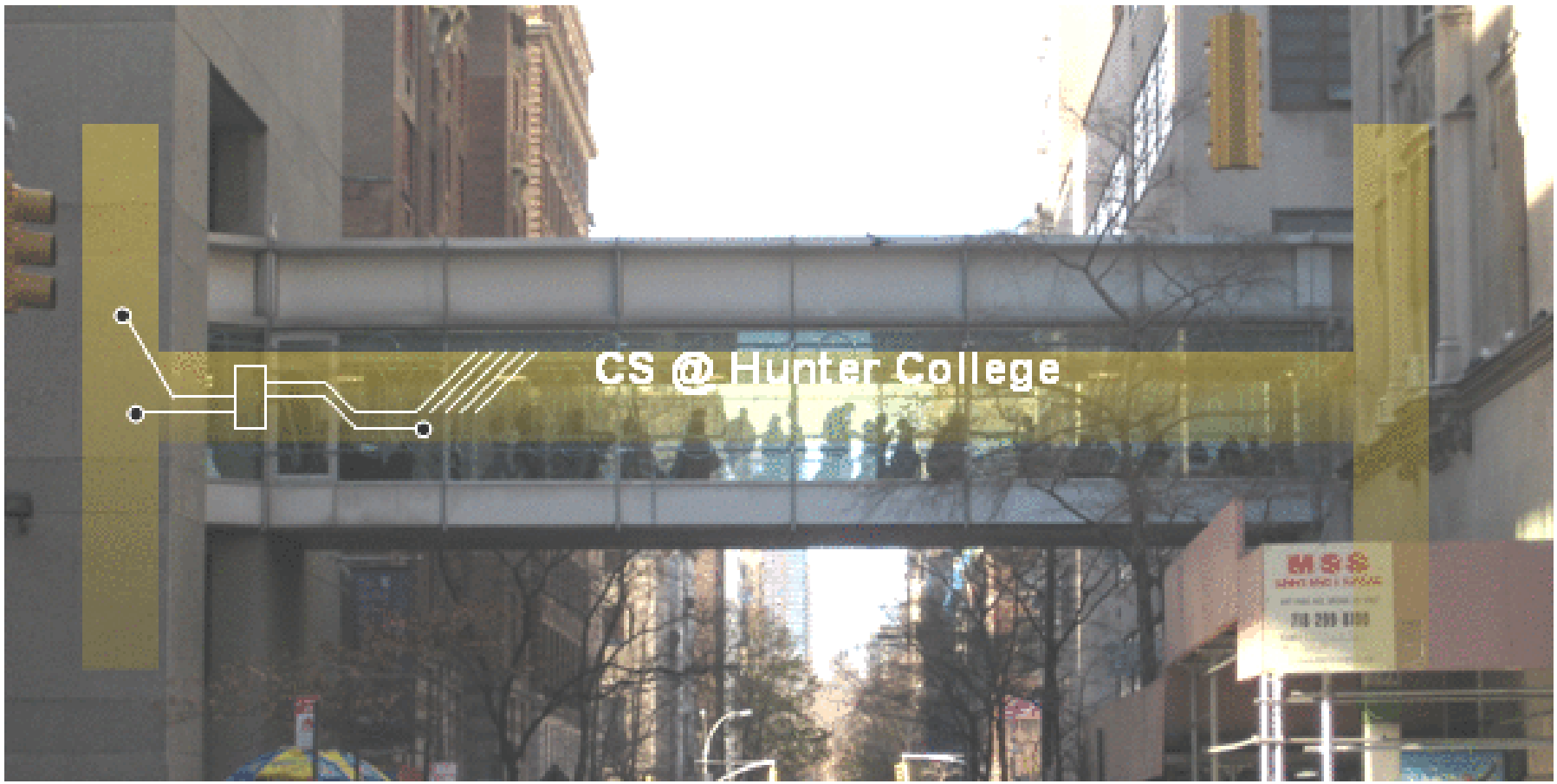
CSci 127: Introduction to Computer Science



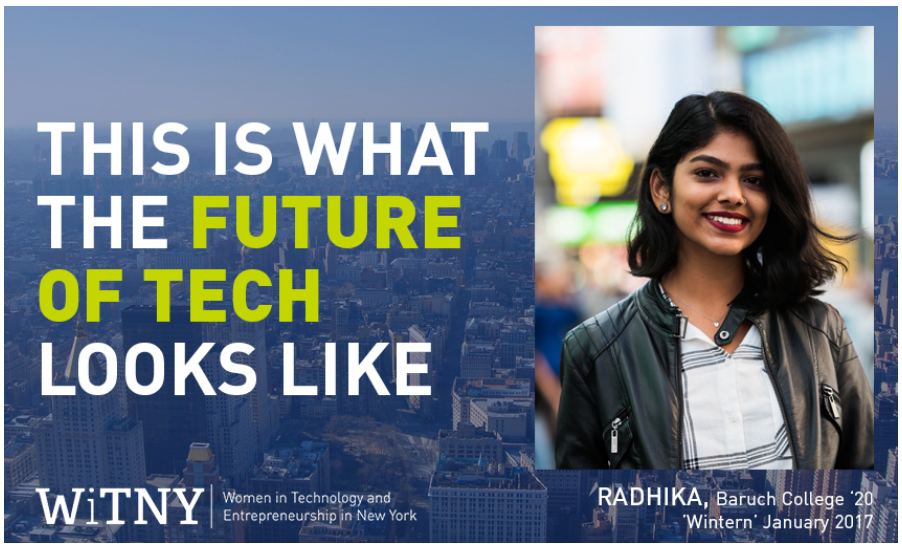
hunter.cuny.edu/csci

Announcements

- Special Guest: President Jennifer Raab!
- Each lecture includes a survey of computing research and tech in NYC.

  Today: Dr. Judy Spitz, Founding Director of Women in Technology & Entrepreneurship in New York (WiTNY)

CS Survey Talk



Dr. Judy Spitz
Founding Director
WiTNY

In Pairs or Triples:

Predict what the code will do:

```
motto = "Mihi Cura Futuri"
l = len(motto)
for i in range(l):
    print(motto[i])
for j in range(l-1,-1,-1):
    print(motto[j])
```

```
import matplotlib.pyplot as plt
import numpy as np
img = plt.imread('csBridge.png')
plt.imshow(img)
plt.show()
height = img.shape[0]
width = img.shape[1]
img2 = img[:height/2, :width/2]
plt.imshow(img2)
plt.show()
```

- And, design a program that asks maps time of day versus last month's 311 complaints.
  (Design only the pseudocode.)

Frequently Asked Questions

From lecture slips & recitation sections.

Frequently Asked Questions

From lecture slips & recitation sections.

- Pandas? Can't we go back to turtles? I like turtles better!

Frequently Asked Questions

From lecture slips & recitation sections.

- Pandas? Can't we go back to turtles? I like turtles better!

  Turtles will reappear, albeit briefly, in Labs 8 & 10.
  We will do more with Pandas since it's an incredibly useful & popular package for structured data. We hope you will soon like Pandas a much as turtles.

Frequently Asked Questions

From lecture slips & recitation sections.

- Pandas? Can't we go back to turtles? I like turtles better!

  Turtles will reappear, albeit briefly, in Labs 8 & 10.
  We will do more with Pandas since it's an incredibly useful & popular package for structured data. We hope you will soon like Pandas a much as turtles.

- Can you explain again when to use brackets and parenthesis?

Frequently Asked Questions

From lecture slips & recitation sections.

- Pandas? Can't we go back to turtles? I like turtles better!

  Turtles will reappear, albeit briefly, in Labs 8 & 10.
  We will do more with Pandas since it's an incredibly useful & popular package for structured data. We hope you will soon like Pandas a much as turtles.

- Can you explain again when to use brackets and parenthesis?

  Parenthesis are for functions: ex: `print("CUNY")` *or* `tess.left(45)`

Frequently Asked Questions

From lecture slips & recitation sections.

- Pandas? Can't we go back to turtles? I like turtles better!

  Turtles will reappear, albeit briefly, in Labs 8 & 10.
  We will do more with Pandas since it's an incredibly useful & popular package for structured data. We hope you will soon like Pandas a much as turtles.

- Can you explain again when to use brackets and parenthesis?

  Parenthesis are for functions: ex: `print("CUNY")` *or* `tess.left(45)`
  Brackets are used for access items in a list or string: ex: `message[3]`

Frequently Asked Questions

From lecture slips & recitation sections.

- Pandas? Can't we go back to turtles? I like turtles better!

  Turtles will reappear, albeit briefly, in Labs 8 & 10.
  We will do more with Pandas since it's an incredibly useful & popular package for structured data. We hope you will soon like Pandas a much as turtles.

- Can you explain again when to use brackets and parenthesis?

  Parenthesis are for functions: ex: `print("CUNY")` *or* `tess.left(45)`
  Brackets are used for access items in a list or string: ex: `message[3]`

- I'd like to do more. Any suggestions?

Frequently Asked Questions

From lecture slips & recitation sections.

- Pandas? Can't we go back to turtles? I like turtles better!

  Turtles will reappear, albeit briefly, in Labs 8 & 10.
  We will do more with Pandas since it's an incredibly useful & popular package for structured data. We hope you will soon like Pandas a much as turtles.

- Can you explain again when to use brackets and parenthesis?

  Parenthesis are for functions: ex: `print("CUNY")` *or* `tess.left(45)`
  Brackets are used for access items in a list or string: ex: `message[3]`

- I'd like to do more. Any suggestions?

  - *Hunter CS has an ACM Chapter & Women in CS Clubs.*

Frequently Asked Questions

From lecture slips & recitation sections.

- Pandas? Can't we go back to turtles? I like turtles better!

  Turtles will reappear, albeit briefly, in Labs 8 & 10.
  We will do more with Pandas since it's an incredibly useful & popular package for structured data. We hope you will soon like Pandas a much as turtles.

- Can you explain again when to use brackets and parenthesis?

  Parenthesis are for functions: ex: `print("CUNY")` *or* `tess.left(45)`
  Brackets are used for access items in a list or string: ex: `message[3]`

- I'd like to do more. Any suggestions?

  - *Hunter CS has an ACM Chapter & Women in CS Clubs.*
  - *Tech Meetups: both via CUNY (next: MongoDB, 11/2) and across city (focused on just about everything tech).*

Frequently Asked Questions

From lecture slips & recitation sections.

- Pandas? Can't we go back to turtles? I like turtles better!

  Turtles will reappear, albeit briefly, in Labs 8 & 10.
  We will do more with Pandas since it's an incredibly useful & popular package for structured data. We hope you will soon like Pandas a much as turtles.

- Can you explain again when to use brackets and parenthesis?

  Parenthesis are for functions: ex: `print("CUNY")` *or* `tess.left(45)`
  Brackets are used for access items in a list or string: ex: `message[3]`

- I'd like to do more. Any suggestions?

  - *Hunter CS has an ACM Chapter & Women in CS Clubs.*
  - *Tech Meetups: both via CUNY (next: MongoDB, 11/2) and across city (focused on just about everything tech).*
  - *Hackathons: upcoming student-focused: Brooklyn Navy Yard 11/10.*

Today's Topics

- CS Survey: Judy Spitz of WiTNY
- Recap: Pandas (Accessing formatted data) & Prep I
- Introduction to Functions
- Final Exam Overview

In Pairs or Triples:

Predict what the code will do:

```
motto = "Mihi Cura Futuri"
l = len(motto)
for i in range(l):
    print(motto[i])
for j in range(l-1,-1,-1):
    print(motto[j])
```

```
import matplotlib.pyplot as plt
import numpy as np
img = plt.imread('csBridge.png')
plt.imshow(img)
plt.show()
height = img.shape[0]
width = img.shape[1]
img2 = img[:height/2, :width/2]
plt.imshow(img2)
plt.show()
```

- And, design a program that asks maps time of day versus last month's 311 complaints.
  (Design only the pseudocode.)

Python Tutor

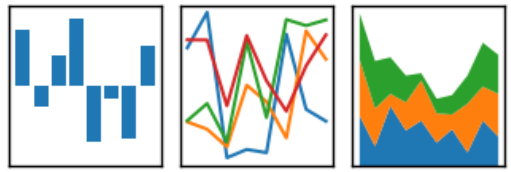
```
motto = "Mihi Cura Futuri"
l = len(motto)
for i in range(l):
    print(motto[i])
for j in range(l-1,-1,-1):
    print(motto[j])
```

(Demo with `pythonTutor`)

Structured Data

pandas

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

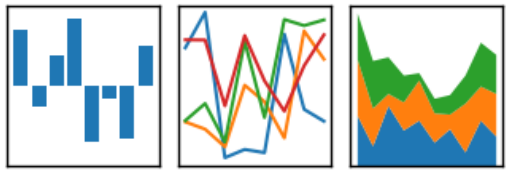


- Common to have data structured in a spread sheet.

Structured Data

pandas

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

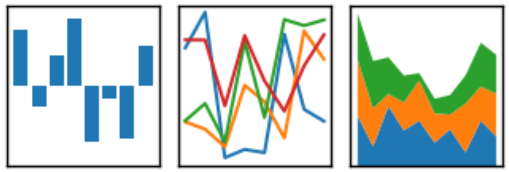


- Common to have data structured in a spread sheet.
- The text file version is called **CSV** for comma separated values.

Structured Data

pandas

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

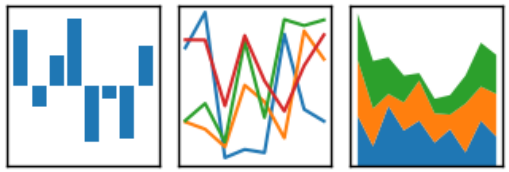


- Common to have data structured in a spread sheet.
- The text file version is called **CSV** for comma separated values.
- Each row is a line; columns are separated by commas.

Structured Data

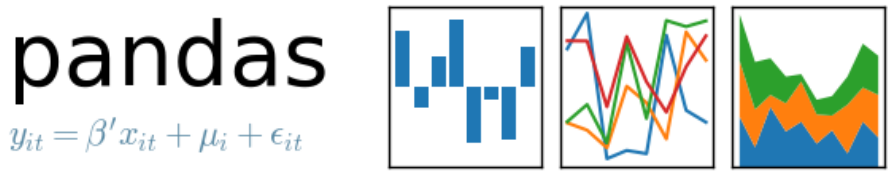
pandas

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



- Common to have data structured in a spread sheet.
- The text file version is called **CSV** for comma separated values.
- Each row is a line; columns are separated by commas.
- We will use the popular Python Data Analysis Library (**Pandas**).

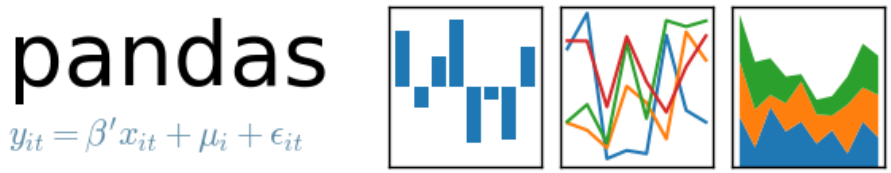
Structured Data



- Common to have data structured in a spread sheet.
- The text file version is called **CSV** for comma separated values.
- Each row is a line; columns are separated by commas.
- We will use the popular Python Data Analysis Library (**Pandas**).
- To use, add to the top of your file:

```
import pandas as pd
```

Structured Data



- Common to have data structured in a spread sheet.
- The text file version is called **CSV** for comma separated values.
- Each row is a line; columns are separated by commas.
- We will use the popular Python Data Analysis Library (**Pandas**).
- To use, add to the top of your file:

```
import pandas as pd
```

- To read in a CSV file:

```
myVar = pd.read_csv("myFile.csv")
```

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
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`

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,,,,,,
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`

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,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`

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()
```

```
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
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

In Lab 6

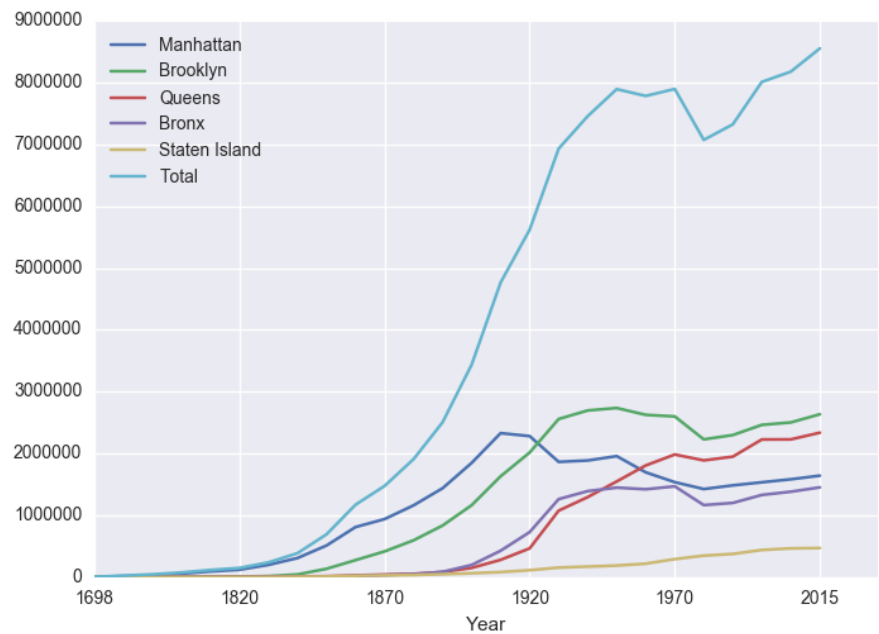
Example: Reading in CSV Files

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

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

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

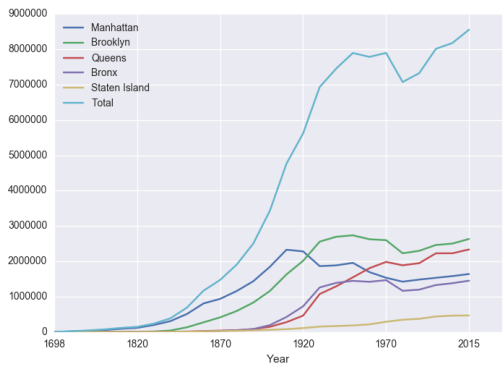
```
Source:  https://en.wikipedia.org/wiki/Demographics_of_New_York_City,,,,,,
All population figures are consistent with present-day boundaries.,,,,,,
First census after the consolidation of the five boroughs,,,,,,
,,,,,,
,,,,,,
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1698,4937,2017,,,727,7681
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

In Lab 6

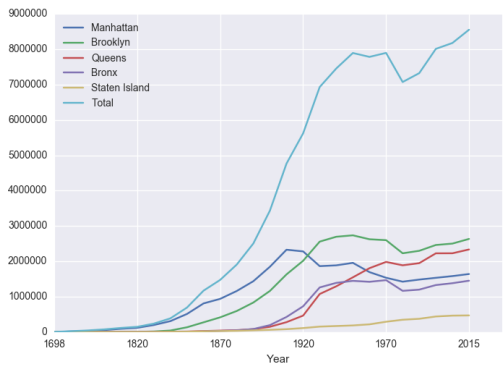


Series in Pandas



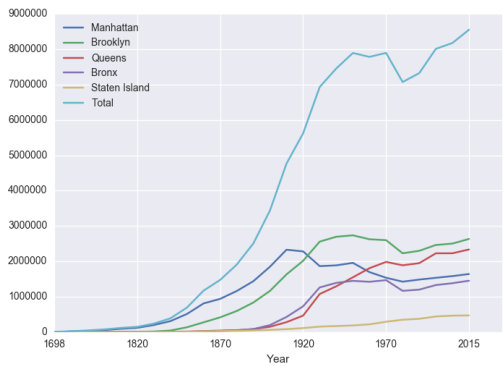
- Series can store a column or row of a DataFrame.

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())
```

In Pairs or Triples:

Predict what the code will do:

```
motto = "Mihi Cura Futuri"
l = len(motto)
for i in range(l):
    print(motto[i])
for j in range(l-1,-1,-1):
    print(motto[j])
```

```
import matplotlib.pyplot as plt
import numpy as np
img = plt.imread('csBridge.png')
plt.imshow(img)
plt.show()
height = img.shape[0]
width = img.shape[1]
img2 = img[:height/2, :width/2]
plt.imshow(img2)
plt.show()
```

- And, design a program that asks maps time of day versus last month's 311 complaints.
  (Design only the pseudocode.)

Design Question

And, design a program that asks maps time of day versus last month's 311 complaints.
(Design only the pseudocode.)

Design Question

And, design a program that asks maps time of day versus last month's 311 complaints.
(Design only the pseudocode.)

How to approach this:

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

Design Question

And, design a program that asks maps time of day versus last month's 311 complaints.
(Design only the pseudocode.)

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

And, design a program that asks maps time of day versus last month's 311 complaints.
(Design only the pseudocode.)

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

And, design a program that asks maps time of day versus last month's 311 complaints.
(Design only the pseudocode.)

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

And, design a program that asks maps time of day versus last month's 311 complaints.
(Design only the pseudocode.)

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. Find data set (great place to look: NYC OpenData).

Design Question

And, design a program that asks maps time of day versus last month's 311 complaints.
(Design only the pseudocode.)

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. Find data set (great place to look: NYC OpenData).
  2. Open up the CSV file.

Design Question

And, design a program that asks maps time of day versus last month's 311 complaints.
(Design only the pseudocode.)

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. Find data set (great place to look: NYC OpenData).
  2. Open up the CSV file.
  3. Count the number of complaints for each time.

Design Question

And, design a program that asks maps time of day versus last month's 311 complaints.
(Design only the pseudocode.)

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. Find data set (great place to look: NYC OpenData).
  2. Open up the CSV file.
  3. Count the number of complaints for each time.
  4. Save the counts in a new column.

Design Question

And, design a program that asks maps time of day versus last month's 311 complaints.
(Design only the pseudocode.)

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. Find data set (great place to look: NYC OpenData).
  2. Open up the CSV file.
  3. Count the number of complaints for each time.
  4. Save the counts in a new column.
  5. Create a plot of time versus counts.

Design Question

And, design a program that asks maps time of day versus last month's 311 complaints.
(Design only the pseudocode.)

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. Find data set (great place to look: NYC OpenData).
  2. Open up the CSV file.
  3. Count the number of complaints for each time.
  4. Save the counts in a new column.
  5. Create a plot of time versus counts.
  6. Display the plot.

Final Prep, #1

(1) Write Python code that prompts the user for distance in kilometers, and prints out the distance in miles.
Useful formula: $miles = 0.621 \cdot kilometers$.

(2) What is the output of the following:

```
a = 4
b = a**2
c = b % 5
d = b // 5
print(a,b,c,d)
a,b = b,c
print(a,b,c,d)
a = b % 2
print(a,b,c,d)
```

Final Prep, #1

(1) Write Python code that prompts the user for distance in kilometers, and prints out the distance in miles.
Useful formula: $miles = 0.621 \cdot kilometers$.

```
# Your name
# Program converts kilometers to miles
```

Final Prep, #1

(1) Write Python code that prompts the user for distance in kilometers, and prints out the distance in miles.
Useful formula: $miles = 0.621 \cdot kilometers$.

```
# Your name
# Program converts kilometers to miles

km = int(input('Enter kilometers'))
```

Final Prep, #1

(1) Write Python code that prompts the user for distance in kilometers, and prints out the distance in miles.
Useful formula: $miles = 0.621 \cdot kilometers$.

```
# Your name
# Program converts kilometers to miles

km = int(input('Enter kilometers'))
miles = 0.621 * km
```

Final Prep, #1

(1) Write Python code that prompts the user for distance in kilometers, and prints out the distance in miles.
Useful formula: $miles = 0.621 \cdot kilometers$.

```
# Your name
# Program converts kilometers to miles

km = int(input('Enter kilometers'))
miles = 0.621 * km

print(km)
```

Python Tutor

(2) What is the output of the following:

```
a = 4
b = a**2
c = b % 5
d = b // 5
print(a,b,c,d)
a,b = b,c
print(a,b,c,d)
a = b % 2
print(a,b,c,d)
```

(Demo with pythonTutor)

Functions

- Functions are a way to break code into pieces, that can be easily reused.

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.

Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`

Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- When you **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:

Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- When you **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:
  Example: `print("Hello", "World")`

Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- When you **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:
  Example: `print("Hello", "World")`
- Can write, or **define** your own functions,

Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- When you **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:
  Example: `print("Hello", "World")`
- Can write, or **define** your own functions, which are stored, until invoked or called.

"Hello, World!" with Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

Python Tutor

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

(Demo with pythonTutor)

In Pairs or Triples:

Predict what the code will do:

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

```
def monthString(monthNum):
    """
    Takes as input a number, monthNum, and
    returns the corresponding month name as a string
    Example: monthString(1) returns "January".
    Assumes that input is an integer ranging from 1
    """

    monthString = ""

    ###################################
    ### FILL IN YOUR CODE HERE      ###
    ### Other than your name above, ###
    ### this is the only section    ###
    ### you change in this program. ###
    ###################################

    return(monthString)

def main():
    n = int(input('Enter the number of the month: '))
    mString = monthString(n)
    print('The month is', mString)
```

Python Tutor

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

(Demo with `pythonTutor`)

IDLE

```
def monthString(monthNum):
    """
    Takes as input a number, monthNum, and
    returns the corresponding month name as a string.
    Example: monthString(1) returns "January".
    Assumes that input is an integer ranging from 1 to 12
    """

    monthString = ""

    ###################################
    ### FILL IN YOUR CODE HERE      ###
    ### Other than your name above, ###
    ### this is the only section    ###
    ### you change in this program. ###
    ###################################

    return(monthString)

def main():
    n = int(input('Enter the number of the month: '))
    mString = monthString(n)
    print('The month is', mString)
```

(Demo with IDLE)

In Pairs or Triples:

Predict what the code will do: main() with turtles

```
#CSci 127 Teaching Staff
#Triangles two ways...
import turtle

def setUp(t, dist, col):
     t.penup()
     t.forward(dist)
     t.pendown()
     t.color(col)

def nestedTriangle(t, side):
     if side > 10:
          for i in range(3):
               t.forward(side)
               t.left(120)
          nestedTriangle(t, side/2)

def fractalTriangle(t, side):
     if side > 10:
          for i in range(3):
               t.forward(side)
               t.left(120)
               fractalTriangle(t, side/2)
```

```
def main():
     nessa = turtle.Turtle()
     setUp(nessa, 100, "violet")
     nestedTriangle(nessa, 160)

     frank = turtle.Turtle()
     setUp(frank, -100, "red")
     fractalTriangle(frank, 160)

if __name__ == "__main__":
     main()
```

IDLE

```
#CSci 127 Teaching Staff
#Triangles two ways...
import turtle

def setUp(t, dist, col):
     t.penup()
     t.forward(dist)
     t.pendown()
     t.color(col)

def nestedTriangle(t, side):
     if side > 10:
          for i in range(3):
               t.forward(side)
               t.left(120)
          nestedTriangle(t, side/2)

def fractalTriangle(t, side):
     if side > 10:
          for i in range(3):
               t.forward(side)
               t.left(120)
               fractalTriangle(t, side/2)
```

(Demo with IDLE)

Recap: Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.

Recap: Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.

Recap: Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`

Recap: Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- When you **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:

Recap: Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- When you **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:
  Example: `print("Hello", "World")`

Recap: Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

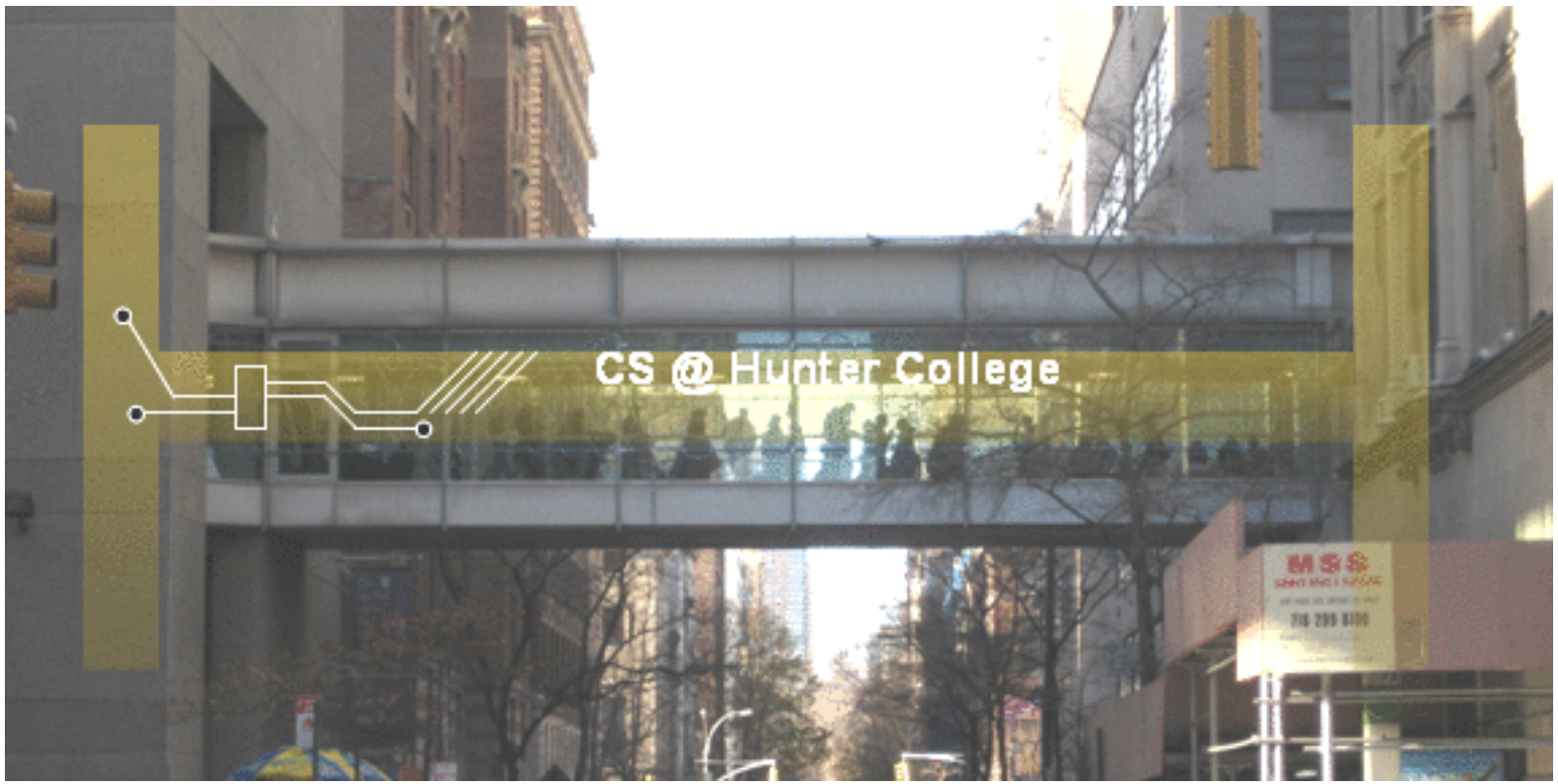
- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- When you **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:
  Example: `print("Hello", "World")`
- Can write, or **define** your own functions,

Recap: Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- When you **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:
  Example: `print("Hello", "World")`
- Can write, or **define** your own functions, which are stored, until invoked or called.

Lecture Slips



- On-line lecture slips: `tinyurl.com/ybgz7bks`

Final Prep

- The last 5 minutes of lecture will be on mock final exam questions.

Final Prep

- The last 5 minutes of lecture will be on mock final exam questions.
- Pull out a sheet of paper, and do as much as you can before class ends.

Final Prep

- The last 5 minutes of lecture will be on mock final exam questions.
- Pull out a sheet of paper, and do as much as you can before class ends.
- Will discuss solutions next lecture.