

# CSci 127: Introduction to Computer Science



# 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
- **Please use class time to ask questions!**

# 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
- **Please use class time to ask questions!**
- Make an office hours appointment for further review!

# Tutoring is Fundamental



# 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

# Challenge

- Need to `0x` in hexcodes (always start with `#`):

# Challenge

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

# Challenge

Need to ll in hexcodes (always start with #):R R G G B B

Black: 0 0 0 0 0 0

# Challenge

Need to ll 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

Need to ll 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

Need to ll 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

Need to ll 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

Need to ll 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

Can specify by name.

# Recap: Colors

Can specify by name.

Can specify by numbers:

# Recap: Colors

Can specify by name.

Can specify by numbers:

- | Amount of Red, Green, and Blue (RGB).

# Recap: Colors

Can specify by name.

Can specify by numbers:

- | Amount of Red, Green, and Blue (RGB).
- | Adding light, not paint:

# Recap: Colors

Can specify by name.

Can specify by numbers:

- | Amount of Red, Green, and Blue (RGB).
- | Adding light, not paint:
  - F Black: 0% red, 0% green, 0% blue

# Recap: Colors

Can specify by name.

Can specify by numbers:

- | Amount of Red, Green, and Blue (RGB).
- | Adding light, not paint:
  - F Black: 0% red, 0% green, 0% blue
  - F White: 100% red, 100% green, 100% blue

# Recap: Colors

Can specify by numbers (RGB):



# Recap: Colors

Can specify by numbers (RGB):

- ┆ Fractions of each:

e.g.  $(1.0, 0, 0)$  is 100% red, no green, and no blue.

# Recap: Colors

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

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

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

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

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

# 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 `matplotlib` and `numpy`

# Images with `matplotlib` and `numpy`

# 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 stu 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 stu 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 stu 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 stu 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 stu 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)
```

# More on numpy arrays

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[:, :, 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:

```
1 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:

```
1 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 ~~pill~~

# Challenge

- 1 Design a 10 by 10 logo for Hunter College that contains a purple ~~pill~~
- 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 ~~line~~
- 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 ~~pill~~
- 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, re ne your steps above into a Python program.

# Design a Hunter Logo

One possible solution:



# Design a Hunter Logo

- 1 Create a 10 by 10 arraylogo , 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.

```
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.

```
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, 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)
logolmg = 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%
logolmg[:, :3, 1] = 0 #Turn the green to 0 for first 3 columns
```

- 3 Set the 3 right columns to be purple.

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

# Translating the Design to Code

- 1 Create a 10 by 10 array, 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)
logolmg = 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%
logolmg[:, :3, 1] = 0 #Turn the green to 0 for first 3 columns
```

- 3 Set the 3 right columns to be purple.

```
logolmg[:, -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, 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)
logolmg = 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%
logolmg[:, :3, 1] = 0 #Turn the green to 0 for first 3 columns
```

- 3 Set the 3 right columns to be purple.

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

- 4 Set the middle 2 rows to be purple.

```
logolmg[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)
logolmg = 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%
logolmg[:,3,1] = 0 #Turn the green to 0 for first 3 columns
```

- 3 Set the 3 right columns to be purple.

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

- 4 Set the middle 2 rows to be purple.

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

- 5 Save `logo` array to `le`.

# 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)
logolmg = 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%
logolmg[:,3,1] = 0 #Turn the green to 0 for first 3 columns
```

- 3 Set the 3 right columns to be purple.

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

- 4 Set the middle 2 rows to be purple.

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

- 5 Save `logo` array to `le`.

```
plt.imsave("logo.png", logolmg) #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 Tutor

(Demo with pythonTutor )

# IDLE

(Demo with IDLE)

# Decisions

# Decisions



# Decisions

(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: reining designs

# Design Challenge: Planes

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

With a slight twist: reining designs

After class:

# Design Challenge: Planes

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

With a slight twist: reining designs

After class:

- | Write down your design .

# Design Challenge: Planes

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

With a slight twist: re nining designs

After class:

- | Write down your design .
- | 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: re-ning designs

After class:

- | Write down your design .
- | 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 .
- | 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 airplane (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 .
- | 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 airplane (Output ) and revise your algorithm .
- | Repeat until you are satisfied with your airplane.

# Recap

In Python, we introduced:



# Recap

In Python, we introduced:

- | Recap: Colors
- | 2D Array & Image Files
- | Decisions

# Weekly Reminders!

Before next lecture, don't forget to:  
Review this week's Lecture and Lab

# Weekly Reminders!

Before next lecture, don't forget to:

Review this week's Lecture and Lab

Take the Lab Quiz on Gradescope by 9pm today

# Weekly Reminders!



Before next lecture, don't forget to:

- Review this week's Lecture and Lab
- Take the Lab Quiz on Gradescope by 9pm today
- Submit this class's 5 programming assignments (programs 16-20)