Command Line and Python Introduction

Jennifer Helsby, Eric Potash
Computation for Public Policy
Lecture 2: January 7, 2016
computationforpolicy.github.io

Today

- Assignment #1!
- Computer architecture
- Basic command line skills
- Python fundamentals
  - Variables and types
  - Basic data structures
  - First programs
- Basic plotting with Python’s matplotlib

Assignment #1

- **Goal:** Get familiar with Python. Learn to do simple Python programming and make simple plots
- Use crime statistics data from the City of Chicago data portal
- Due in 1 week: January 14, 2016
- `https://computationforpolicy.github.io/assignments/01.html`
- Submit via email to the course email account

Introduction

In this assignment you will practice basic data manipulation and plotting using Python. We will use City of Chicago crime statistics from the open data portal as our data source. First, go to the data portal and view the crime statistics data available:

https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2

Since there's quite a lot of data, for this assignment let's filter down the data a bit so we have a more manageable dataset.

Go to Filter and apply a Filter on the Date column to select crimes after 12/01/2015 12:00:00 AM. This should produce a dataset with ~18,000 crimes (approximate as the data may be updated as you do the assignment). Now, select Export, and download the data in 'csv' format, a text format where each field is separated by a comma.

Your solution will be a Python script that does the following basic computations. Use whichever modules you think are appropriate. Denote the solution to each part with a comment, e.g.:

```
# Part I: Question 1
code goes here
```

To get credit for your answer, you must show in full the code that produced the answer.

Part I: Data Manipulation

Question 1: Write a piece of code that reads in the data to a format that you can use.

Let's now look at some descriptive statistics.

Question 2: How often did a crime result in an arrest?

Question 3: Which types of crimes most often result in arrest?

Question 4: What are the number of weapons violations (one of the Primary Types) per district?

Question 5: What are the number of arrests per days of the week? Which day of the week has the most arrests?

Part 2: Basic Plotting

In this part we'll make a few plots to visualize the data.

Question 6: Make bar charts that show (a) the result of Question 4 and (b) Question 5.

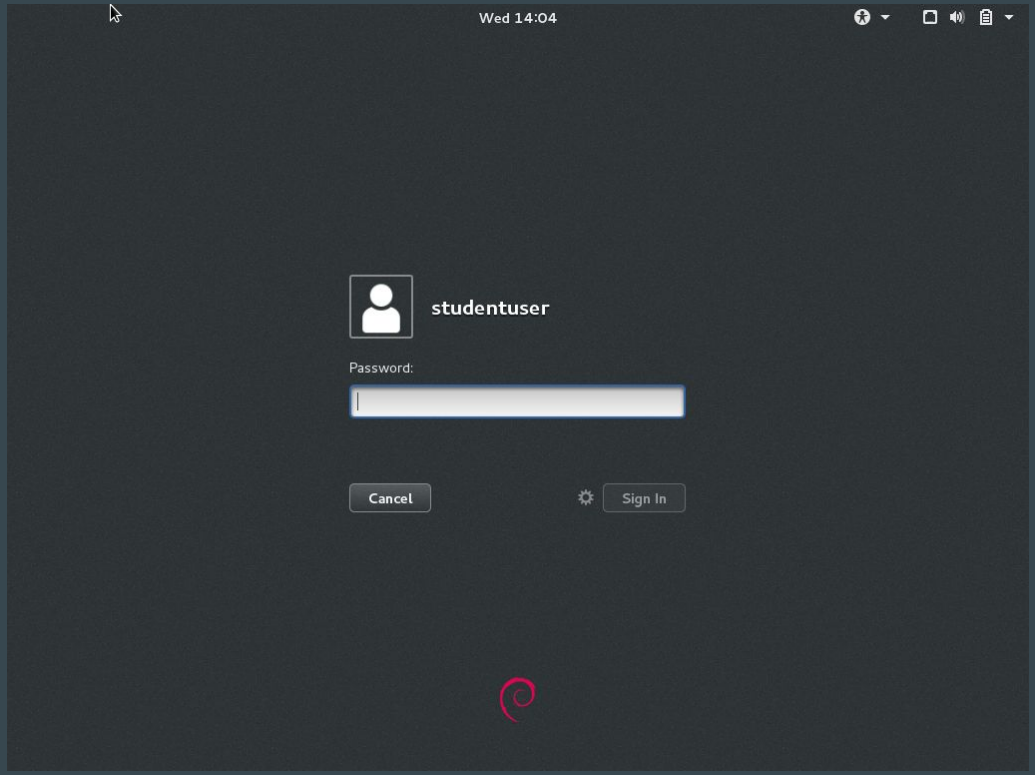
Question 7: Make a scatter plot of latitude versus longitude (we'll get more into making real maps later in the course) for those crimes where the Primary Type was: deceptive pratice.

Question 8: Make a histogram that shows the number of arrests per beat. The Y-axis should show Frequency and the X-axis should show the count of arrests in that beat.

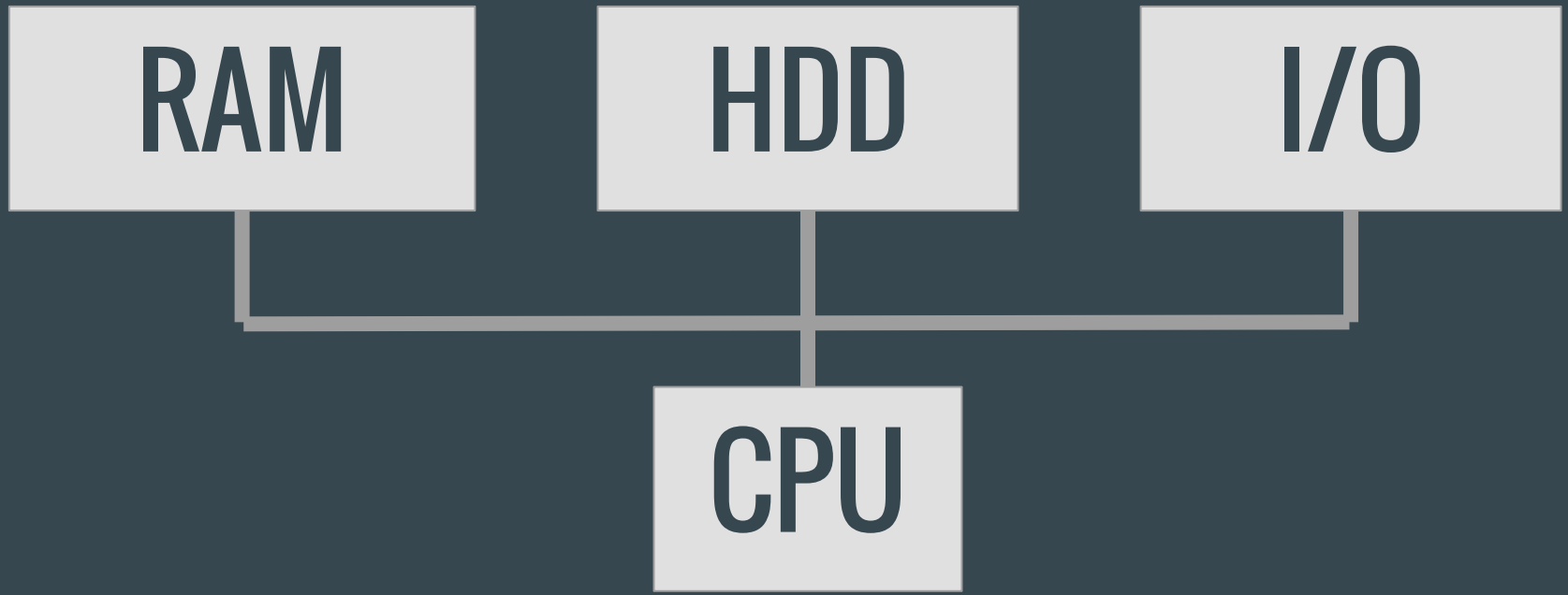
Challenge 1 (ungraded): Make a timeseries plot of arrests per date in that month

Class Virtual Machine

- Class VM download: Link on Chalk
- Load with VirtualBox (available for Mac, Windows, GNU/Linux)
- Feel free to install software yourself



Hardware



Software

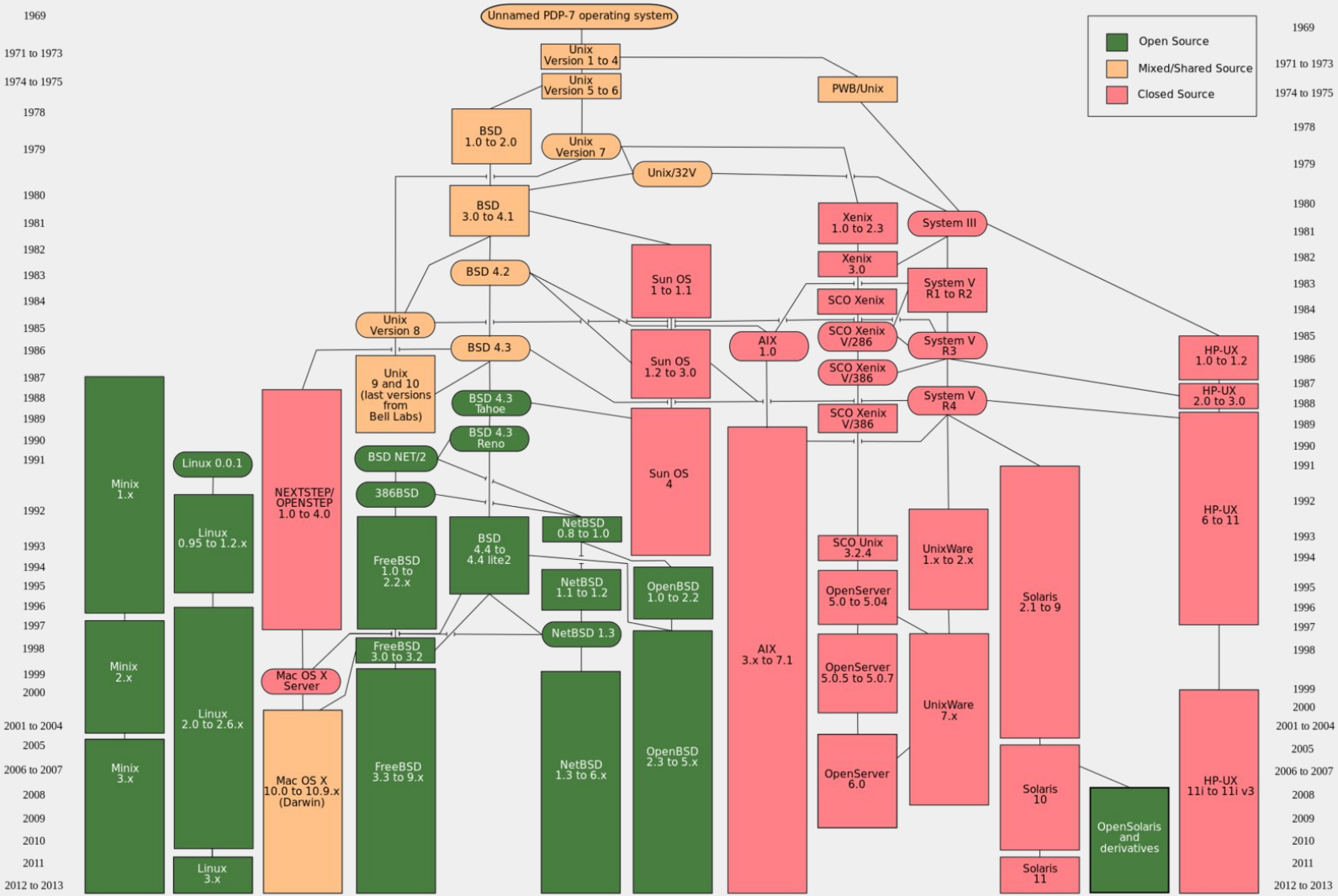
Adapted from Silberschatz, Galvin and Gagne

Operating System

- Does things that the user doesn’t need/want to deal with
- Makes the system more efficient and convenient to use
- Intermediary between the hardware and user

Unix and its variants

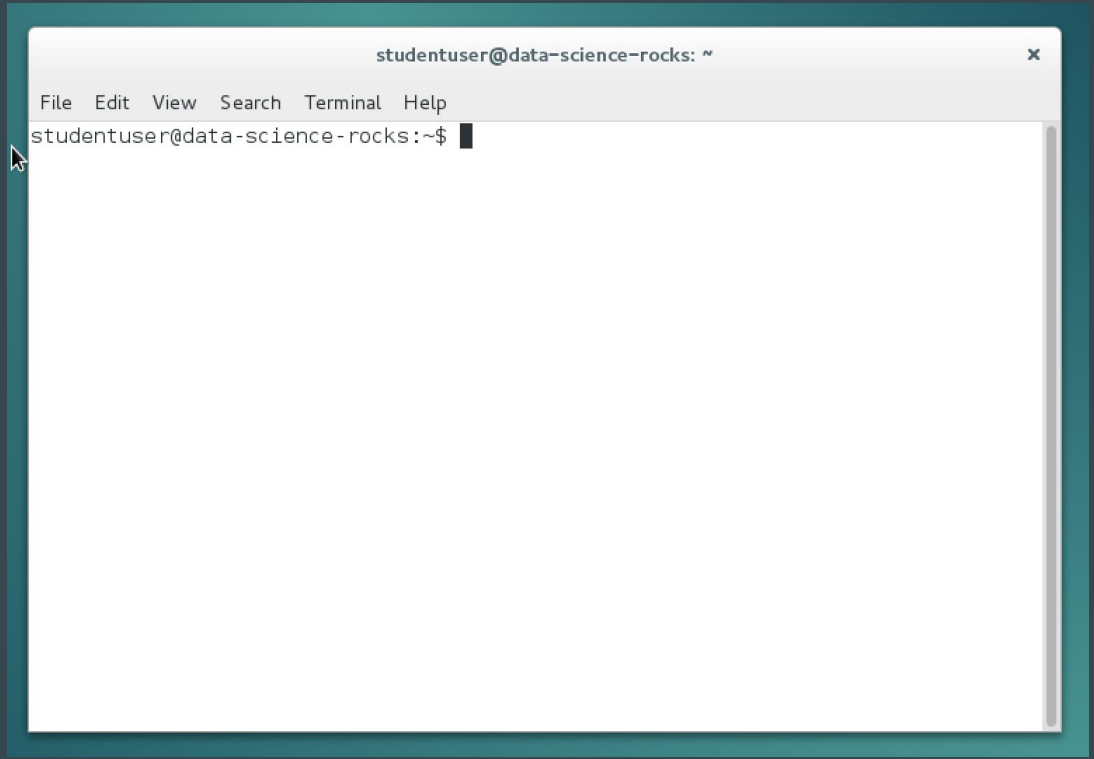
- Unix is family of extremely popular operating systems
- Variants:
  - GNU/Linux
  - Mac OS X
  - BSD Unix



http://www.computerworld.com/article/2524660/operating-systems/the-unix-family-tree.html

Command Line: Mac OS X

Command Line: Debian GNU/Linux



Command Line Syntax

```
$ commandname --flags -a -b -c arg1 arg2
```

Command Line: Basic Unix Commands 1

- `ls`: List contents of directory
- `pwd`: Print current working directory
- `cd`: Change directory
  - `cd directory_name`
- `mkdir`: Make new directory
  - `mkdir new_directory_name`
- `cp`: Copy a file or directory (with `-r`)
  - `cp original_file copy_file`
- `cat`: print to screen (really concatenate)
  - `cat file1`

Command Line: Basic Unix Commands 2

- `rm`: Remove file
  - `rm file_to_delete`
- `rmdir`: Remove directory
  - `rmdir directory_to_delete`
- `nano`: Text editor in terminal
  - `nano name_of_textfile`
- `wget`: Get file from the web
  - `wget http://mywebsite.com/dataset.csv`

Use rm with caution!!

Navigating Files

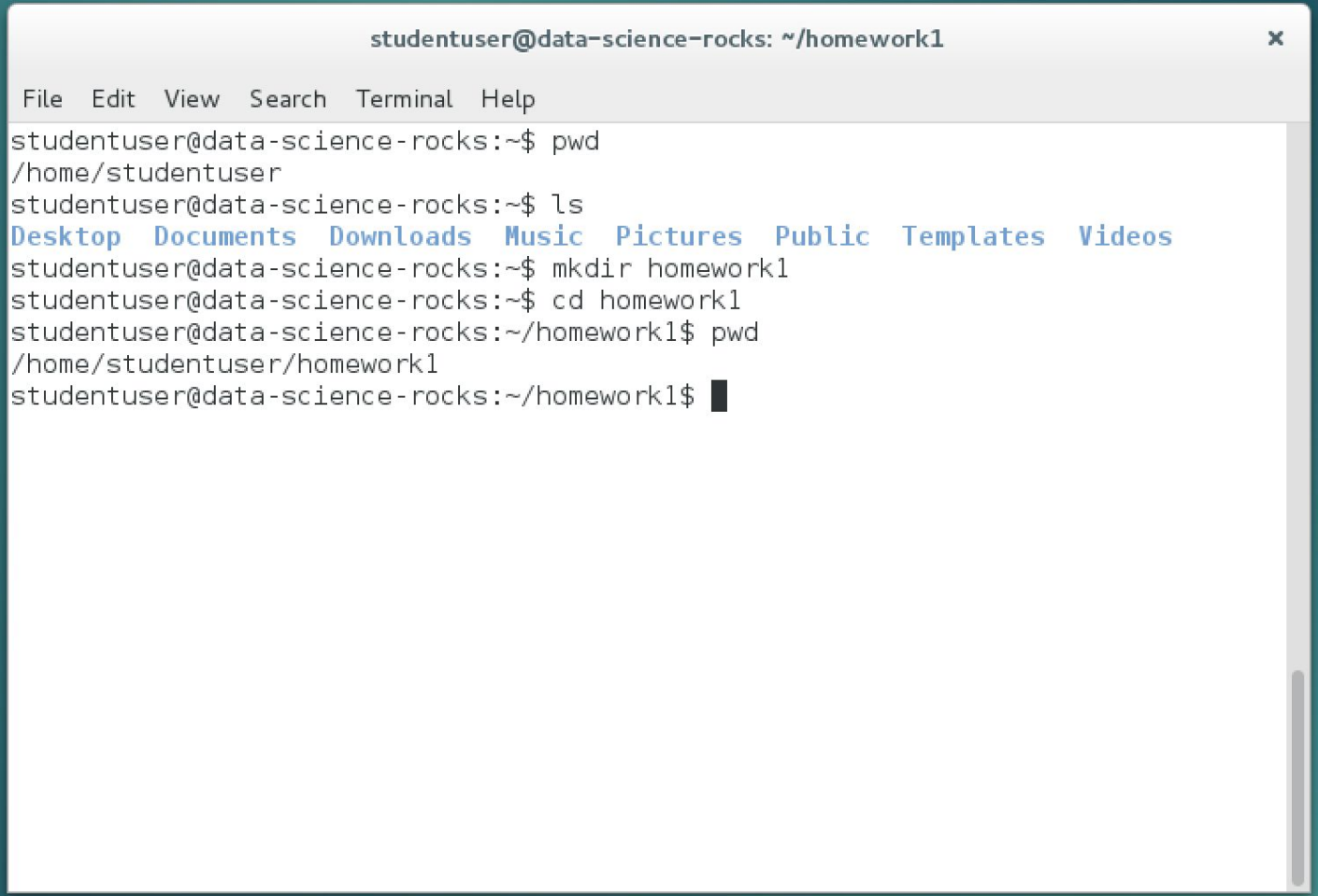
- Look at first few or last few lines using `head` and `tail`:

```
tail -5 Crimes_-_2001_to_present.csv   # last 5 lines

head -5 Crimes_-_2001_to_present.csv   # first 5 lines
```

  (especially important for very large files)

- Search through files looking for certain strings using `grep`:

```
grep 'S ESSEX AVE' Crimes_-_2001_to_present.csv
```


studentuser@data-science-rocks: ~/homework1
File Edit View Search Terminal Help
studentuser@data-science-rocks:~$ pwd
/home/studentuser
studentuser@data-science-rocks:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
studentuser@data-science-rocks:~$ mkdir homework1
studentuser@data-science-rocks:~$ cd homework1
studentuser@data-science-rocks:~/homework1$ pwd
/home/studentuser/homework1
studentuser@data-science-rocks:~/homework1$

Notation

`~`	Default directory is a user's *home directory*, e.g. `/home/studentuser` on the VM
`.`	Current directory
`..`	Parent directory

Notation

`~`	Default directory is a user's *home directory*, e.g. `/home/studentuser` on the VM
`.`	Current directory
`..`	Parent directory

What directory is `~/..`?

Redirecting Output

`cmd1 | cmd2`	Send output of `cmd1` as input to `cmd2`
`cmd1 > output.txt`	Send output of `cmd1` to a new file
`cmd1 >> debug.log`	Append output of `cmd1` to an existing file

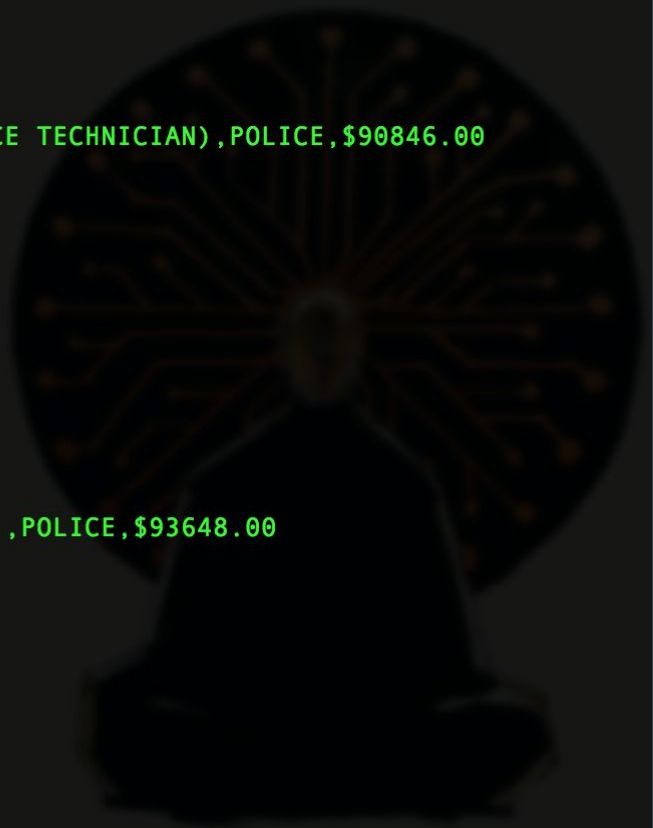
Example: Redirection

- Download the current employee names, salaries, and position titles of city employeesfrom the City of Chicago data portal (`https://data.cityofchicago.org/`)
- Let's count the number of police officers in the dataset using grep and redirection:

```
Thu Jan 07 10:59 ☢ Computation and Public Policy (☞ﾜ゚)☞ $ grep 'POLICE OFFICER' Cu
rrent_Employee_Names__Salaries__and_Position_Titles.csv | wc -l
   10634
```

- `wc -l` counts the number of lines
- Can also redirect to `less` and page through the results:

```
Thu Jan 07 10:59 ☢ Computation and Public Policy (☞ﾜ゚)☞ $ grep 'POLICE OFFICER' Current_Employee_
Names__Salaries__and_Position_Titles.csv | less
```



```
"AARON,  KARINA",POLICE OFFICER,POLICE,$80778.00
"ABBATE,  TERRY M",POLICE OFFICER,POLICE,$86520.00
"ABDALLAH,  ZAID",POLICE OFFICER,POLICE,$69684.00
"ABDELHADI,  ABDALMAHD",POLICE OFFICER,POLICE,$80778.00
"ABDELMAJEID,  AZIZ",POLICE OFFICER,POLICE,$80778.00
"ABEJERO,  JASON V",POLICE OFFICER,POLICE,$86520.00
"ABRAHAM,  NANCY A",POLICE OFFICER,POLICE,$46206.00
"ABRAMS,  HENRY L",POLICE OFFICER,POLICE,$92316.00
"ABRON,  FLOYD",POLICE OFFICER,POLICE,$86520.00
"ABSTON,  KATHY A",POLICE OFFICER,POLICE,$89718.00
"ABUDAYEH,  ELIAS",POLICE OFFICER,POLICE,$46206.00
"ABUZANAT,  ABDALLA H",POLICE OFFICER (ASSIGNED AS EVIDENCE TECHNICIAN),POLICE,$90846.00
"ACCARDO,  JENNIFER A",POLICE OFFICER,POLICE,$83616.00
"ACCARDO,  THOMAS J",POLICE OFFICER,POLICE,$83616.00
"ACEVEDO,  AARON F",POLICE OFFICER,POLICE,$80778.00
"ACEVEDO,  ALEJANDRO R",POLICE OFFICER,POLICE,$80778.00
"ACEVEDO,  BIENVENIDO",POLICE OFFICER,POLICE,$86520.00
"ACEVEDO,  CLARISSA",POLICE OFFICER,POLICE,$92316.00
"ACEVEDO,  EDWARD",POLICE OFFICER,POLICE,$83616.00
"ACEVEDO,  ERIC",POLICE OFFICER,POLICE,$46206.00
"ACEVEDO,  ILIA",POLICE OFFICER,POLICE,$69684.00
"ACEVEDO,  JANETE",POLICE OFFICER,POLICE,$80778.00
"ACEVEDO,  JASON M",POLICE OFFICER,POLICE,$83616.00
"ACEVEDO,  JEFF",POLICE OFFICER,POLICE,$89718.00
"ACEVEDO,  MARCO A",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"ACEVEDO,  MARTIN J",POLICE OFFICER,POLICE,$86520.00
"ACEVEDO,  NADINE M",POLICE OFFICER,POLICE,$83616.00
"ACEVEDO,  RAFAEL",POLICE OFFICER,POLICE,$83616.00
"ACEVES,  JUAN C",POLICE OFFICER,POLICE,$83616.00
"ACEVEZ,  ANTHONY E",POLICE OFFICER,POLICE,$86520.00
"ACHILLY,  LISA S",POLICE OFFICER,POLICE,$86520.00
"ACOSTA,  JESSE A",POLICE OFFICER,POLICE,$89718.00
"ACOSTA,  MICHELE D",POLICE OFFICER,POLICE,$86520.00
"ACSVECS,  ZAIREH",POLICE OFFICER,POLICE,$80778.00
"ADAIR,  KENNETH R",POLICE OFFICER,POLICE,$83616.00
"ADAMIAK,  SIMON P",POLICE OFFICER,POLICE,$83616.00
"ADAMIK,  JEFFREY S",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
```

Example: Redirection

- Take only the detectives and redirect the output to a new text file:

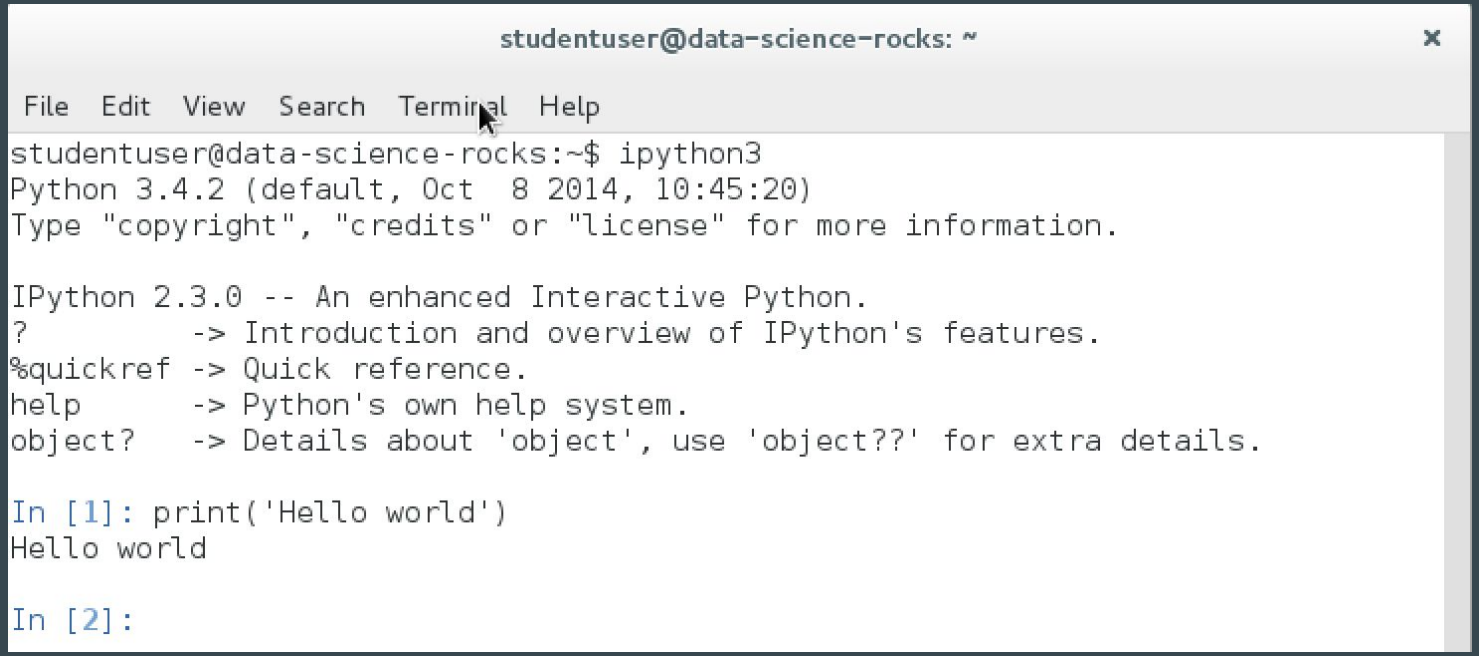
```
Thu Jan 07 11:09 ☢ Computation and Public Policy (☞ﾜﾟ)☞ $ grep 'DETECTIVE' Current
_Employee_Names__Salaries__and_Position_Titles.csv > detectives
```

```
"WALSH,  DONNA M",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"WALSH,  MICHAEL J",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"WALSH,  PATRICIA L",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"WASHINGTON,  TONI M",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$99888.00
"WATHEN,  VICTOR L",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"WATSON,  ARLESEUIA N",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"WEBER,  MATTHEW J",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"WEBER,  RUBEN",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"WEDDINGTON JR,  ARNOLD E",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$97044.00
"WEISSER,  JOHN",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"WEITZMAN,  JOSHUA C",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$97044.00
"WENSERITT,  TODD R",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$97044.00
"WEXLER,  PHILIP R",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$97044.00
"WICKERY,  KEVIN J",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$99888.00
"WIEDENSKI,  MARK J",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"WIGGINS,  KENNETH L",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$97044.00
"WIKTOREK,  MARK C",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$99888.00
"WILBORN,  DESTRY M",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"WILLIAMS,  ALLECIA J",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$99888.00
"WILLIAMS,  ESTHER S",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$99888.00
"WILLIAMS,  KAREN L",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$99888.00
"WILLIAMS,  MARZEEK",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$97044.00
"WILLIAMSON,  ANNE V",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$99888.00
"WILLIAMS,  RAPUNZEL A",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$99888.00
"WILLIAMS,  SUSAN M",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"WILLIAMS,  TENICIA",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"WISNIEWSKI,  GARY A",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$97044.00
"WOJCIK,  STEPHEN J",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"WOOD,  MICHELE A",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$90456.00
"WOOLEY,  CHANETE R",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"WO,  RANDALL K",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$97044.00
"WRONKOWSKI,  ANTHONY R",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$99888.00
"XANOS,  NICHOLAS S",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"YANAGIDATE,  ALAN I",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$99888.00
"YAVERSKI,  BRIAN T",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$97044.00
"YNIGUEZ,  JOHN J",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$97044.00
"YURISICH,  DONALD M",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$93648.00
"ZALEWSKI,  JOHN E",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$97044.00
"ZAWILA,  MARK A",POLICE OFFICER (ASSIGNED AS DETECTIVE),POLICE,$97044.00
```

Python

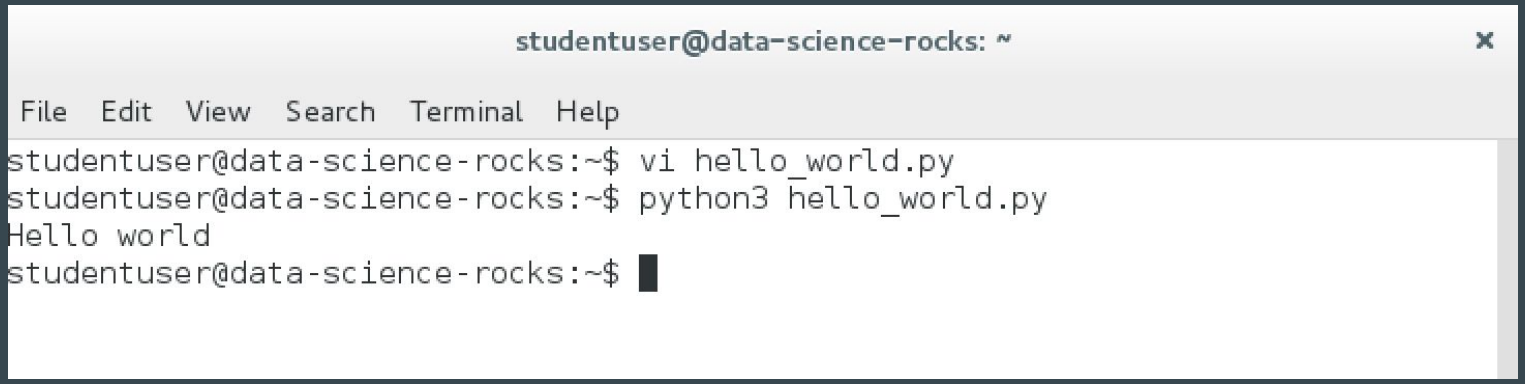
- Popular and easy to learn interpreted language
- Two versions of Python commonly used:
  - 2.7x - Run with `python` and `ipython`
  - 3.x (recommended) - Run with `python3` and `ipython3`
- Running Python code:
  - From command line
  - From interpreter

Python: Interpreter



Quit Python with `quit()` or CTRL-D

Python: Command Line



Python: Basic Data Types

- Integer: `2`
- String: `“this is a string”` or `‘this is a string’`
- List: `[2, 3, 5, 7]` or `[‘list’, ‘of’, ‘strings’]`
- Floating point: `3.1`
- Boolean: `True`, `False`
- Dictionary: `{‘alice’: ‘apples’, ‘bob’: ‘oranges’}`

Python: Variables

- *Assign* values to variables using the assignment operator (=):

```
In [26]: num_students = 12

In [27]: num_assignments = 7
```

Python: Arithmetic

- Simple integer arithmetic:

```
In [26]: num_students = 12

In [27]: num_assignments = 7

In [28]: num_to_grade = num_students * num_assignments

In [29]: num_to_grade
Out[29]: 84
```

- Arithmetic operators:

  `+`, `-`, `*`, `/`, `//` (discard remainder), `%` (mod, return remainder of division), ** (raise to the power)

Python: String Manipulation

- Assign a string using “stringhere”, ‘stringhere’, “””stringhere”””, ‘’’stringhere’’’:

```
In [2]: dog_name = "Rover"
```

- Indexing a string using square brackets:

index	0	1	2	3	4
dog_name	‘R’	‘o’	‘v’	‘e’	‘r’

```
In [2]: dog_name = "Rover"

In [3]: dog_name[0]
Out[3]: 'R'
```

Python: String Manipulation

- Concatenate (join strings together) with `+` and repeat them N times with `* N`:

```
In [9]: my_string = 'A' + 'B' + 'C' + 'D'

In [10]: my_string * 8
Out[10]: 'ABCDABCDABCDABCDABCDABCDABCDABCD'
```

Python: String Methods 1

- Split a string on spaces using `split()`:

```
In [24]: test_string = "The cat in the hat"

In [25]: test_string.split()
Out[25]: ['The', 'cat', 'in', 'the', 'hat']
```

- Split on another character, `char`, using `split(char)`:

```
In [26]: test_string.split('i')
Out[26]: ['The cat ', 'n the hat']
```

Python: String Methods 2

- Return the uppercase and lowercase versions of a string with `upper()` and `lower()`:

```
In [27]: test_string.upper()
Out[27]: 'THE CAT IN THE HAT'

In [28]: test_string.lower()
Out[28]: 'the cat in the hat'
```

- Replace a substring with another using `replace()`:

```
In [29]: test_string.replace('cat', 'dog')
Out[29]: 'The dog in the hat'
```

Python: Lists

- *Assign* lists with comma-separated values in square brackets:

```
In [14]: favorite_numbers = [2, 3, 5, 7, 11]
```

- Index with square brackets:

index	0	1	2	3	4
favorite_numbers	2	3	5	7	11

```
In [15]: favorite_numbers[0]
Out[15]: 2
```

Python: Length of list or string

- Determine length of list with:

```
In [16]: len(favorite_numbers)
Out[16]: 5
```

- Similarly with strings:

```
In [32]: test_string = "The cat in the hat"

In [33]: len(test_string)
Out[33]: 18
```

Python: List Methods 1

- Append a new value with `append()` or multiple values with `extend()`:

index	0	1	2	3	4	5
favorite_numbers	2	3	5	7	11	13

```
In [17]: favorite_numbers.append(13)

In [18]: favorite_numbers
Out[18]: [2, 3, 5, 7, 11, 13]
```

- And then pop them off with `pop()`:

```
In [39]: favorite_numbers.pop()
Out[39]: 13

In [40]: favorite_numbers
Out[40]: [2, 3, 5, 7, 11]
```

Python: List Methods 2

- Reverse list with `reverse()`:

```
In [5]: types_of_pizza = ['cheese', 'hawaiian', 'veggie']

In [6]: types_of_pizza.reverse()

In [7]: types_of_pizza
Out[7]: ['veggie', 'hawaiian', 'cheese']
```

- Sort lists with `sort()`:

```
In [44]: dice_throws = [2, 3, 1, 5, 3, 6, 2]

In [45]: dice_throws.sort()

In [46]: dice_throws
Out[46]: [1, 2, 2, 3, 3, 5, 6]
```

Python: Dictionaries

- Dictionaries are key value stores
- Define using: `{‘key1’: value1, ‘key2’: value2}`
- Reference a given entry using square brackets

```
In [47]: food_to_order = {'Alice': 'apple', 'Bob': 'bananas'}

In [48]: food_to_order['Alice']
Out[48]: 'apple'
```

Python: Boolean

- Comparison operators: <= (less than or equal to), >=, == (equal to), >, <, != (not equal to)

```
In [53]: is_tuesday = False

In [54]: is_lecture_today = True

In [55]: is_lecture_today == False
Out[55]: False

In [56]: is_lecture_today == True
Out[56]: True
```

- Combine with and and or:

```
In [60]: is_tuesday == True and is_lecture_today == True
Out[60]: False

In [61]: is_tuesday != True and is_lecture_today == True
Out[61]: True
```

Python: Data Types

- Determine data types using `type()`:

```
In [9]: my_string = 'A' + 'B' + 'C' + 'D'

In [10]: my_string * 8
Out[10]: 'ABCDABCDABCDABCDABCDABCDABCDABCD'

In [11]: type(my_string)
Out[11]: str
```

Python: Loading Python modules

- Wide array of modules in Python available for almost any computational task
- Use existing code in these libraries when you can to prevent reinventing the wheel
- Libraries are imported into the Python environment using the `import` statement:

```
import math
```

Other Python Tidbits

- Whitespace (tabs, spaces) is meaningful
  - Will talk more about this next time
- Comment with #

```
# this is a python comment

my_var = ‘this is not a python comment’

# comments will be ignored by the interpreter

my_string = ‘but you can put # in a string’
```

Important Python Packages for Scientific Computing

- Scipy:
  - Library of codes for scientific computing (linear algebra, stats, interpolation, ...)
  - `import scipy`
- Numpy:
  - Library for fast computations on large arrays
  - `import numpy as np`
- pandas:
  - Popular data analysis toolkit
  - `import pandas as pd`
- matplotlib
  - Most popular package for plotting
  - `import matplotlib.pyplot as plt`

Numpy ndarray

- `ndarray`: n-dimensional array

```
In [2]: import numpy as np

In [3]: example = np.array([2, 3, 5, 7, 11])

In [4]: example
Out[4]: array([ 2,  3,  5,  7, 11])
```

- Useful for linear algebra and image manipulation

Numpy: Useful Functions

- `np.linspace()`: Makes linearly spaced bins

```
In [5]: b = np.linspace(0, 6, 7)

In [6]: b
Out[6]: array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.])
```

- Generate random numbers with `np.rand.random()`:

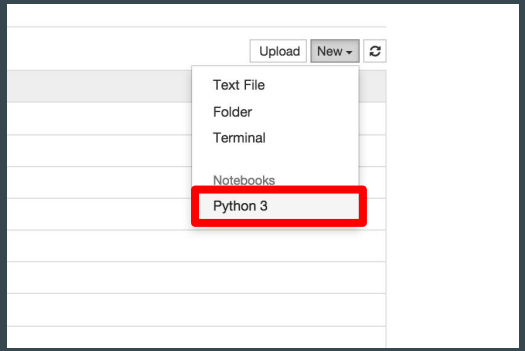
```
In [11]: np.random.rand(4)
Out[11]: array([ 0.08600364,  0.46749484,  0.28960941,  0.80695897])
```

- Make specific matrices with `np.ones()` and `np.zeros()`:

```
In [5]: np.zeros((3, 3))
Out[5]:
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
```

IPython Notebook

- Useful way of doing data analysis (collaborative)
- Launch with: `ipython notebook`
- Can do Python programming in browser environment



localhost:8888/notebooks/Untitled.ipynb?kernel_name=python3

jupyter Untitled Last Checkpoint: a few seconds ago (unsaved changes)

File Edit View Insert Cell Kernel Help Python 3

Code Cell Toolbar: None

In [1]:
```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [ ]:

IPython: Integrating plotting

- Enable plotting inside the notebook with: `% matplotlib inline`

```
In [1]: %matplotlib inline
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
```

Pandas

- Extremely useful data analysis package perfect for analysis in social science, policy, etc.
- Two fundamental objects:
  - DataFrame: 2D object
  - Series: 1D object
- Nicely handles missing data, alignment of data, merging and joining datasets, handling timeseries data

Pandas: Basic Data Handling

- Read in a CSV file into a Pandas DataFrame:

```
In [2]: salaries = pd.read_csv('Current_Employee_Names__Salaries__and_Position_Titles.csv')
```

- Pandas DataFrame:

```
In [4]: salaries
```

Out[4]:

	Name	Position Title	Department	Employee Annual Salary
0	AARON, ELVIA J	WATER RATE TAKER	WATER MGMNT	$88968.00
1	AARON, JEFFERY M	POLICE OFFICER	POLICE	$80778.00
2	AARON, KARINA	POLICE OFFICER	POLICE	$80778.00
3	AARON, KIMBERLEI R	CHIEF CONTRACT EXPEDITER	GENERAL SERVICES	$84780.00
4	ABAD JR, VICENTE M	CIVIL ENGINEER IV	WATER MGMNT	$104736.00
5	ABARCA, ANABEL	ASST TO THE ALDERMAN	CITY COUNCIL	$70764.00
6	ABARCA, EMMANUEL	GENERAL LABORER - DSS	STREETS & SAN	$40560.00
7	ABBATACOLA, ROBERT J	ELECTRICAL MECHANIC	AVIATION	$91520.00
8	ABBATEMARCO, JAMES J	FIRE ENGINEER	FIRE	$90456.00
9	ABBATE, TERRY M	POLICE OFFICER	POLICE	$86520.00

Pandas: Columns (Series)

- Refer to a single column using square brackets:

```
In [5]: salaries['Employee Annual Salary']

Out[5]: 0          $88968.00
        1          $80778.00
        2          $80778.00
        3          $84780.00
        4         $104736.00
        5          $70764.00
        6          $40560.00
        7          $91520.00
        8          $90456.00
        9          $86520.00
        10          $2756.00
        11         $43920.00
        12         $72468.00
        13         $69684.00
        14         $80778.00
        15         $98244.00
        16         $80778.00
        17         $87720.00
        18        $106104.00
```

Pandas: Munging Data

- Need floating point numbers only so that we can make a histogram of salaries
- Problem 1: NaNs

```
32177     $86520.00
32178     $83616.00
32179     $86520.00
32180    $110352.00
32181           NaN
```

- Problem 2: Strings

```
In [8]: type(salaries['Employee Annual Salary'][0])
Out[8]: str
```

Pandas: Munging Data

- **Solution 1**: Use `dropna()` to drop rows with NaNs:

```
In [9]: salaries['Employee Annual Salary'].dropna(inplace=True)
```

- **Solution 2:** Strip off dollar signs with `lstrip('$')` and represent as floating point with `astype(float)`:

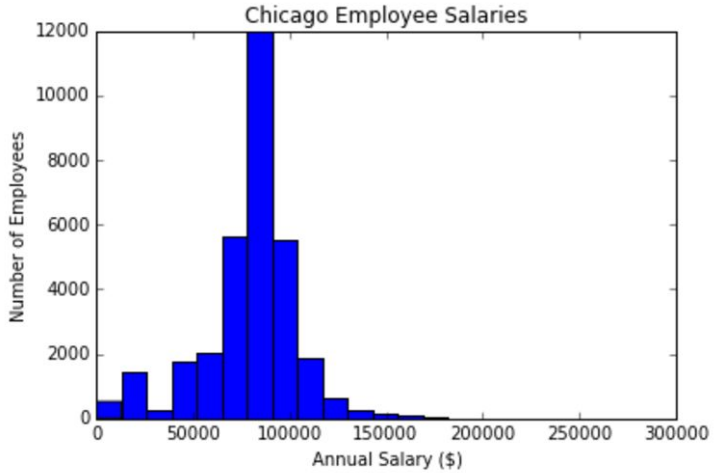
```
In [10]: chicago_employees = salaries['Employee Annual Salary'].str.lstrip('$').astype(float)

In [13]: type(chicago_employees[0])
Out[13]: numpy.float64
```

- We can now visualize this data!

Matplotlib Basic Plotting: Histogram

- Histogram with `plt.hist()`:

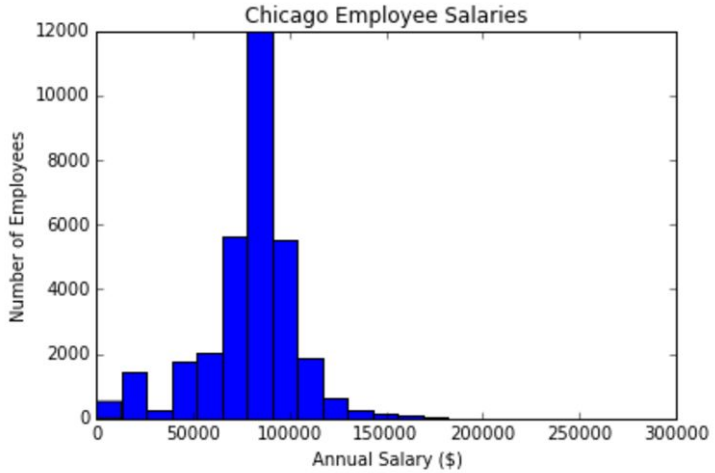
In [25]:
```
n, bins, patches = plt.hist(chicago_employees, 20)

plt.xlabel('Annual Salary ($)')
plt.ylabel('Number of Employees')
plt.title('Chicago Employee Salaries')
plt.show()
```



Matplotlib Basic Plotting: Histogram

- Histogram with `plt.hist()`:

```
In [25]: n, bins, patches = plt.hist(chicago_employees, 20)

         plt.xlabel('Annual Salary ($)')
         plt.ylabel('Number of Employees')
         plt.title('Chicago Employee Salaries')
         plt.show()
```

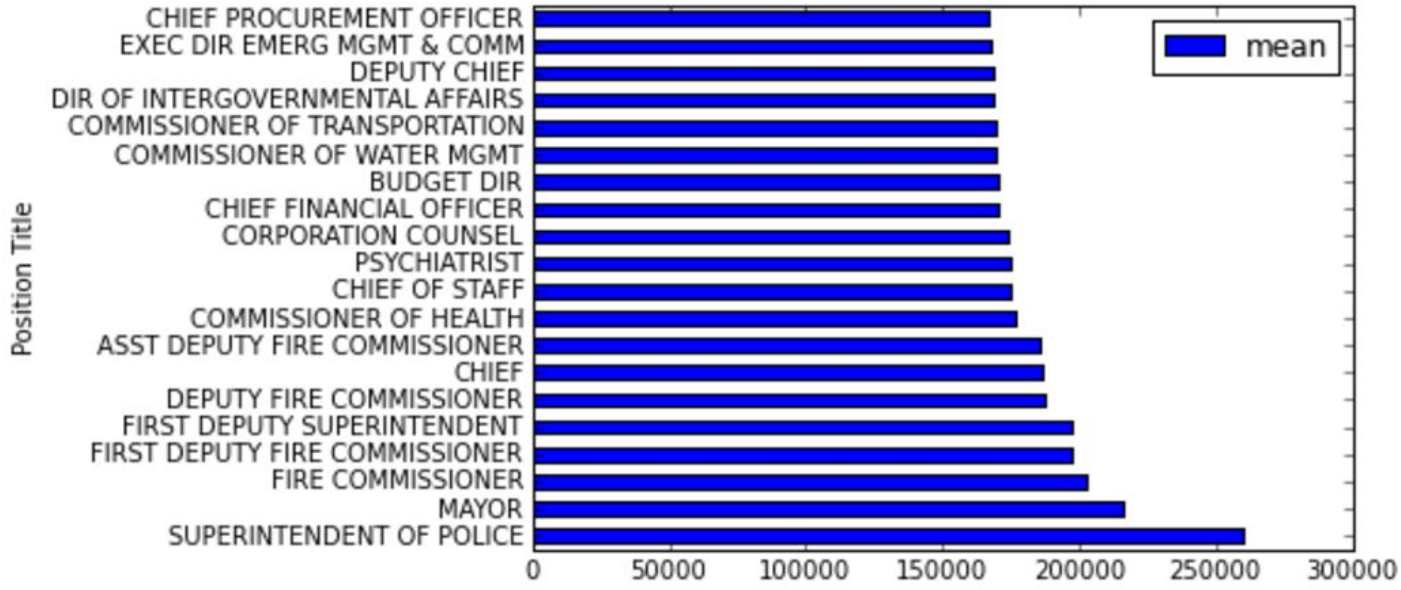



```
In [27]: np.mean(chicago_employees)

Out[27]: 79167.525938908046

In [28]: np.max(chicago_employees)

Out[28]: 260004.0
```

Pandas plotting with `plot()`: Bar Chart

```
In [68]: top_jobs.plot(kind='barh')
```

```
Out[68]: <matplotlib.axes._subplots.AxesSubplot at 0x108cd0a20>
```



Pandas Supported Plot Types

- Pass string to `kind='plot type here'` to tell pandas which plot to make
- Supported plot types you will find useful:
  - Bar charts: 'bar' or 'barh'
  - Histograms: 'hist'
  - Scatter plots: 'scatter'
  - Pie charts: 'pie'