

# CSci 127: Introduction to Computer Science



[hunter.cuny.edu/csci](http://hunter.cuny.edu/csci)

# Frequently Asked Questions

From email.

# Frequently Asked Questions

From email.

- Could you spend more time on circuits/logical expressions/truth tables/decisions?

# Frequently Asked Questions

From email.

- Could you spend more time on circuits/logical expressions/truth tables/decisions?

*Yes, we will start with that today.*

# Frequently Asked Questions

From email.

- Could you spend more time on circuits/logical expressions/truth tables/decisions?  
*Yes, we will start with that today.*
- I still don't get indices and the brackets. Could you spend more time on that?

# Frequently Asked Questions

From email.

- **Could you spend more time on circuits/logical expressions/truth tables/decisions?**  
*Yes, we will start with that today.*
- **I still don't get indices and the brackets. Could you spend more time on that?**  
*Yes, we will, since*
  - 1) *it's fundamental, and*
  - 2) *the same ideas are used for accessing formatted data (today's topic).*

# Frequently Asked Questions

From email.

- **Could you spend more time on circuits/logical expressions/truth tables/decisions?**  
*Yes, we will start with that today.*
- **I still don't get indices and the brackets. Could you spend more time on that?**  
*Yes, we will, since*  
1) *it's fundamental, and*  
2) *the same ideas are used for accessing formatted data (today's topic).*
- **I still don't get what is meant by input?**

# Frequently Asked Questions

From email.

- **Could you spend more time on circuits/logical expressions/truth tables/decisions?**  
*Yes, we will start with that today.*
- **I still don't get indices and the brackets. Could you spend more time on that?**  
*Yes, we will, since*  
1) *it's fundamental, and*  
2) *the same ideas are used for accessing formatted data (today's topic).*
- **I still don't get what is meant by input?**  
*Input is data provided to a program each time it runs, it may change at each run.  
In this course we have used the `input()` function.*

# Today's Topics



- Recap: Logical Expressions & Circuits
- Design: Cropping Images
- Accessing Formatted Data

# Today's Topics



- **Recap: Logical Expressions & Circuits**
- Design: Cropping Images
- Accessing Formatted Data

# Recap: Logical Operators

## and

in1	and	in2	<i>returns:</i>
False	and	False	False
False	and	True	False
True	and	False	False
True	and	True	True

# Recap: Logical Operators

**and**

in1	in2	<i>returns:</i>
False	and	False
False	and	True
True	and	False
True	and	True

**or**

in1	in2	<i>returns:</i>
False	or	False
False	or	True
True	or	False
True	or	True

# Recap: Logical Operators

**and**

in1		in2	<i>returns:</i>
False	and	False	False
False	and	True	False
True	and	False	False
True	and	True	True

**or**

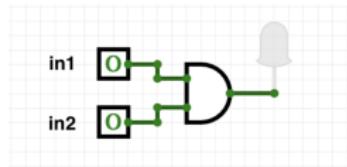
in1		in2	<i>returns:</i>
False	or	False	False
False	or	True	True
True	or	False	True
True	or	True	True

**not**

	in1	<i>returns:</i>
not	False	True
not	True	False

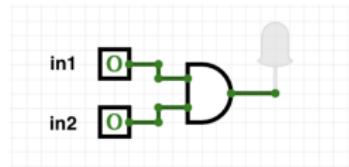
# Logical Operators & Circuits

- Each logical operator (and, or, & not) can be used to join together expressions.



# Logical Operators & Circuits

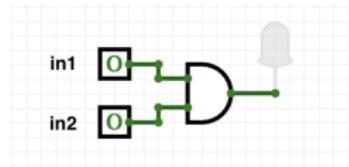
- Each logical operator (and, or, & not) can be used to join together expressions.



Example:  $\text{in1} \text{ and } \text{in2}$

# Logical Operators & Circuits

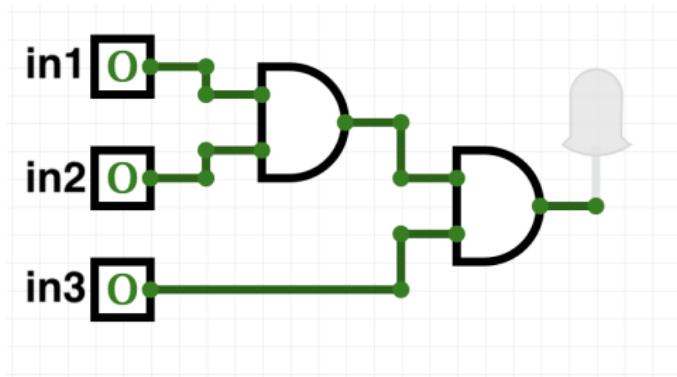
- Each logical operator (and, or, & not) can be used to join together expressions.



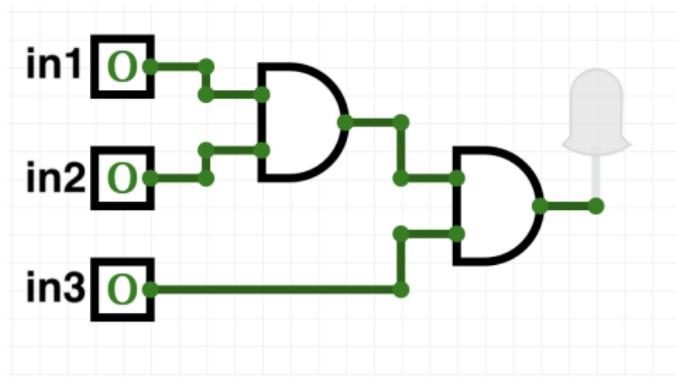
Example:  $\text{in1} \text{ and } \text{in2}$

- Each logical operator (and, or, & not) has a corresponding logical circuit that can be used to join together inputs.

# Examples: Logical Circuit



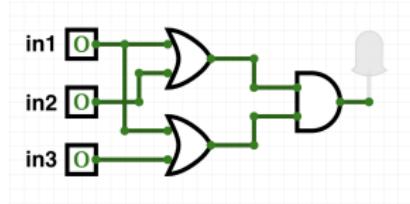
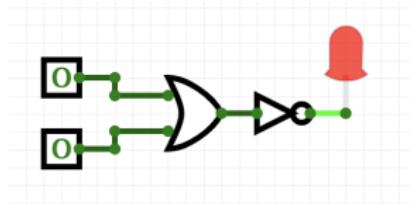
## Examples: Logical Circuit



$(in1 \text{ and } in2) \text{ and } in3$

# More Circuit Examples

*Examples from last lecture:*



Draw a circuit that corresponds to each logical expression:

- $\text{not}(\text{in1 or in2})$
- $(\text{in1 or in2}) \text{ and } (\text{in1 or in3})$
- $(\text{not(in1 and not in2)}) \text{ or } (\text{in1 and (in2 and in3)})$

# Challenge:

Predict what the code will do:

```
x = 6
y = x % 4
w = y**3
z = w // 2
print(x,y,w,z)
x,y = y,w
print(x,y,w,z)
x = y / 2
print(x,y,w,z)

sports = ["Field Hockey", "Swimming", "Water Polo"]
mess = "Qoauxca BrletRce crcx qvBnqa ocUxk"
result = ""
for i in range(len(mess)):
    if i % 3 == 0:
        print(mess[i])
        result = result + mess[i]
print(sports[1], result)
```

# Python Tutor

```
x = 6
y = x % 4
w = y**3
z = w // 2
print(x,y,w,z)      (Demo with pythonTutor)
x,y = y,w
print(x,y,w,z)
x = y / 2
print(x,y,w,z)
```

# Today's Topics



- Recap: Logical Expressions & Circuits
- **Design: Cropping Images**
- Accessing Formatted Data
- CS Survey: Astrophysics and astropy

# Challenge: Design Question

From Final Exam, Fall 2017, V4, #6.



Design an algorithm that reads in an image and displays the lower left corner of the image.

# Challenge: Design Question

From Final Exam, Fall 2017, V4, #6.



Design an algorithm that reads in an image and displays the lower left corner of the image.

**Input:**

**Output:**

**Process:** (*Brainstorm for a “To Do” list to accomplish this.*)

# Design Question

Design a program that asks the user for an image and then display the upper left quarter of the image. (First, design the pseudocode, and if time, expand to a Python program.)

# Design Question

Design a program that asks the user for an image and then display the upper left quarter of the image. (First, design the pseudocode, and if time, expand to a Python program.)

How to approach this:

- Create a “To Do” list of what your program has to accomplish.

# Design Question

Design a program that asks the user for an image and then display the upper left quarter of the image. (First, design the pseudocode, and if time, expand to a Python program.)

How to approach this:

- Create a “To Do” list of what your program has to accomplish.
- Read through the problem, and break it into “To Do” items.

# Design Question

Design a program that asks the user for an image and then display the upper left quarter of the image. (First, design the pseudocode, and if time, expand to a Python program.)

How to approach this:

- Create a “To Do” list of what your program has to accomplish.
- Read through the problem, and break it into “To Do” items.
- Don’t worry if you don’t know how to do all the items you write down.

# Design Question

Design a program that asks the user for an image and then display the upper left quarter of the image. (First, design the pseudocode, and if time, expand to a Python program.)

How to approach this:

- Create a “To Do” list of what your program has to accomplish.
- Read through the problem, and break it into “To Do” items.
- Don’t worry if you don’t know how to do all the items you write down.
- Example:

# Design Question

Design a program that asks the user for an image and then display the upper left quarter of the image. (First, design the pseudocode, and if time, expand to a Python program.)

How to approach this:

- Create a “To Do” list of what your program has to accomplish.
- Read through the problem, and break it into “To Do” items.
- Don’t worry if you don’t know how to do all the items you write down.
- Example:
  - ① Import libraries.

# Design Question

Design a program that asks the user for an image and then display the upper left quarter of the image. (First, design the pseudocode, and if time, expand to a Python program.)

How to approach this:

- Create a “To Do” list of what your program has to accomplish.
- Read through the problem, and break it into “To Do” items.
- Don’t worry if you don’t know how to do all the items you write down.
- Example:
  - ① Import libraries.
  - ② Ask user for an image name.

# Design Question

Design a program that asks the user for an image and then display the upper left quarter of the image. (First, design the pseudocode, and if time, expand to a Python program.)

How to approach this:

- Create a “To Do” list of what your program has to accomplish.
- Read through the problem, and break it into “To Do” items.
- Don’t worry if you don’t know how to do all the items you write down.
- Example:
  - ① Import libraries.
  - ② Ask user for an image name.
  - ③ Read in image.

# Design Question

Design a program that asks the user for an image and then display the upper left quarter of the image. (First, design the pseudocode, and if time, expand to a Python program.)

How to approach this:

- Create a “To Do” list of what your program has to accomplish.
- Read through the problem, and break it into “To Do” items.
- Don’t worry if you don’t know how to do all the items you write down.
- Example:
  - ① Import libraries.
  - ② Ask user for an image name.
  - ③ Read in image.
  - ④ Figure out size of image.

# Design Question

Design a program that asks the user for an image and then display the upper left quarter of the image. (First, design the pseudocode, and if time, expand to a Python program.)

How to approach this:

- Create a “To Do” list of what your program has to accomplish.
- Read through the problem, and break it into “To Do” items.
- Don’t worry if you don’t know how to do all the items you write down.
- Example:
  - ① Import libraries.
  - ② Ask user for an image name.
  - ③ Read in image.
  - ④ Figure out size of image.
  - ⑤ Make a new image that’s half the height and half the width.

# Design Question

Design a program that asks the user for an image and then display the upper left quarter of the image. (First, design the pseudocode, and if time, expand to a Python program.)

How to approach this:

- Create a “To Do” list of what your program has to accomplish.
- Read through the problem, and break it into “To Do” items.
- Don’t worry if you don’t know how to do all the items you write down.
- Example:
  - ① Import libraries.
  - ② Ask user for an image name.
  - ③ Read in image.
  - ④ Figure out size of image.
  - ⑤ Make a new image that’s half the height and half the width.
  - ⑥ Display the new image.

# In Pairs or Triples: Design Question



- ① Import libraries.

# In Pairs or Triples: Design Question



- ① Import libraries.

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

# In Pairs or Triples: Design Question



- ① Import libraries.

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

- ② Ask user for an image name.

# In Pairs or Triples: Design Question



- ① Import libraries.

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

- ② Ask user for an image name.

```
inF = input('Enter file name: ')
```

# In Pairs or Triples: Design Question



- ① Import libraries.

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

- ② Ask user for an image name.

```
inF = input('Enter file name: ')
```

- ③ Read in image.

# In Pairs or Triples: Design Question



- ① Import libraries.

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

- ② Ask user for an image name.

```
inF = input('Enter file name: ')
```

- ③ Read in image.

```
img = plt.imread(inF) #Read in image from inF
```

# In Pairs or Triples: Design Question



- ① Import libraries.

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

- ② Ask user for an image name.

```
inF = input('Enter file name: ')
```

- ③ Read in image.

```
img = plt.imread(inF) #Read in image from inF
```

- ④ Figure out size of image.

# In Pairs or Triples: Design Question



- ① Import libraries.

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

- ② Ask user for an image name.

```
inF = input('Enter file name: ')
```

- ③ Read in image.

```
img = plt.imread(inF) #Read in image from inF
```

- ④ Figure out size of image.

```
height = img.shape[0] #Get height  
width = img.shape[1] #Get width
```

# In Pairs or Triples: Design Question



The City University of New York



- ① Import libraries.

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

- ② Ask user for an image name.

```
inF = input('Enter file name: ')
```

- ③ Read in image.

```
img = plt.imread(inF) #Read in image from inF
```

- ④ Figure out size of image.

```
height = img.shape[0] #Get height  
width = img.shape[1] #Get width
```

- ⑤ Make a new image that's half the height and half the width.

# In Pairs or Triples: Design Question



- ① Import libraries.

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

- ② Ask user for an image name.

```
inF = input('Enter file name: ')
```

- ③ Read in image.

```
img = plt.imread(inF) #Read in image from inF
```

- ④ Figure out size of image.

```
height = img.shape[0] #Get height  
width = img.shape[1] #Get width
```

- ⑤ Make a new image that's half the height and half the width.

```
img2 = img[height//2:, :width//2] #Crop to lower left corner
```

# In Pairs or Triples: Design Question



The City University of New York



- ① Import libraries.

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

- ② Ask user for an image name.

```
inF = input('Enter file name: ')
```

- ③ Read in image.

```
img = plt.imread(inF) #Read in image from inF
```

- ④ Figure out size of image.

```
height = img.shape[0] #Get height  
width = img.shape[1] #Get width
```

- ⑤ Make a new image that's half the height and half the width.

```
img2 = img[height//2:, :width//2] #Crop to lower left corner
```

- ⑥ Display the new image.

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

# Today's Topics



- Recap: Logical Expressions & Circuits
- Design: Cropping Images
- **Accessing Formatted Data**
- CS Survey: Astrophysics and astropy

# Structured Data

College	Undergraduate		
	Full-time	Part-time	Total
Baruch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,633	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

- Common to have data structured in a spread sheet.

# Structured Data

College	Undergraduate		
	Full-time	Part-time	Total
Baruch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,633	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

- Common to have data structured in a spread sheet.
- In the example above, we have the first line that says “Undergraduate”.

# Structured Data

College	Undergraduate		
	Full-time	Part-time	Total
Baruch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,633	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

- Common to have data structured in a spread sheet.
- In the example above, we have the first line that says “Undergraduate”.
- Next line has the titles for the columns.

# Structured Data

College	Undergraduate		
	Full-time	Part-time	Total
Baruch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,633	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

- Common to have data structured in a spread sheet.
- In the example above, we have the first line that says “Undergraduate”.
- Next line has the titles for the columns.
- Subsequent lines have a college and attributes about the college.

# Structured Data

College	Undergraduate		
	Full-time	Part-time	Total
Baruch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,633	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

- Common to have data structured in a spread sheet.
- In the example above, we have the first line that says “Undergraduate”.
- Next line has the titles for the columns.
- Subsequent lines have a college and attributes about the college.
- Python has several ways to read in such data.

# Structured Data

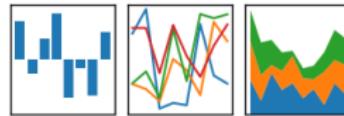
College	Undergraduate		
	Full-time	Part-time	Total
Baruch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,633	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

- Common to have data structured in a spread sheet.
- In the example above, we have the first line that says “Undergraduate”.
- Next line has the titles for the columns.
- Subsequent lines have a college and attributes about the college.
- Python has several ways to read in such data.
- We will use the popular Python Data Analysis Library (**Pandas**).

# Structured Data

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

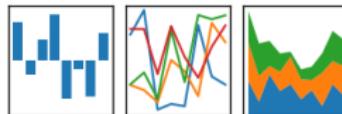


- We will use the popular Python Data Analysis Library (**Pandas**).

# Structured Data

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

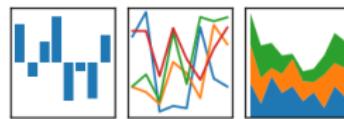


- We will use the popular Python Data Analysis Library (**Pandas**).
- Open source and freely available (part of anaconda distribution).

# Structured Data

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

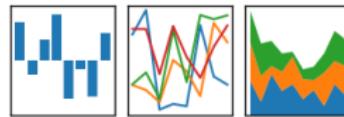


- We will use the popular Python Data Analysis Library (**Pandas**).
- Open source and freely available (part of anaconda distribution).
- See Lab 1 for directions on downloading it to your home machine.

# Structured Data

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

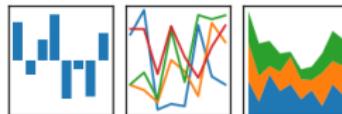


- We will use the popular Python Data Analysis Library (**Pandas**).
- Open source and freely available (part of anaconda distribution).
- See Lab 1 for directions on downloading it to your home machine.
- If you can't install on your computer, it is supported in  
<https://repl.it/>

# Structured Data

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



- We will use the popular Python Data Analysis Library (**Pandas**).
- Open source and freely available (part of anaconda distribution).
- See Lab 1 for directions on downloading it to your home machine.
- If you can't install on your computer, it is supported in  
<https://repl.it/>
- To use, add to the top of your program:

```
import pandas as pd
```

# CSV Files

College	Undergraduate		
	Full-time	Part-time	Total
Baruch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,633	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

- Excel .xls files have much extra formatting.

# CSV Files

College	Undergraduate		
	Full-time	Part-time	Total
Baruch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,633	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

- Excel .xls files have much extra formatting.
- The text file version is called **CSV** for comma separated values.

# CSV Files

College	Undergraduate		
	Full-time	Part-time	Total
Baruch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,633	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

- Excel .xls files have much extra formatting.
- The text file version is called **CSV** for comma separated values.
- Each row is a line in the file.

# CSV Files

College	Undergraduate		
	Full-time	Part-time	Total
Baruch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,633	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

- Excel .xls files have much extra formatting.
- The text file version is called **CSV** for comma separated values.
- Each row is a line in the file.
- Columns are separated by commas on each line.

# CSV Files

Source: [https://en.wikipedia.org/wiki/Demographics\\_of\\_New\\_York\\_City](https://en.wikipedia.org/wiki/Demographics_of_New_York_City),  
All population figures are consistent with present-day boundaries.,,  
First census after the consolidation of the five boroughs.,,  
.,,,  
.,,,  
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total  
1698,4937,2017,,,727,7681  
1771,21863,3623,,,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6642,1755,4563,79215  
1810,96373,8303,7444,2267,5347,119734  
1820,123706,11187,8246,2782,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312710,47613,14480,5346,10965,391114  
1850,515547,138882,18593,8032,15061,696115  
1860,813669,279122,32903,23593,25492,1174779  
1870,942292,419921,45468,37393,33029,1478103  
1880,1164673,599495,56559,51980,38991,1911698  
1890,1441216,838547,87050,88908,51693,2507414  
1900,1850093,1166582,152999,200507,67021,3437202  
1910,2331542,1634351,284041,430980,85969,4766883  
1920,2284103,2018356,469042,732016,116531,5620048  
1930,1867312,2560401,1079129,1265258,158346,6930446  
1940,1889924,2698285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550849,1451277,191555,7891957  
1960,1698281,2627319,1809578,1424815,221991,7781984  
1970,1539233,2602012,1986473,1471701,295443,7894862  
1980,1428285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,378977,7322564  
2000,1537195,2465326,2229379,1332650,443728,8008278  
2010,1585873,2504700,2230722,1385108,468730,8175133  
2015,1644518,2636735,2339150,1455444,474558,8550405

nycHistPop.csv

# Reading in CSV Files

College	Undergraduate		
	Full-time	Part-time	Total
Baruch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,633	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

- To read in a CSV file: `myVar = pd.read_csv("myFile.csv")`

# Reading in CSV Files

College	Undergraduate		
	Full-time	Part-time	Total
Baruch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,633	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

- To read in a CSV file: `myVar = pd.read_csv("myFile.csv")`
- Pandas has its own type, **DataFrame**, that is perfect for holding a sheet of data.

# Reading in CSV Files

College	Undergraduate		
	Full-time	Part-time	Total
Baruch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,633	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

- To read in a CSV file: `myVar = pd.read_csv("myFile.csv")`
- Pandas has its own type, **DataFrame**, that is perfect for holding a sheet of data.
- Often abbreviated: `df`.

# Reading in CSV Files

College	Undergraduate		
	Full-time	Part-time	Total
Baruch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,633	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

- To read in a CSV file: `myVar = pd.read_csv("myFile.csv")`
- Pandas has its own type, **DataFrame**, that is perfect for holding a sheet of data.
- Often abbreviated: `df`.
- It also has **Series**, that is perfect for holding a row or column of data.

# Example: Reading in CSV Files

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total  
1690,4937,103,727,768,422  
1771,21843,36231,,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,67541,6254,6840,2000,49734  
1820,123704,11487,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,18013,14049,5346,10965,391114  
1850,35549,12854,18850,5346,10965,391115  
1860,513469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33021,1911801  
1890,1367711,66711,6348,51980,33021,1911804  
1900,1850593,1165852,152999,200567,67621,2437202  
1910,223342,1634351,284041,430980,8569,476683  
1920,2211103,2018354,44601,44601,73201,11651,50048  
1930,1667128,1667128,1667128,1667128,1667128,4930446  
1940,1889924,2698285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690101,2738175,1550949,1451277,191555,7981984  
1970,1539231,2460705,1471071,1471071,135443,7981984  
1980,1426285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,378977,7322564  
2000,1537195,2485326,2229379,1332450,419782,8080879  
2010,1583873,2504705,2216722,1385108,474558,8175133  
2015,1444518,2636733,2339150,1459446,474558,8059405
```

nycHistPop.csv

In Lab 6

# Example: Reading in CSV Files

```
import matplotlib.pyplot as plt  
import pandas as pd
```

Source: [https://en.wikipedia.org/wiki/Demographics\\_of\\_New\\_York\\_City](https://en.wikipedia.org/wiki/Demographics_of_New_York_City),  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total  
1690,4937,2017,...,727,7881  
1771,21843,36231,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,67541,6200,6800,1750,4573,83934  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,311510,19113,14000,5346,10965,391114  
1850,355441,21800,18800,5346,10965,43115  
1860,613469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33029,1911801  
1890,1364473,66000,5653,51980,33029,2081534  
1900,185093,116582,152999,200567,67621,2437202  
1910,223342,1634351,28441,430980,8569,476683  
1920,2210103,2018354,44607,44607,73201,11651,50048  
1930,2671137,2490285,1297634,1297634,1297634,4930446  
1940,1889924,2490285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690101,2738175,1550949,1451277,191555,7981984  
1970,1539231,2460705,1471071,1471071,135443,7691846  
1980,1428285,2230936,1891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1320789,378977,7322564  
2000,1537195,2485326,2223379,1332450,419782,8080879  
2010,1583873,2504705,2272722,1385108,447515,8175133  
2015,1444018,2436733,2339150,1459446,474558,8059405

nycHistPop.csv

In Lab 6

# Example: Reading in CSV Files

```
import matplotlib.pyplot as plt  
import pandas as pd
```

```
pop = pd.read_csv('nycHistPop.csv', skiprows=5)
```

```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Bronx,Brooklyn,Queens,Bronx,Staten Island,Totals  
1690,4937,2037,727,788,1000  
1771,21843,3623,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,67500,6200,6800,1800,4500,89734  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3032,7082,242278  
1840,312110,18013,14000,5346,10965,391114  
1850,355400,218000,185000,5346,10965,50115  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,599400,5653,51980,33091,1911801  
1890,1364473,629000,5653,51980,33091,2101534  
1900,1850993,116582,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,2210110,2018354,44600,44600,73201,11651,501088  
1930,2671110,2210110,44600,44600,73201,11651,501088  
1940,1889924,2698285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550949,1451277,191555,7991957  
1960,1690101,2738175,1550949,1451277,191555,7991957  
1970,1690101,2738175,1550949,1451277,191555,7991984  
1980,1426285,2230936,1891325,1472701,135443,7691846  
1990,1487536,2300664,1951598,1320789,138977,7322564  
2000,1537195,2485326,2223379,1332450,143798,8080879  
2010,1583873,2504705,2272722,1385108,1457513,8175133  
2015,1444518,2640733,2339150,1459444,174558,8059405
```

nycHistPop.csv

In Lab 6

# Example: Reading in CSV Files

```
import matplotlib.pyplot as plt
import pandas as pd

pop = pd.read_csv('nycHistPop.csv', skiprows=5)

pop.plot(x="Year")
plt.show()

Year,Population
1698,1037,2017,727,7181
1771,21843,3623,2847,28423
1790,33131,4549,6159,1781,3827,49447
1800,60515,5740,6442,1755,4543,75955
1810,69000,5340,6240,1725,4343,79334
1820,123704,11487,8246,2792,6135,152056
1830,202589,20535,9049,3032,7082,242278
1840,312110,19113,14049,5348,10965,391114
1850,355400,18800,13850,5148,10000,401115
1860,813449,279122,32903,23593,25492,217477
1870,942292,419921,45468,37393,33029,1479103
1880,1164473,59940,5653,51980,33091,1911801
1890,1380000,60000,56000,51980,33091,1911804
1900,1850093,116582,152999,200567,67621,2437202
1910,2233142,1634351,2841,430980,8569,476683
1920,2210110,2018354,4460,430981,72021,11651,50048
1930,2671110,2018354,4460,430981,72021,11651,50048
1940,1889924,2698285,1297634,1394711,174441,7454995
1950,1960101,2738175,1550949,1451277,191555,7991957
1960,1696010,2738175,1550949,1451277,191555,7991957
1970,1696010,2738175,1550949,1451277,191555,7991984
1970,1696010,2738175,1550949,1451277,191555,7991984
1980,1426285,2230936,1891325,1168972,352121,7071639
1990,1487536,2300664,1951598,1203789,378977,7322564
2000,1537195,2485326,2229379,1332450,419728,8080879
2010,1583873,2504705,2237722,1385108,474558,8559405
2015,1444518,2546733,2339150,1459444,474558,8559405
```

nycHistPop.csv

In Lab 6

# Example: Reading in CSV Files

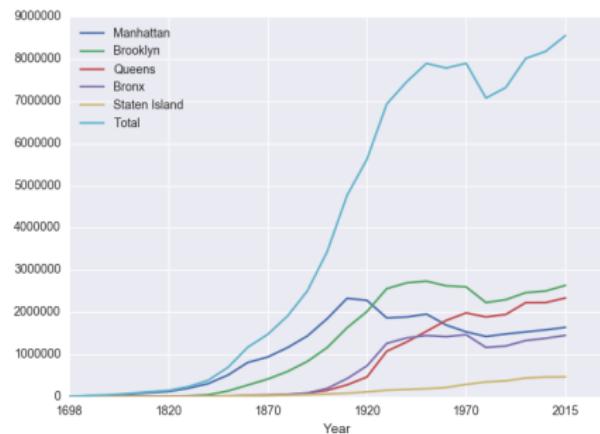
```
import matplotlib.pyplot as plt  
import pandas as pd
```

```
pop = pd.read_csv('nycHistPop.csv', skiprows=5)
```

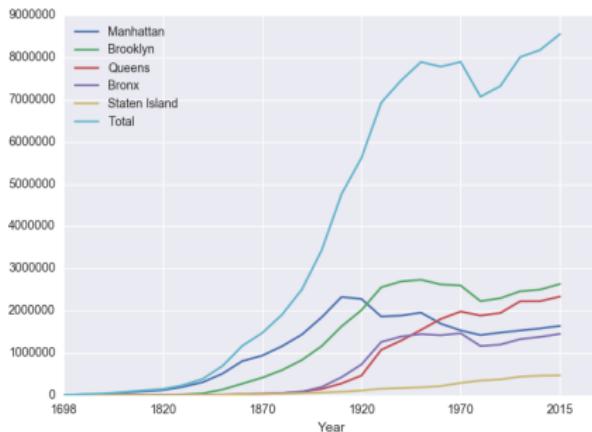
```
Source: https://en.wikipedia.org/wiki/Demographics_of_New_York_City.....  
All population figures are consistent with present-day boundaries.....  
First census after the consolidation of the five boroughs.....  
.....  
Year,Population  
1699,4937,2017,...,727,7181  
1771,21843,36231,...,2847,28423  
1790,33131,4549,6159,1781,3827,49447  
1800,60515,5740,6442,1755,4543,75955  
1810,71031,6354,7041,1811,5147,89734  
1820,123704,11187,8246,2792,6135,152056  
1830,202589,20535,9049,3023,7082,242278  
1840,312110,20113,14041,5348,10965,391114  
1850,455491,21890,18851,6851,13001,51154  
1860,813469,279122,32903,23593,25492,174777  
1870,942292,419921,45468,37393,33029,1479103  
1880,1164473,59943,5653,51980,33091,1911801  
1890,1400000,720000,68000,55000,40000,200000  
1900,1850093,1165852,152999,200567,67621,2437202  
1910,2233142,1634351,2841,430980,8569,476683  
1920,2210110,2018354,44601,73201,11651,50048  
1930,2671110,2489128,35254,5583,5000,4930446  
1940,1889924,2690285,1297634,1394711,174441,7454995  
1950,1960101,2738175,1550849,1451277,191555,7991957  
1960,1690101,2738175,1550849,1451277,191555,7981984  
1970,1639231,2465701,1472701,1235443,7984642  
1980,1426285,2230936,11891325,1168972,352121,7071639  
1990,1487536,2300664,1951598,1203789,378977,7322564  
2000,1537195,2485326,2229379,1332650,413728,8080879  
2010,1583873,2504705,2216722,1385108,413728,8175133  
2015,1444518,2636733,2339150,1459446,474558,8059405
```

nycHistPop.csv

In Lab 6

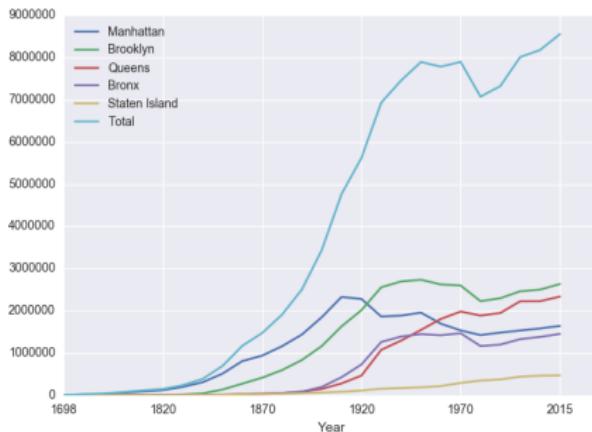


# Series in Pandas



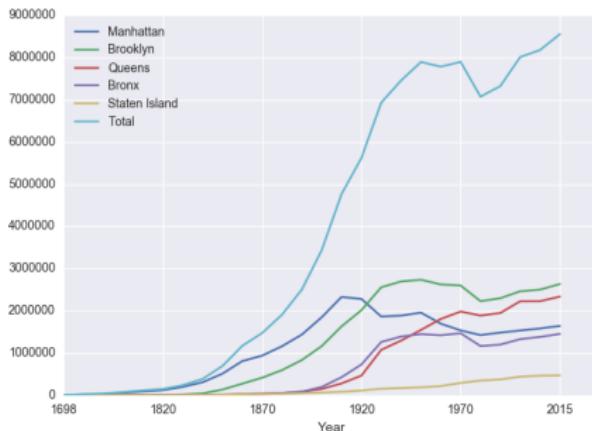
- Series can store a column or row of a DataFrame.

# Series in Pandas



- Series can store a column or row of a DataFrame.
- Example: `pop["Manhattan"]` is the Series corresponding to the column of Manhattan data.

# Series in Pandas



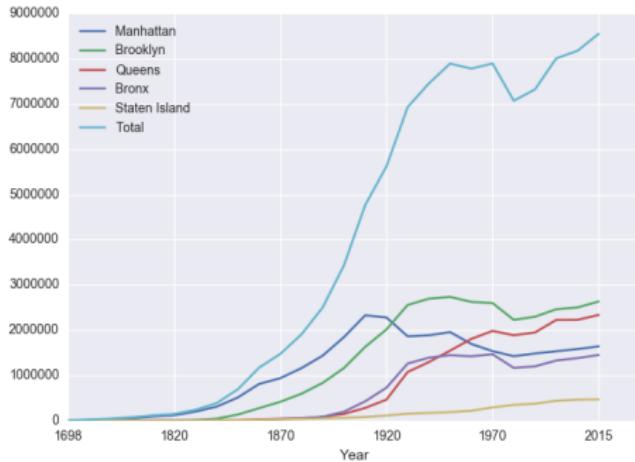
- Series can store a column or row of a DataFrame.
- Example: `pop["Manhattan"]` is the Series corresponding to the column of Manhattan data.
- Example:  

```
print("The largest number living in the Bronx is",
      pop["Bronx"].max())
```

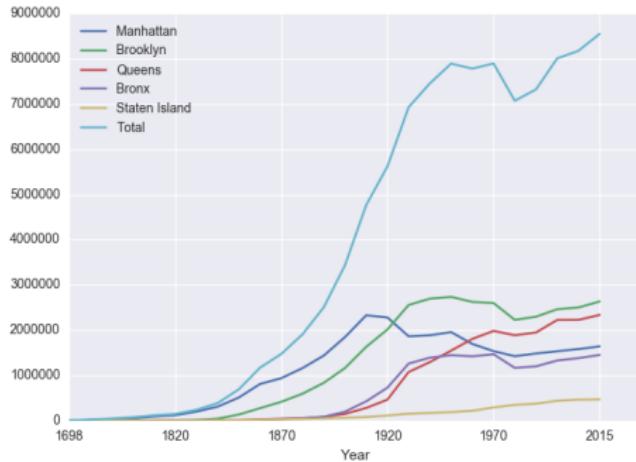
# Challenge:

Predict what the following will do:

- `print("Queens:", pop["Queens"].min())`



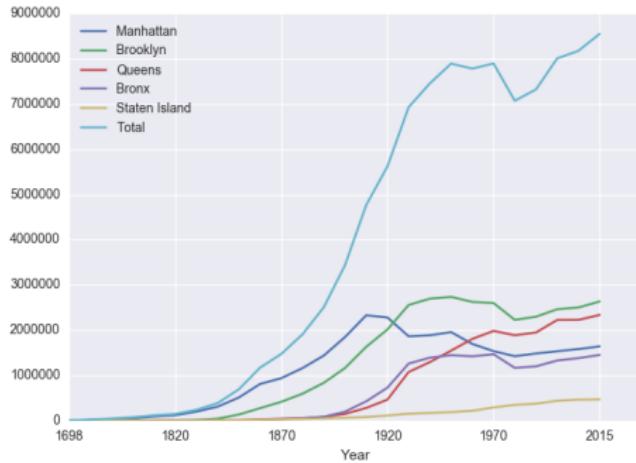
# Challenge:



Predict what the following will do:

- `print("Queens:", pop["Queens"].min())`
- `print("S I:", pop["Staten Island"].mean())`

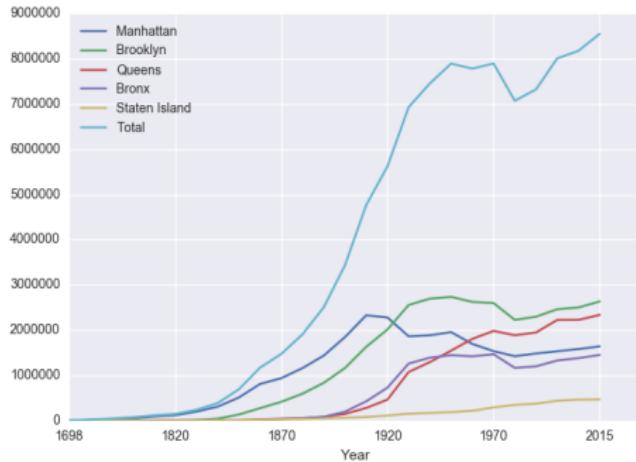
# Challenge:



Predict what the following will do:

- `print("Queens:", pop["Queens"].min())`
- `print("S I:", pop["Staten Island"].mean())`
- `print("S I:", pop["Staten Island"].std())`

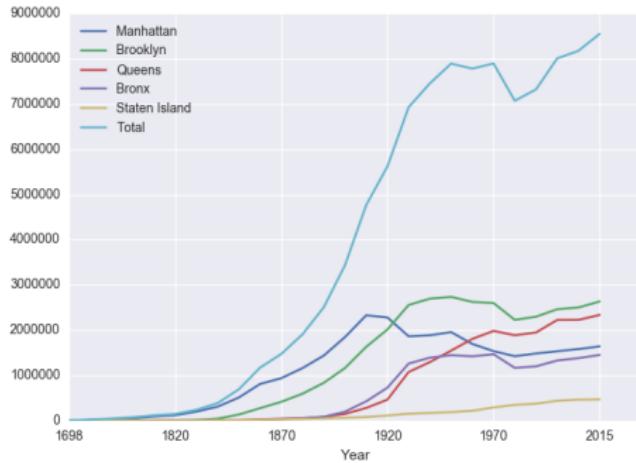
# Challenge:



Predict what the following will do:

- `print("Queens:", pop["Queens"].min())`
- `print("S I:", pop["Staten Island"].mean())`
- `print("S I:", pop["Staten Island"].std())`
- `pop.plot.bar(x="Year")`

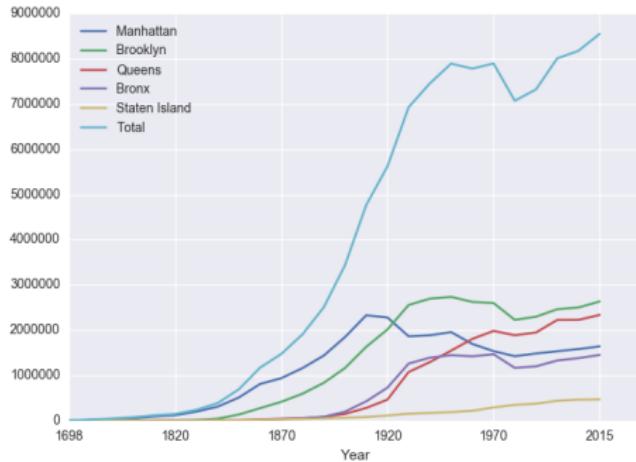
# Challenge:



Predict what the following will do:

- `print("Queens:", pop["Queens"].min())`
- `print("S I:", pop["Staten Island"].mean())`
- `print("S I:", pop["Staten Island"].std())`
- `pop.plot.bar(x="Year")`
- `pop.plot.scatter(x="Brooklyn", y= "Total")`

# Challenge:



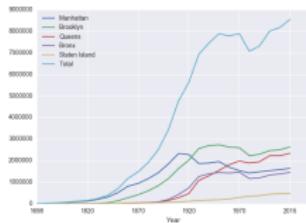
Predict what the following will do:

- `print("Queens:", pop["Queens"].min())`
- `print("S I:", pop["Staten Island"].mean())`
- `print("S I:", pop["Staten Island"].std())`
- `pop.plot.bar(x="Year")`
- `pop.plot.scatter(x="Brooklyn", y= "Total")`
- `pop["Fraction"] = pop["Bronx"]/pop["Total"]`

# Solutions

Predict what the following will do:

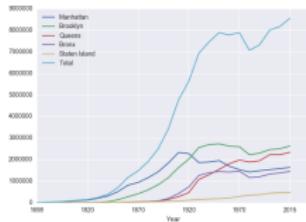
- `print("Queens:", pop["Queens"].min())`



# Solutions

Predict what the following will do:

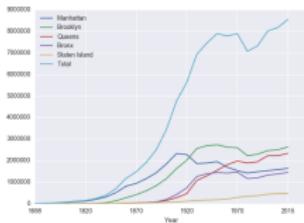
- `print("Queens:", pop["Queens"].min())`  
*Minimum value in the column with label "Queens".*



# Solutions

Predict what the following will do:

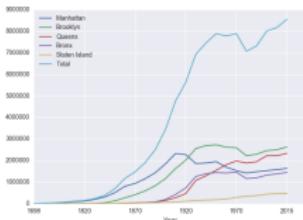
- `print("Queens:", pop["Queens"].min())`  
*Minimum value in the column with label "Queens".*
- `print("S I:", pop["Staten Island"].mean())`



# Solutions

Predict what the following will do:

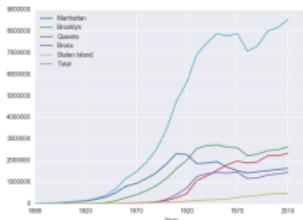
- `print("Queens:", pop["Queens"].min())`  
*Minimum value in the column with label "Queens".*
- `print("S I:", pop["Staten Island"].mean())`  
*Average of values in the column "Staten Island".*



# Solutions

Predict what the following will do:

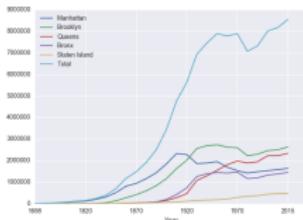
- `print("Queens:", pop["Queens"].min())`  
*Minimum value in the column with label "Queens".*
- `print("S I:", pop["Staten Island"].mean())`  
*Average of values in the column "Staten Island".*
- `print("S I :", pop["Staten Island"].std())`



# Solutions

Predict what the following will do:

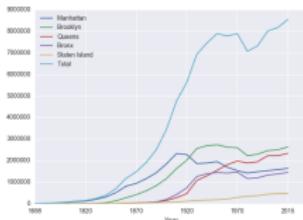
- `print("Queens:", pop["Queens"].min())`  
*Minimum value in the column with label "Queens".*
- `print("S I:", pop["Staten Island"].mean())`  
*Average of values in the column "Staten Island".*
- `print("S I :", pop["Staten Island"].std())`  
*Standard deviation of values in the column "Staten Island".*



# Solutions

Predict what the following will do:

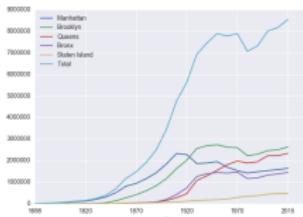
- `print("Queens:", pop["Queens"].min())`  
*Minimum value in the column with label "Queens".*
- `print("S I:", pop["Staten Island"].mean())`  
*Average of values in the column "Staten Island".*
- `print("S I :", pop["Staten Island"].std())`  
*Standard deviation of values in the column "Staten Island".*
- `pop.plot.bar(x="Year")`



# Solutions

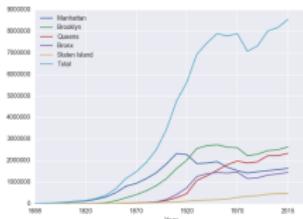
Predict what the following will do:

- `print("Queens:", pop["Queens"].min())`  
*Minimum value in the column with label "Queens".*
- `print("S I:", pop["Staten Island"].mean())`  
*Average of values in the column "Staten Island".*
- `print("S I :", pop["Staten Island"].std())`  
*Standard deviation of values in the column "Staten Island".*
- `pop.plot.bar(x="Year")`  
*Bar chart with x-axis "Year".*



# Solutions

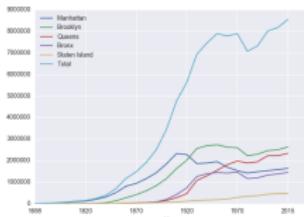
Predict what the following will do:



- `print("Queens:", pop["Queens"].min())`  
*Minimum value in the column with label "Queens".*
- `print("S I:", pop["Staten Island"].mean())`  
*Average of values in the column "Staten Island".*
- `print("S I :", pop["Staten Island"].std())`  
*Standard deviation of values in the column "Staten Island".*
- `pop.plot.bar(x="Year")`  
*Bar chart with x-axis "Year".*
- `pop.plot.scatter(x="Brooklyn", y= "Total")`

# Solutions

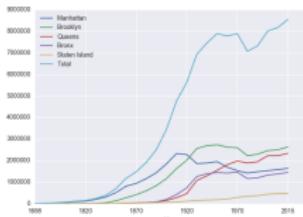
Predict what the following will do:



- `print("Queens:", pop["Queens"].min())`  
*Minimum value in the column with label "Queens".*
- `print("S I:", pop["Staten Island"].mean())`  
*Average of values in the column "Staten Island".*
- `print("S I :", pop["Staten Island"].std())`  
*Standard deviation of values in the column "Staten Island".*
- `pop.plot.bar(x="Year")`  
*Bar chart with x-axis "Year".*
- `pop.plot.scatter(x="Brooklyn", y= "Total")`  
*Scatter plot of Brooklyn versus Total values.*

# Solutions

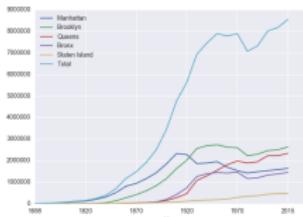
Predict what the following will do:



- `print("Queens:", pop["Queens"].min())`  
*Minimum value in the column with label "Queens".*
- `print("S I:", pop["Staten Island"].mean())`  
*Average of values in the column "Staten Island".*
- `print("S I :", pop["Staten Island"].std())`  
*Standard deviation of values in the column "Staten Island".*
- `pop.plot.bar(x="Year")`  
*Bar chart with x-axis "Year".*
- `pop.plot.scatter(x="Brooklyn", y= "Total")`  
*Scatter plot of Brooklyn versus Total values.*
- `pop["Fraction"] = pop["Bronx"]/pop["Total"]`

# Solutions

Predict what the following will do:



- `print("Queens:", pop["Queens"].min())`  
*Minimum value in the column with label "Queens".*
- `print("S I:", pop["Staten Island"].mean())`  
*Average of values in the column "Staten Island".*
- `print("S I :", pop["Staten Island"].std())`  
*Standard deviation of values in the column "Staten Island".*
- `pop.plot.bar(x="Year")`  
*Bar chart with x-axis "Year".*
- `pop.plot.scatter(x="Brooklyn", y= "Total")`  
*Scatter plot of Brooklyn versus Total values.*
- `pop["Fraction"] = pop["Bronx"]/pop["Total"]`  
*New column with the fraction of population that lives in the Bronx.*

# Challenge:

Write a complete Python program that reads in the file, `cunyF2016.csv`, and produces a scatter plot of full-time versus part-time enrollment.

College	Undergraduate		
	Full-time	Part-time	Total
Banach	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,087	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,833	16,526
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

`cunyF2016.csv`

# Challenge:

Write a complete Python program that reads in the file, `cunyF2016.csv`, and produces a scatter plot of full-time versus part-time enrollment.

College	Undergraduate		
	Full-time	Part-time	Total
Banach	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,087	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,833	16,526
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

`cunyF2016.csv`

# Challenge:

Write a complete Python program that reads in the file, `cunyF2016.csv`, and produces a scatter plot of full-time versus part-time enrollment.

College	Undergraduate		
	Full-time	Part-time	Total
Banach	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,833	16,526
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

`cunyF2016.csv`

*Solution:*

- 1 *Include pandas & pyplot libraries.*

# Challenge:

Write a complete Python program that reads in the file, cunyF2016.csv, and produces a scatter plot of full-time versus part-time enrollment.

College	Undergraduate		
	Full-time	Part-time	Total
Banach	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,833	16,526
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

cunyF2016.csv

*Solution:*

① *Include pandas & pyplot libraries.*

② *Read in the CSV file.*

# Challenge:

Write a complete Python program that reads in the file, `cunyF2016.csv`, and produces a scatter plot of full-time versus part-time enrollment.

College	Undergraduate		
	Full-time	Part-time	Total
Banach	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,833	16,526
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

`cunyF2016.csv`

*Solution:*

- ① *Include pandas & pyplot libraries.*
- ② *Read in the CSV file.*
- ③ *Set up a scatter plot.*

# Challenge:

College	Undergraduate		
	Full-time	Part-time	Total
Banach	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Medgar Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,693	4,833	16,526
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

cunyF2016.csv

Write a complete Python program that reads in the file, cunyF2016.csv, and produces a scatter plot of full-time versus part-time enrollment.

*Solution:*

- ① *Include pandas & pyplot libraries.*
- ② *Read in the CSV file.*
- ③ *Set up a scatter plot.*
- ④ *Display plot.*

# Challenge:

Write a complete Python program that reads in the file, `cunyF2016.csv`, and produces a scatter plot of full-time versus part-time enrollment.

College	Undergraduates		
	Full-time	Part-time	Total
Bauch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Melvin Evers	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,884	4,833	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

`cunyF2016.csv`

# Challenge:

Write a complete Python program that reads in the file, `cunyF2016.csv`, and produces a scatter plot of full-time versus part-time enrollment.

College	Undergraduates		
	Full-time	Part-time	Total
Bauch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Melgar Evans	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,884	4,833	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

`cunyF2016.csv`

*Solution:*

- ① *Include pandas & pyplot libraries.*

# Challenge:

Write a complete Python program that reads in the file, `cunyF2016.csv`, and produces a scatter plot of full-time versus part-time enrollment.

College	Undergraduates		
	Full-time	Part-time	Total
Baumgardt	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Melvin Evans	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,884	4,833	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

`cunyF2016.csv`

*Solution:*

- ① *Include pandas & pyplot libraries.*

```
import matplotlib.pyplot as plt  
import pandas as pd
```

# Challenge:

Write a complete Python program that reads in the file, `cunyF2016.csv`, and produces a scatter plot of full-time versus part-time enrollment.

College	Undergraduates		
	Full-time	Part-time	Total
Bauch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Melvin Evans	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,884	4,833	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

`cunyF2016.csv`

*Solution:*

- ① *Include pandas & pyplot libraries.*

```
import matplotlib.pyplot as plt  
import pandas as pd
```

- ② *Read in the CSV file.*

# Challenge:

Write a complete Python program that reads in the file, `cunyF2016.csv`, and produces a scatter plot of full-time versus part-time enrollment.

College	Undergraduates		
	Full-time	Part-time	Total
Baumgardt	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Melvin Evans	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,884	4,833	16,326
Satellite Island	9,584	2,948	12,532
York	5,066	3,192	8,258

`cunyF2016.csv`

*Solution:*

- ① *Include pandas & pyplot libraries.*  
`import matplotlib.pyplot as plt`  
`import pandas as pd`
- ② *Read in the CSV file.*  
`pop=pd.read_csv('cunyF2016.csv',skiprows=1)`

# Challenge:

Write a complete Python program that reads in the file, `cunyF2016.csv`, and produces a scatter plot of full-time versus part-time enrollment.

College	Undergraduates		
	Full-time	Part-time	Total
Bauch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Melvin Evans	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,884	4,833	16,326
Satellite Island	9,584	2,948	12,532
York	5,066	3,192	8,258

`cunyF2016.csv`

*Solution:*

- ① *Include pandas & pyplot libraries.*

```
import matplotlib.pyplot as plt  
import pandas as pd
```

- ② *Read in the CSV file.*

```
pop=pd.read_csv('cunyF2016.csv',skiprows=1)
```

- ③ *Set up a scatter plot.*

# Challenge:

Write a complete Python program that reads in the file, `cunyF2016.csv`, and produces a scatter plot of full-time versus part-time enrollment.

College	Undergraduates		
	Full-time	Part-time	Total
Baumgardt	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Melvin Evans	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,884	4,833	16,326
Satellite Island	9,584	2,948	12,532
York	5,066	3,192	8,258

`cunyF2016.csv`

*Solution:*

- ① *Include pandas & pyplot libraries.*

```
import matplotlib.pyplot as plt  
import pandas as pd
```

- ② *Read in the CSV file.*

```
pop=pd.read_csv('cunyF2016.csv',skiprows=1)
```

- ③ *Set up a scatter plot.*

```
pop.plot.scatter(x="Full-time",y="Part-time")
```

# Challenge:

Write a complete Python program that reads in the file, `cunyF2016.csv`, and produces a scatter plot of full-time versus part-time enrollment.

College	Undergraduate		
	Full-time	Part-time	Total
Bauch	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Melvin Evans	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,884	4,833	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

`cunyF2016.csv`

*Solution:*

- ① *Include pandas & pyplot libraries.*  
`import matplotlib.pyplot as plt`  
`import pandas as pd`
- ② *Read in the CSV file.*  
`pop=pd.read_csv('cunyF2016.csv',skiprows=1)`
- ③ *Set up a scatter plot.*  
`pop.plot.scatter(x="Full-time",y="Part-time")`
- ④ *Display plot.*

# Challenge:

Write a complete Python program that reads in the file, `cunyF2016.csv`, and produces a scatter plot of full-time versus part-time enrollment.

College	Undergraduates		
	Full-time	Part-time	Total
Baumgardt	11,288	3,922	15,210
Brooklyn	10,198	4,208	14,406
City	10,067	3,250	13,317
Hunter	12,223	4,500	16,723
John Jay	9,831	2,843	12,674
Lehman	6,600	4,720	11,320
Melvin Evans	4,760	2,059	6,819
NYCCT	10,912	6,370	17,282
Queens	11,884	4,833	16,326
Staten Island	9,584	2,948	12,532
York	5,066	3,192	8,258

`cunyF2016.csv`

*Solution:*

- ① *Include pandas & pyplot libraries.*

```
import matplotlib.pyplot as plt  
import pandas as pd
```

- ② *Read in the CSV file.*

```
pop=pd.read_csv('cunyF2016.csv',skiprows=1)
```

- ③ *Set up a scatter plot.*

```
pop.plot.scatter(x="Full-time",y="Part-time")
```

- ④ *Display plot.*

```
plt.show()
```

# groupby()

Sometimes you have **recurring values in a column** and you want to examine the data for a particular value.

Rain in Australia				
Date	Location	MinTemp	MaxTemp	Rainfall
12/1/08	Albury	13.4	22.9	0.6
5/22/15	BadgerysCree	11	15.6	1.6
3/17/11	BadgerysCree	18.1	25.8	16.6
7/27/10	Cobar	5.3	17.2	0
9/5/10	Moree	12.1	19.8	23.4
1/23/12	CoffsHarbour	20	24.4	28
7/15/11	Moree	2.8	19	0
1/28/10	Newcastle	22.2	28	0
12/2/15	Moree	20.1	32	4.8
***				

AustraliaRain.csv

# groupby()

Sometimes you have **recurring values in a column** and you want to examine the data for a particular value.

*For example, to find the average rainfall at each location:*

Rain in Australia				
Date	Location	MinTemp	MaxTemp	Rainfall
12/1/08	Albury	13.4	22.9	0.6
5/22/15	BadgerysCree	11	15.6	1.6
3/17/11	BadgerysCree	18.1	25.8	16.6
7/27/10	Cobar	5.3	17.2	0
9/5/10	Moree	12.1	19.8	23.4
1/23/12	CoffsHarbour	20	24.4	28
7/15/11	Moree	2.8	19	0
1/28/10	Newcastle	22.2	28	0
12/2/15	Moree	20.1	32	4.8
***				

AustraliaRain.csv

# groupby()

Sometimes you have **recurring values in a column** and you want to examine the data for a particular value.

*For example, to find the average rainfall at each location:*

Rain in Australia				
Date	Location	MinTemp	MaxTemp	Rainfall
12/1/08	Albury	13.4	22.9	0.6
5/22/15	BadgerysCree	11	15.6	1.6
3/17/11	BadgerysCree	18.1	25.8	16.6
7/27/10	Cobar	5.3	17.2	0
9/5/10	Moree	12.1	19.8	23.4
1/23/12	CoffsHarbour	20	24.4	28
7/15/11	Moree	2.8	19	0
1/28/10	Newcastle	22.2	28	0
12/2/15	Moree	20.1	32	4.8
***				

AustraliaRain.csv

## ① Import libraries.

```
import pandas as pd
```

# groupby()

Sometimes you have **recurring values in a column** and you want to examine the data for a particular value.

*For example, to find the average rainfall at each location:*

Rain in Australia				
Date	Location	MinTemp	MaxTemp	Rainfall
12/1/08	Albury	13.4	22.9	0.6
5/22/15	BadgerysCree	11	15.6	1.6
3/17/11	BadgerysCree	18.1	25.8	16.6
7/27/10	Cobar	5.3	17.2	0
9/5/10	Moree	12.1	19.8	23.4
1/23/12	CoffsHarbour	20	24.4	28
7/15/11	Moree	2.8	19	0
1/28/10	Newcastle	22.2	28	0
12/2/15	Moree	20.1	32	4.8
***				

AustraliaRain.csv

- ① Import libraries.

```
import pandas as pd
```

- ② Read in the CSV file.

```
rain =  
pd.read_csv('AustraliaRain.csv', skiprows=1)
```

# groupby()

Sometimes you have **recurring values in a column** and you want to examine the data for a particular value.

*For example, to find the average rainfall at each location:*

Rain in Australia				
Date	Location	MinTemp	MaxTemp	Rainfall
12/1/08	Albury	13.4	22.9	0.6
5/22/15	BadgerysCree	11	15.6	1.6
3/17/11	BadgerysCree	18.1	25.8	16.6
7/27/10	Cobar	5.3	17.2	0
9/5/10	Moree	12.1	19.8	23.4
1/23/12	CoffsHarbour	20	24.4	28
7/15/11	Moree	2.8	19	0
1/28/10	Newcastle	22.2	28	0
12/2/15	Moree	20.1	32	4.8
***				

AustraliaRain.csv

- ① Import libraries.  
`import pandas as pd`
- ② Read in the CSV file.  
`rain = pd.read_csv('AustraliaRain.csv', skiprows=1)`
- ③ Group the data by location.  
`groupAvg = rain.groupby('Location')`

# groupby()

Sometimes you have **recurring values in a column** and you want to examine the data for a particular value.

*For example, to find the average rainfall at each location:*

Rain in Australia				
Date	Location	MinTemp	MaxTemp	Rainfall
12/1/08	Albury	13.4	22.9	0.6
5/22/15	BadgerysCree	11	15.6	1.6
3/17/11	BadgerysCree	18.1	25.8	16.6
7/27/10	Cobar	5.3	17.2	0
9/5/10	Moree	12.1	19.8	23.4
1/23/12	CoffsHarbour	20	24.4	28
7/15/11	Moree	2.8	19	0
1/28/10	Newcastle	22.2	28	0
12/2/15	Moree	20.1	32	4.8
***				

AustraliaRain.csv

- ① Import libraries.  
`import pandas as pd`
- ② Read in the CSV file.  
`rain = pd.read_csv('AustraliaRain.csv', skiprows=1)`
- ③ Group the data by location.  
`groupAvg = rain.groupby('Location')`
- ④ Print the average rainfall at each location.  
`print(groupAvg['Rainfall'].mean())`

# groupby()

Rain in Australia				
Date	Location	MinTemp	MaxTemp	Rainfall
12/1/08	Albury	13.4	22.9	0.6
5/22/15	BadgersCree	11	15.6	1.6
3/17/11	BadgersCree	18.1	25.8	16.6
7/27/10	Cobar	5.3	17.2	0
9/5/10	Moree	12.1	19.8	23.4
1/23/12	CoffsHarbour	20	24.4	28
7/15/11	Moree	2.8	19	0
1/28/10	Newcastle	22.2	28	0
12/2/15	Moree	20.1	32	4.8
***				

AustraliaRain.csv

Adelaide	1.572185
Albany	2.255073
Albury	1.925710
AliceSprings	0.869355
BadgersCreek	2.207925
Ballarat	1.688830
Bendigo	1.621452
Brisbane	3.160536
Cairns	5.765317
Canberra	1.735038
Cobar	1.129262
CoffsHarbour	5.054592
Darntmoor	2.148554

Sometimes you have **recurring values in a column** and you want to examine the data for a particular value.

For example, to find the average rainfall at each location:

- 1 Import libraries.

```
import pandas as pd
```

- 2 Read in the CSV file.

```
rain =  
pd.read_csv('AustraliaRain.csv', skiprows=1)
```

- 3 Group the data by location.

```
groupAvg = rain.groupby('Location')
```

- 4 Print the average rainfall at each location.

```
print(groupAvg['Rainfall'].mean())
```

# groupby()

Sometimes you have **recurring values in a column** and you want to examine the data for a particular value.

*For example, to find the average rainfall at one location, e.g. Albury:*

Rain in Australia				
Date	Location	MinTemp	MaxTemp	Rainfall
1/2/19 Albury		13.4	22.9	0.6
5/2/11 Albury/Creek		11	18.6	1.6
3/1/11 Albury/Creek		18	23.8	14.6
7/2/10 Colac		5.3	17.2	0
9/5/10 Moree		12.1	19.8	23.4
12/5/11 Port Lincoln		20	24	20
7/1/11 Moree		2.8	19	0
1/2/11 Newcastle		22.2	28	0
1/2/19 Moree		20.1	32	4.8
***				

AustraliaRain.csv

# groupby()

Sometimes you have **recurring values in a column** and you want to examine the data for a particular value.

*For example, to find the average rainfall at one location, e.g. Albury:*

Date	Location	MinTemp	MaxTemp	Rainfall
12/1/98	Albury	13.4	22.9	0.6
5/2/11	BalgownieCreek	11	18.6	1.6
3/1/10	BalgownieCreek	18	23.8	14.6
7/2/10	Cobar	5.3	17.2	0
9/5/10	Moresby	12.1	19.8	23.4
12/2/11	NewcastleHarbour	20	24	20
7/1/11	Moresby	2.8	19	0
1/2/11	Newcastle	22.2	28	0
1/2/11	Moresby	20.1	32	4.8
...				

AustraliaRain.csv

- ① Import libraries.

```
import pandas as pd
```

- ② Read in the CSV file.

```
rain =  
pd.read_csv('AustraliaRain.csv', skiprows=1)
```

- ③ Group the data by location get data for group Albury.

```
AlburyAvg =  
rain.groupby('Location').get_group('Albury')
```

# groupby()

Sometimes you have **recurring values in a column** and you want to examine the data for a particular value.

*For example, to find the average rainfall at one location, e.g. Albury:*

Date	Location	MinTemp	MaxTemp	Rainfall
1/2/95	Albury	13.4	22.9	0.6
5/2/11	BalgownieCreek	11	18.6	1.6
3/1/10	BalgownieCreek	18	23.8	0.6
7/2/10	Cobar	5.3	17.2	0
9/5/10	Moresby	12.1	19.8	23.4
12/2/11	NewcastleHarbour	20	24	20
7/1/11	Moresby	2.8	19	0
1/2/91	Newcastle	22.2	28	0
1/2/91	Moresby	20.1	32	4.8
...				

AustraliaRain.csv

- ① Import libraries.

```
import pandas as pd
```

- ② Read in the CSV file.

```
rain =  
pd.read_csv('AustraliaRain.csv', skiprows=1)
```

- ③ Group the data by location get data for group Albury.

```
AlburyAvg =  
rain.groupby('Location').get_group('Albury')
```

- ④ Print the average rainfall in Albury.

```
print(AlburyAvg['Rainfall'].mean())
```

# groupby()

Sometimes you have **recurring values in a column** and you want to examine the data for a particular value.

*For example, to find the average rainfall at one location, e.g. Albury:*

Rain in Australia				
Date	Location	MinTemp	MaxTemp	Rainfall
12/1/08	Albury	13.4	22.9	0.8
5/2/11	Bateman's Creek	11	16.6	1.6
5/2/11	Bogong High Plains	16.3	27.1	54.8
7/2/11	Cobar	5.3	17.2	0
9/5/10	Melbourne	12.1	19.8	23.4
10/1/11	Maitland	20	24.6	26
7/1/11	Morne	2.8	19	0
1/2/10	Newcastle	22.2	28	0
12/2/11	Morne	20.1	32	4.8
...				

AustraliaRain.csv

1.9257104647275156

- ① Import libraries.

```
import pandas as pd
```

- ② Read in the CSV file.

```
rain =  
pd.read_csv('AustraliaRain.csv', skiprows=1)
```

- ③ Group the data by location get data for group Albury.

```
AlburyAvg =  
rain.groupby('Location').get_group('Albury')
```

- ④ Print the average rainfall in Albury.

```
print(AlburyAvg['Rainfall'].mean())
```

# Design Challenge

Stars						
Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class
3068	0.0024	0.17	16.12	Brown Dwarf	Red	M
25000	0.056	0.0084	10.58	White Dwarf	Blue White	B
2650	0.00069	0.11	17.45	Brown Dwarf	Red	M
11790	0.00015	0.011	12.59	White Dwarf	Yellowish White	F
15276	1136	7.2	-1.97	Main Sequence	Blue-white	B
5800	0.81	0.9	5.05	Main Sequence	yellow-white	F
16500	0.013	0.014	11.89	White Dwarf	Blue White	B
3192	0.00362	0.1967	13.53	Red Dwarf	Red	M
6380	1.35	0.98	2.93	Main Sequence	yellow-white	F
3834	272000	1183	-9.2	Hypergiant	Red	M

- Design an algorithm that:
  - ▶ Prints the luminosity of the brightest star.
  - ▶ Prints the temperature of the coldest star.
  - ▶ Prints the average radius of a Hypergiant.

# Design Challenge - Solution

Stars						
Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class
3068	0.0024	0.17	16.12	Brown Dwarf	Red	M
25000	0.056	0.0084	10.58	White Dwarf	Blue White	B
2650	0.00069	0.11	17.45	Brown Dwarf	Red	M
11790	0.00015	0.011	12.59	White Dwarf	Yellowish White	F
15276	1136	7.2	-1.97	Main Sequence	Blue-white	B
5800	0.81	0.9	5.05	Main Sequence	yellow-white	F
16500	0.013	0.014	11.89	White Dwarf	Blue White	B
3192	0.00362	0.1967	13.53	Red Dwarf	Red	M
6380	1.35	0.98	2.93	Main Sequence	yellow-white	F
3834	272000	1183	-9.2	Hypergiant	Red	M

- **Libraries:** pandas

# Design Challenge - Solution

Stars						
Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class
3068	0.0024	0.17	16.12	Brown Dwarf	Red	M
25000	0.056	0.0084	10.58	White Dwarf	Blue White	B
2650	0.00069	0.11	17.45	Brown Dwarf	Red	M
11790	0.00015	0.011	12.59	White Dwarf	Yellowish White	F
15276	1136	7.2	-1.97	Main Sequence	Blue-white	B
5800	0.81	0.9	5.05	Main Sequence	yellow-white	F
16500	0.013	0.014	11.89	White Dwarf	Blue White	B
3192	0.00362	0.1967	13.53	Red Dwarf	Red	M
6380	1.35	0.98	2.93	Main Sequence	yellow-white	F
3834	272000	1183	-9.2	Hypergiant	Red	M

- **Libraries:** pandas
- **Process:**
  - ▶ Print **max** of '**Luminosity**' column

# Design Challenge - Solution

Stars						
Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class
3068	0.0024	0.17	16.12	Brown Dwarf	Red	M
25000	0.056	0.0084	10.58	White Dwarf	Blue White	B
2650	0.00069	0.11	17.45	Brown Dwarf	Red	M
11790	0.00015	0.011	12.59	White Dwarf	Yellowish White	F
15276	1136	7.2	-1.97	Main Sequence	Blue-white	B
5800	0.81	0.9	5.05	Main Sequence	yellow-white	F
16500	0.013	0.014	11.89	White Dwarf	Blue White	B
3192	0.00362	0.1967	13.53	Red Dwarf	Red	M
6380	1.35	0.98	2.93	Main Sequence	yellow-white	F
3834	272000	1183	-9.2	Hypergiant	Red	M

- **Libraries:** pandas
- **Process:**
  - ▶ Print **max** of '**Luminosity**' column
  - ▶ Print **min** of '**Temperature**' column

# Design Challenge - Solution

Stars						
Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class
3068	0.0024	0.17	16.12	Brown Dwarf	Red	M
25000	0.056	0.0084	10.58	White Dwarf	Blue White	B
2650	0.00069	0.11	17.45	Brown Dwarf	Red	M
11790	0.00015	0.011	12.59	White Dwarf	Yellowish White	F
15276	1136	7.2	-1.97	Main Sequence	Blue-white	B
5800	0.81	0.9	5.05	Main Sequence	yellow-white	F
16500	0.013	0.014	11.89	White Dwarf	Blue White	B
3192	0.00362	0.1967	13.53	Red Dwarf	Red	M
6380	1.35	0.98	2.93	Main Sequence	yellow-white	F
3834	272000	1183	-9.2	Hypergiant	Red	M

- **Libraries:** pandas

- **Process:**

- ▶ Print **max** of '**Luminosity**' column
- ▶ Print **min** of '**Temperature**' column
- ▶ **groupby** '**Star Type**' and take **averages**, then print **max** of '**Radius**' column

# Design Challenge - Solution

Stars						
Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class
3068	0.0024	0.17	16.12	Brown Dwarf	Red	M
25000	0.056	0.0084	10.58	White Dwarf	Blue White	B
2650	0.00069	0.11	17.45	Brown Dwarf	Red	M
11790	0.00015	0.011	12.59	White Dwarf	Yellowish White	F
15276	1136	7.2	-1.97	Main Sequence	Blue-white	B
5800	0.81	0.9	5.05	Main Sequence	yellow-white	F
16500	0.013	0.014	11.89	White Dwarf	Blue White	B
3192	0.00362	0.1967	13.53	Red Dwarf	Red	M
6380	1.35	0.98	2.93	Main Sequence	yellow-white	F
3834	272000	1183	-9.2	Hypergiant	Red	M

- **Libraries:** pandas

- **Process:**

- ▶ Print **max** of '**Luminosity**' column
- ▶ Print **min** of '**Temperature**' column
- ▶ **groupby** '**Star Type**' and take **averages**, then print **max of 'Radius'** column
- ▶ OR **groupby** '**Star Type**' and get group '**Hypergiant**' to print **average 'Radius'**

# Design Challenge - Code

- **Libraries:** pandas

```
import pandas as pd  
stars = pd.read_csv('Stars.csv')
```

# Design Challenge - Code

- **Libraries:** pandas

```
import pandas as pd  
stars = pd.read_csv('Stars.csv')
```

- **Process:**

- ▶ Print **max** of '**Luminosity**' column

```
print(stars['Luminosity(L/Lo)'].max())
```

# Design Challenge - Code

- **Libraries:** pandas

```
import pandas as pd  
stars = pd.read_csv('Stars.csv')
```

- **Process:**

- ▶ Print **max** of '**Luminosity**' column

```
print(stars['Luminosity(L/Lo)'].max())
```

- ▶ Prints **min** of '**Temperature**' column and store it in temp variable

```
print(stars['Temperature (K)'].min())
```

# Design Challenge - Code

- **Libraries:** pandas

```
import pandas as pd  
stars = pd.read_csv('Stars.csv')
```

- **Process:**

- ▶ Print **max** of '**Luminosity**' column

```
print(stars['Luminosity(L/Lo)'].max())
```

- ▶ Prints **min** of '**Temperature**' column and store it in temp variable

```
print(stars['Temperature (K)').min())
```

- ▶ **groupby** '**Star Type**' and take **averages**, then print **max** of '**Radius**' column

```
print(stars.groupby('Star type')\n    .mean()['Radius(R/Ro)'].max())
```

# Design Challenge - Code

- **Libraries:** pandas

```
import pandas as pd  
stars = pd.read_csv('Stars.csv')
```

- **Process:**

- ▶ Print **max** of '**Luminosity**' column

```
print(stars['Luminosity(L/Lo)'].max())
```

- ▶ Prints **min** of '**Temperature**' column and store it in temp variable

```
print(stars['Temperature (K)').min())
```

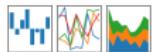
- ▶ OR **groupby** '**Star Type**' and **get group** '**Hypergiant**' to print average '**Radius**'

```
print(stars.groupby('Star type')\n      .get_group('Hypergiant').mean()['Radius(R/Ro)'])
```

# Recap

- Recap: Logical Expressions & Circuits

pandas

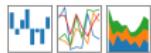


$y_t = \beta^T x_t + \mu_t + \epsilon_t$

# Recap

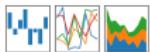
- Recap: Logical Expressions & Circuits
- Accessing Formatted Data:
  - ▶ Pandas library has elegant solutions for accessing & analyzing structured data.

pandas



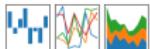
# Recap

pandas  
 $y_t = \beta^T x_t + \mu_t + \epsilon_t$



- Recap: Logical Expressions & Circuits
- Accessing Formatted Data:
  - ▶ Pandas library has elegant solutions for accessing & analyzing structured data.
  - ▶ Can manipulate individual columns or rows ('Series').

# Recap

pandas 

- Recap: Logical Expressions & Circuits
- Accessing Formatted Data:
  - ▶ Pandas library has elegant solutions for accessing & analyzing structured data.
  - ▶ Can manipulate individual columns or rows ('Series').
  - ▶ Has useful functions for the entire sheet ('DataFrame') such as plotting.