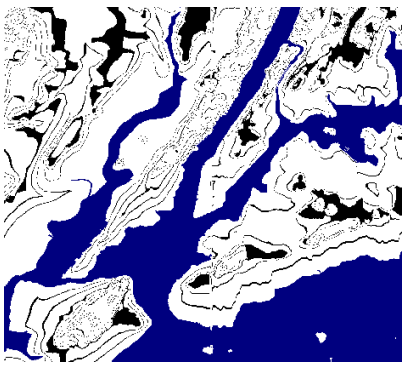
MfA: Python in the City

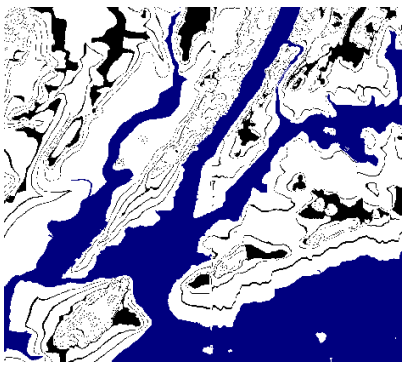
Katherine St. John
City University of New York
American Museum of Natural History

Outline



- Welcome and Introductions
- Workshop Overview
- Design Challenge: Flood Maps
  - Data: Elevations from NOAA
  - Packages: `numpy` and `pyplot`
  - Concepts: Colors and Arrays
  - Putting It Altogether
- Break
- Variations on the Theme
- Design a Challenge
- Wrap Up

Outline



- Welcome and Introductions
- Workshop Overview
- Design Challenge: Flood Maps
  - Data: Elevations from NOAA
  - Packages: `numpy` and `pyplot`
  - Concepts: Colors and Arrays
  - Putting It Altogether
- Break
- Variations on the Theme
- Design a Challenge
- Wrap Up

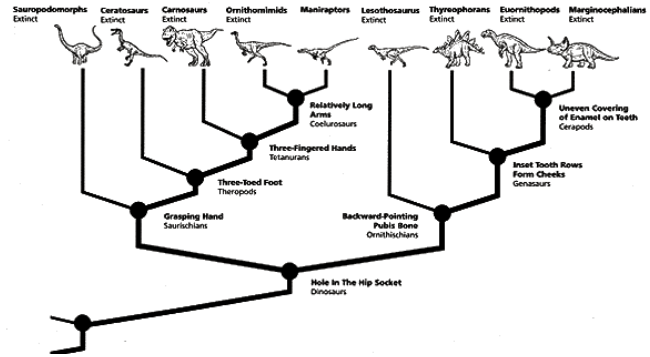
Welcome and Introductions

Welcome and Introductions

Katherine St. John

- Professor, City University of New York
- Research Associate, American Museum of Natural History
- Postdoctoral Work at University of Texas and University of Pennsylvania
- PhD, UCLA
- MA, Johns Hopkins University
- AB, Smith College

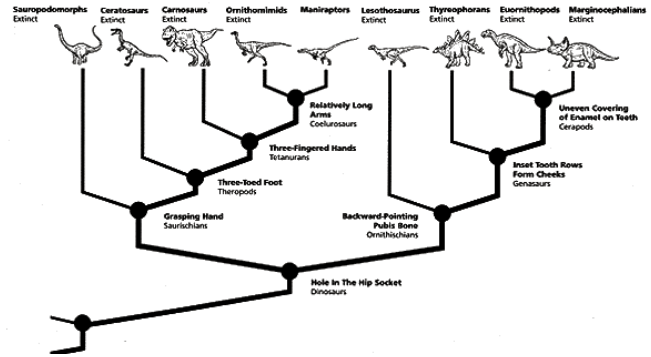
Welcome and Introductions



Katherine St. John

- Professor, City University of New York
- Research Associate, American Museum of Natural History
- Postdoctoral Work at University of Texas and University of Pennsylvania
- PhD, UCLA
- MA, Johns Hopkins University
- AB, Smith College

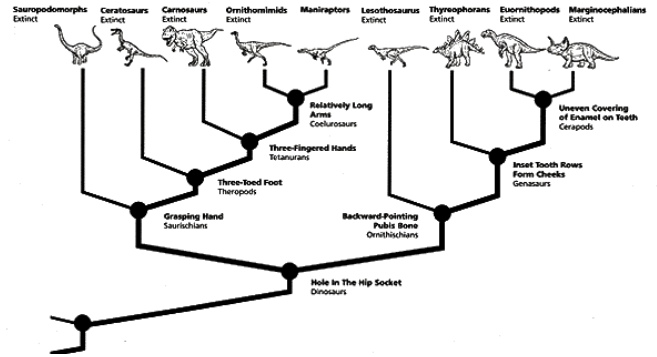
Welcome and Introductions



Katherine St. John

- Professor, City University of New York
- Research Associate, American Museum of Natural History
- Postdoctoral Work at University of Texas and University of Pennsylvania
- PhD, UCLA
- MA, Johns Hopkins University
- AB, Smith College

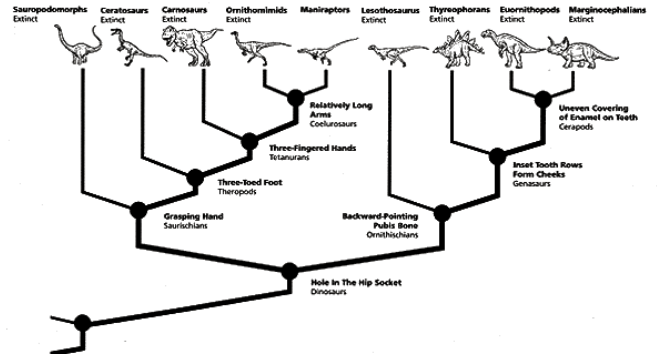
Welcome and Introductions



Katherine St. John

- Professor, City University of New York
- Research Associate, American Museum of Natural History
- Postdoctoral Work at University of Texas and University of Pennsylvania
- PhD, UCLA
- MA, Johns Hopkins University
- AB, Smith College

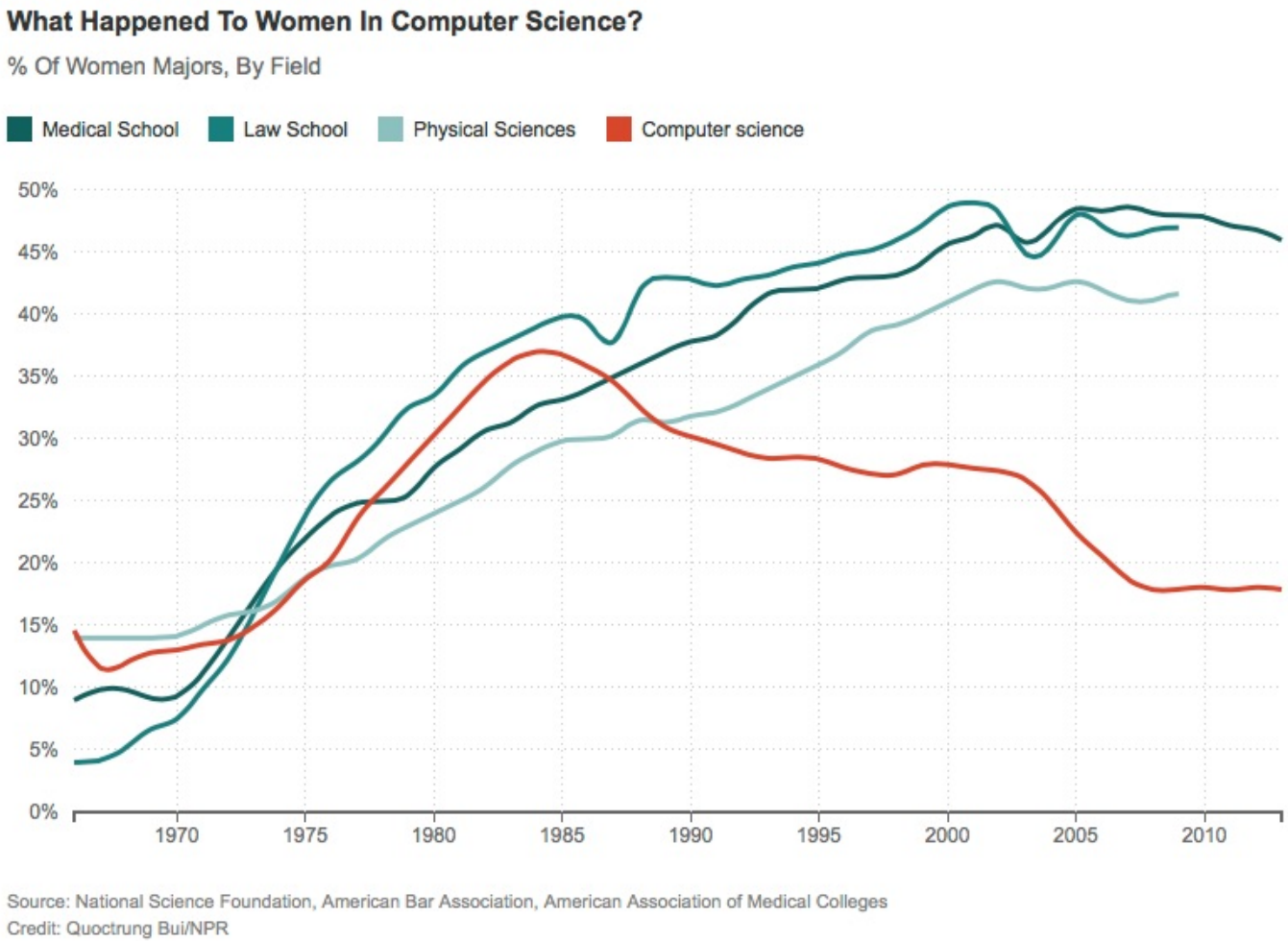
Welcome and Introductions



Katherine St. John

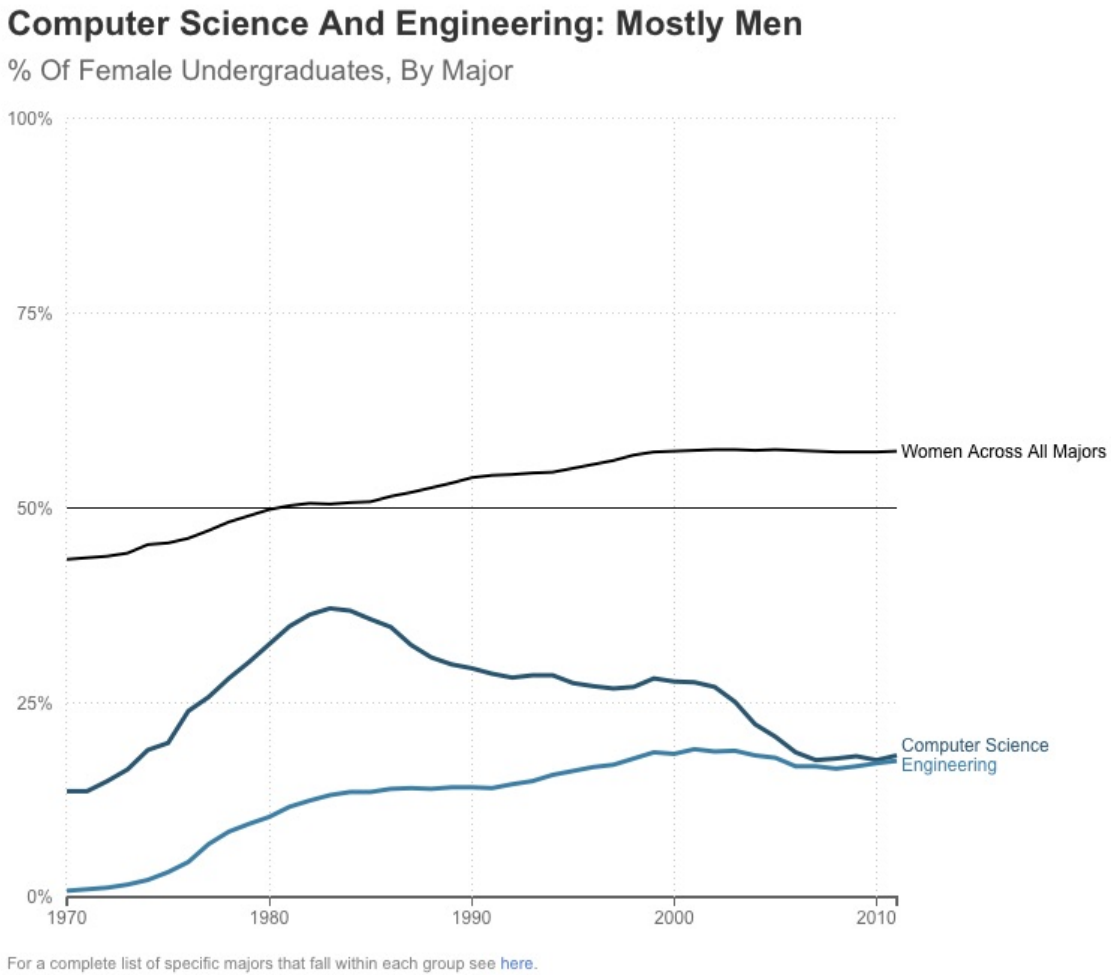
- Professor, City University of New York
- Research Associate, American Museum of Natural History
- Postdoctoral Work at University of Texas and University of Pennsylvania
- PhD, UCLA
- MA, Johns Hopkins University
- AB, Smith College

Why: Historical Trends: Women Majors



NPR Planet Money, October 2014

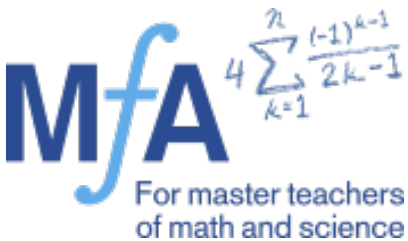
Why: Historical Trends: Computer Science & Engineering



NPR Planet Money, October 2014

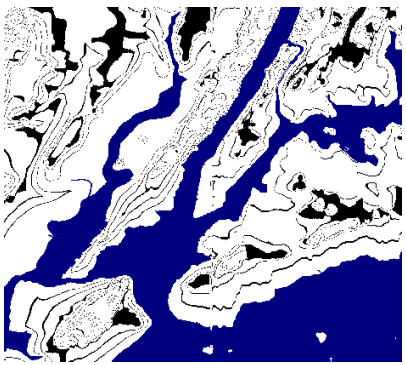
Introductions

Your name, what you teach, and something you're passionate about...



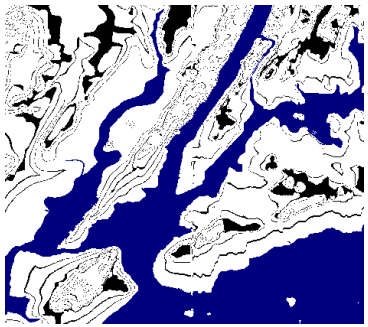
Adam	Bradley	Brandon	Brian
Christa	Conor	Daisy	David
Derek	George	Izagma	Joshua
Julia	Katie	Laura	Luis
Marisa	Matthew	Michael	Paul
Peter	Tom	Thomas	Topher

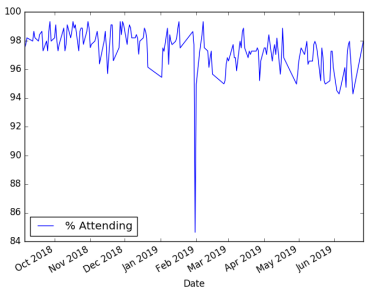
Outline

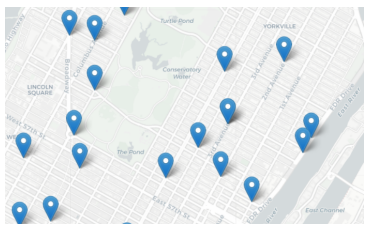


- Welcome and Introductions
- Workshop Overview
- Design Challenge: Flood Maps
  - Data: Elevations from NOAA
  - Packages: `numpy` and `pyplot`
  - Concepts: Colors and Arrays
  - Putting It Altogether
- Break
- Variations on the Theme
- Design a Challenge
- Wrap Up

Workshop Overview



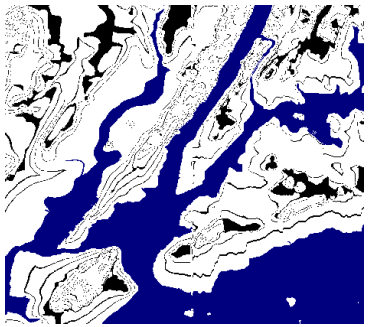


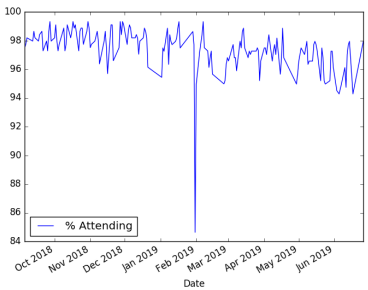


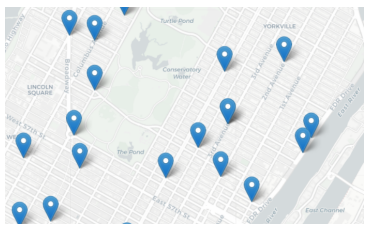
Three sessions:

1. Flood Maps (arrays & images)
2. Noisiest Street (structured data, file I/O)
3. Mapping Collisions (using objects, mapping coordinates)

Workshop Overview







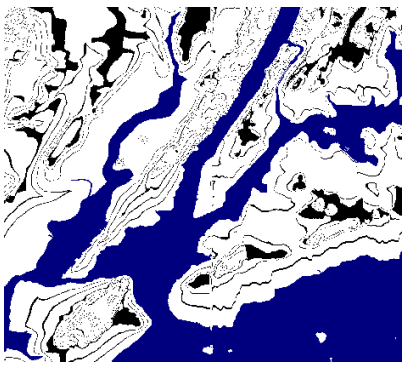
Three sessions:

1. Flood Maps (arrays & images)
2. Noisiest Street (structured data, file I/O)
3. Mapping Collisions (using objects, mapping coordinates)

Each session:

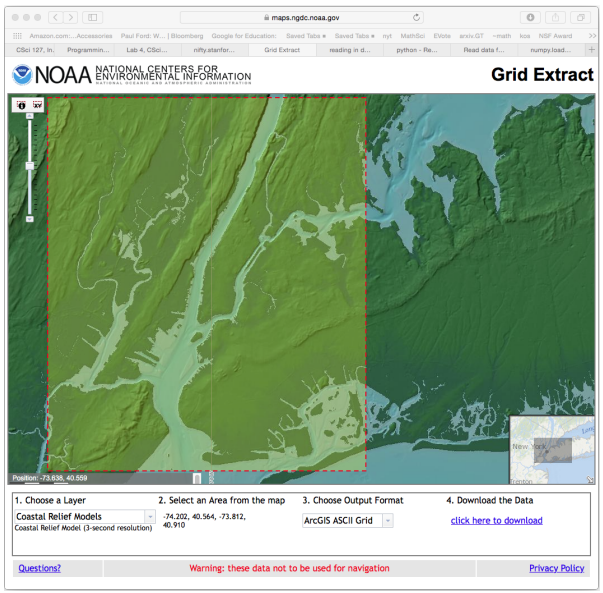
- Design Challenge
  - Analyze a publicly available dataset
  - Introduce computing concepts & packages
  - Write a program to solve the problem
- Variations on the theme
- Design a Challenge

Outline



- Welcome and Introductions
- Workshop Overview
- Design Challenge: Flood Maps
  - Data: Elevations from NOAA
  - Packages: `numpy` and `pyplot`
  - Concepts: Colors and Arrays
  - Putting It Altogether
- Break
- Variations on the Theme
- Design a Challenge
- Wrap Up

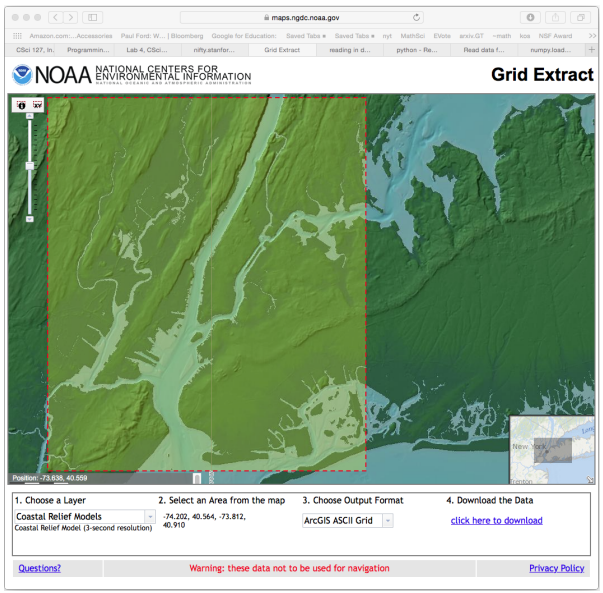
Design Challenge: Flood Maps



Today's opening design challenge is:

How much of New York City (and surrounds) will be flooded in the next big storm?

Design Challenge: Flood Maps



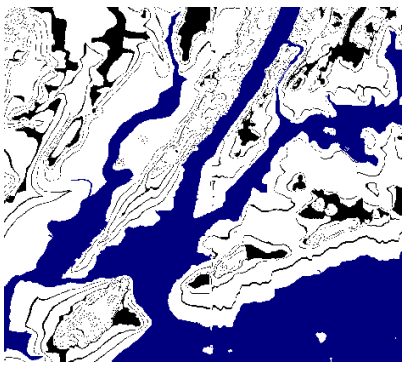
Today's opening design challenge is:

> *How much of New York City (and surrounds) will be flooded in the next big storm?*

Some questions to discuss with your group to get started:

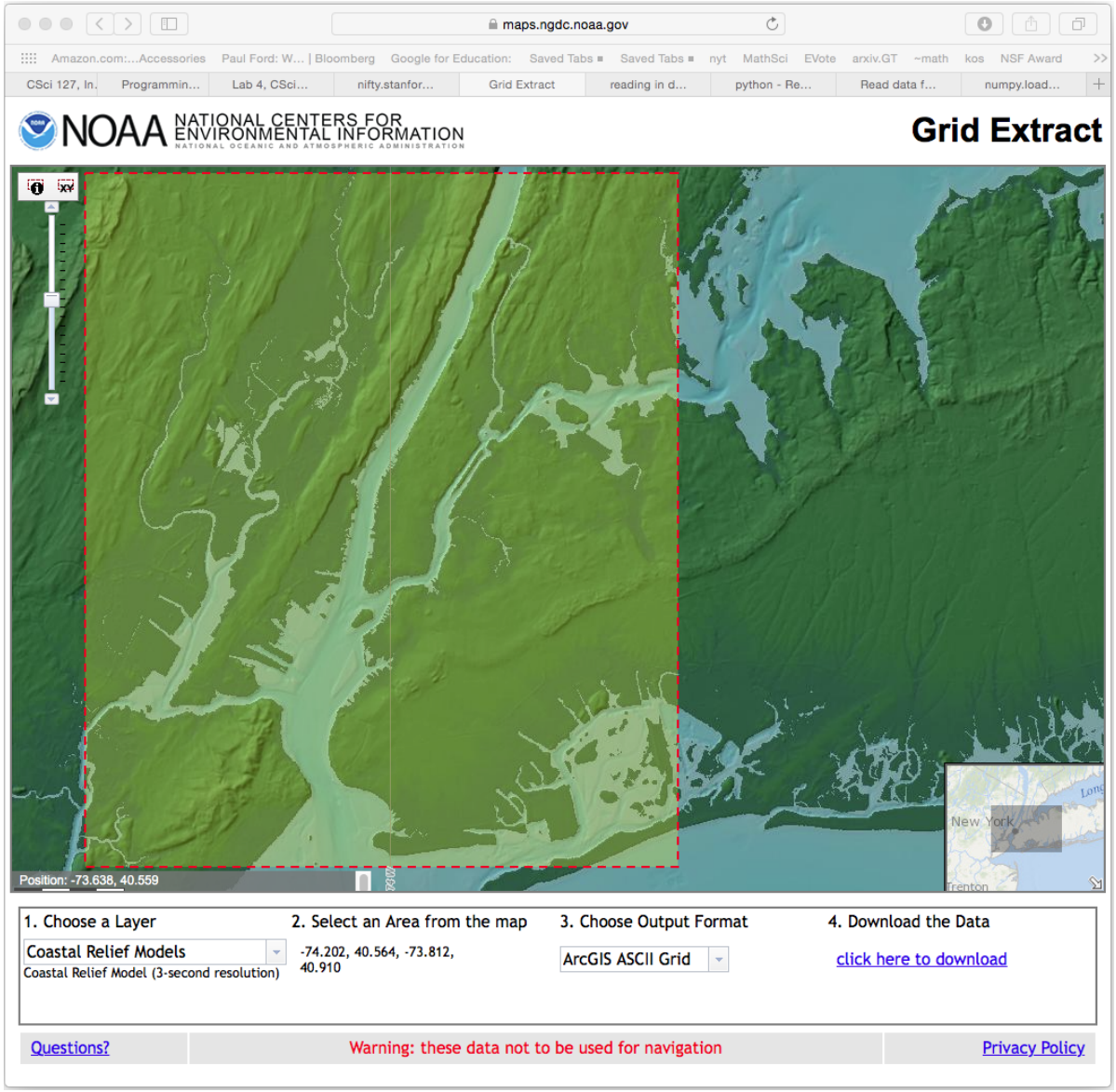
- What is the input? What data would you need to determine flooding?
- What is the output? How can you best present your results?

Outline



- Welcome and Introductions
- Workshop Overview
- Design Challenge: Flood Maps
  - Data: Elevations from NOAA
  - Packages: `numpy` and `pyplot`
  - Concepts: Colors and Arrays
  - Putting It Altogether
- Break
- Variations on the Theme
- Design a Challenge
- Wrap Up

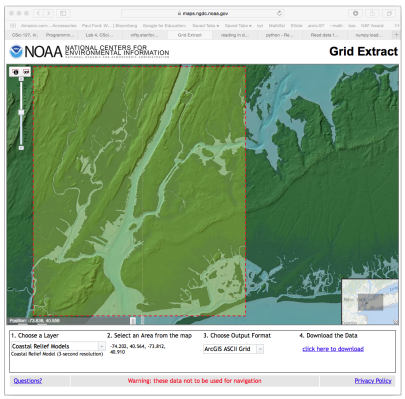
Data: Elevations from NOAA



Data: Elevations from NOAA

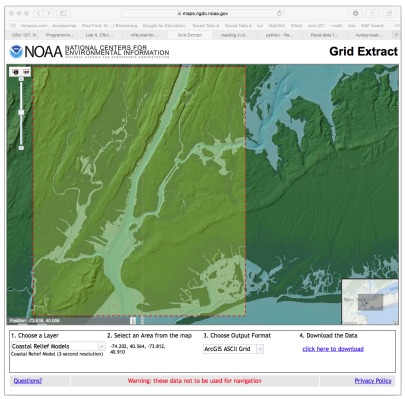
`https://maps.ngdc.noaa.gov/viewers/wcs-client/`

- Your turn: download a dataset.



Data: Elevations from NOAA

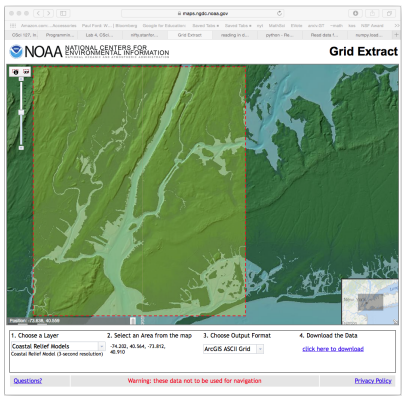
`https://maps.ngdc.noaa.gov/viewers/wcs-client/`



- Your turn: download a dataset.
- The US National Oceanic and Atmospheric Administration (NOAA) provides a searchable database of elevation data.

Data: Elevations from NOAA

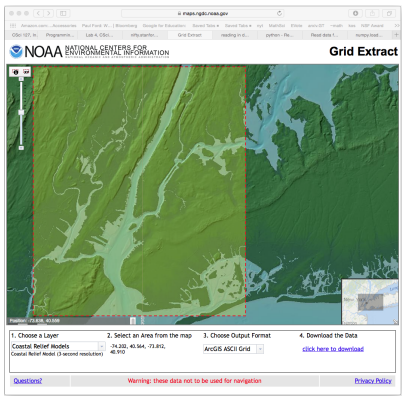
`https://maps.ngdc.noaa.gov/viewers/wcs-client/`



- Your turn: download a dataset.
- The US National Oceanic and Atmospheric Administration (NOAA) provides a searchable database of elevation data.
- To extract a grid of elevations, used:
  - Layer: Coastal Relief Models

Data: Elevations from NOAA

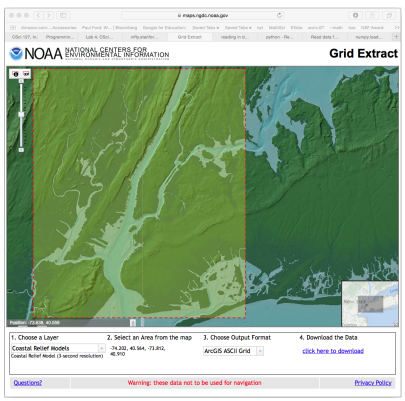
`https://maps.ngdc.noaa.gov/viewers/wcs-client/`



- Your turn: download a dataset.
- The US National Oceanic and Atmospheric Administration (NOAA) provides a searchable database of elevation data.
- To extract a grid of elevations, used:
  - Layer: Coastal Relief Models
  - Select a region (clunky, but works)

Data: Elevations from NOAA

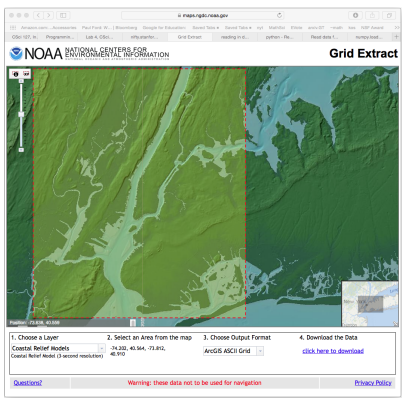
`https://maps.ngdc.noaa.gov/viewers/wcs-client/`



- Your turn: download a dataset.
- The US National Oceanic and Atmospheric Administration (NOAA) provides a searchable database of elevation data.
- To extract a grid of elevations, used:
  - Layer: Coastal Relief Models
  - Select a region (clunky, but works)
  - Output format: ArcGIS ASCII Grid

Data: Elevations from NOAA

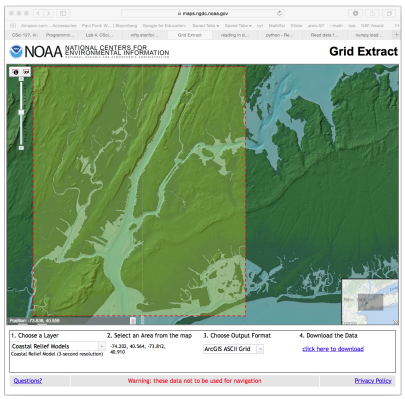
`https://maps.ngdc.noaa.gov/viewers/wcs-client/`



- Your turn: download a dataset.
- The US National Oceanic and Atmospheric Administration (NOAA) provides a searchable database of elevation data.
- To extract a grid of elevations, used:
  - Layer: Coastal Relief Models
  - Select a region (clunky, but works)
  - Output format: ArcGIS ASCII Grid
  - Downloaded text file of elevations

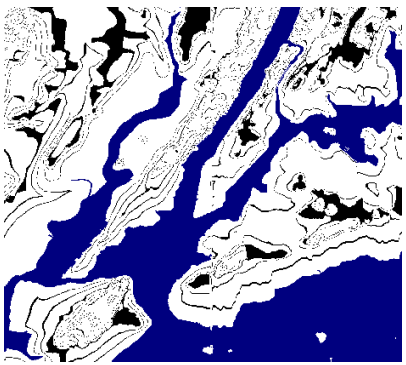
Data: Elevations from NOAA

`https://maps.ngdc.noaa.gov/viewers/wcs-client/`



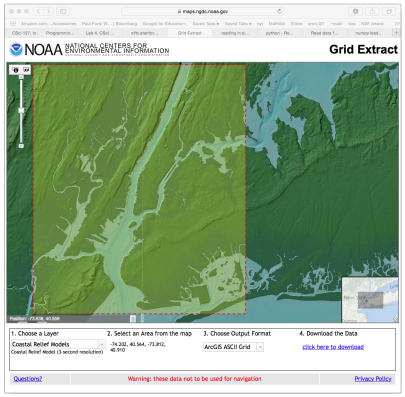
- Your turn: download a dataset.
- The US National Oceanic and Atmospheric Administration (NOAA) provides a searchable database of elevation data.
- To extract a grid of elevations, used:
  - Layer: Coastal Relief Models
  - Select a region (clunky, but works)
  - Output format: ArcGIS ASCII Grid
  - Downloaded text file of elevations
- Open in text editor and delete metadata (5 lines: # of rows and Columns, lat, lon) and rename as a text file, `elevationsNYC.txt`.

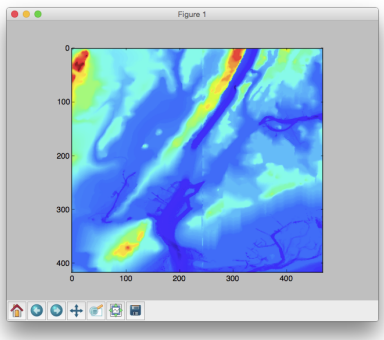
Outline

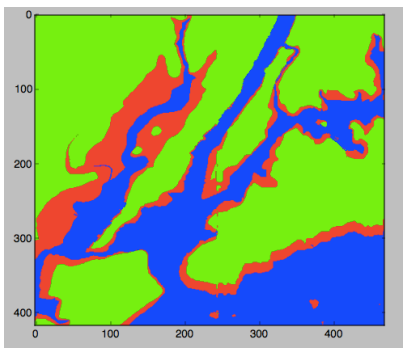


- Welcome and Introductions
- Workshop Overview
- Design Challenge: Flood Maps
  - Data: Elevations from NOAA
  - Packages: `numpy` and `pyplot`
  - Concepts: Colors and Arrays
  - Putting It Altogether
- Break
- Variations on the Theme
- Design a Challenge
- Wrap Up

Useful Packages: `numpy` and `pyplot`

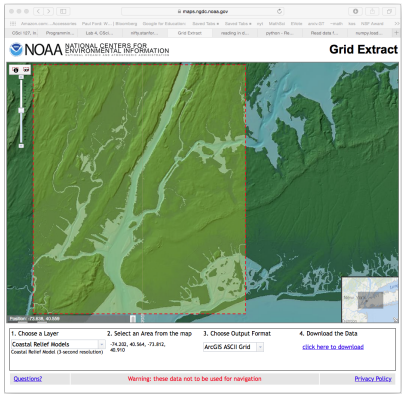


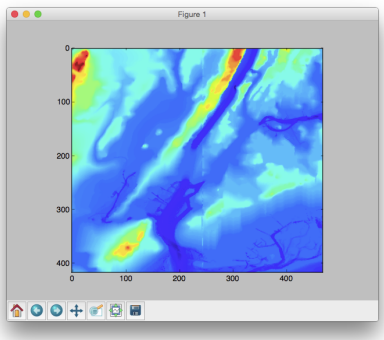


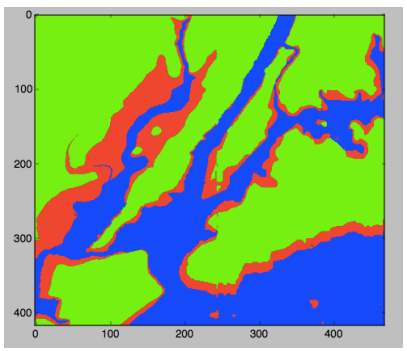


We will use two popular packages:

Useful Packages: `numpy` and `pyplot`



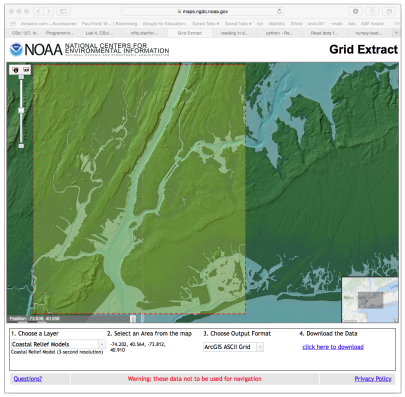


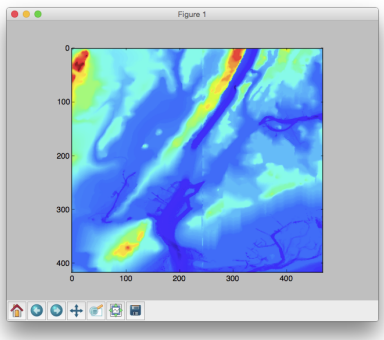


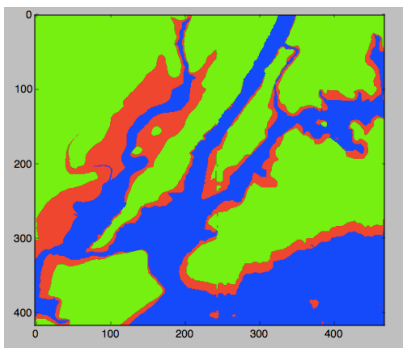
We will use two popular packages:

- `numpy`: numerical analysis package

Useful Packages: `numpy` and `pyplot`



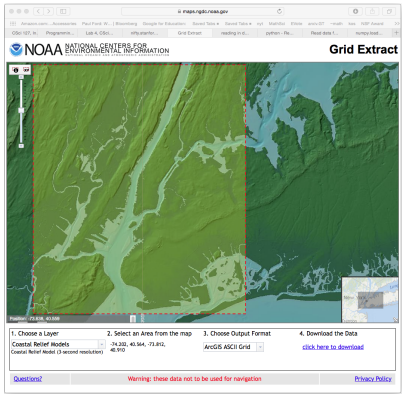


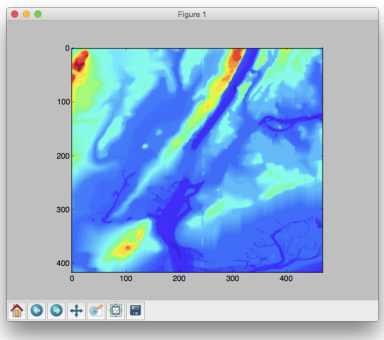


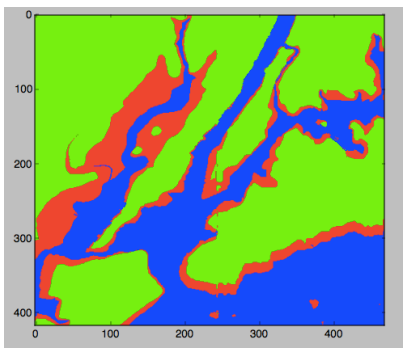
We will use two popular packages:

- `numpy`: numerical analysis package
- Standard abbreviation: `np`

Useful Packages: `numpy` and `pyplot`





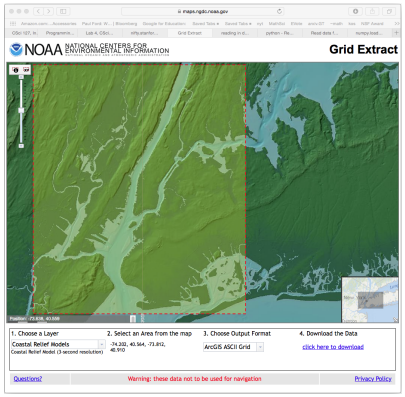


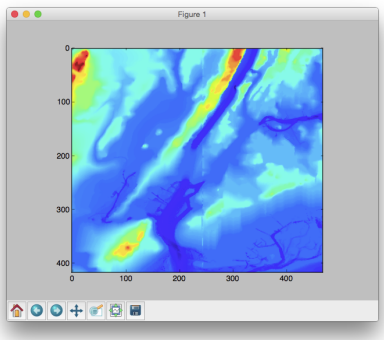
We will use two popular packages:

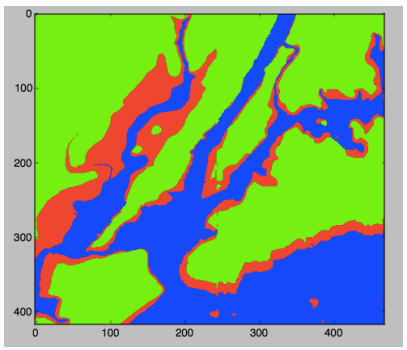
- `numpy`: numerical analysis package
- Standard abbreviation: `np`

```
import numpy as np
```

Useful Packages: `numpy` and `pyplot`







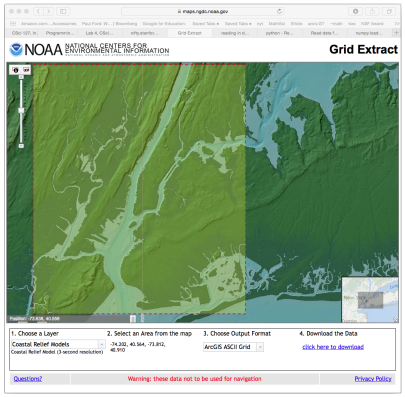
We will use two popular packages:

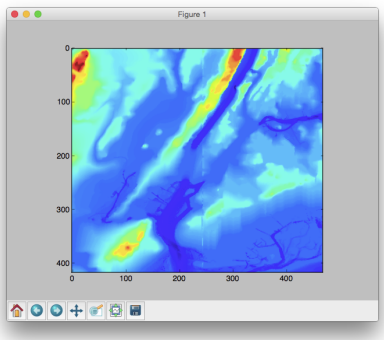
- `numpy`: numerical analysis package
- Standard abbreviation: `np`

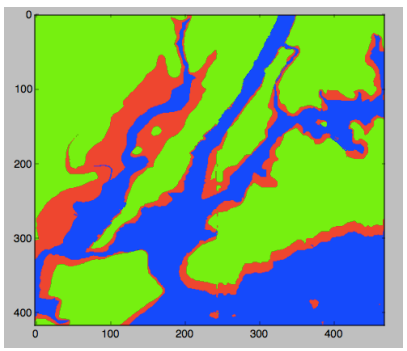
```
import numpy as np
```

- `pyplot`: part of `matplotlib` for making graphs and plots

Useful Packages: `numpy` and `pyplot`







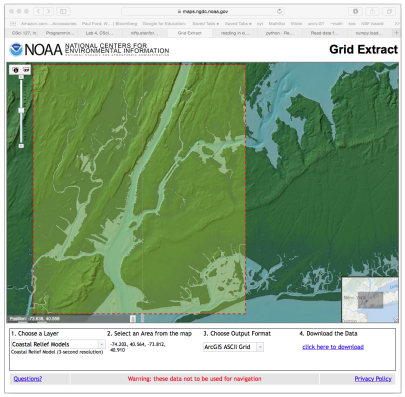
We will use two popular packages:

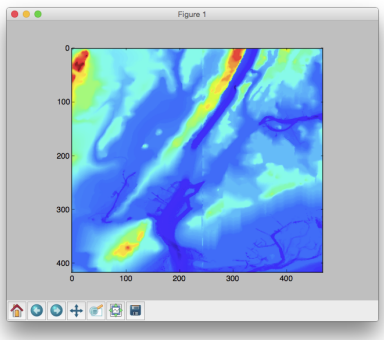
- `numpy`: numerical analysis package
- Standard abbreviation: `np`

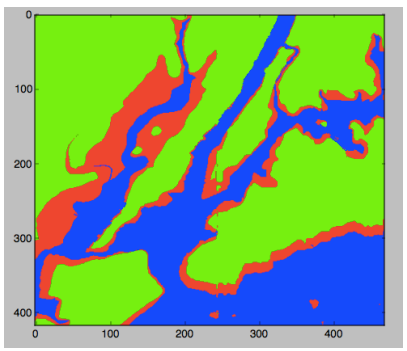
```
import numpy as np
```

- `pyplot`: part of `matplotlib` for making graphs and plots
- Standard abbreviation: `plt`

Useful Packages: `numpy` and `pyplot`







We will use two popular packages:

- `numpy`: numerical analysis package
- Standard abbreviation: `np`

```
import numpy as np
```

- `pyplot`: part of `matplotlib` for making graphs and plots
- Standard abbreviation: `plt`

```
import matplotlib.pyplot as plt
```

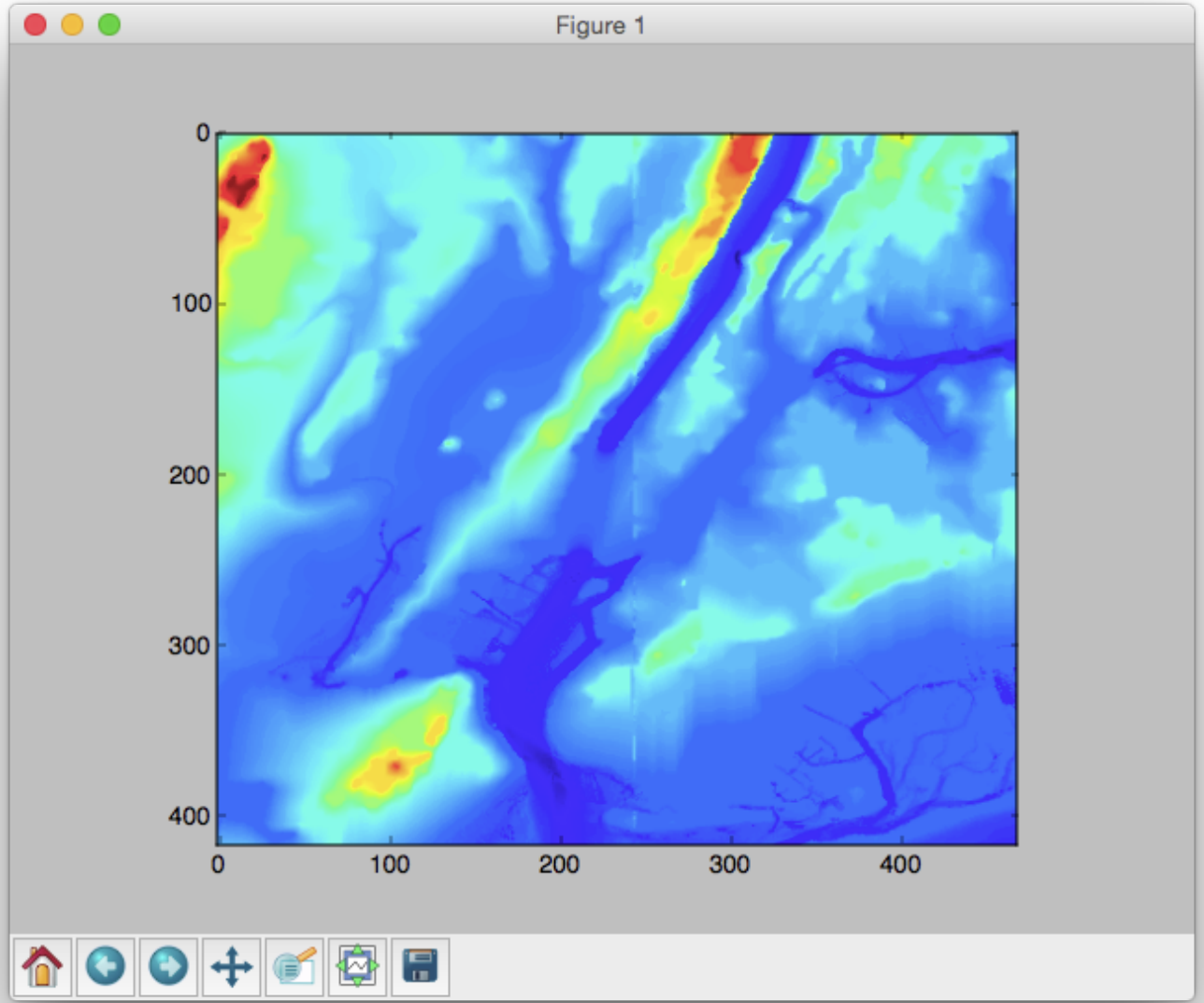
Your Turn: Using `plt`

Open IDLE (or favorite editor), and create the following short program:

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

#Read in the data to an array, called elevations:
elevations = np.loadtxt('elevationsNYC.txt')

#Load the array into matplotlib.pyplot:
plt.imshow(elevations)

#Display the plot:
plt.show()
```

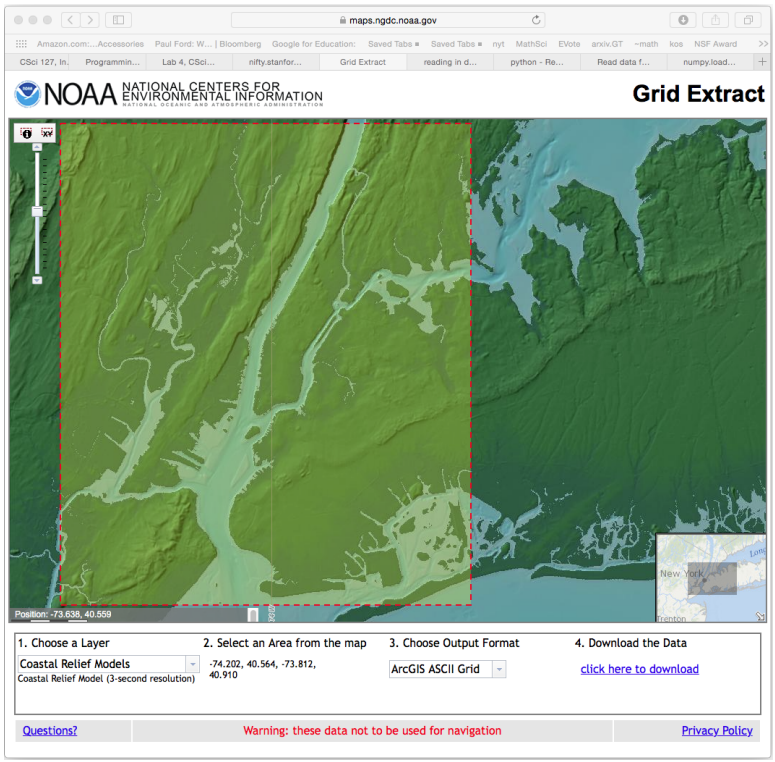
Your Turn: Using `plt`

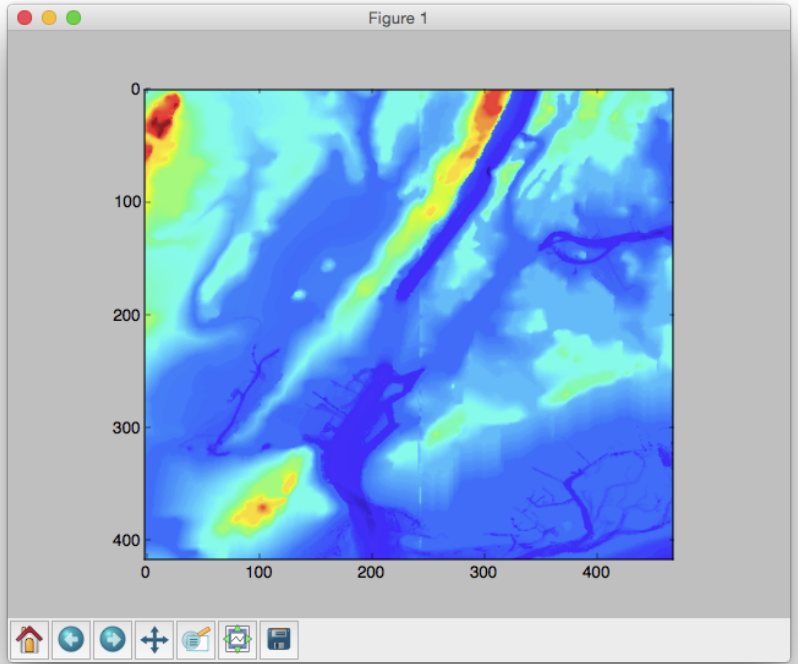
Run your program:



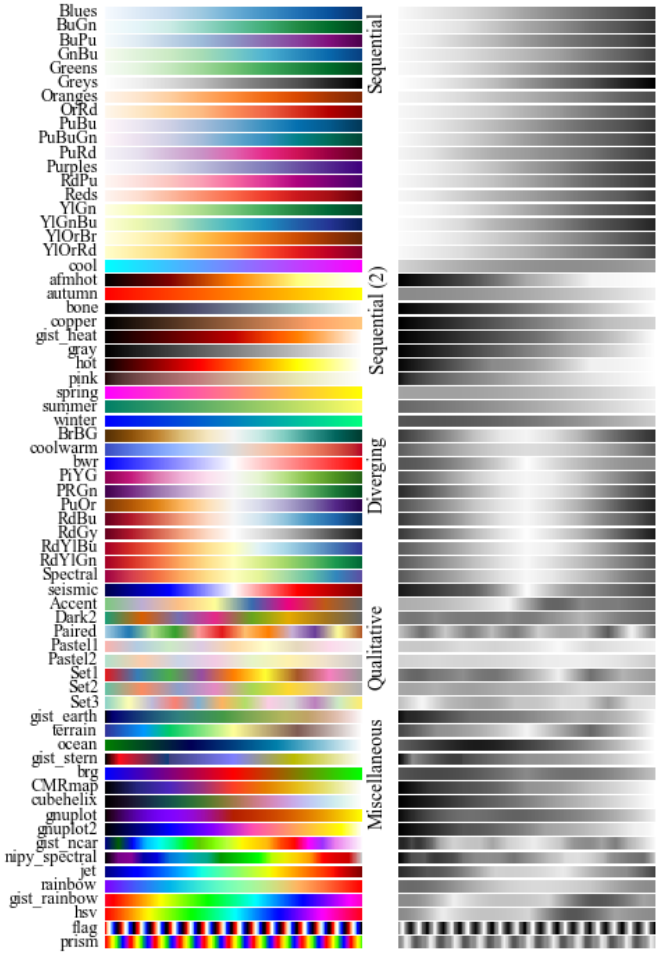
Color Maps

Comparing the NOAA image with the one you created:





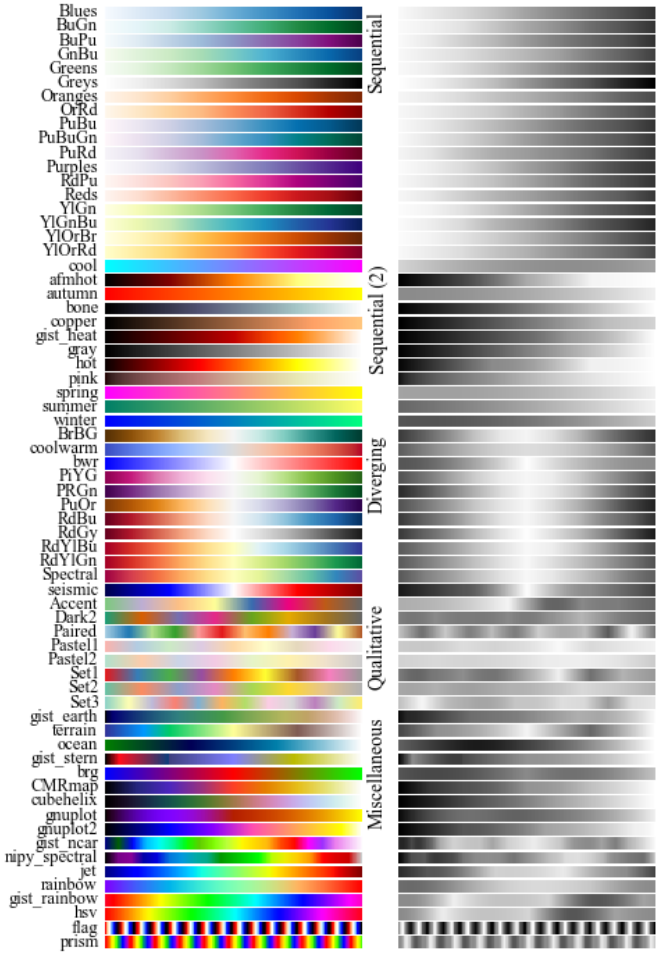
Color Maps



astrodsg blog

- The default ('jet') for displaying arrays is a color map that assigns:

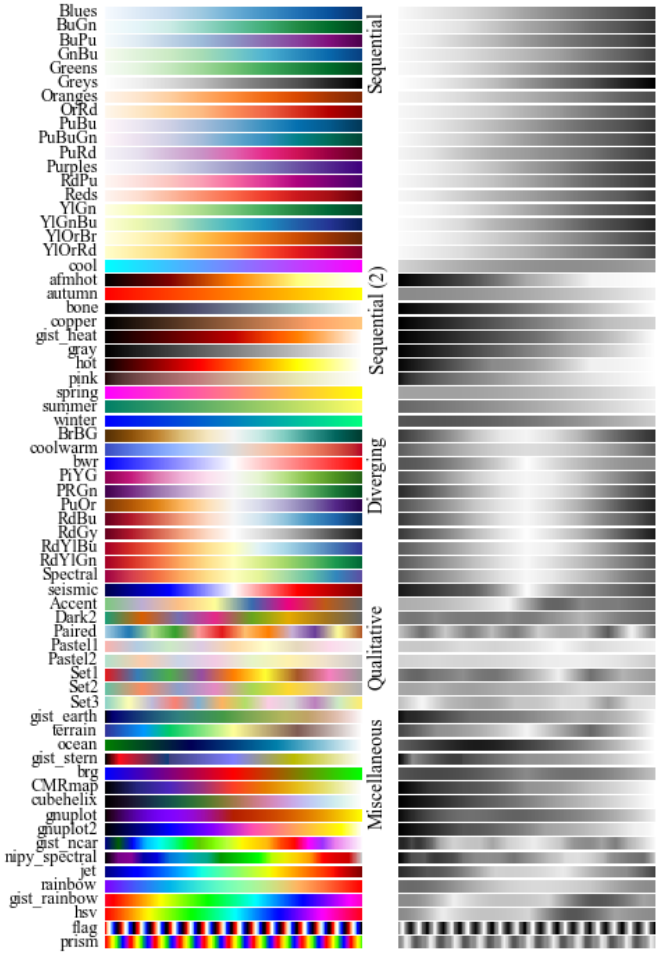
Color Maps



astrodsg blog

- The default ('jet') for displaying arrays is a color map that assigns:
  - the lowest values are blue,

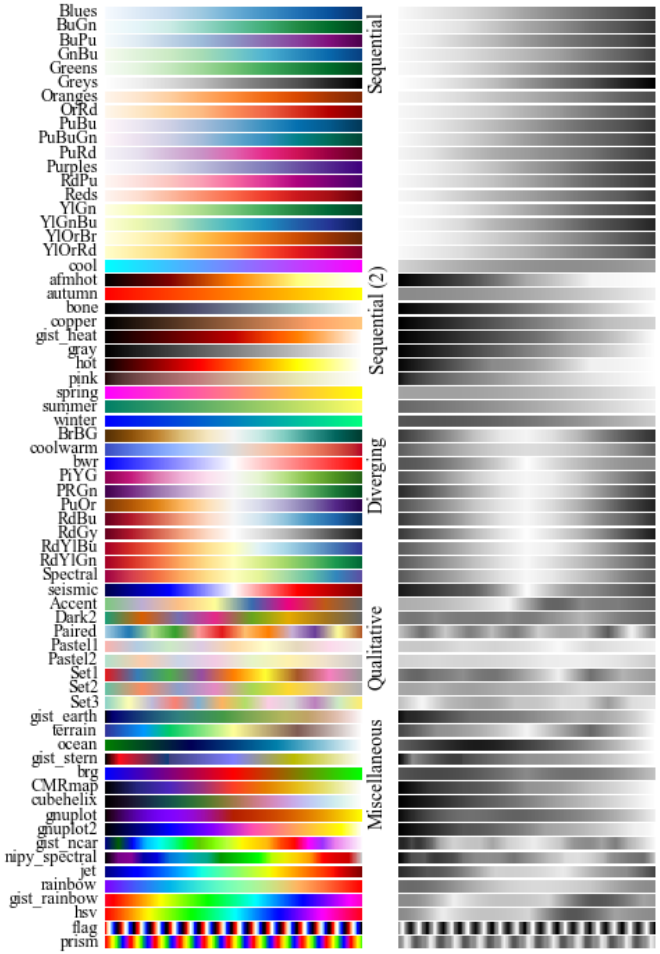
Color Maps



astrodsg blog

- The default ('jet') for displaying arrays is a color map that assigns:
  - the lowest values are blue,
  - the highest values are red, and

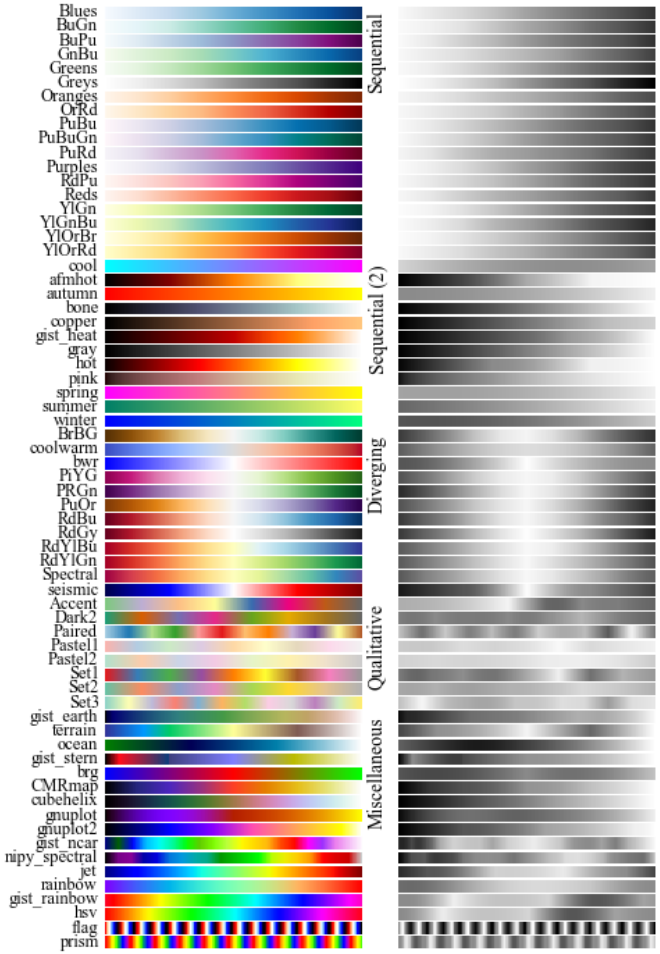
Color Maps



astrodsg blog

- The default ('jet') for displaying arrays is a color map that assigns:
  - the lowest values are blue,
  - the highest values are red, and
  - smoothly map the values in between.

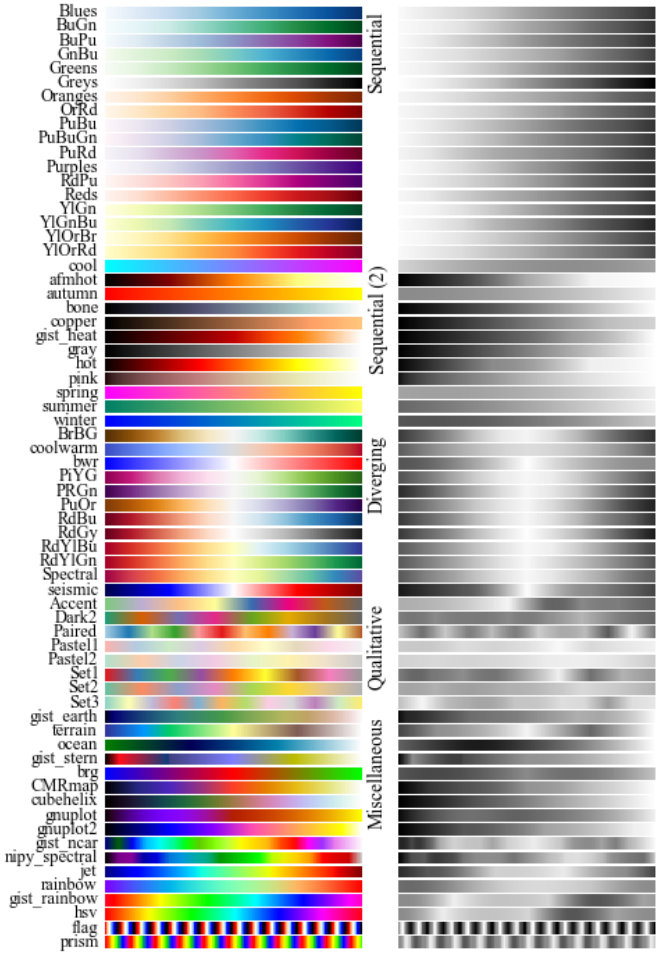
Color Maps



astrodsg blog

- The default ('jet') for displaying arrays is a color map that assigns:
  - the lowest values are blue,
  - the highest values are red, and
  - smoothly map the values in between.
- Many other color maps available.

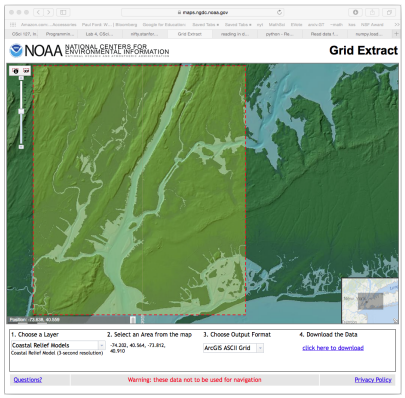
Color Maps

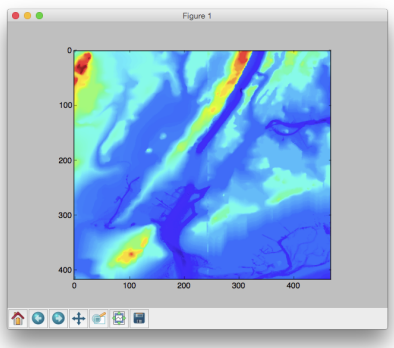


astrodsg blog

- The default ('jet') for displaying arrays is a color map that assigns:
  - the lowest values are blue,
  - the highest values are red, and
  - smoothly map the values in between.
- Many other color maps available.
- Equally space the coloring & do not treat 0 ("sea level") as a special value.

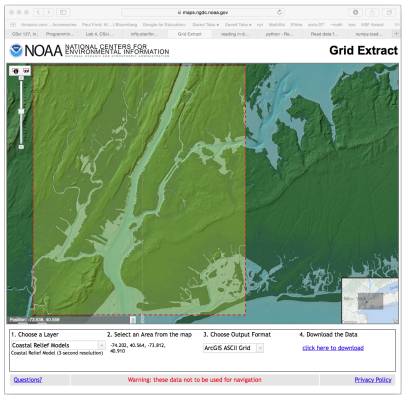
Refining our Coloring

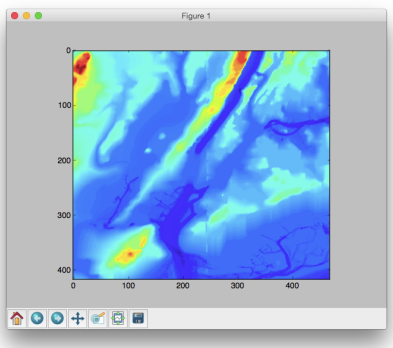




Using the storm surge of Hurricane Sandy in 2012:

Refining our Coloring

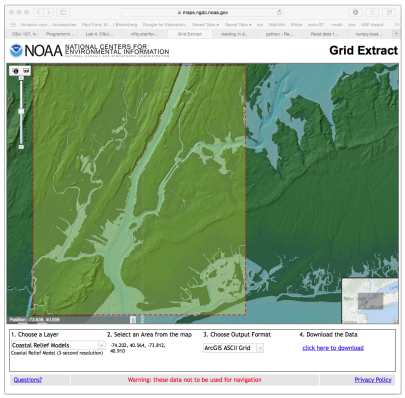


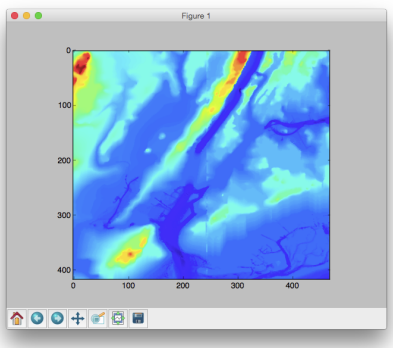


Using the storm surge of Hurricane Sandy in 2012:

- If the elevation is less than or equal to 0:

Refining our Coloring

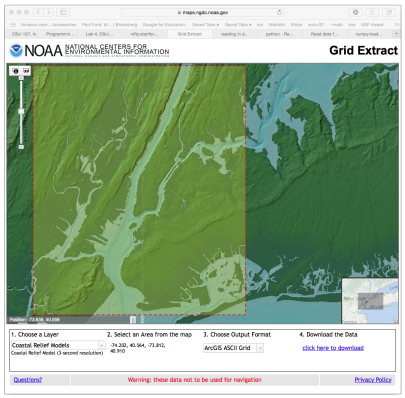


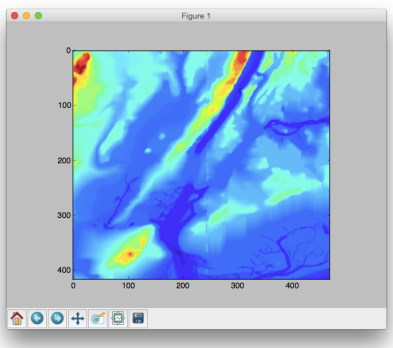


Using the storm surge of Hurricane Sandy in 2012:

- If the elevation is less than or equal to 0:
  - Below sea level,
  - Color the pixel blue

Refining our Coloring

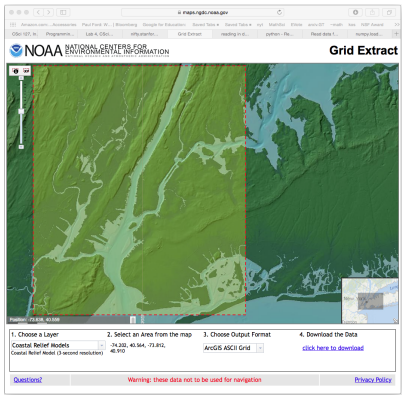


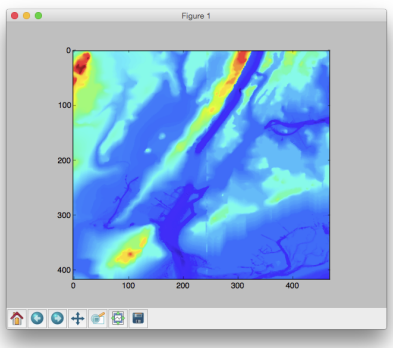


Using the storm surge of Hurricane Sandy in 2012:

- If the elevation is less than or equal to 0:
  - Below sea level,
  - Color the pixel blue
- Else if the elevation is less than or equal to 6:

Refining our Coloring

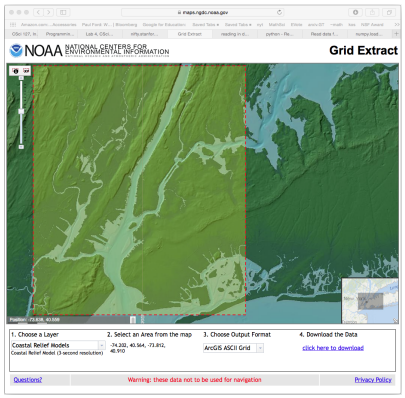


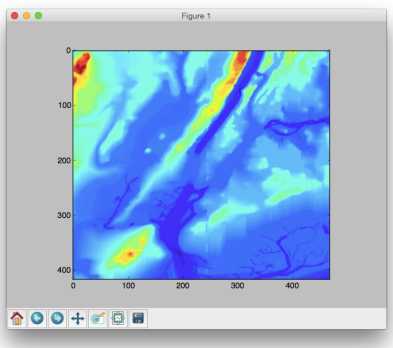


Using the storm surge of Hurricane Sandy in 2012:

- If the elevation is less than or equal to 0:
  - Below sea level,
  - Color the pixel blue
- Else if the elevation is less than or equal to 6:
  - In storm surge (flooding likely)
  - Color the pixel red

Refining our Coloring

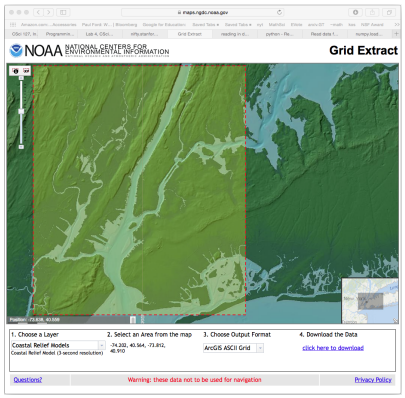


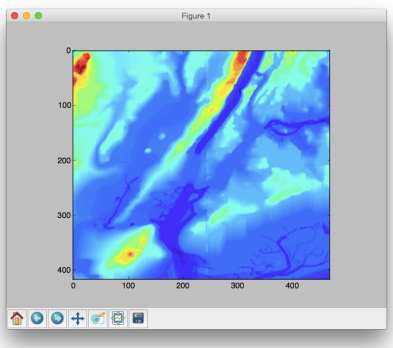


Using the storm surge of Hurricane Sandy in 2012:

- If the elevation is less than or equal to 0:
  - Below sea level,
  - Color the pixel blue
- Else if the elevation is less than or equal to 6:
  - In storm surge (flooding likely)
  - Color the pixel red
- Else:

Refining our Coloring

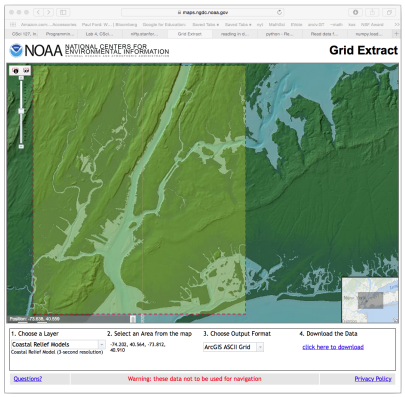


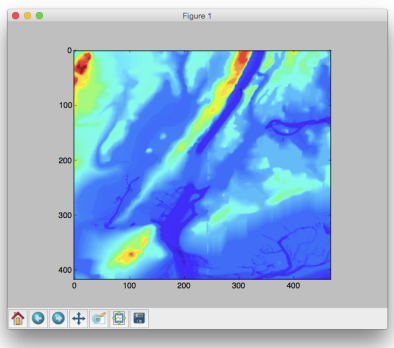


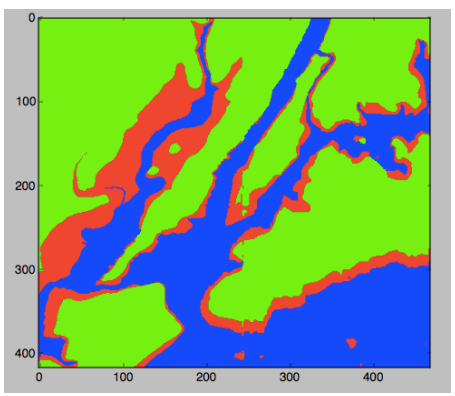
Using the storm surge of Hurricane Sandy in 2012:

- If the elevation is less than or equal to 0:
  - Below sea level,
  - Color the pixel blue
- Else if the elevation is less than or equal to 6:
  - In storm surge (flooding likely)
  - Color the pixel red
- Else:
  - Above the 6 foot storm surge
  - Color the pixel green

Refining our Coloring



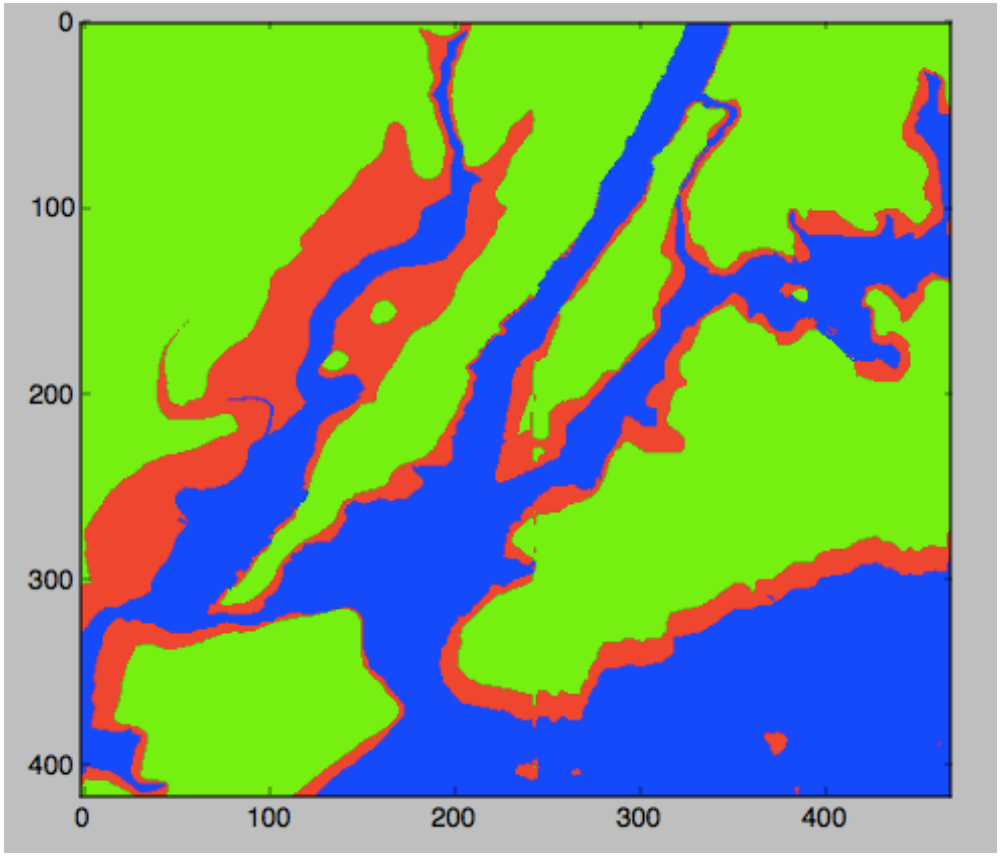




Using the storm surge of Hurricane Sandy in 2012:

- If the elevation is less than or equal to 0:
  - Below sea level,
  - Color the pixel blue
- Else if the elevation is less than or equal to 6:
  - In storm surge (flooding likely)
  - Color the pixel red
- Else:
  - Above the 6 foot storm surge
  - Color the pixel green

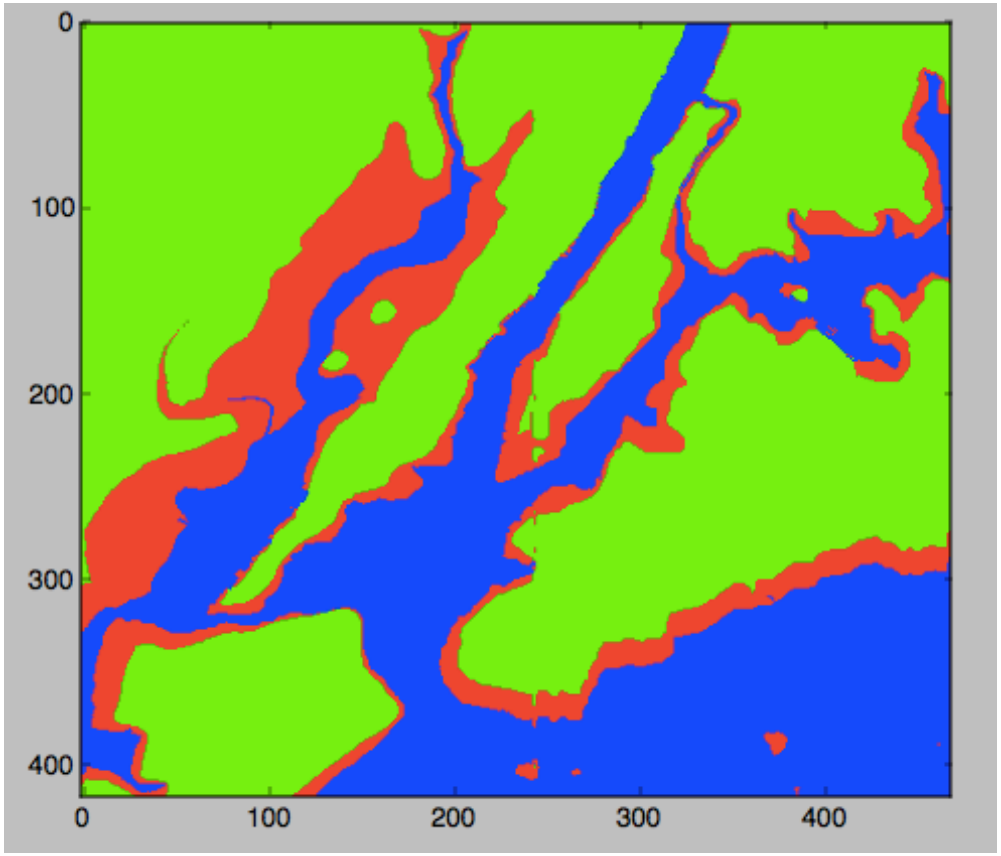
Before Translating to Python...



Need to explain:

- Color

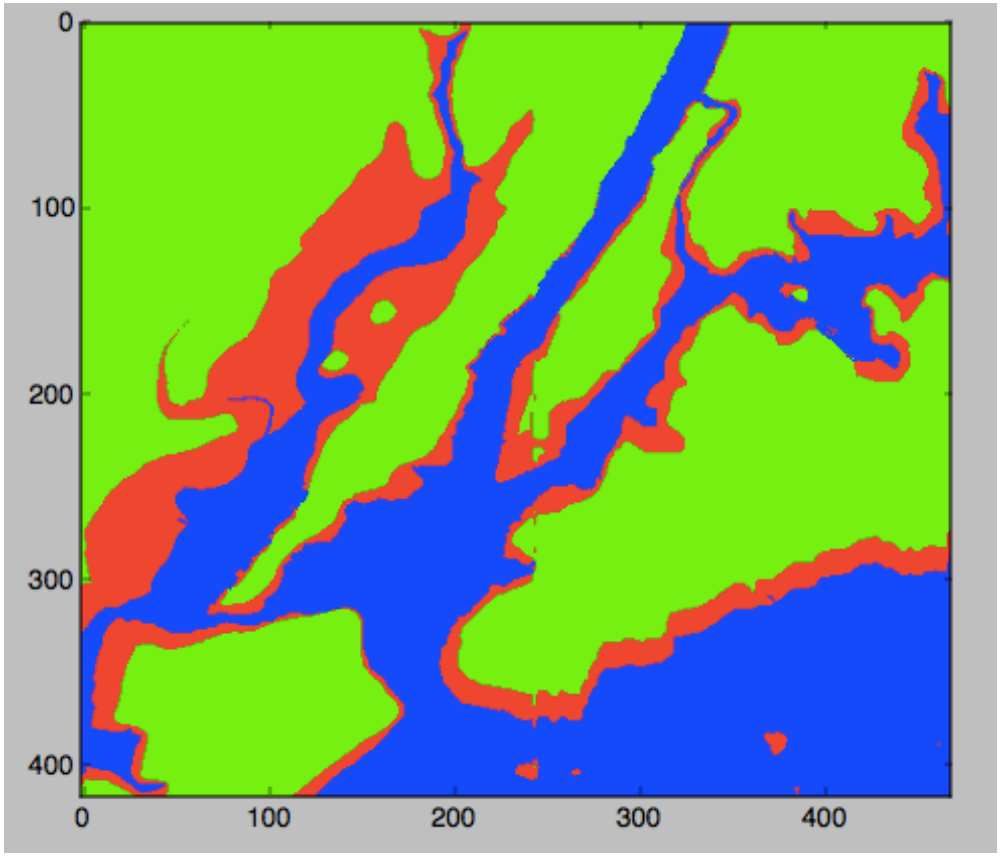
Before Translating to Python...



Need to explain:

- Color (usually done week before with turtles)

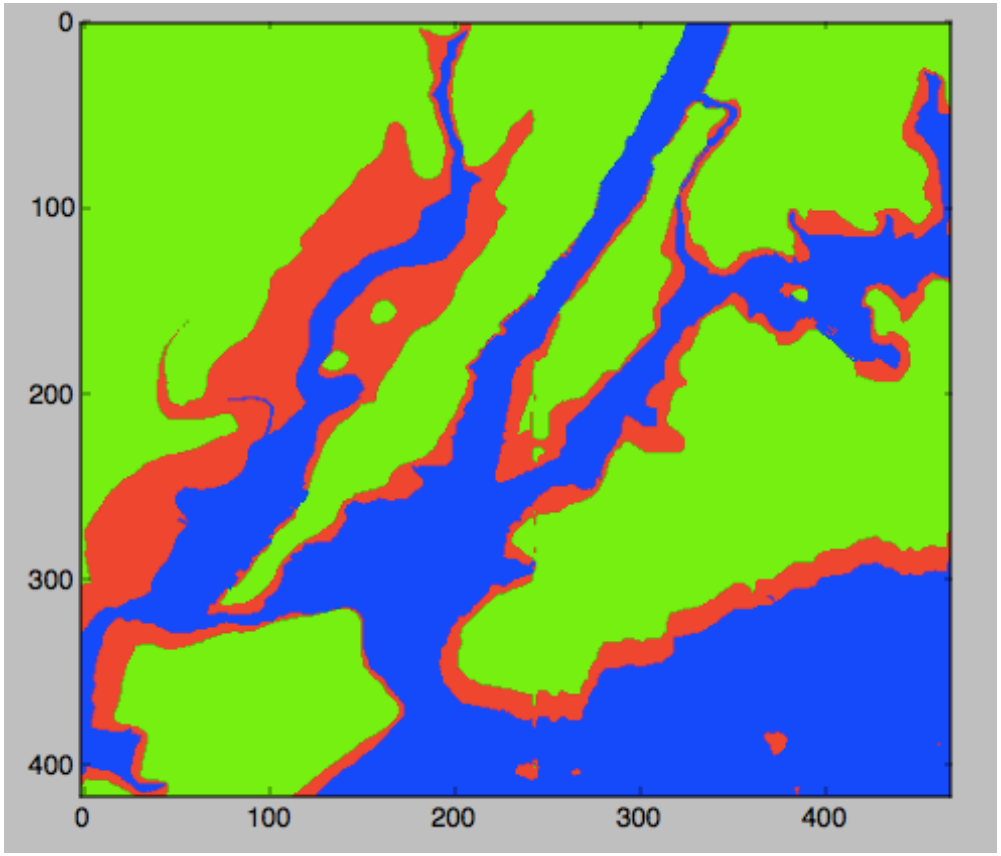
Before Translating to Python...



Need to explain:

- Color (usually done week before with turtles) and
- Arrays

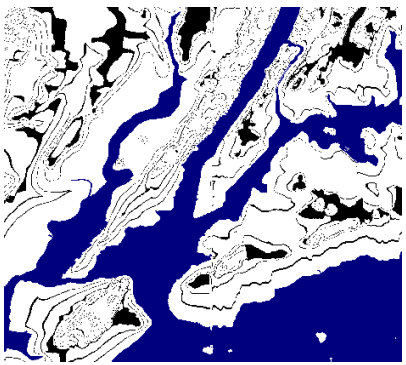
Before Translating to Python...



Need to explain:

- Color (usually done week before with turtles) and
- Arrays (two stage approach: bare minimum, then more details later).

Outline



- Welcome and Introductions
- Workshop Overview
- Design Challenge: Flood Maps
  - Data: Elevations from NOAA
  - Packages: `numpy` and `pyplot`
  - Concepts: Colors and Arrays
  - Putting It Altogether
- Break
- Variations on the Theme
- Design a Challenge
- Wrap Up

Colors

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

- Can specify by name.

Colors

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

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

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).

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:

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

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

Colors

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

- Can specify by numbers (RGB):

Colors

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

- Can specify by numbers (RGB):
  - Fractions of each:

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.

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:

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.

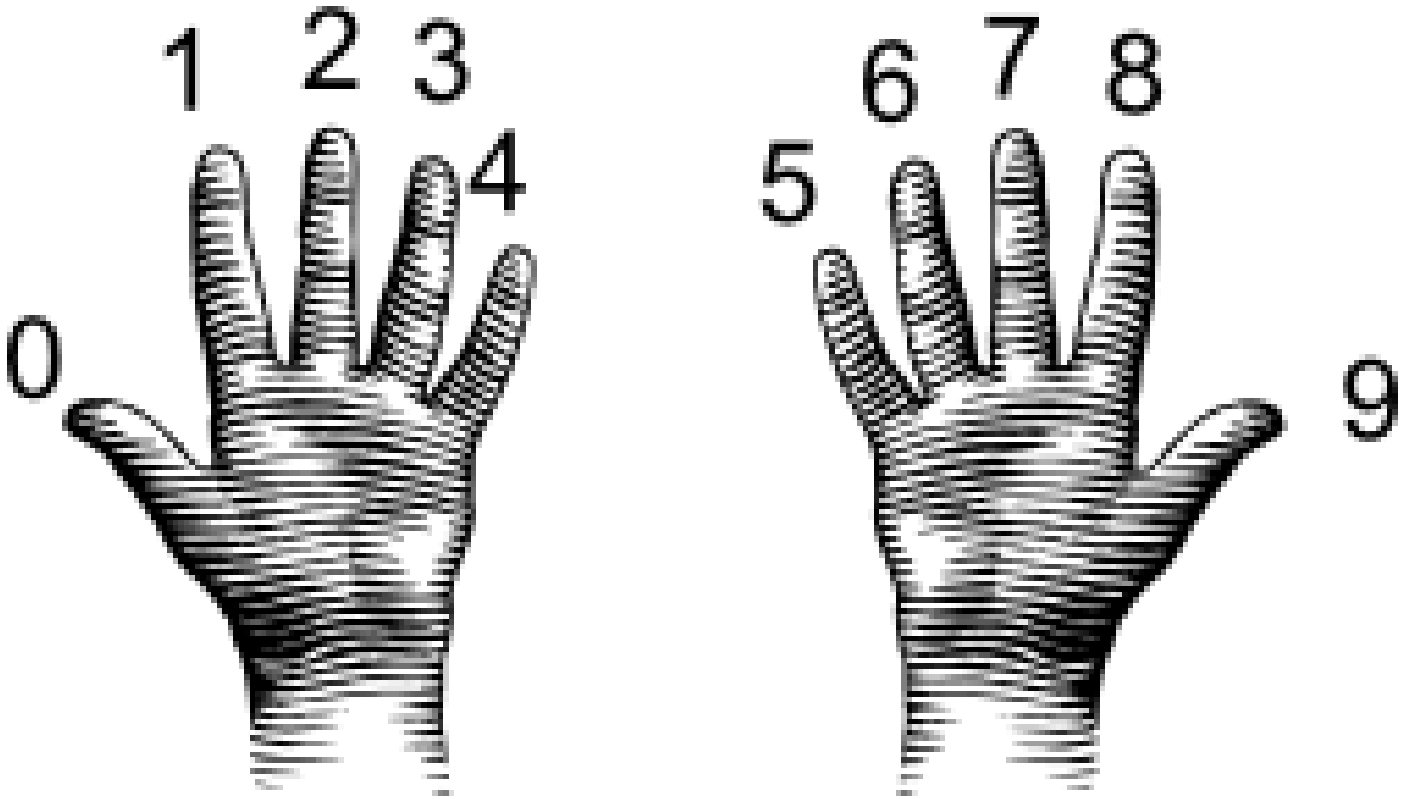
Colors

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)...

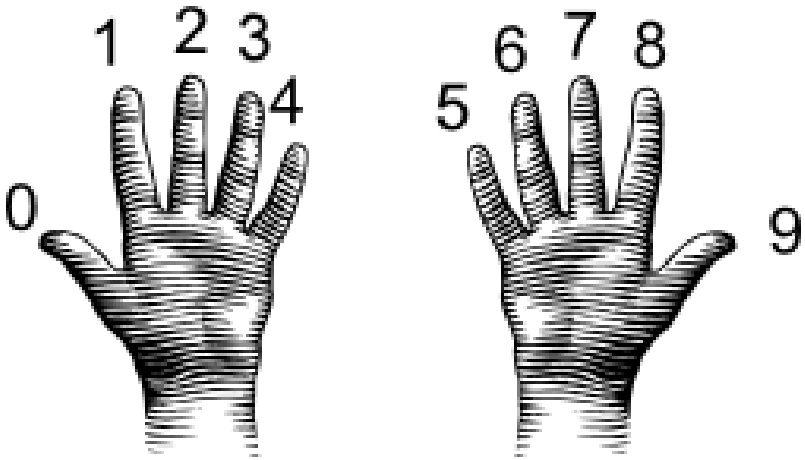
Decimal & Hexadecimal Numbers

Counting with 10 digits:



(from i-programmer.info)

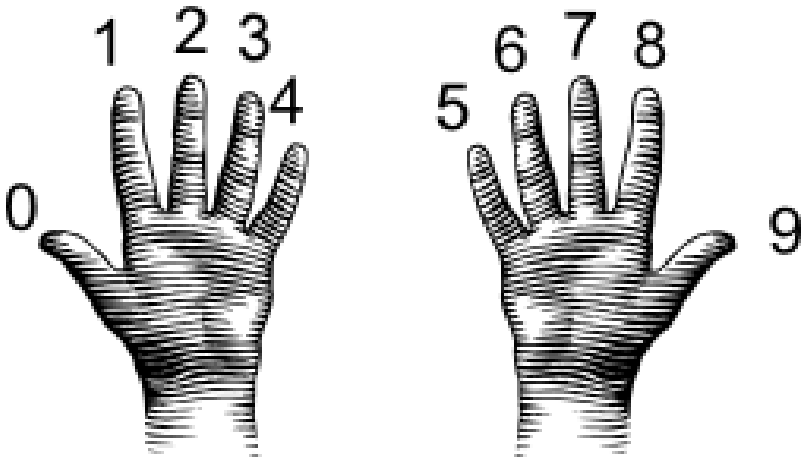
Decimal



(from i-programmer.info)

Decimal

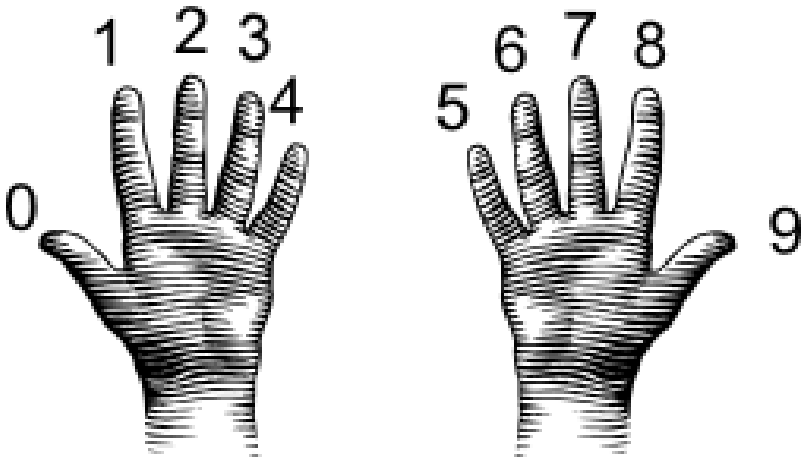
00 01 02 03 04 05 06 07 08 09



(from i-programmer.info)

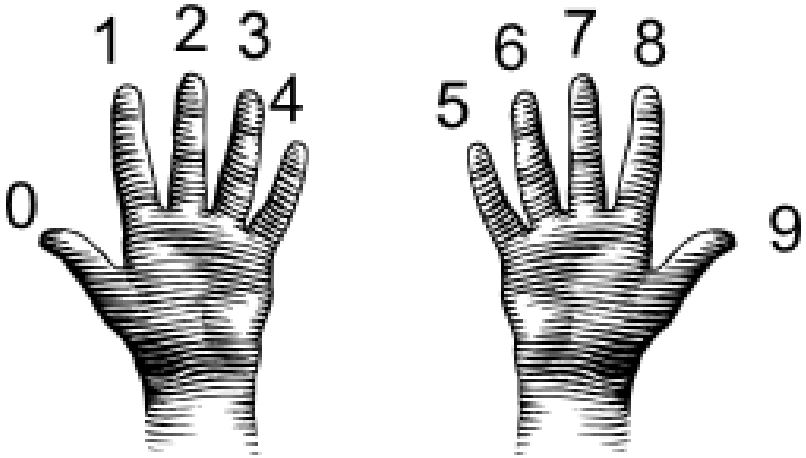
Decimal

00 01 02 03 04 05 06 07 08 09
10 11 12 13 14 15 16 17 18 19



(from i-programmer.info)

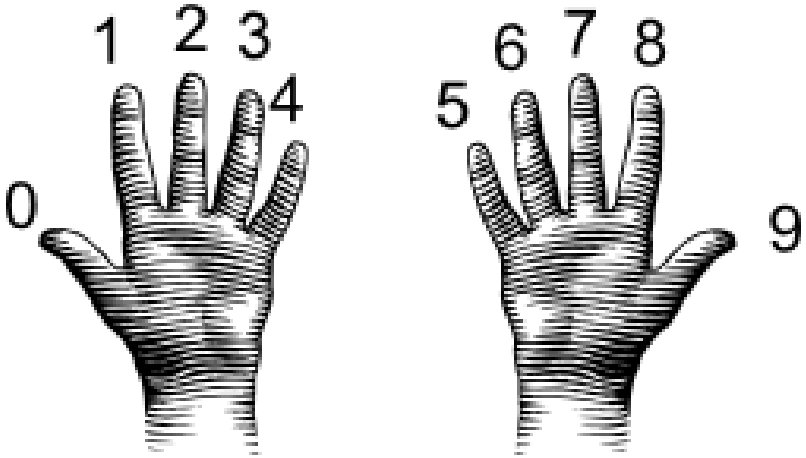
Decimal



(from i-programmer.info)

00 01 02 03 04 05 06 07 08 09
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29

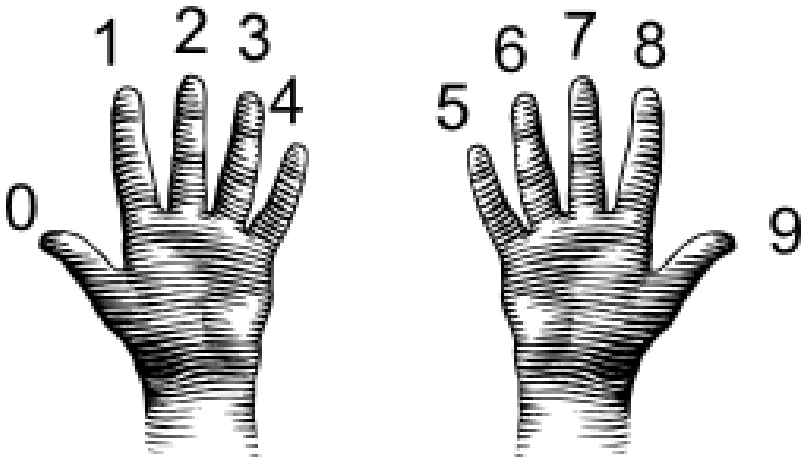
Decimal



(from i-programmer.info)

00 01 02 03 04 05 06 07 08 09
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39

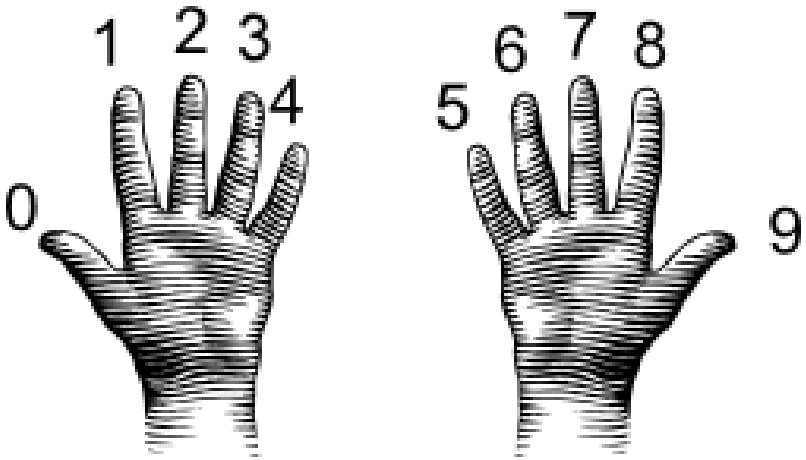
Decimal



(from i-programmer.info)

00 01 02 03 04 05 06 07 08 09
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49

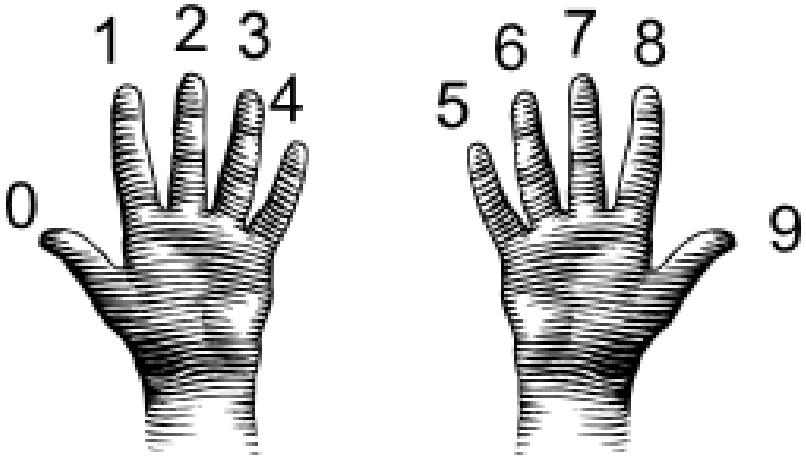
Decimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59

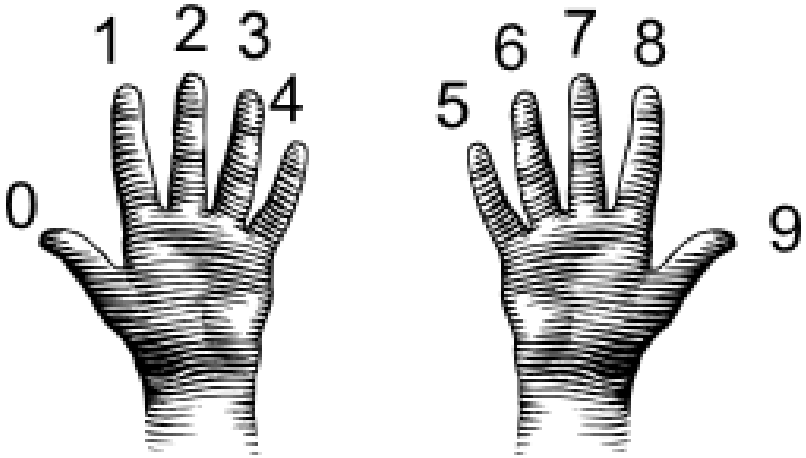
Decimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69

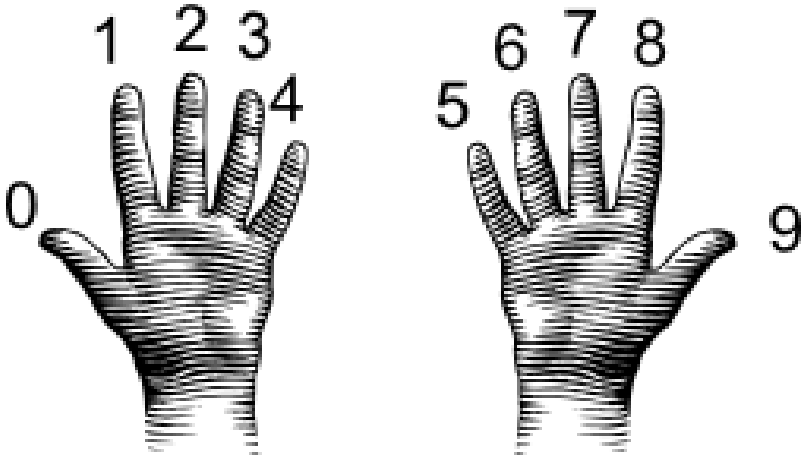
Decimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79

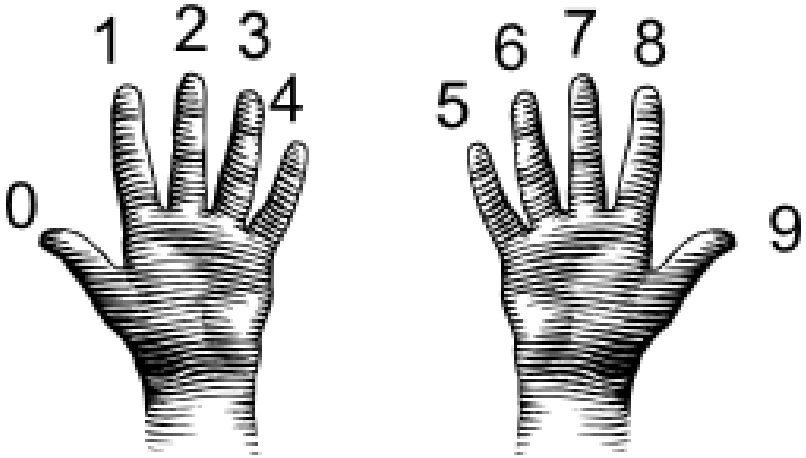
Decimal



(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89

Decimal

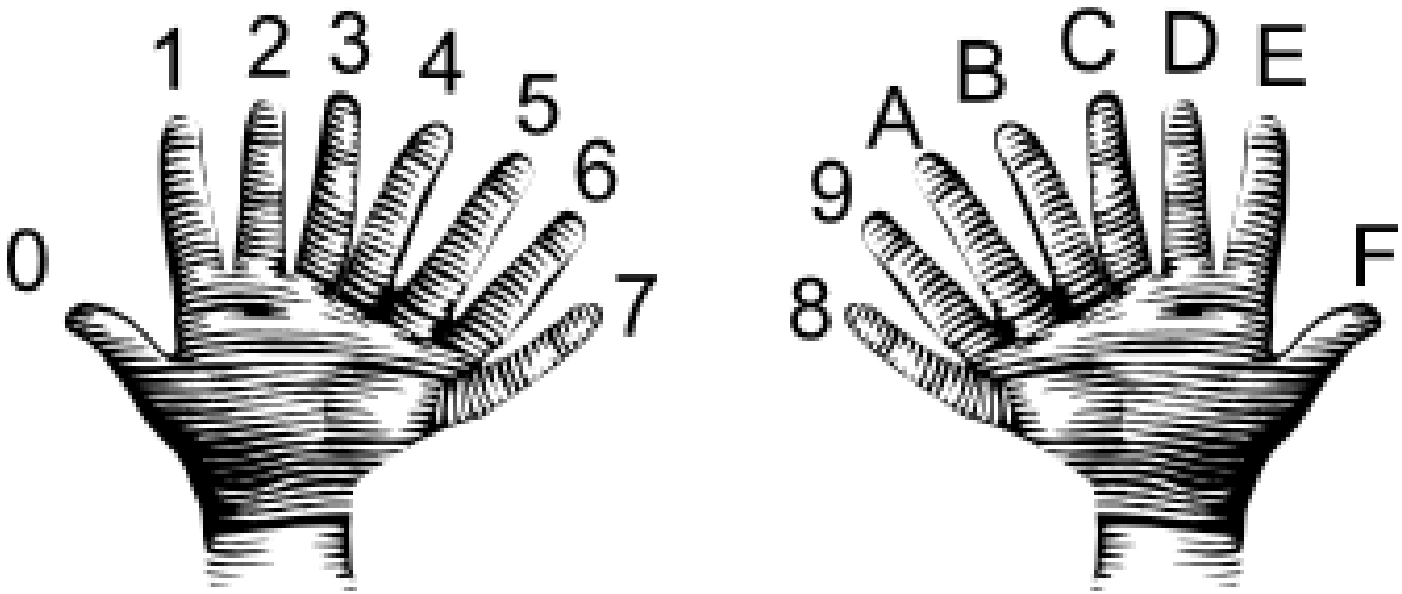


(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

Decimal & Hexadecimal Numbers

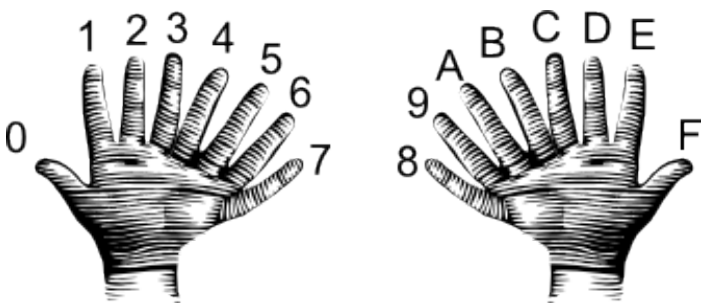
Counting with 16 digits:



(from i-programmer.info)

Hexadecimal

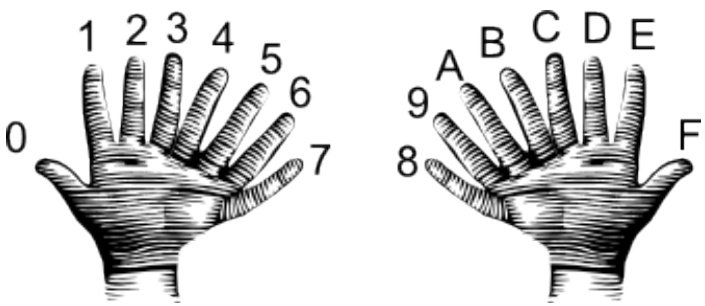
```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
```



(from i-programmer.info)

Hexadecimal

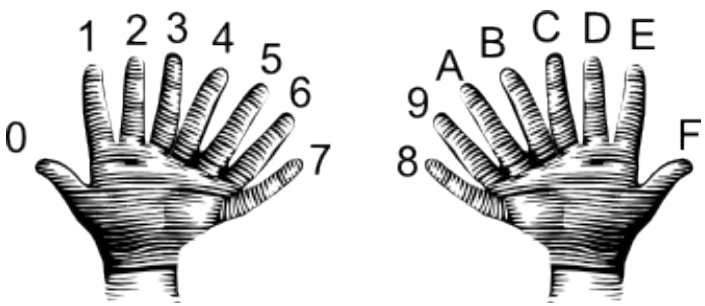
```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
```



(from i-programmer.info)

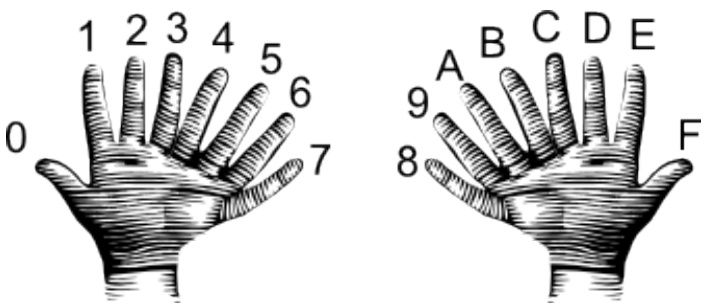
Hexadecimal

```
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
```



(from i-programmer.info)

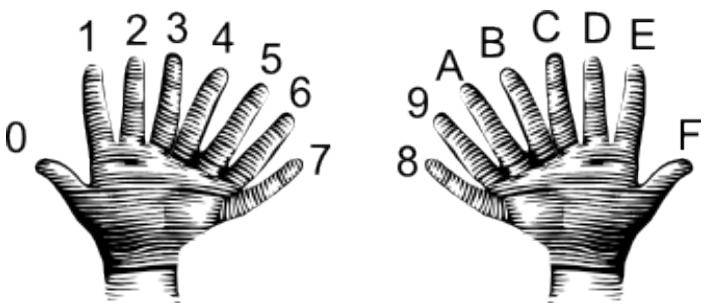
Hexadecimal



(from i-programmer.info)

```
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
```

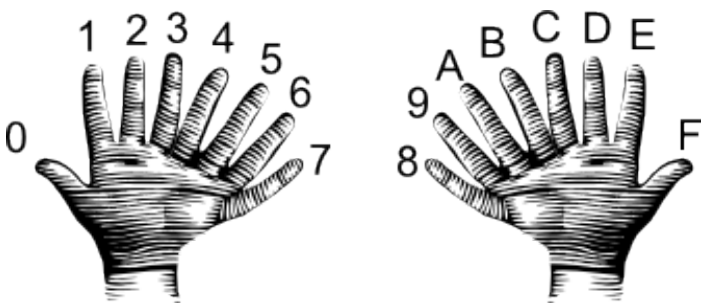
Hexadecimal



(from i-programmer.info)

```
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
```

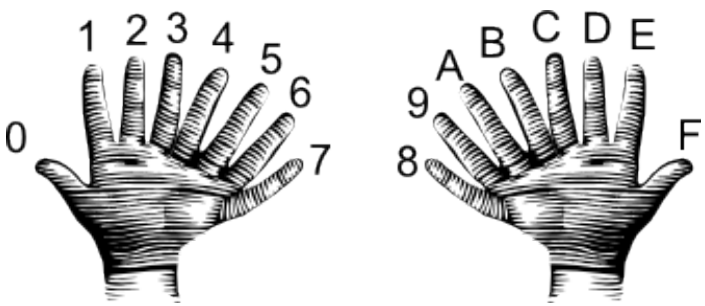
Hexadecimal



(from i-programmer.info)

```
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
```

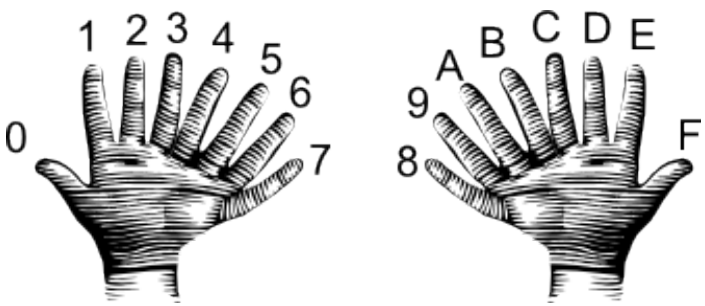
Hexadecimal



(from i-programmer.info)

```
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
```

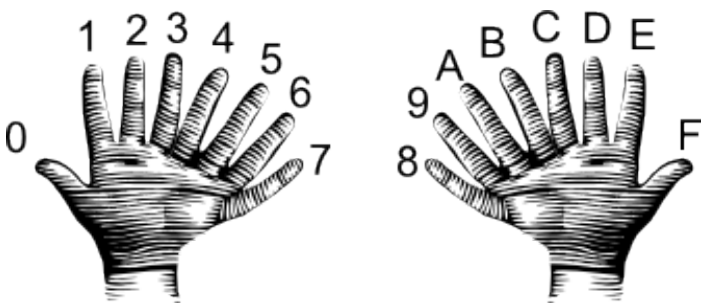
Hexadecimal



(from i-programmer.info)

```
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
```

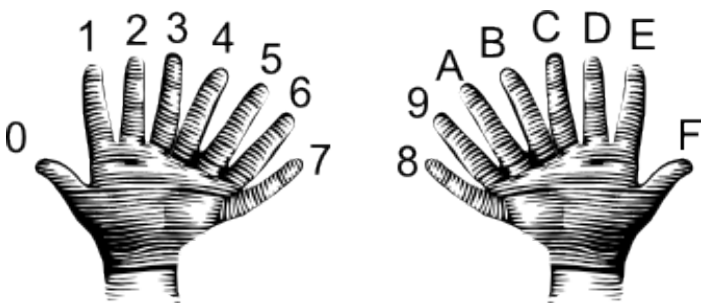
Hexadecimal



(from i-programmer.info)

```
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
```

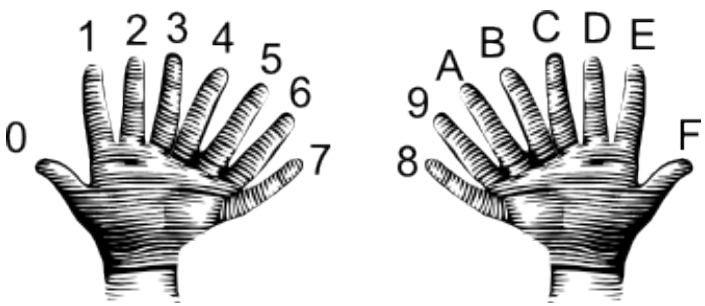
Hexadecimal



(from i-programmer.info)

```
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
```

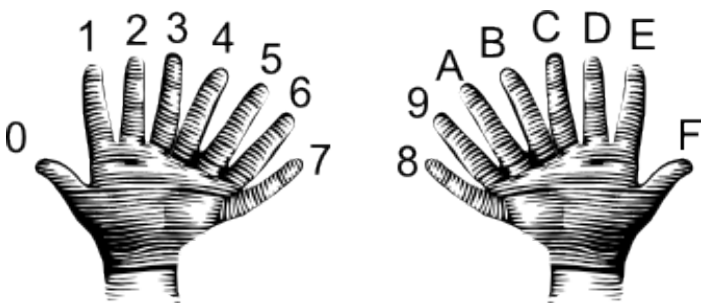
Hexadecimal



(from i-programmer.info)

```
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
```

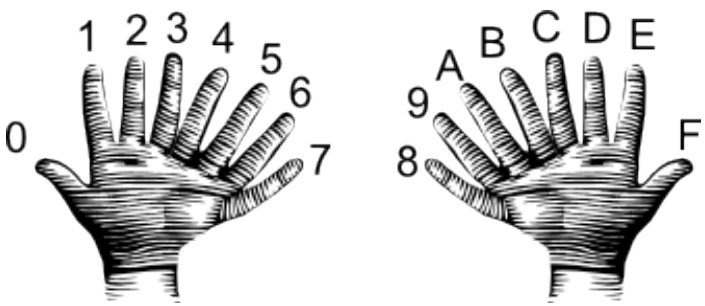
Hexadecimal



(from i-programmer.info)

```
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
```

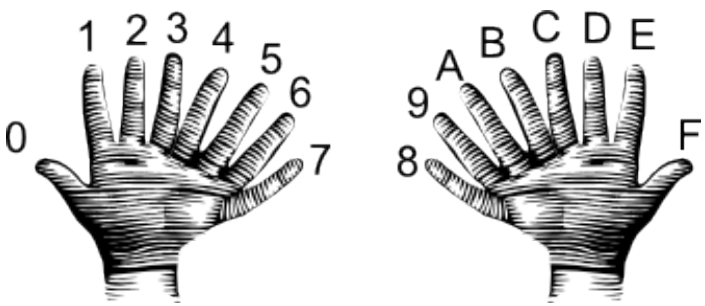
Hexadecimal



(from i-programmer.info)

```
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
```

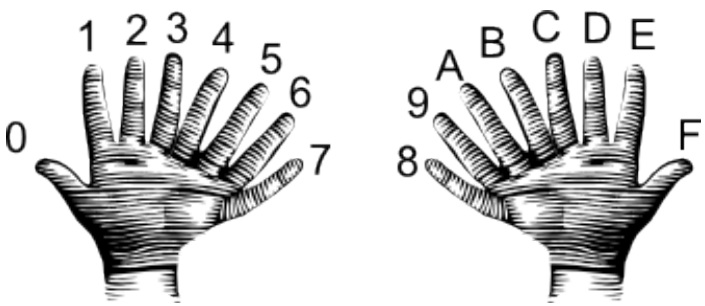
Hexadecimal



(from i-programmer.info)

```
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
```

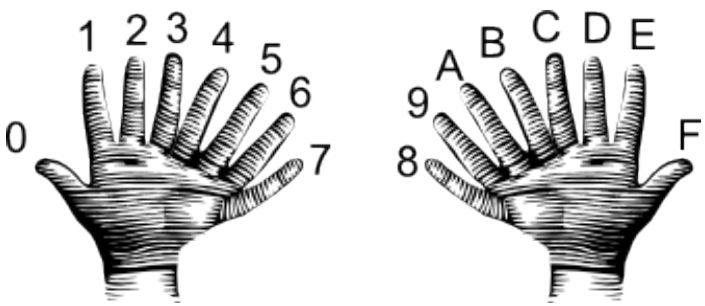
Hexadecimal



(from i-programmer.info)

```
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
```

Hexadecimal



(from i-programmer.info)

```
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.

Images

- For each picture element ('pixel'), we store the amount of red, the amount of blue, and the amount of green.

Images

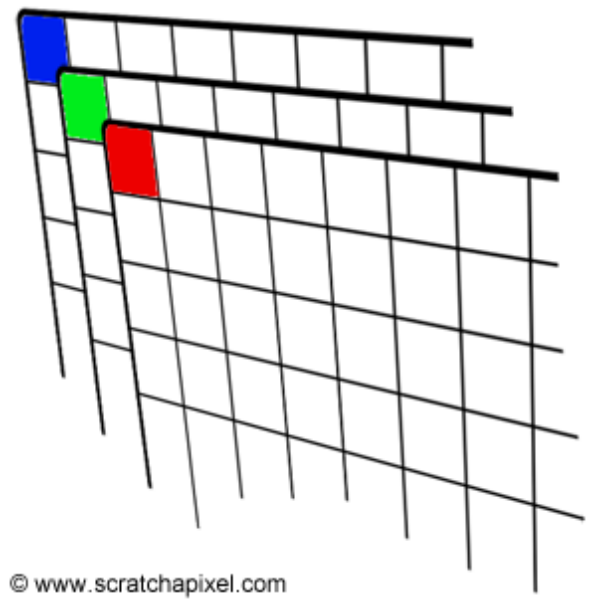
- For each picture element ('pixel'), we store the amount of red, the amount of blue, and the amount of green.
- Can access the color component of each pixel: e.g. `img[row,col,0]` for red portion of `(row,col)`.

Images

- For each picture element ('pixel'), we store the amount of red, the amount of blue, and the amount of green.
- Can access the color component of each pixel: e.g. `img[row,col,0]` for red portion of `(row,col)`.
- Can view the image as 3 sheets of graph paper:

Images

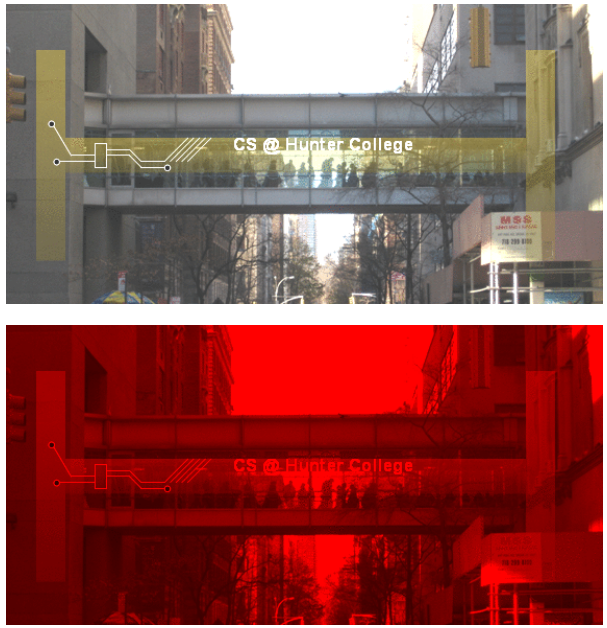
- For each picture element ('pixel'), we store the amount of red, the amount of blue, and the amount of green.
- Can access the color component of each pixel: e.g. `img[row,col,0]` for red portion of `(row,col)`.
- Can view the image as 3 sheets of graph paper:



Images

- For each picture element ('pixel'), we store the amount of red, the amount of blue, and the amount of green.
- Can access the color component of each pixel: e.g. `img[row,col,0]` for red portion of `(row,col)`.
- Can view the image as 3 sheets of graph paper:

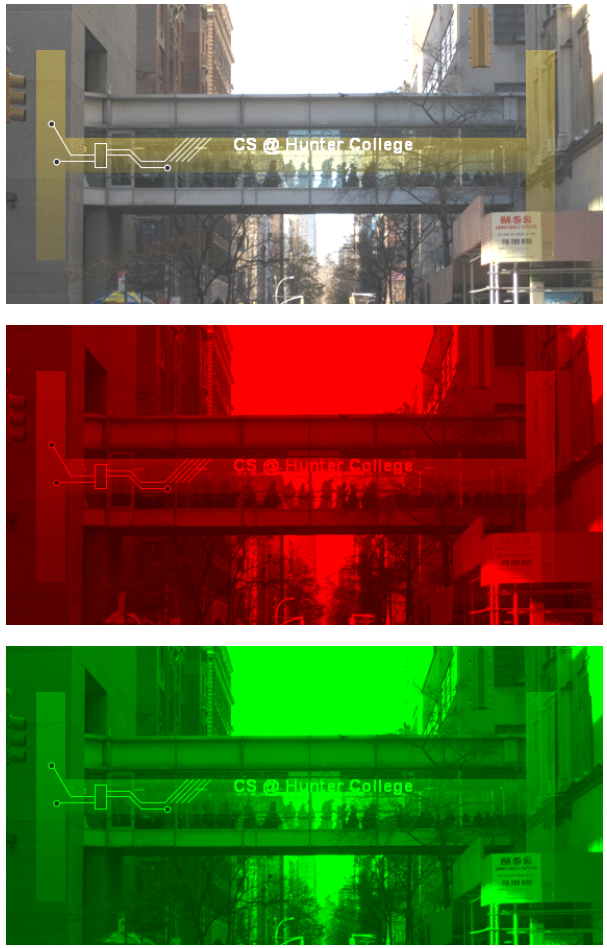
Images



- For each picture element ('pixel'), we store the amount of red, the amount of blue, and the amount of green.
- Can access the color component of each pixel: e.g. `img[row,col,0]` for red portion of `(row,col)`.
- Can view the image as 3 sheets of graph paper:

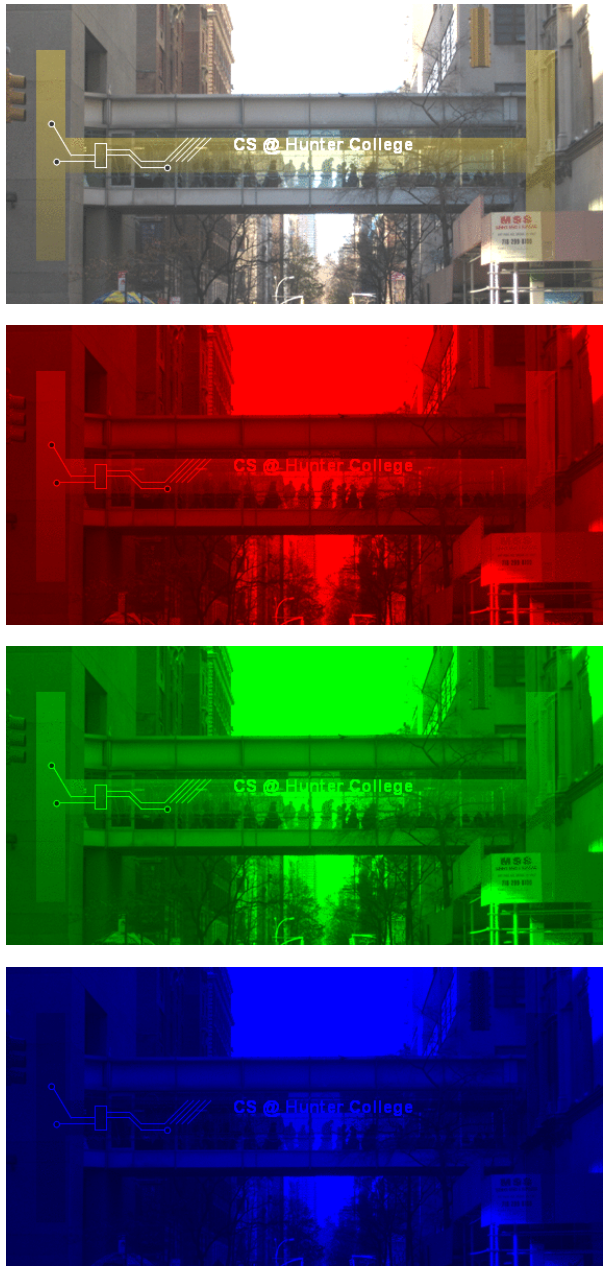
Images

- For each picture element ('pixel'), we store the amount of red, the amount of blue, and the amount of green.
- Can access the color component of each pixel: e.g. `img[row,col,0]` for red portion of `(row,col)`.
- Can view the image as 3 sheets of graph paper:

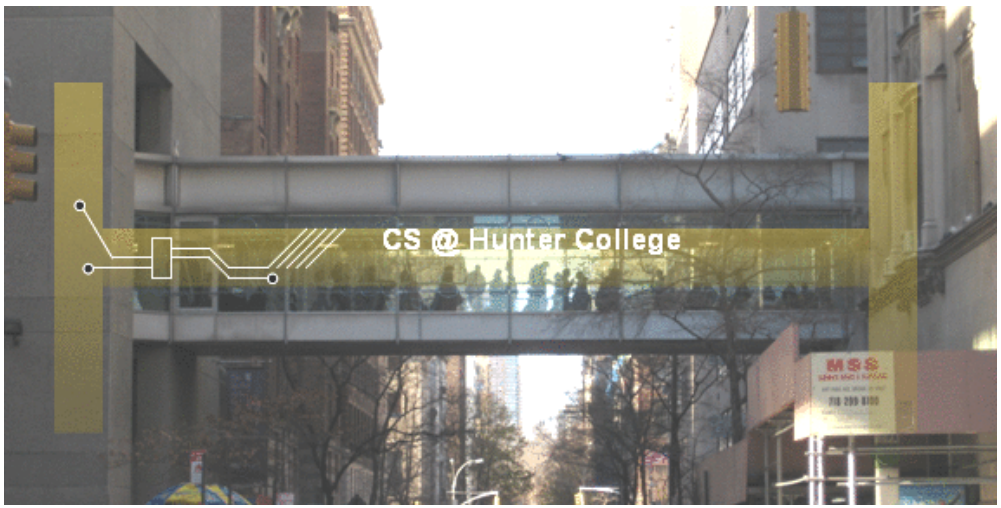


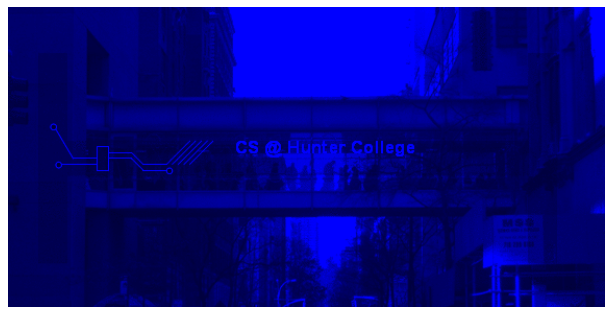
Images

- For each picture element ('pixel'), we store the amount of red, the amount of blue, and the amount of green.
- Can access the color component of each pixel: e.g. `img[row,col,0]` for red portion of `(row,col)`.
- Can view the image as 3 sheets of graph paper:



Images

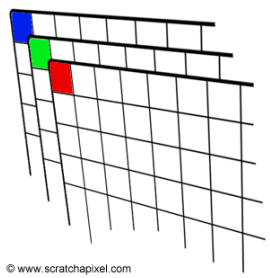




This image has 287 rows, 573 columns, and 4 color channels (for red, green, blue, and a 4th for how transparent).

Creating Images

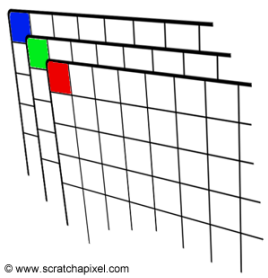
To create an image from scratch:



Creating Images

To create an image from scratch:

1. Import the libraries.

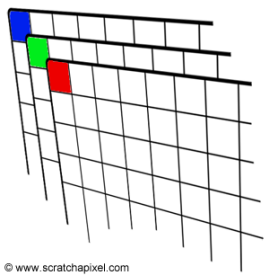


Creating Images

To create an image from scratch:

1. Import the libraries.

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



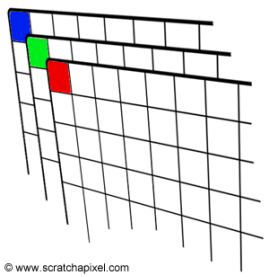
Creating Images

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



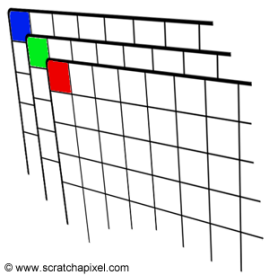
Creating Images

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

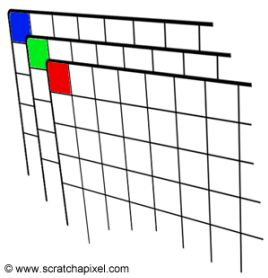
To create an image from scratch:

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) )
```



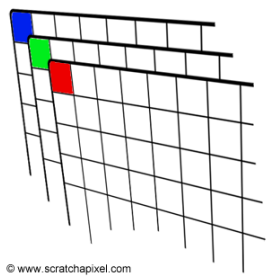
Creating Images

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):



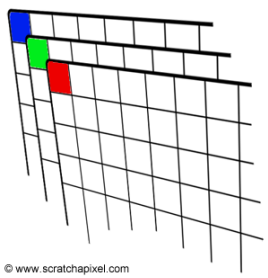
Creating Images

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) )`



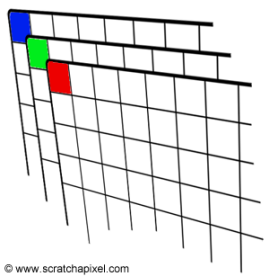
Creating Images

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*



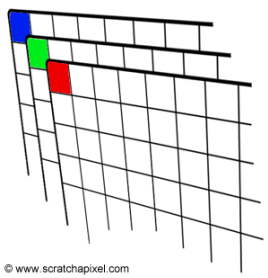
Creating Images

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

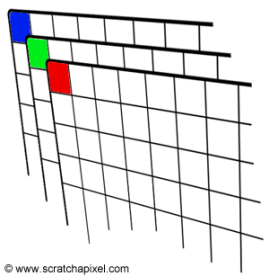
To create an image from scratch:

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:

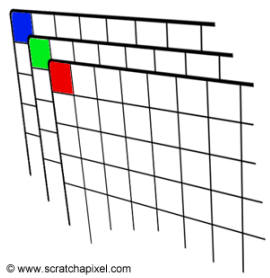
1. Import the libraries.

```
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.

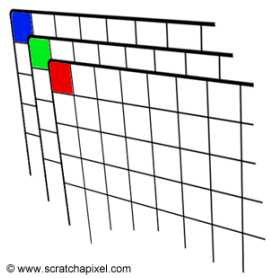
```
import matplotlib.pyplot as plt
import numpy as np
```

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:

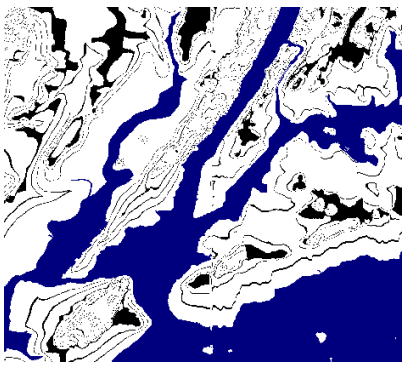
```
plt.imshow(img)
plt.show()
```

5. And save your image:

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

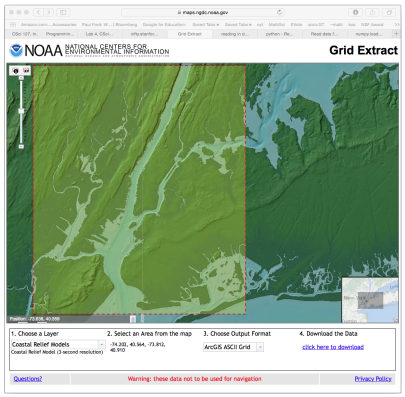


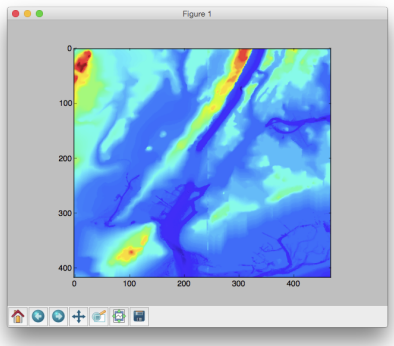
Outline

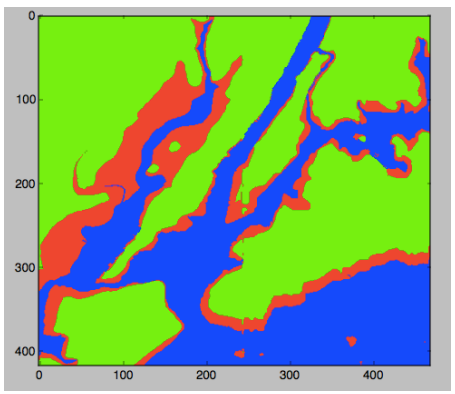


- Welcome and Introductions
- Workshop Overview
- Design Challenge: Flood Maps
  - Data: Elevations from NOAA
  - Packages: `numpy` and `pyplot`
  - Concepts: Colors and Arrays
  - Putting It Altogether
- Break
- Variations on the Theme
- Design a Challenge
- Wrap Up

Putting It Altogether: Design







Using the storm surge of Hurricane Sandy in 2012:

- If the elevation is less than or equal to 0:
  - Below sea level,
  - Color the pixel blue
- Else if the elevation is less than or equal to 6:
  - In storm surge (flooding likely)
  - Color the pixel red
- Else:
  - Above the 6 foot storm surge
  - Color the pixel green

Setting up the Image

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

#Read in the data to an array, called elevations:
elevations = np.loadtxt('elevationsNYC.txt')

#Take the shape (dimensions) of the elevations
#  and add another dimension to hold the 3 color channels:
mapShape = elevations.shape + (3,)

#Create a blank image that's all zeros:
floodMap = np.zeros(mapShape)
```

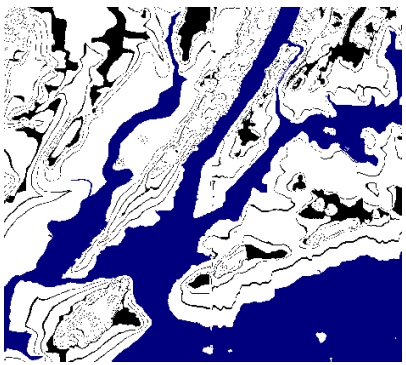
Looping through Elevations

```
for row in range(mapShape[0]):
    for col in range(mapShape[1]):
        if elevations[row,col] <= 0:
            #Below sea level
           floodMap[row,col,2] = 1.0     #Set the blue channel to 100%
        elif elevations[row,col] <= 6:
            #Below the storm surge of Hurricane Sandy (flooding likely)
           floodMap[row,col,0] = 1.0     #Set the red channel to 100%
        else:
            #Above the 6 foot storm surge and didn't flood
            floodMap[row,col,1] = 1.0   #Set the green channel to 100%
```

Putting it Altogether

```
#Import the libraries for arrays and displaying images:
import numpy as np
import matplotlib.pyplot as plt
#Read in the data to an array, called elevations:
elevations = np.loadtxt('elevationsNYC.txt')
#Take the shape (dimensions) of the elevations
#  and add another dimension to hold the 3 color channels:
mapShape = elevations.shape + (3,)

#Create a blank image that's all zeros:
floodMap = np.zeros(mapShape)

for row in range(mapShape[0]):
    for col in range(mapShape[1]):
        if elevations[row,col] <= 0:
            #Below sea level
            floodMap[row,col,2] = 1.0     #Set the blue channel to 100%
        elif elevations[row,col] <= 6:
            #Below the storm surge of Hurricane Sandy (flooding likely)
            floodMap[row,col,0] = 1.0     #Set the red channel to 100%
        else:
            #Above the 6 foot storm surge and didn't flood
            floodMap[row,col,1] = 1.0   #Set the green channel to 100%

#Load the flood map image into matplotlib.pyplot:
plt.imshow(floodMap)
#Display the plot:
plt.show()
#Save the image:
plt.imsave('floodMap.png', floodMap)
```

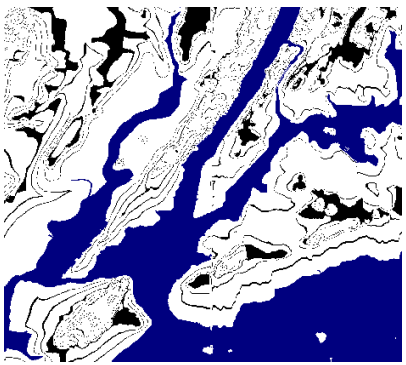
Outline



- Welcome and Introductions
- Workshop Overview
- Design Challenge: Flood Maps
  - Data: Elevations from NOAA
  - Packages: `numpy` and `pyplot`
  - Concepts: Colors and Arrays
  - Putting It Altogether
- Break
- Variations on the Theme
- Design a Challenge
- Wrap Up

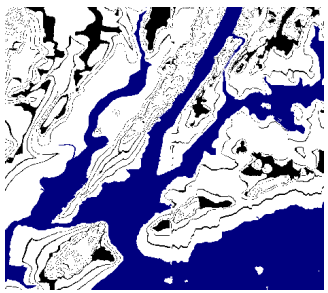
Break

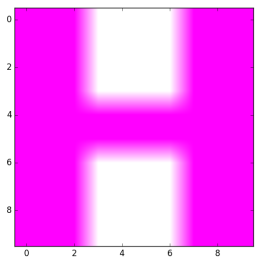
Outline



- Welcome and Introductions
- Workshop Overview
- Design Challenge: Flood Maps
  - Data: Elevations from NOAA
  - Packages: `numpy` and `pyplot`
  - Concepts: Colors and Arrays
  - Putting It Altogether
- Break
- Variations on the Theme
- Design a Challenge
- Wrap Up

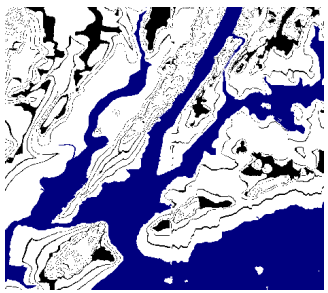
Variations on the Theme

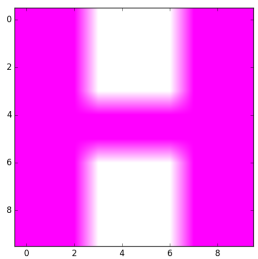




- Coastline Map
- Topographic Map
- School Logo

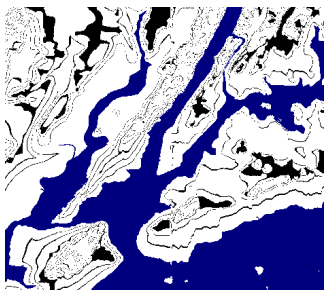
Variations on the Theme

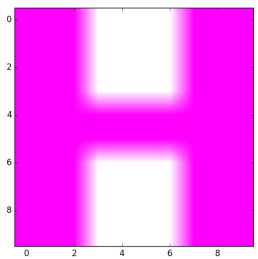




- Coastline Map: Modify your program:
  - If the elevation ≤ 0, color the pixel blue.
  - If the elevation $== 1$, color the pixel light gray.
  - Else, the pixel should be colored dark gray.

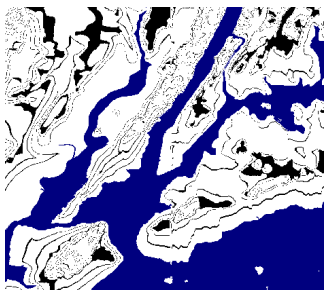
Variations on the Theme

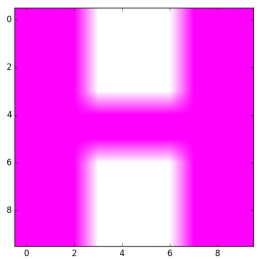




- Coastline Map: Modify your program:
  - If the elevation ≤ 0, color the pixel blue.
  - If the elevation $== 1$, color the pixel light gray.
  - Else, the pixel should be colored dark gray.
- Topographic Map: Modify your program:
  - If the elevation ≤ 0, color the pixel blue.
  - If the elevation is divisible by 10, color the pixel black.
  - Else, the pixel should be colored light gray.

Variations on the Theme





- Coastline Map: Modify your program:
  - If the elevation ≤ 0, color the pixel blue.
  - If the elevation $== 1$, color the pixel light gray.
  - Else, the pixel should be colored dark gray.
- Topographic Map: Modify your program:
  - If the elevation ≤ 0, color the pixel blue.
  - If the elevation is divisible by 10, color the pixel black.
  - Else, the pixel should be colored light gray.
- School Logo: Design a 10x10 grid with your school logo (use only white and one other color).

Variations on the Theme: Coastline Map

- Coastline Map: Modify your program:
  - If the elevation ≤ 0, color the pixel blue.
  - If the elevation $== 1$, color the pixel light gray.
  - Else, the pixel should be colored dark gray.

Variations on the Theme: Coastline Map

```
for row in range(mapShape[0]):
    for col in range(mapShape[1]):
        if elevations[row,col] <= 0:
            #Below sea level
            floodMap[row,col,2] = 0.5
        elif elevations[row,col] == 1:
            floodMap[row,col,0] = 0.75
            floodMap[row,col,1] = 0.75
            floodMap[row,col,2] = 0.75
        else:
            floodMap[row,col,0] = 0.5
            floodMap[row,col,1] = 0.5
            floodMap[row,col,2] = 0.5
```

Variations on the Theme: Topographic Map

- Topographic Map: Modify your program:
  - If the elevation ≤ 0, color the pixel blue.
  - If the elevation is divisible by 10, color the pixel black.
  - Else, the pixel should be colored light gray.

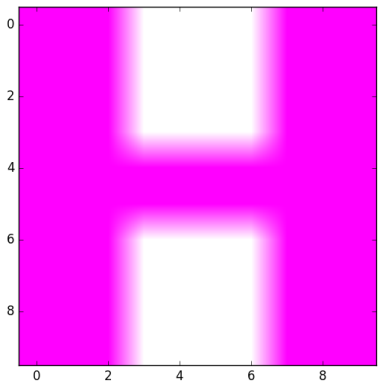
Variations on the Theme: Topographic Map

- Topographic Map: Modify your program:
  - If the elevation ≤ 0, color the pixel blue.
  - If the elevation is divisible by 10, color the pixel black.
  - Else, the pixel should be colored light gray.

Variations on the Theme: Topographic Map

```
for row in range(mapShape[0]):
    for col in range(mapShape[1]):
        if elevations[row,col] <= 0:
            #Below sea level
            floodMap[row,col,2] = blue
        elif elevations[row,col] %10 == 0:
            floodMap[row,col,0] = 0.0
            floodMap[row,col,1] = 0.0
            floodMap[row,col,2] = 0.0
        else:
            floodMap[row,col,0] = 1.0
            floodMap[row,col,1] = 1.0
            floodMap[row,col,2] = 1.0
```

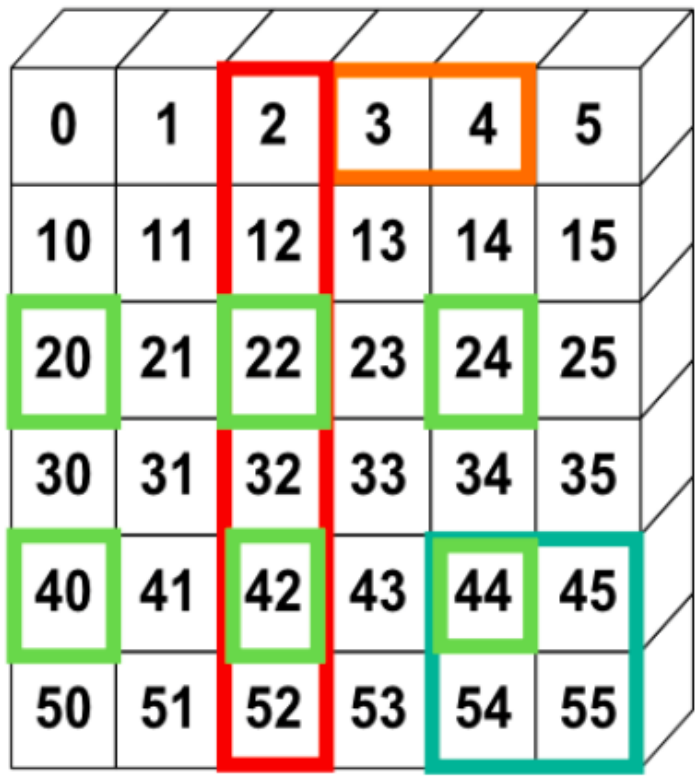
Variations on the Theme



- School Logo: Design a 10x10 grid with your school logo (use only white and one other color).

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]])
```

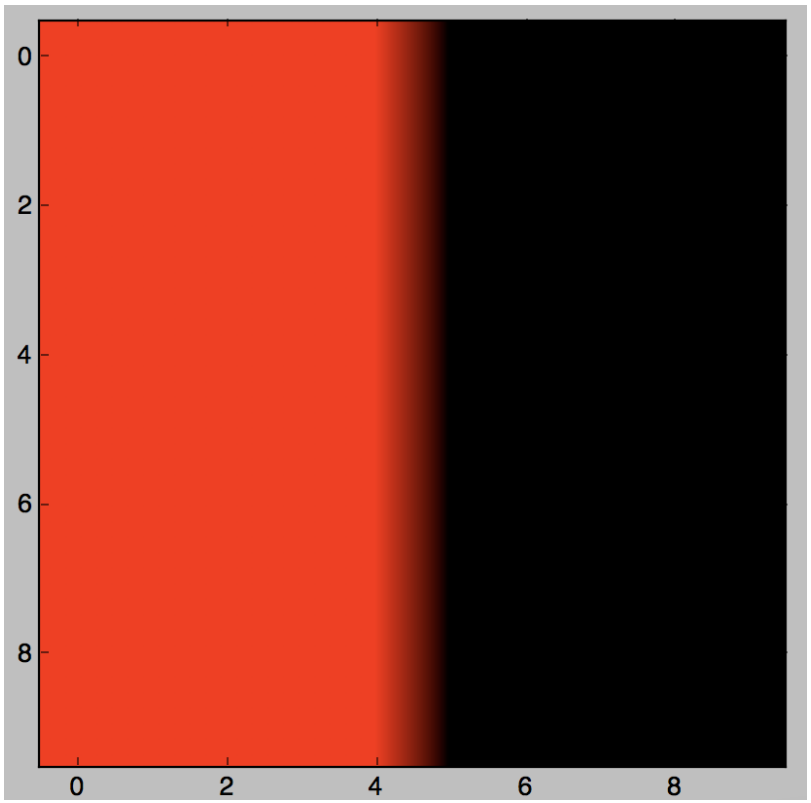


numpy tutorial

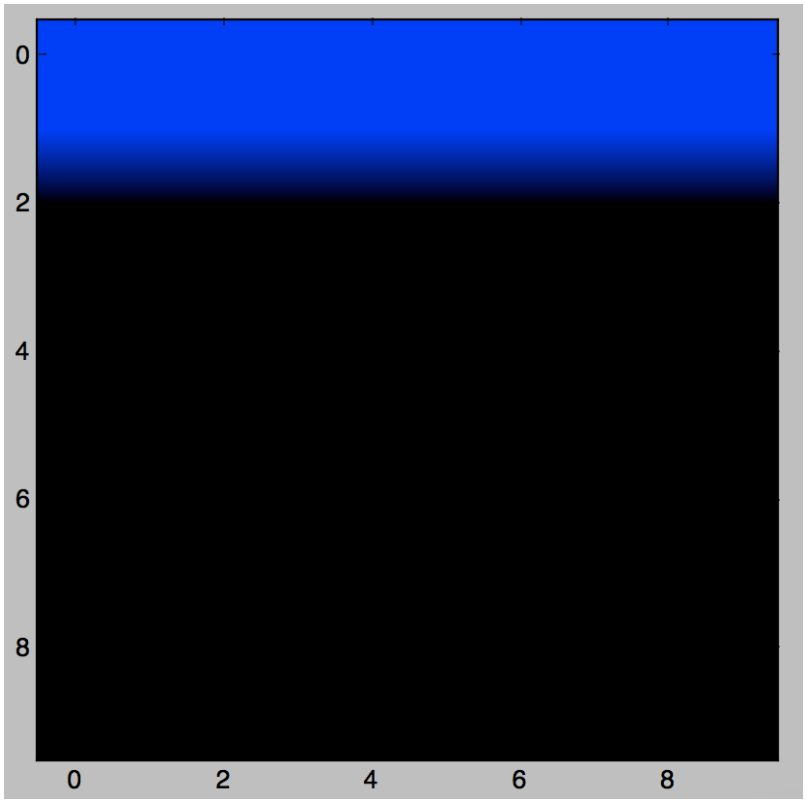
Slicing & Image Examples

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

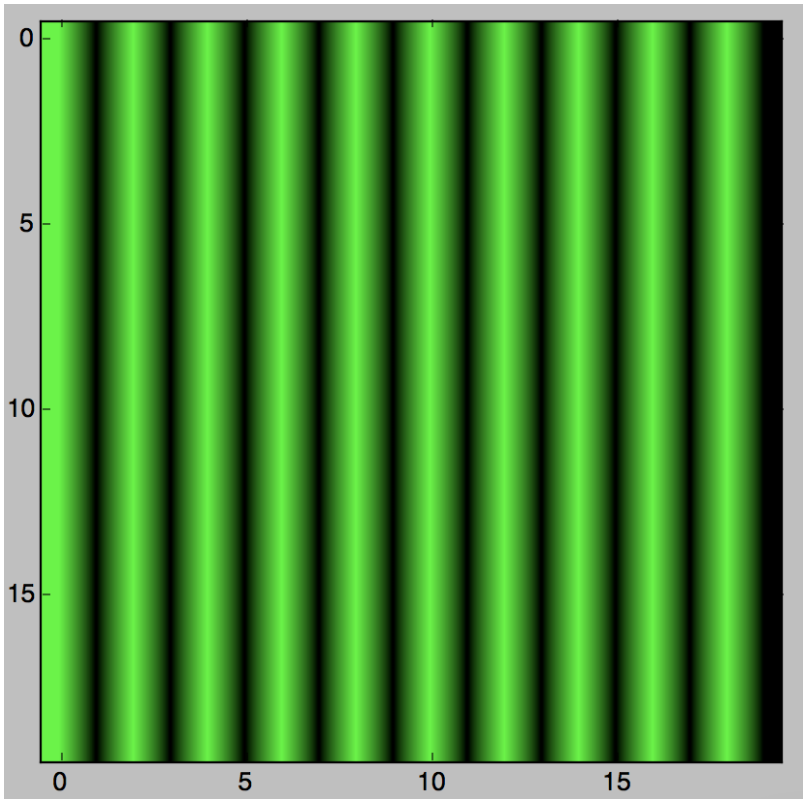
Slicing & Image Examples

- 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
    ```


    ```

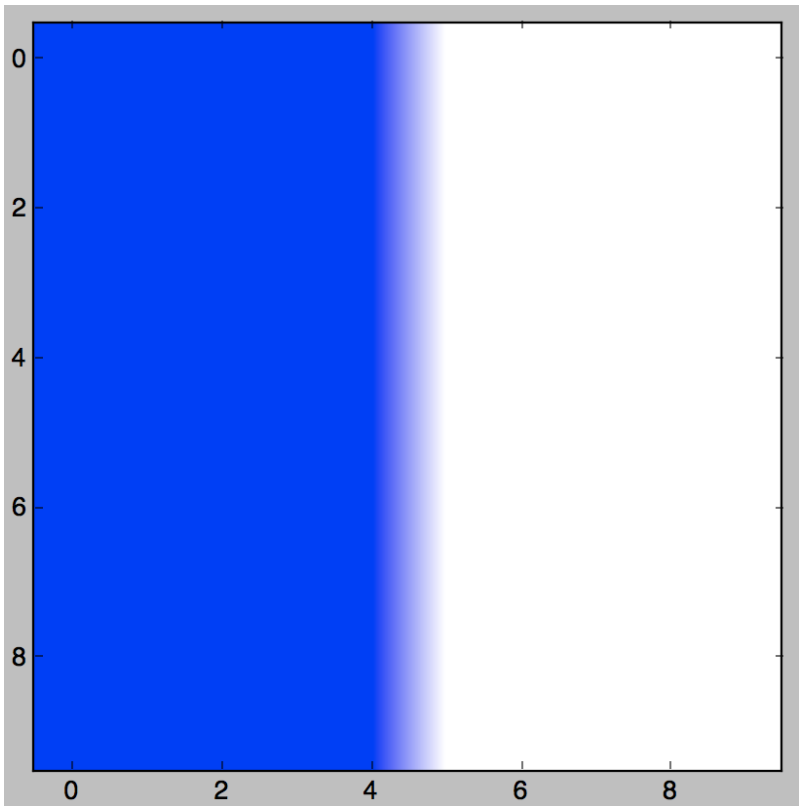
In Pairs or Triples

- 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
    ```
    ```

In Pairs or Triples

- 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
    ```

In Pairs or Triples

- 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
    ```
    ```

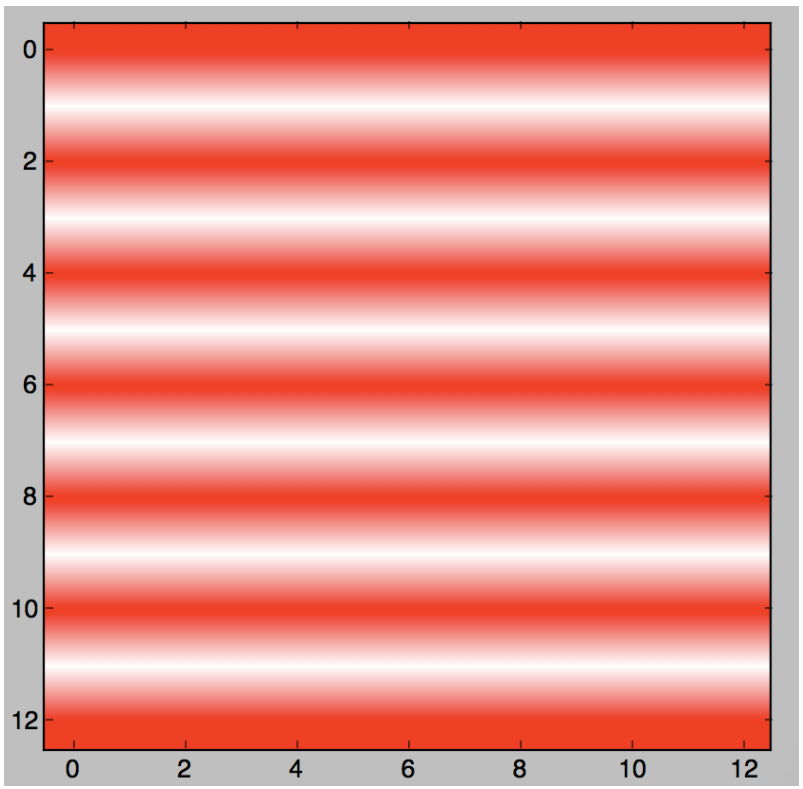
In Pairs or Triples

- 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
    ```
    ```

In Pairs or Triples

- 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
    ```


    ```

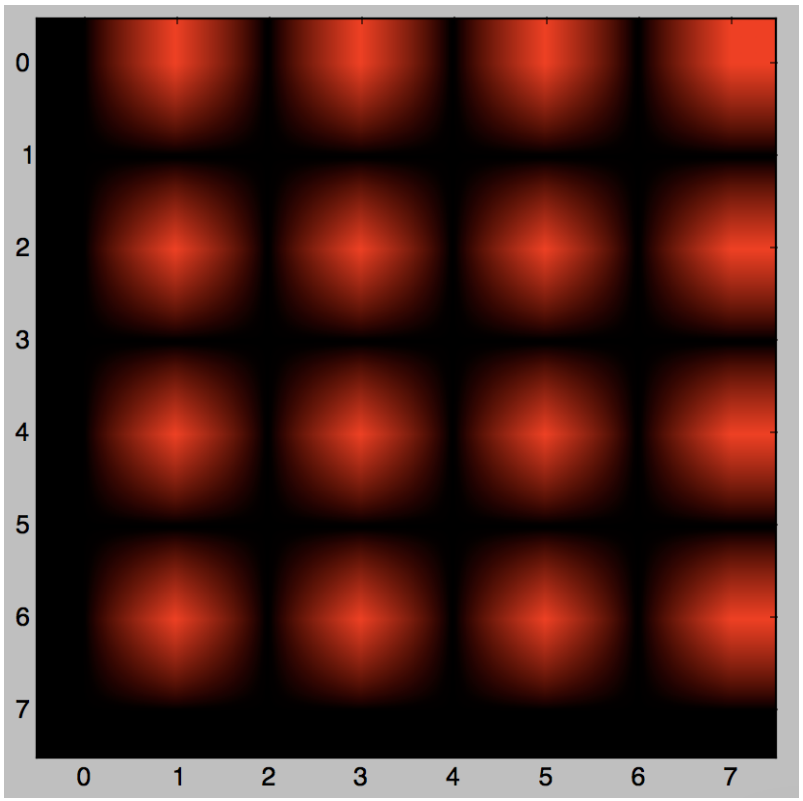
In Pairs or Triples

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

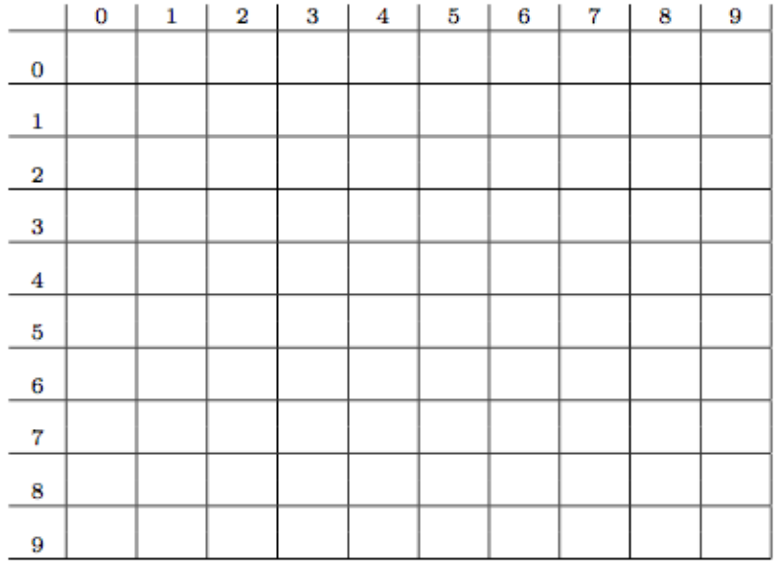
  - ```
    num = int(input('Enter size '))
    img = np.ones( (num,num,3) )
    img[::2,:,1:]  = 0
    ```
    ```

In Pairs or Triples

- 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
    ```


    ```

In Pairs or Triples

- 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
    ```
    ```

In Pairs or Triples

- 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
    ```


    ```

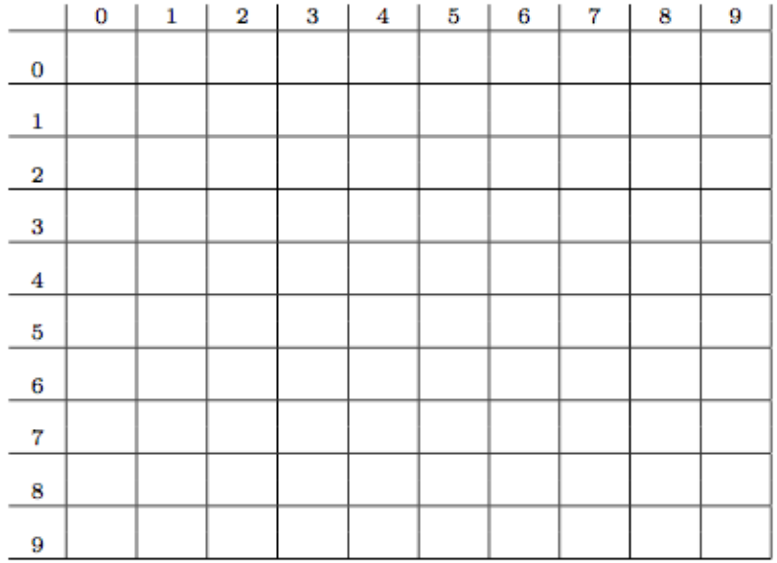
Side Challenge: School Logo



	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5										
6										
7										
8										
9										

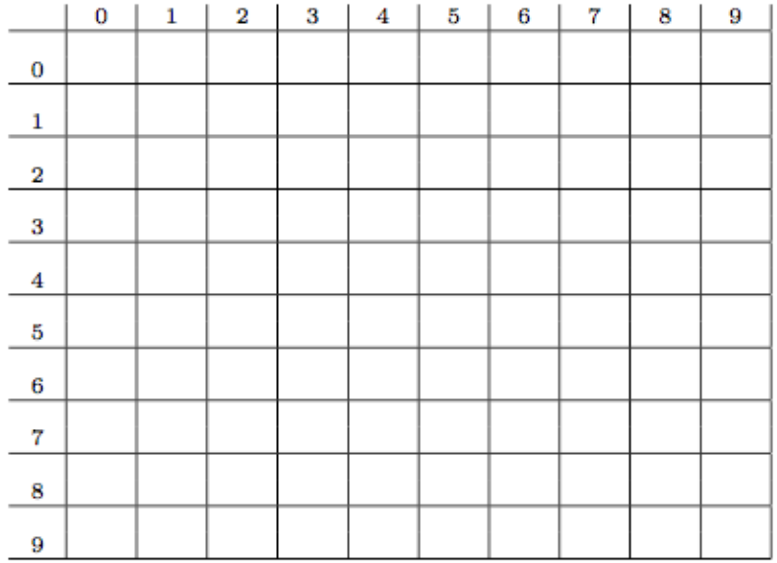
1. Design a 10 by 10 logo for Hunter College that contains a purple 'H'.

Side Challenge: School Logo



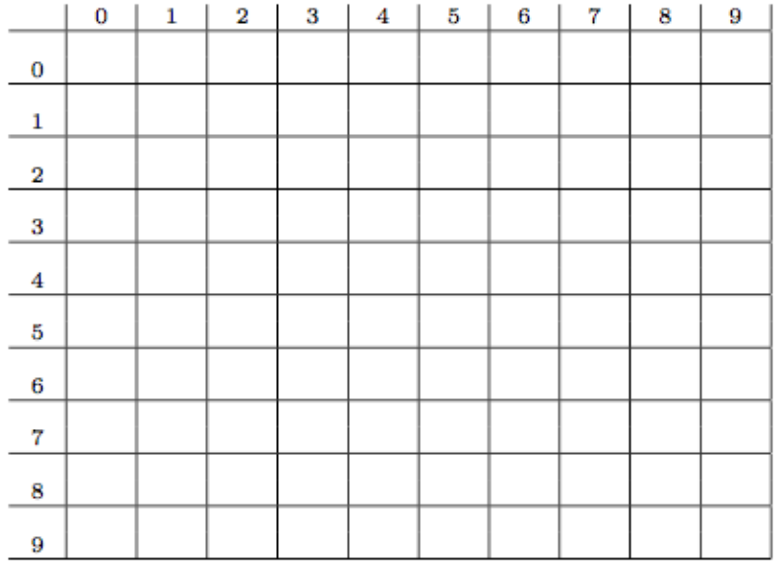
1. Design a 10 by 10 logo for Hunter College that contains a purple 'H'.
2. Your logo should only contain the colors purple and white.

Side Challenge: School Logo



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.

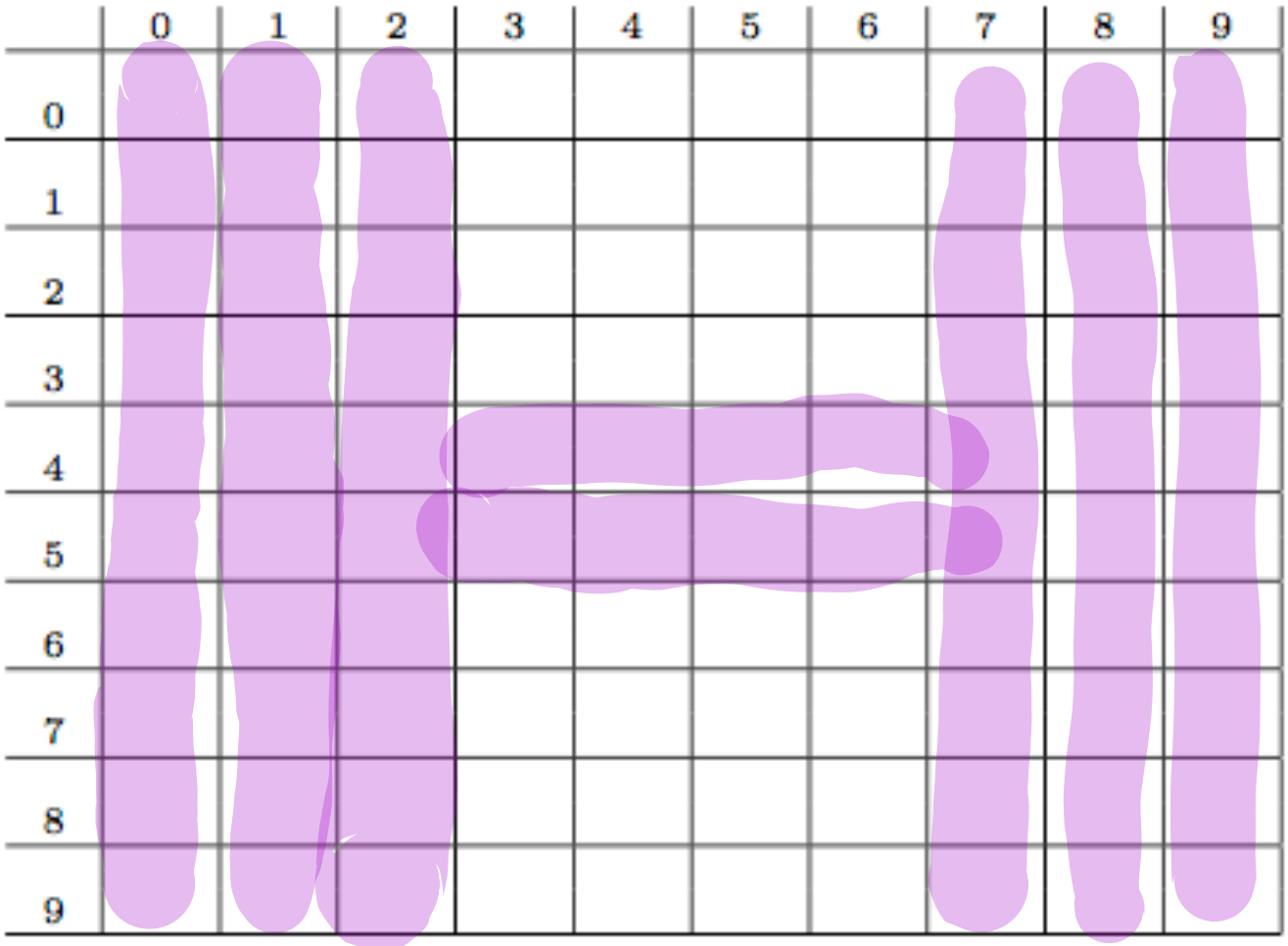
Side Challenge: School Logo



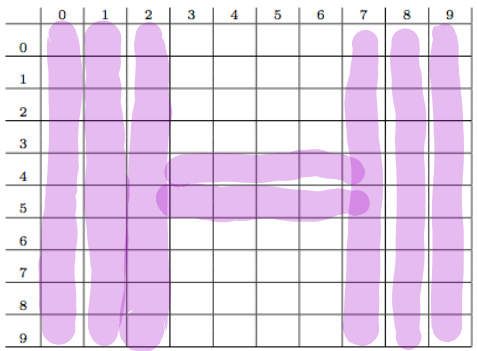
1. Design a 10 by 10 logo for Hunter College that contains a purple 'H'.
2. Your logo should only contain the colors purple and white.
3. How can you make Python draw the logo?
   Write down a "To Do" list of things you need to do.
4. If time, refine your steps above into a Python program.

Design a Hunter Logo

One possible solution:

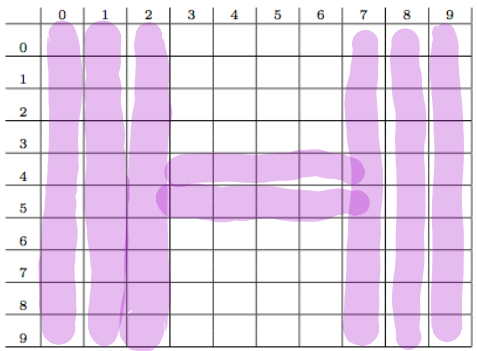


Design a Hunter Logo



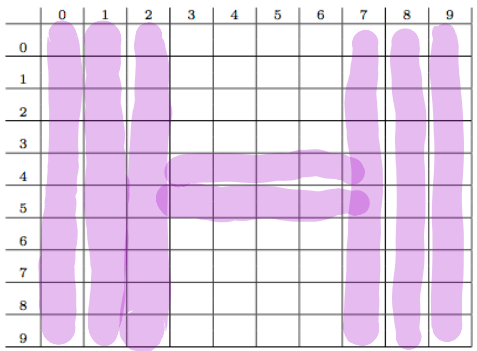
1. Create a 10 by 10 array, `logo`, that starts out as all white pixels.

Design a Hunter Logo



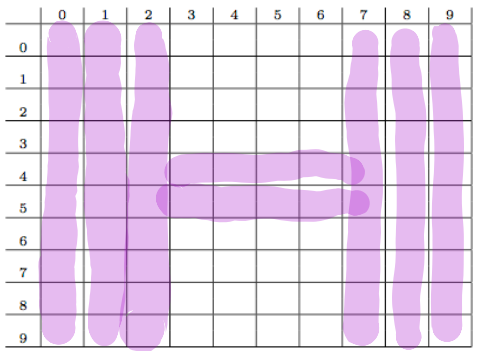
1. Create a 10 by 10 array, `logo`, that starts out as all white pixels.
2. Set the 3 left columns to be purple.

Design a Hunter Logo



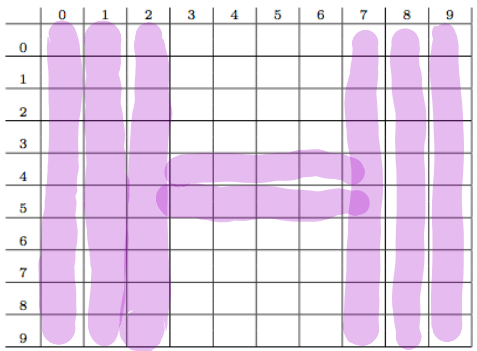
1. Create a 10 by 10 array, `logo`, that starts out as all white pixels.
2. Set the 3 left columns to be purple.
3. Set the 3 right columns to be purple.

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.

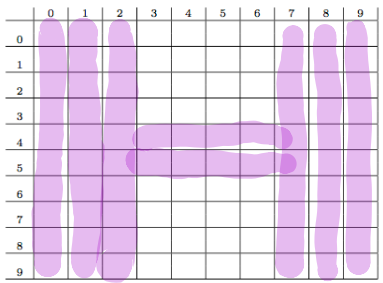
Design a Hunter Logo



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

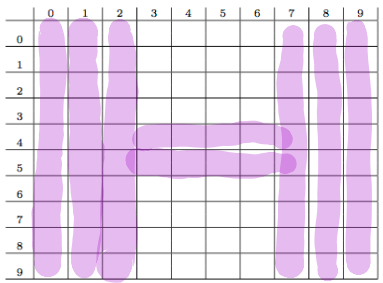
1. Create a 10 by 10 array, `logo`, that starts out as all white pixels.



Translating the Design to Code

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

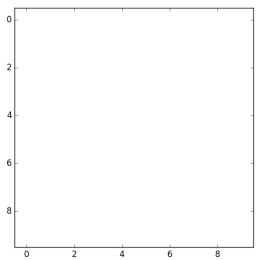
```
import matplotlib.pyplot as plt #import libraries for plotting
import numpy as np              #and for arrays (to hold images)
logoImg = np.ones((10,10,3))    #10x10 array with 3 sheets of 1's
```



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
```

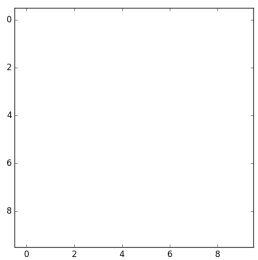


Translating the Design to Code

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.



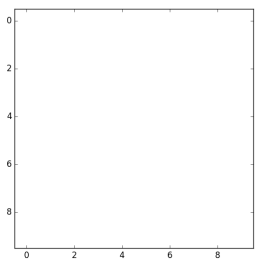
Translating the Design to Code

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

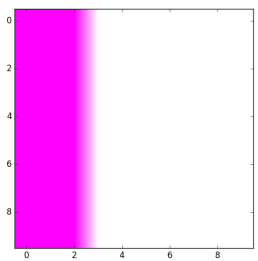
```
import matplotlib.pyplot as plt #import libraries for plotting
import numpy as np              #and for arrays (to hold images)
logoImg = np.ones((10,10,3))    #10x10 array with 3 sheets of 1's
```

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



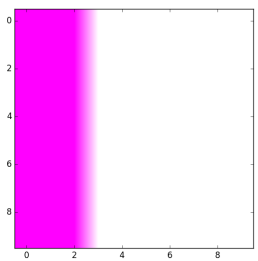
1. Create a 10 by 10 array, `logo`, that starts out as all white pixels.

```
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
```

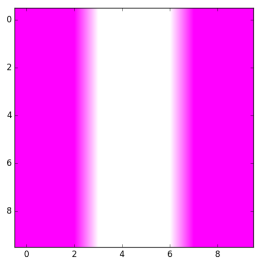
2. Set the 3 left columns to be purple.

```
#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, `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
```

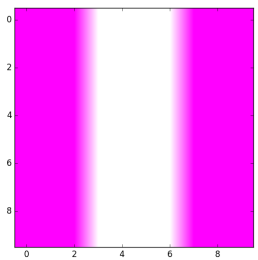
2. Set the 3 left columns to be purple.

```
#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.

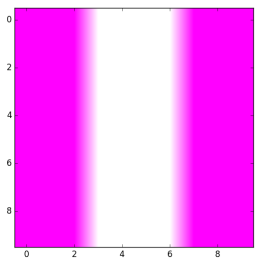
```
#To make purple, we'll keep red and blue at 100% and turn green to 0%
logoImg[:,:3,1] = 0 #Turn the green to 0 for first 3 columns
```

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
```

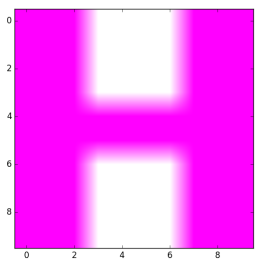
3. Set the 3 right columns to be purple.

```
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
```

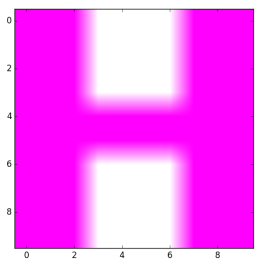
3. Set the 3 right columns to be purple.

```
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.

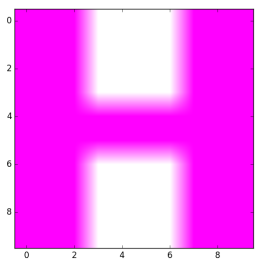
```
logoImg[:,-3:,1] = 0 #Turn the green to 0 for last 3 columns
```

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
```

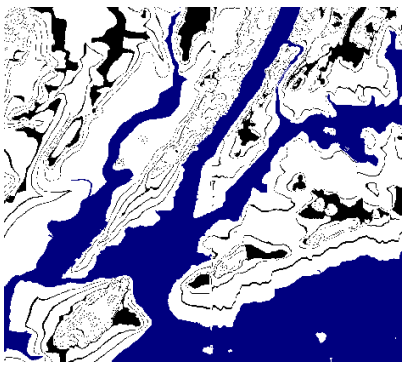
4. Set the middle 2 rows to be purple.

```
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
```

Outline



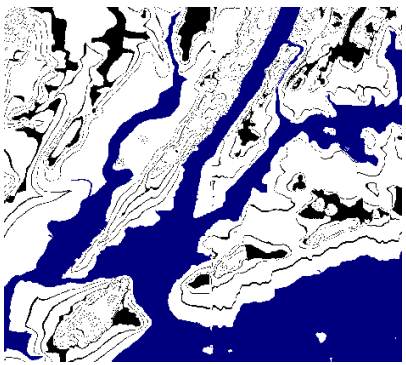
- Welcome and Introductions
- Workshop Overview
- Design Challenge: Flood Maps
  - Data: Elevations from NOAA
  - Packages: `numpy` and `pyplot`
  - Concepts: Colors and Arrays
  - Putting It Altogether
- Break
- Variations on the Theme
- Design a Challenge
- Wrap Up

Design a Challenge

With your group, brainstorm about a design challenge that:

- Creates an image with 2 or more colors.
- Can be based on a data file or implement a student design.

Outline



- Welcome and Introductions
- Workshop Overview
- Design Challenge: Flood Maps
  - Data: Elevations from NOAA
  - Packages: `numpy` and `pyplot`
  - Concepts: Colors and Arrays
  - Putting It Altogether
- Break
- Variations on the Theme
- Design a Challenge
- Wrap Up

Wrap Up

- Design a flood map of NYC.

Wrap Up

- Design a flood map of NYC.
  - Used NOAA data.

Wrap Up

- Design a flood map of NYC.
  - Used NOAA data.
  - Introduced Colors and Arrays.

Wrap Up

- Design a flood map of NYC.
  - Used NOAA data.
  - Introduced Colors and Arrays.
  - Introduced standard plotting `plt` & numerical analysis packages `np`.

Wrap Up

- Design a flood map of NYC.
  - Used NOAA data.
  - Introduced Colors and Arrays.
  - Introduced standard plotting `plt` & numerical analysis packages `np`.
- Variations on the theme: pixel art (logos), topo maps, & coastlines.

Wrap Up

- Design a flood map of NYC.
  - Used NOAA data.
  - Introduced Colors and Arrays.
  - Introduced standard plotting `plt` & numerical analysis packages `np`.
- Variations on the theme: pixel art (logos), topo maps, & coastlines.
- Designed your own challenges.

Wrap Up

- Design a flood map of NYC.
  - Used NOAA data.
  - Introduced Colors and Arrays.
  - Introduced standard plotting `plt` & numerical analysis packages `np`.
- Variations on the theme: pixel art (logos), topo maps, & coastlines.
- Designed your own challenges.

- See you in two weeks!