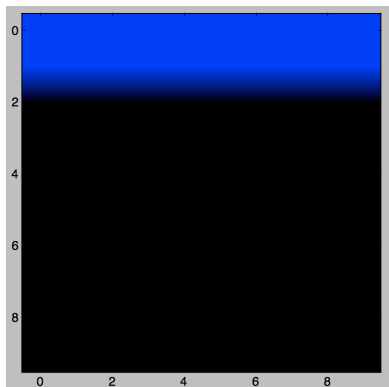    img[::2,::2,0] = 1
  ```

Challenge

- 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) )
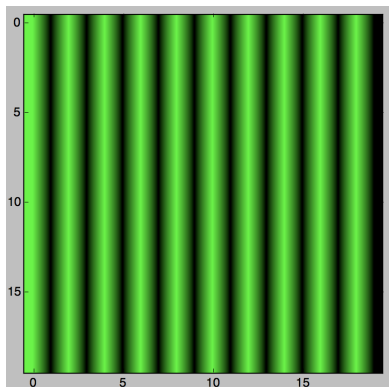    img[0:10,0:5,0:2] = 0
  ```

Challenge

- 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

- 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

- 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

- 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
    ```
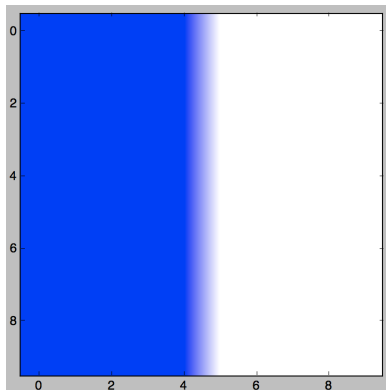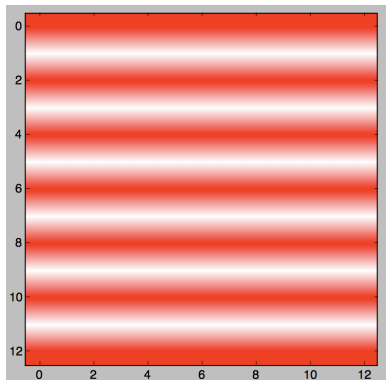
Challenge

- 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

1. Design a 10 by 10 logo for Hunter College that contains a purple 'H'.

Challenge

1. Design a 10 by 10 logo for Hunter College that contains a purple 'H'.
2. Your logo should only contain the colors purple and white.

Challenge

1. Design a 10 by 10 logo for Hunter College that contains a purple 'H'.
2. Your logo should only contain the colors purple and white.
3. How can you make Python draw the logo?
   Write down a "To Do" list of things you need to do.

Challenge

1. Design a 10 by 10 logo for Hunter College that contains a purple 'H'.
2. Your logo should only contain the colors purple and white.
3. How can you make Python draw the logo?
   Write down a "To Do" list of things you need to do.
4. If time, refine your steps above into a Python program.

Design a Hunter Logo

One possible solution:

Design a Hunter Logo

1. Create a 10 by 10 array, logo, that starts out as all white pixels.

Design a Hunter Logo

1. Create a 10 by 10 array, logo, that starts out as all white pixels.
2. Set the 3 left columns to be purple.

Design a Hunter Logo

1. Create a 10 by 10 array, logo, that starts out as all white pixels.
2. Set the 3 left columns to be purple.
3. Set the 3 right columns to be purple.

Design a Hunter Logo

1. Create a 10 by 10 array, logo, that starts out as all white pixels.
2. Set the 3 left columns to be purple.
3. Set the 3 right columns to be purple.
4. Set the middle 2 rows to be purple.

Design a Hunter Logo

1. Create a 10 by 10 array, logo, that starts out as all white pixels.
2. Set the 3 left columns to be purple.
3. Set the 3 right columns to be purple.
4. Set the middle 2 rows to be purple.
5. Save logo array to a file.

Translating the Design to Code

1. Create a 10 by 10 array, `logo`, that starts out as all white pixels.

Translating the Design to Code

1. Create a 10 by 10 array, `logo`, that starts out as all white pixels.

```
import matplotlib.pyplot as plt #import libraries for plotting
import numpy as np              #and for arrays (to hold images)
logoImg = np.ones((10,10,3))    #10x10 array with 3 sheets of 1's
```

Translating the Design to Code

1. Create a 10 by 10 array, logo, that starts out as all white pixels.

```
import matplotlib.pyplot as plt #import libraries for plotting
import numpy as np              #and for arrays (to hold images)
logoImg = np.ones((10,10,3))    #10x10 array with 3 sheets of 1's
```

Translating the Design to Code

1. Create a 10 by 10 array, logo, that starts out as all white pixels.

```
import matplotlib.pyplot as plt #import libraries for plotting
import numpy as np              #and for arrays (to hold images)
logoImg = np.ones((10,10,3))    #10x10 array with 3 sheets of 1's
```

2. Set the 3 left columns to be purple.

Translating the Design to Code

1. Create a 10 by 10 array, `logo`, that starts out as all white pixels.

```
import matplotlib.pyplot as plt #import libraries for plotting
import numpy as np              #and for arrays (to hold images)
logoImg = np.ones((10,10,3))    #10x10 array with 3 sheets of 1's
```

2. Set the 3 left columns to be purple.

```
#To make purple, we'll keep red and blue at 100% and turn green to 0%
logoImg[:,:3,1] = 0 #Turn the green to 0 for first 3 columns
```

Translating the Design to Code

1. Create a 10 by 10 array, `logo`, that starts out as all white pixels.
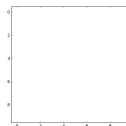
```python
import matplotlib.pyplot as plt #import libraries for plotting
import numpy as np              #and for arrays (to hold images)
logoImg = np.ones((10,10,3))    #10x10 array with 3 sheets of 1's
```

2. Set the 3 left columns to be purple.

```python
#To make purple, we'll keep red and blue at 100% and turn green to 0%
logoImg[:,:3,1] = 0 #Turn the green to 0 for first 3 columns
```

Translating the Design to Code

1. Create a 10 by 10 array, logo, that starts out as all white pixels.

```
import matplotlib.pyplot as plt  #import libraries for plotting
import numpy as np               #and for arrays (to hold images)
logoImg = np.ones((10,10,3))     #10x10 array with 3 sheets of 1's
```
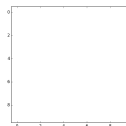
2. Set the 3 left columns to be purple.

```
#To make purple, we'll keep red and blue at 100% and turn green to 0%
logoImg[:,:3,1] = 0 #Turn the green to 0 for first 3 columns
```

3. Set the 3 right columns to be purple.

```
logoImg[:,-3:,1] = 0 #Turn the green to 0 for last 3 columns
```
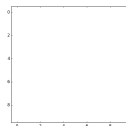
Translating the Design to Code

1. Create a 10 by 10 array, `logo`, that starts out as all white pixels.

```
import matplotlib.pyplot as plt #import libraries for plotting
import numpy as np               #and for arrays (to hold images)
logoImg = np.ones((10,10,3))     #10x10 array with 3 sheets of 1's
```

2. Set the 3 left columns to be purple.

```
#To make purple, we'll keep red and blue at 100% and turn green to 0%
logoImg[:,:3,1] = 0 #Turn the green to 0 for first 3 columns
```
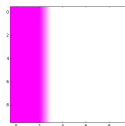

3. Set the 3 right columns to be purple.

```
logoImg[:,-3:,1] = 0 #Turn the green to 0 for last 3 columns
```

Translating the Design to Code

1. Create a 10 by 10 array, `logo`, that starts out as all white pixels.
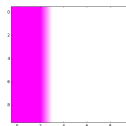
```
import matplotlib.pyplot as plt #import libraries for plotting
import numpy as np              #and for arrays (to hold images)
logoImg = np.ones((10,10,3))    #10x10 array with 3 sheets of 1's
```

2. Set the 3 left columns to be purple.

```
#To make purple, we'll keep red and blue at 100% and turn green to 0%
logoImg[:,:3,1] = 0 #Turn the green to 0 for first 3 columns
```

3. Set the 3 right columns to be purple.

```
logoImg[:,-3:,1] = 0 #Turn the green to 0 for last 3 columns
```

4. Set the middle 2 rows to be purple.

Translating the Design to Code

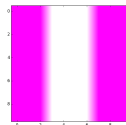1. Create a 10 by 10 array, `logo`, that starts out as all white pixels.

```
import matplotlib.pyplot as plt #import libraries for plotting
import numpy as np              #and for arrays (to hold images)
logoImg = np.ones((10,10,3))    #10x10 array with 3 sheets of 1's
```

2. Set the 3 left columns to be purple.

```
#To make purple, we'll keep red and blue at 100% and turn green to 0%
logoImg[:,:3,1] = 0 #Turn the green to 0 for first 3 columns
```

3. Set the 3 right columns to be purple.

```
logoImg[:,-3:,1] = 0 #Turn the green to 0 for last 3 columns
```

4. Set the middle 2 rows to be purple.

```
logoImg[4:6,:,1] = 0 #Turn the green to 0 for middle rows
```

Translating the Design to Code

1. Create a 10 by 10 array, `logo`, that starts out as all white pixels.

```
import matplotlib.pyplot as plt #import libraries for plotting
import numpy as np              #and for arrays (to hold images)
logoImg = np.ones((10,10,3))    #10x10 array with 3 sheets of 1's
```
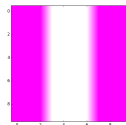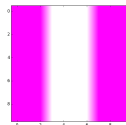
2. Set the 3 left columns to be purple.

```
#To make purple, we'll keep red and blue at 100% and turn green to 0%
logoImg[:,:3,1] = 0 #Turn the green to 0 for first 3 columns
```


3. Set the 3 right columns to be purple.

```
logoImg[:,-3:,1] = 0 #Turn the green to 0 for last 3 columns
```

4. Set the middle 2 rows to be purple.

```
logoImg[4:6,:,1] = 0 #Turn the green to 0 for middle rows
```
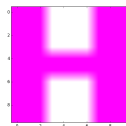
Translating the Design to Code

1. Create a 10 by 10 array, `logo`, that starts out as all white pixels.

```
import matplotlib.pyplot as plt #import libraries for plotting
import numpy as np              #and for arrays (to hold images)
logoImg = np.ones((10,10,3))    #10x10 array with 3 sheets of 1's
```

2. Set the 3 left columns to be purple.

```
#To make purple, we'll keep red and blue at 100% and turn green to 0%
logoImg[:,:3,1] = 0 #Turn the green to 0 for first 3 columns
```

3. Set the 3 right columns to be purple.

```
logoImg[:,-3:,1] = 0 #Turn the green to 0 for last 3 columns
```

4. Set the middle 2 rows to be purple.

```
logoImg[4:6,:,1] = 0 #Turn the green to 0 for middle rows
```
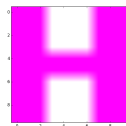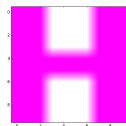
5. Save logo array to file.

Translating the Design to Code

1. Create a 10 by 10 array, `logo`, that starts out as all white pixels.

```
import matplotlib.pyplot as plt #import libraries for plotting
import numpy as np              #and for arrays (to hold images)
logoImg = np.ones((10,10,3))    #10x10 array with 3 sheets of 1's
```

2. Set the 3 left columns to be purple.

```
#To make purple, we'll keep red and blue at 100% and turn green to 0%
logoImg[:,:3,1] = 0 #Turn the green to 0 for first 3 columns
```

3. Set the 3 right columns to be purple.

```
logoImg[:,-3:,1] = 0 #Turn the green to 0 for last 3 columns
```

4. Set the middle 2 rows to be purple.

```
logoImg[4:6,:,1] = 0 #Turn the green to 0 for middle rows
```

5. Save logo array to file.

```
plt.imsave("logo.png", logoImg) #Save the image to logo.png
```

Today's Topics

- Recap: Colors
- 2D Arrays & Image Files
- **Decisions**
- Design Challenge: Airplanes

Challenge

Predict what these will do (novel concepts):

```python
yearBorn = int(input('Enter year born: '))
if yearBorn < 1946:
    print("Greatest Generation")
elif yearBorn <= 1964:
    print("Baby Boomer")
elif yearBorn <= 1984:
    print("Generation X")
elif yearBorn <= 2004:
    print("Millennial")
else:
    print("TBD")

x = int(input('Enter number: '))
if x % 2 == 0:
    print('Even number')
else:
    print('Odd number')
```

```python
import turtle

tess = turtle.Turtle()
myWin = turtle.Screen()          #The graphics window
commands = input("Please enter a command string: ")

for ch in commands:
    #perform action indicated by the character
    if ch == 'F':                #move forward
        tess.forward(50)
    elif ch == 'L':              #turn left
        tess.left(90)
    elif ch == 'R':              #turn right
        tess.right(90)
    elif ch == '^':              #lift pen
        tess.penup()
    elif ch == 'v':              #lower pen
        tess.pendown()
    elif ch == 'B':              #go backwards
        tess.backward(50)
    elif ch == 'r':              #turn red
        tess.color("red")
    elif ch == 'g':              #turn green
        tess.color("green")
    elif ch == 'b':              #turn blue
        tess.color("blue")
    else:                        #for any other character
        print("Error: do not know the command:", c)
```

Python Tutor

```python
yearBorn = int(input('Enter year born: '))
if yearBorn < 1946:
    print("Greatest Generation")
elif yearBorn <= 1964:
    print("Baby Boomer")
elif yearBorn <= 1984:
    print("Generation X")
elif yearBorn <= 2004:
    print("Millennial")
else:
    print("TBD")

x = int(input('Enter number: '))
if x % 2 == 0:
    print('Even number')
else:
    print('Odd number')
```

(Demo with `pythonTutor`)

IDLE

```python
import turtle

tess = turtle.Turtle()
myWin = turtle.Screen()          #The graphics window
commands = input("Please enter a command string: ")

for ch in commands:
    #perform action indicated by the character
    if ch == 'F':                #move forward
        tess.forward(50)
    elif ch == 'L':              #turn left
        tess.left(90)
    elif ch == 'R':              #turn right
        tess.right(90)
    elif ch == '^':              #lift pen
        tess.penup()
    elif ch == 'v':              #lower pen
        tess.pendown()
    elif ch == 'B':              #go backwards
        tess.backward(50)
    elif ch == 'r':              #turn red
        tess.color("red")
    elif ch == 'g':              #turn green
        tess.color("green")
    elif ch == 'b':              #turn blue
        tess.color("blue")
    else:                        #for any other character
        print("Error: do not know the command:", c)
```

(Demo with `IDLE`)

Decisions

```python
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")
```

Lecture 4

Decisions

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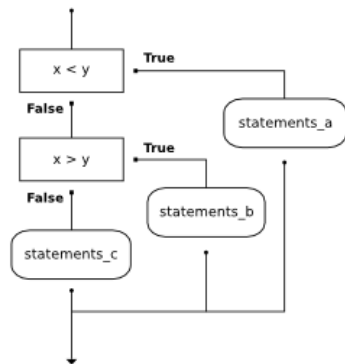
Decisions
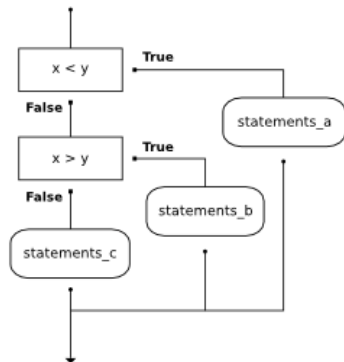
```python
if x < y:
    print("x is less than y")
elif x > y:
    print("x is greater than y")
else:
    print("x and y must be equal")
```
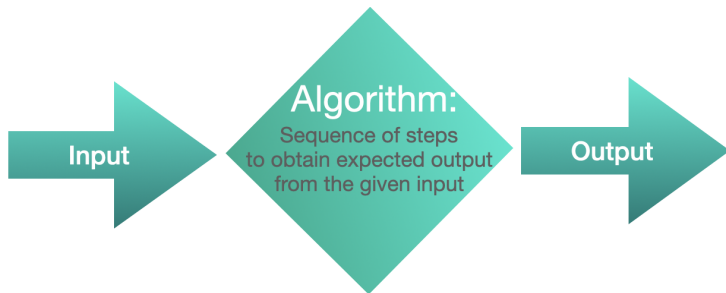

(This was just a first glance, will do much more on decisions over the next several weeks.)

Today's Topics

- Recap: Colors
- 2D Arrays & Image Files
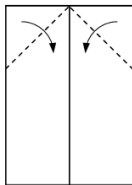- Decisions
- **Design Challenge: Airplanes**
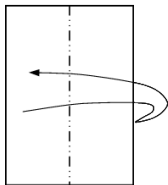
What is an Algorithm?

Characteristics of an Algorithm
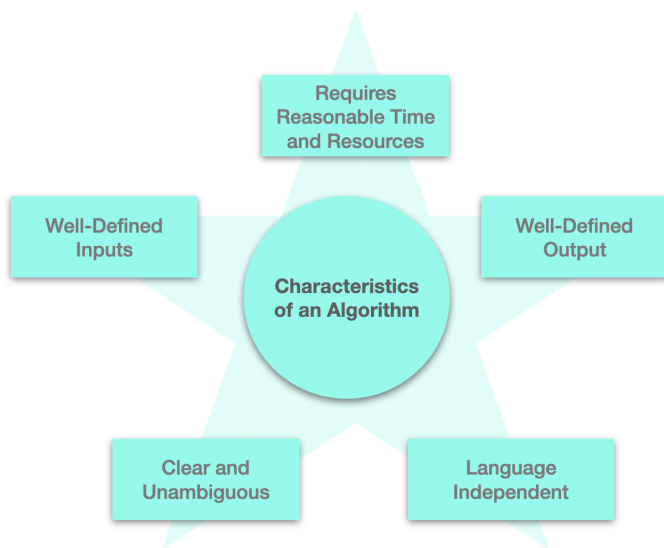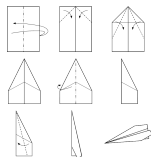
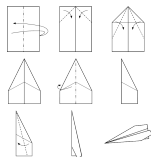# Algorithm Design Cycle
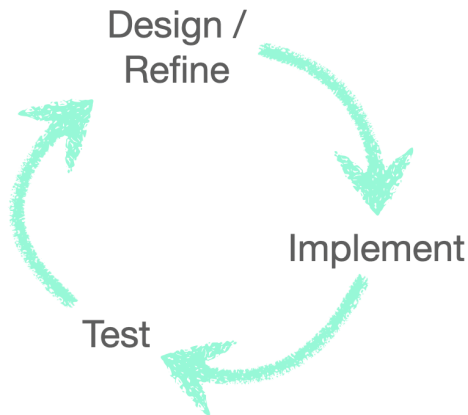
Design Challenge: Planes
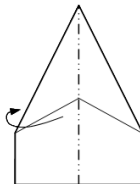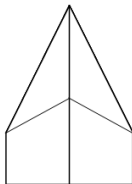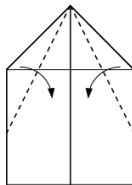
Design Challenge: Planes

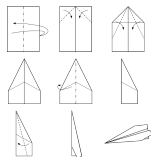- A classic write-an-algorithm challenge for introductory programming.

Design Challenge: Planes

- A classic write-an-algorithm challenge for introductory programming.
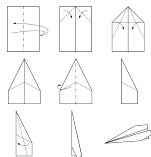
- With a slight twist:

Design Challenge: Planes

- A classic write-an-algorithm challenge for introductory programming.

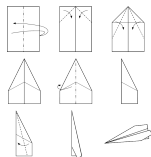- With a slight twist: refining designs

Design Challenge: Planes

- A classic write-an-algorithm challenge for introductory programming.

- With a slight twist: refining designs

- After class:

Design Challenge: Planes

- A classic write-an-algorithm challenge for introductory programming.

- With a slight twist: refining designs

- After class:
    - Write down your design (different style from the one in Lecture Quiz).
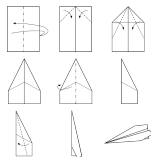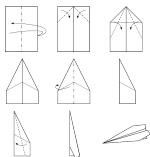
Design Challenge: Planes

- A classic write-an-algorithm challenge for introductory programming.

- With a slight twist: refining designs

- After class:
  - ▶ Write down your design (different style from the one in Lecture Quiz).
  - ▶ Exchange your design and a blank sheet of paper (**Input**) with a family member or friend.
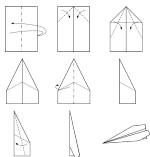
Design Challenge: Planes

- A classic write-an-algorithm challenge for introductory programming.

- With a slight twist: refining designs

- After class:

  - Write down your design (different style from the one in Lecture Quiz).
  - Exchange your design and a blank sheet of paper (**Input**) with a family member or friend.
  - Ask them to follow your design to build an airplane **without consulting you**.

Design Challenge: Planes

- A classic write-an-algorithm challenge for introductory programming.

- With a slight twist: refining designs

- After class:

  ▶ Write down your design (different style from the one in Lecture Quiz).

  ▶ Exchange your design and a blank sheet of paper (**Input**) with a family member or friend.

  ▶ Ask them to follow your design to build an airplane **without consulting you**.

  ▶ When they are done, observe the folded airplange (**Output**) and **revise your algorithm**.

Design Challenge: Planes

- A classic write-an-algorithm challenge for introductory programming.

- With a slight twist: refining designs

- After class:
  - ▶ Write down your design (different style from the one in Lecture Quiz).
  - ▶ Exchange your design and a blank sheet of paper (**Input**) with a family member or friend.
  - ▶ Ask them to follow your design to build an airplane **without consulting you**.
  - ▶ When they are done, observe the folded airplange (**Output**) and **revise your algorithm**.
  - ▶ Repeat until you are satisfied with your airplane.
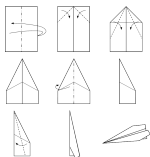
Lecture Quiz

- Log-in to Gradescope
- Find LECTURE 4 Quiz
- Take the quiz
- **You have 5 minutes**

Recap
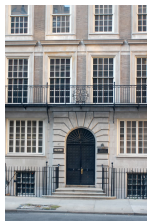

- In Python, we introduced:

Recap

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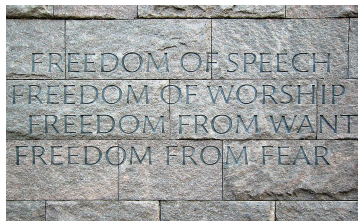   - 2D Array & Image Files
    - Decisions

Practice Quiz & Final Questions
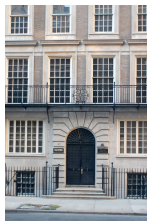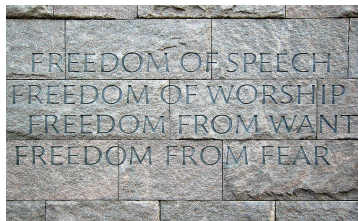

(NYTimes)

(Hunter College)

(FDR 4 FP)

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.

Practice Quiz & Final Questions
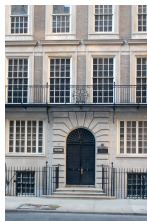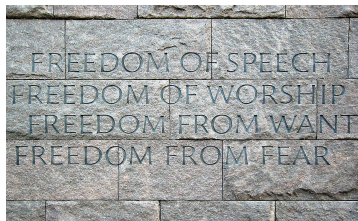

(NYTimes)

(Hunter College)

(FDR 4 FP)

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).

Practice Quiz & Final Questions
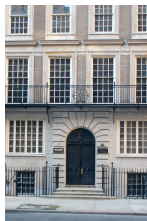

(NYTimes)

(Hunter College)

(FDR 4 FP)

FREEDOM OF SPEECH
FREEDOM OF WORSHIP
FREEDOM FROM WANT
FREEDOM FROM FEAR

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
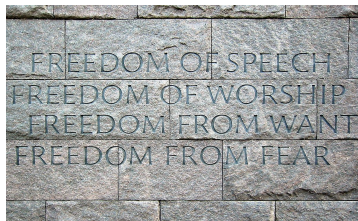- Lightning rounds:

Practice Quiz & Final Questions
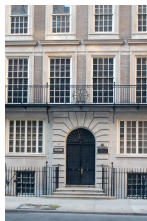

(NYTimes)

(Hunter College)

(FDR 4 FP)

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;
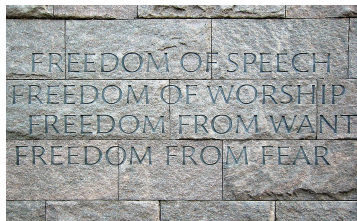
Practice Quiz & Final Questions

(NYTimes)   (Hunter College)   (FDR 4 FP)

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
    - write as much you can for 60 seconds;
    - followed by answer; and

Practice Quiz & Final Questions

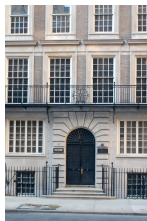(NYTimes)   (Hunter College)   (FDR 4 FP)

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and
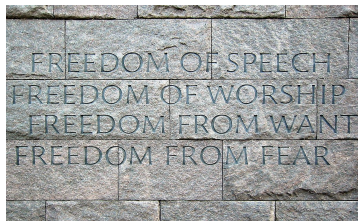 - repeat.

Practice Quiz & Final Questions

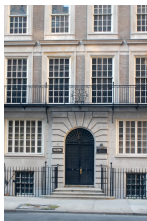(NYTimes)   (Hunter College)   (FDR 4 FP)

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
    ▶ write as much you can for 60 seconds;
    ▶ followed by answer; and
    ▶ repeat.
- Past exams are on the webpage (under Final Exam Information).
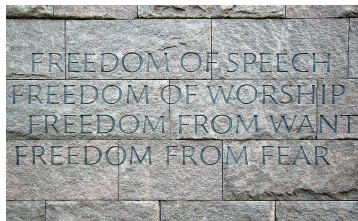
Practice Quiz & Final Questions

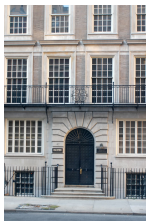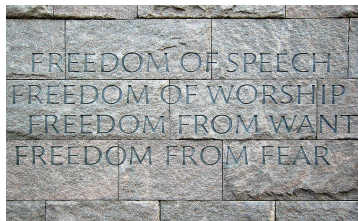(NYTimes)   (Hunter College)   (FDR 4 FP)

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
    - write as much you can for 60 seconds;
    - followed by answer; and
    - repeat.
- Past exams are on the webpage (under Final Exam Information).
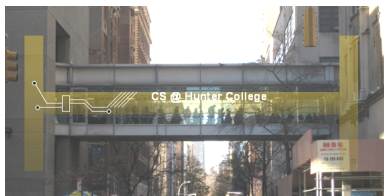- We're starting with Fall 2019, Version 1.

Weekly Reminders!

Before next lecture, don't forget to:

- Work on this week's Online Lab

Weekly Reminders!

Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
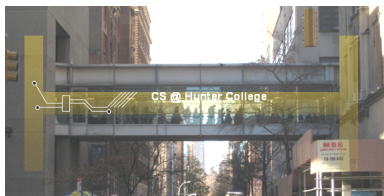
Weekly Reminders!

Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
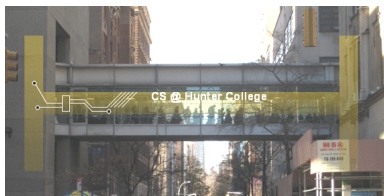- Take the Lab Quiz on Gradescope by 6pm on Wednesday

Weekly Reminders!

Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
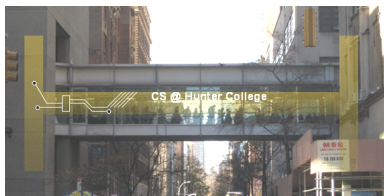- Submit this week's 5 programming assignments (programs 16-20)

Weekly Reminders!

Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 5 programming assignments (programs 16-20)
- At any point, visit our Drop-In Tutoring 11am-5pm for help!!!

Weekly Reminders!

Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 5 programming assignments (programs 16-20)
- At any point, visit our Drop-In Tutoring 11am-5pm for help!!!
- Take the Lecture Preview on Blackboard on Monday (or no later than 10am on Tuesday)