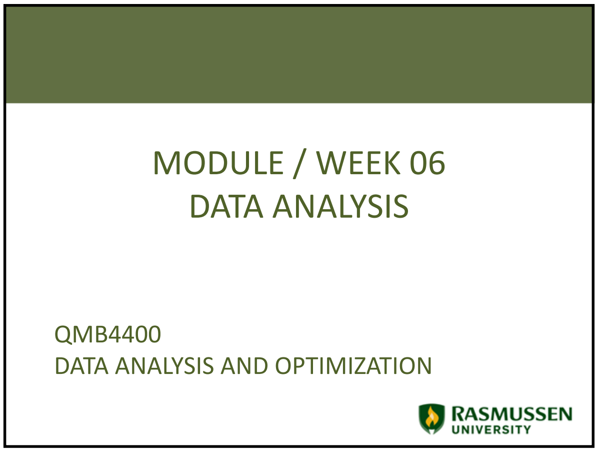
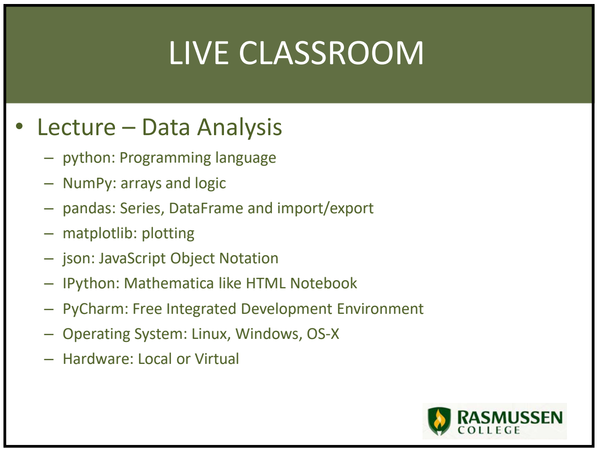


MODULE / WEEK 06
DATA ANALYSIS

QMB4400
DATA ANALYSIS AND OPTIMIZATION

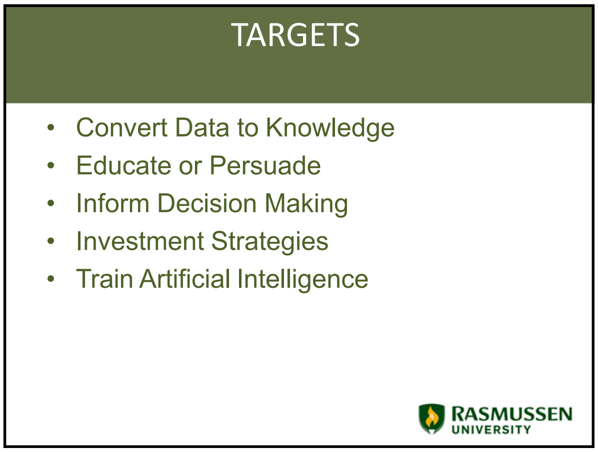
1



LIVE CLASSROOM

- Lecture – Data Analysis
  - python: Programming language
  - NumPy: arrays and logic
  - pandas: Series, DataFrame and import/export
  - matplotlib: plotting
  - json: JavaScript Object Notation
  - IPython: Mathematica like HTML Notebook
  - PyCharm: Free Integrated Development Environment
  - Operating System: Linux, Windows, OS-X
  - Hardware: Local or Virtual

2



TARGETS

- Convert Data to Knowledge
- Educate or Persuade
- Inform Decision Making
- Investment Strategies
- Train Artificial Intelligence

3

Series Example

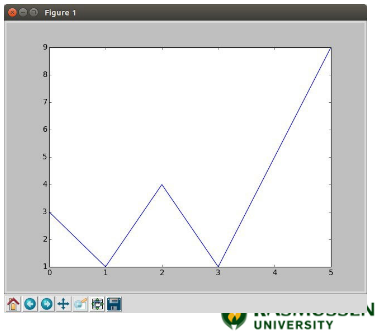
```
#!/usr/bin/env python

# use standard import conventions
import numpy as np
from pandas import DataFrame, Series
import pandas as pd
import matplotlib.pyplot as plt

def main():
      s1 = Series([ 3, 1, 4, 1, 5, 9 ])
      print s1
      print "s1 at index 2" + "%d" % s1[2]
      s2 = Series([ 3, 1, 4, 1, 5, 9 ], index =['a', 'b', 'c', 'd', 'e', 'f'])
      print s2
      print "s2 at index 'c'" + "%d" % s2['c']  plt.plot(s2)
      plt.show()
      return

if__name____== "_main_":  main()
```

RASMUSSEN UNIVERSITY

4

Hello World Output

```
0   3
1   1
2   4
3   1
4   5
5   9
dtype: int64
s1 at index 24
a   3
b.  1
c.  4
d.  1
e.  5
f.  9
dtype: int64
s2 at index 'c'4
```



RASMUSSEN UNIVERSITY

5

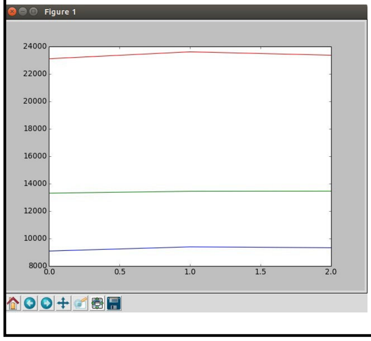
DataFrame Example

```
#!/usr/bin/env python
import numpy as np
from pandas import DataFrame, Series
import pandas as pd
import matplotlib.pyplot as plt

def dataframe_example():
    # live births for Wayne, Oakland, and Macomb counties 2012, 2013,
    2014  # source http://www.mdch.state.mi.us/
    data = {
        "Wayne":    [ 23109, 23612, 23366 ],
        "Oakland":  [ 13307, 13445, 13454 ],
        "Macomb":   [  9089,  9394,  9332 ]
    }
    dataframe = DataFrame(data, index = [ 2012, 2013, 2014 ] )

    print dataframe
    plt.plot(dataframe)
    plt.show()
    return

if___name__== "_main_":
```

RASMUSSEN UNIVERSITY

6

dataframe_example Output



```
      Macomb Oakland Wayne
2012    9089   13307  23109
2013    9394   13445  23612
2014    9332   13454  23366

[3 rows x 3 columns]
```

RASMUSSEN UNIVERSITY

7

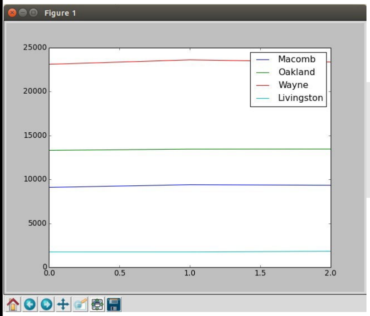
DataFrame: Adding a Column & Legend

```
#!/usr/bin/env python
import numpy as np
from pandas import DataFrame, Series
import pandas as pd
import matplotlib.pyplot as plt

def dataframe_example():
    # live births for Wayne, Oakland, and Macomb counties 2012, 2013,
    2014  # source http://www.mdch.state.mi.us/
    data = {
        "Wayne":    [ 23109, 23612, 23366 ],
        "Oakland":  [ 13307, 13445, 13454 ],
        "Macomb":   [  9089,  9394,  9332 ]
    }
    dataframe = DataFrame(data, index = [ 2012, 2013, 2014 ] )

    dataframe["Livingston"] = [1739,1738,1813]
    print dataframe
    plt.plot(dataframe)
    plt.legend(dataframe.keys())
    plt.show()
    return

if __name__== "_main__":
    dataframe_example()
```

RASMUSSEN UNIVERSITY

8

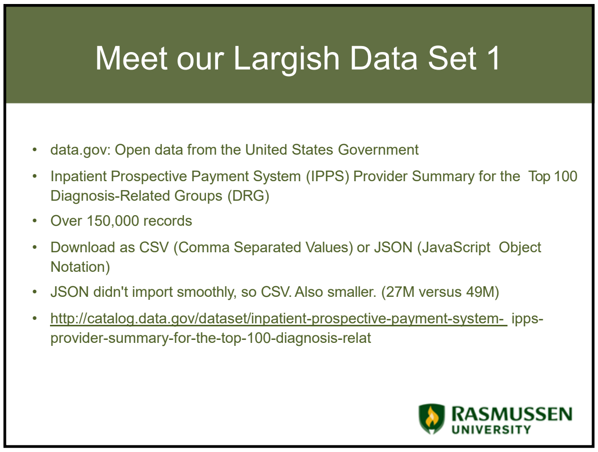
dataframe_example Output



```
Macomb Oakland Wayne  Livingston
2012    9089    13307  23109    1739
2013    9394    13445  23612    1738
2014    9332    13454  23366    1813

[3 rows x 4 columns]
```

RASMUSSEN UNIVERSITY

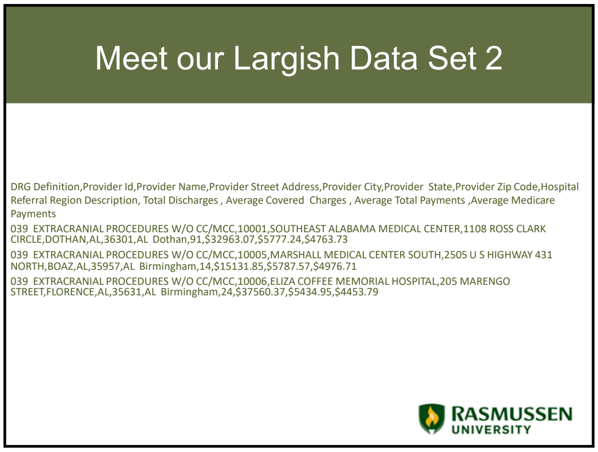
9



Meet our Largish Data Set 1

- data.gov: Open data from the United States Government
- Inpatient Prospective Payment System (IPPS) Provider Summary for the Top 100 Diagnosis-Related Groups (DRG)
- Over 150,000 records
- Download as CSV (Comma Separated Values) or JSON (JavaScript Object Notation)
- JSON didn't import smoothly, so CSV. Also smaller. (27M versus 49M)
- http://catalog.data.gov/dataset/inpatient-prospective-payment-system- ipps-provider-summary-for-the-top-100-diagnosis-relat

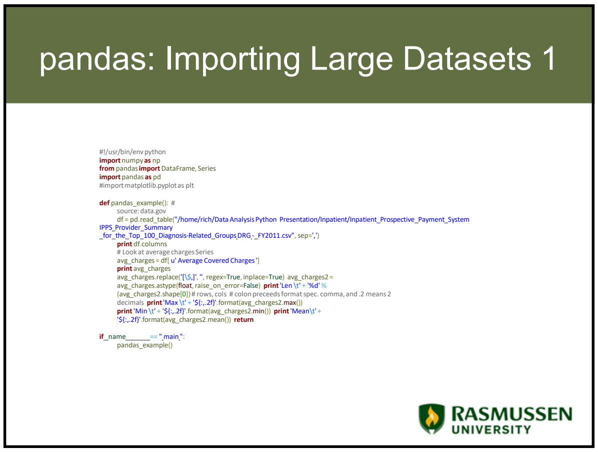
10



Meet our Largish Data Set 2

```
DRG Definition,Provider Id,Provider Name,Provider Street Address,Provider City,Provider  State,Provider Zip Code,Hospital
Referral Region Description, Total Discharges , Average Covered  Charges , Average Total Payments ,Average Medicare
Payments
039  EXTRACRANIAL PROCEDURES W/O CC/MCC,10001,SOUTHEAST ALABAMA MEDICAL CENTER,1108 ROSS CLARK
CIRCLE,DOTHAN,AL,36301,AL  Dothan,91,$32963.07,$5777.24,$4763.73
039  EXTRACRANIAL PROCEDURES W/O CC/MCC,10005,MARSHALL MEDICAL CENTER SOUTH,2505 U S HIGHWAY 431
NORTH,BOAZ,AL,35957,AL  Birmingham,14,$15131.85,$5787.57,$4976.71
039  EXTRACRANIAL PROCEDURES W/O CC/MCC,10006,ELIZA COFFEE MEMORIAL HOSPITAL,205 MARENGO
STREET,FLORENCE,AL,35631,AL  Birmingham,24,$37560.37,$5434.95,$4453.79
```

11



pandas: Importing Large Datasets 1

```
#!/usr/bin/env python
import numpy as np
from pandas import DataFrame, Series
import pandas as pd
#import matplotlib.pyplot as plt

def pandas_example():  #
        source: data.gov
        df = pd.read_table("/home/rich/Data Analysis Python  Presentation/Inpatient/Inpatient_Prospective_Payment_System
IPPS_Provider_Summary
_for_the_Top_100_Diagnosis-Related_Groups DRG -_FY2011.csv", sep=',')
        print df.columns
        # Look at average charges Series
        avg_charges = df[ u' Average Covered Charges ']
        print avg_charges
        avg_charges.replace('[\$,]', '', regex=True, inplace=True)  avg_charges2 =
        avg_charges.astype(float, raise_on_error=False)  print 'Len \t' + '%d' %
        (avg_charges2.shape[0]) # rows, cols  # colon preceeds format spec. comma, and .2 means 2
        decimals  print 'Max \t' + '${:,.2f}'.format(avg_charges2.max())
        print 'Min \t' + '${:,.2f}'.format(avg_charges2.min())  print 'Mean\t' +
        '${:,.2f}'.format(avg_charges2.mean())  return

if_name_______== "_main_":
        pandas_example()
```

12

pandas: Importing Large Datasets 2

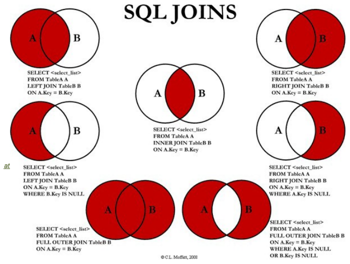
```
Index([u'DRG Definition', u'Provider Id', u'Provider Name',
       u'Provider Street Address', u'Provider City', u'Provider State',
       u'Provider Zip Code', u'Hospital Referral Region Description',
       u' Total Discharges ', u' Average Covered Charges ',
       u' Average Total Payments ', u'Average Medicare Payments'],
      dtype='object')
0         $32963.07
1         $15131.85
            ...

163063    $28873.09
163064    $15042.00
Name:  Average Covered Charges , dtype:
                                 object
Len     163065
Max     $929,118.90
Min     $2,459.40
Mean    $36,133.95
```

13

Merging Data

- combine data sets by linking rows
- many to one merge
- overlapping column names are used as keys
- inner join by default

diagram source: http://www.codeproject.com/Articles/33052/Visual-Represent at



```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

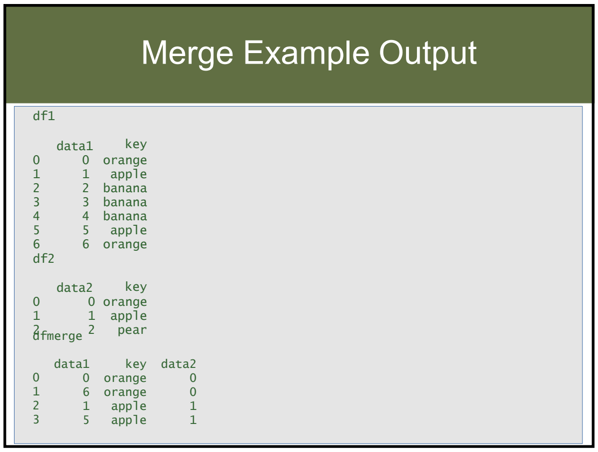
© C.L. Moffatt, 2008

14

Merge Example

```
#!/usr/bin/env python
import numpy as np
from pandas import DataFrame, Series
import pandas as pd

df1 = DataFrame({ 'key': [ 'orange', 'apple', 'bannana', 'banana', 'banana',  'apple', 'orange' ],
                  'data1' : range(7)})

df2 = DataFrame({         'key'    : [ 'orange', 'apple', 'pear'],
                          'data2'     : range(3) })

print     'df1'
print     '-------'
print df1
print 'df2'  print '----
---'  print df2

dfmerge = pd.merge(df1,df2, on='key')

print         'dfmerge'
print  '-------'    print
dfmerge
```

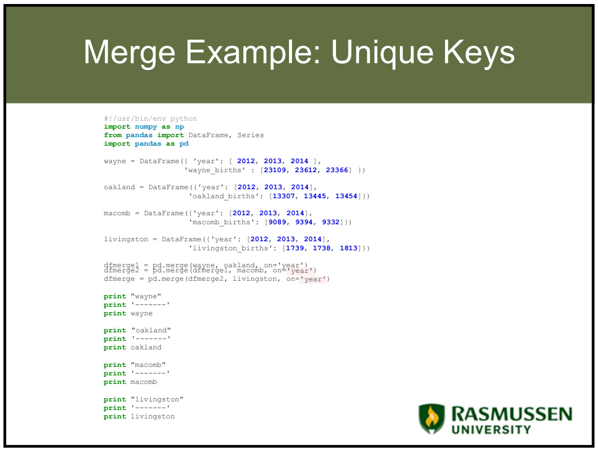
15



Merge Example Output

```
df1

   data1     key
0      0  orange
1      1   apple
2      2  banana
3      3  banana
4      4  banana
5      5   apple
6      6  orange
df2

   data2     key
0      0  orange
1      1   apple
2      2    pear
dfmerge

   data1     key  data2
0      0  orange      0
1      6  orange      0
2      1   apple      1
3      5   apple      1
```

16

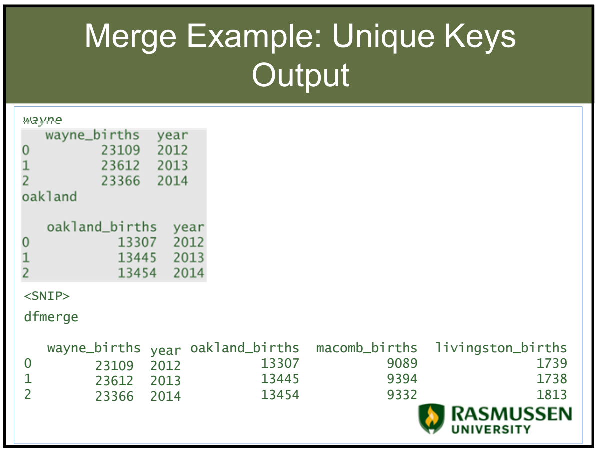


Merge Example: Unique Keys

```
#!/usr/bin/env python
import numpy as np
from pandas import DataFrame, Series
import pandas as pd

wayne = DataFrame({ 'year': [ 2012, 2013, 2014 ],
                  'wayne_births' : [23109, 23612, 23366] })

oakland = DataFrame({'year': [2012, 2013, 2014],
                   'oakland_births': [13307, 13445, 13454]})

macomb = DataFrame({'year': [2012, 2013, 2014],
                   'macomb_births': [9089, 9394, 9332]})

livingston = DataFrame({'year': [2012, 2013, 2014],
                   'livingston_births': [1739, 1738, 1813]})

dfmerge1 = pd.merge(wayne, oakland, on='year')
dfmerge2 = pd.merge(dfmerge1, macomb, on='year')
dfmerge = pd.merge(dfmerge2, livingston, on='year')

print "wayne"
print '-------'
print wayne

print "oakland"
print '-------'
print oakland

print "macomb"
print '-------'
print macomb

print "livingston"
print '-------'
print livingston
```

RASMUSSEN UNIVERSITY

17



Merge Example: Unique Keys Output

```
wayne
   wayne_births  year
0         23109  2012
1         23612  2013
2         23366  2014
oakland

   oakland_births  year
0           13307  2012
1           13445  2013
2           13454  2014
<SNIP>
dfmerge

   wayne_births  year  oakland_births  macomb_births  livingston_births
0         23109  2012           13307           9089               1739
1         23612  2013           13445           9394               1738
2         23366  2014           13454           9332               1813
```

RASMUSSEN UNIVERSITY

18

Understanding GroupBy

- *Split-Apply-Combine*
- *Split* data into groups based on keys (Provider Name, Provider State, Procedure Name, ..)
- *Apply* A function is applied to each group (e.g. average, sum, count)
- *Combine* The results of the "apply" functions are combined to form a new object.

19

Merge Example: Average Covered Costs

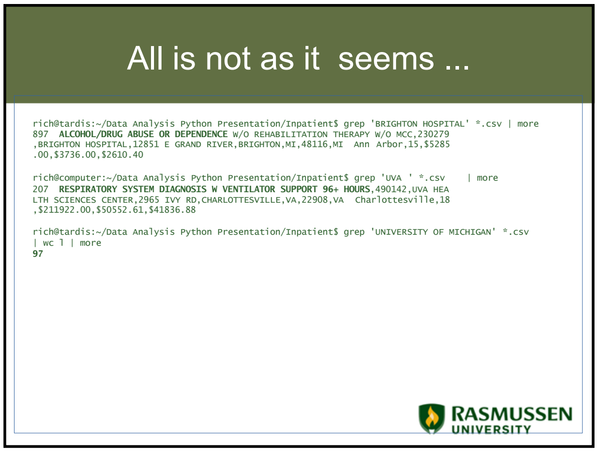
```
#!/usr/bin/env python
import numpy as np
from pandas import DataFrame, Series
import pandas as pd

def convert_acc(value):
    v2 = value.replace('$','')
    f = float(v2)
    return f

def pandas_example2():
    # source: data.gov
    df = pd.read_table("/home/rich/Data Analysis Python
Presentation/Inpatient/Inpatient Prospective Payment System IPPS_Provider_Summary
_for_the_Top_100_Diagnosis-Related_Groups_DRG_-_FY2011.csv",
                        sep=',',
                        converters= { u' Average Covered Charges ': convert_acc } )

    grouped = df[u' Average Covered Charges '].groupby([df[u'Provider Id'],
df[u'Provider Name']])
    means = grouped.mean()
    print means
    meansdf = DataFrame(means)
    print meansdf.sort(u' Average Covered Charges ')
    pass

if___name___== "_main__":
    pandas_example2()
```

20

Average Covered Costs Output

```
                                                     Average Covered Charges
Provider Id Provider Name
450813      COMMUNITY GENERAL HOSPITAL                            2995.610000
250079      SHARKEY ISSAQUENA COMMUNITY HOSPITAL                  3369.955000
450746      KNOX COUNTY HOSPITAL                                  3677.000000
110209      TURNING POINT HOSPITAL                                3720.430000
450270      LAKE WHITNEY MEDICAL CENTER                           3906.842727
190161      W O MOSS REGIONAL MEDICAL CENTER                      4059.250000
390025      KENSINGTON HOSPITAL                                   4108.750000
220062      ADCARE HOSPITAL OF WORCESTER INC                      4227.460000
190208      EAST CARROLL PARISH HOSPITAL                          4318.224444
<SNIP>
230279      BRIGHTON HOSPITAL                                     5285.000000
360247      WOODS AT PARKSIDE,THE                                 5384.680000
10097       ELMORE COMMUNITY HOSPITAL                             5404.585556
...                                                                       ...
50197       SEQUOIA HOSPITAL                                     99682.389216
50153       O'CONNOR HOSPITAL                                    99812.639589
50002       ST ROSE HOSPITAL                                    100844.518519
50380       GOOD SAMARITAN HOSPITAL                             101206.971111
50742       OLYMPIA MEDICAL CENTER                              102538.674091
<SNIP>
50367       NORTHBAY MEDICAL CENTER                             138504.546230
50441       STANFORD HOSPITAL                                   138818.649770
50464       DOCTORS MEDICAL CENTER                              144695.833286
310025      BAYONNE HOSPITAL CENTER                             147441.334000
490142      UVA HEALTH SCIENCES CENTER                          211922.000000

[3337 rows x 1 columns]
```

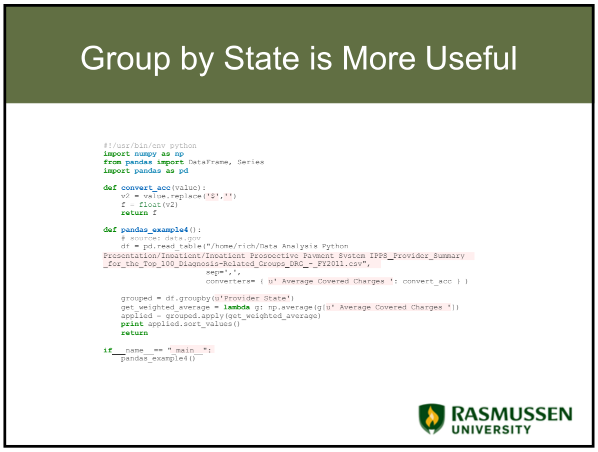
21



All is not as it seems ...

```
rich@tardis:~/Data Analysis Python Presentation/Inpatient$ grep 'BRIGHTON HOSPITAL' *.csv | more
897  ALCOHOL/DRUG ABUSE OR DEPENDENCE W/O REHABILITATION THERAPY W/O MCC,230279
,BRIGHTON HOSPITAL,12851 E GRAND RIVER,BRIGHTON,MI,48116,MI  Ann Arbor,15,$5285
.00,$3736.00,$2610.40

rich@computer:~/Data Analysis Python Presentation/Inpatient$ grep 'UVA ' *.csv    | more
207  RESPIRATORY SYSTEM DIAGNOSIS W VENTILATOR SUPPORT 96+ HOURS,490142,UVA HEA
LTH SCIENCES CENTER,2965 IVY RD,CHARLOTTESVILLE,VA,22908,VA  Charlottesville,18
,$211922.00,$50552.61,$41836.88

rich@tardis:~/Data Analysis Python Presentation/Inpatient$ grep 'UNIVERSITY OF MICHIGAN' *.csv
| wc l | more
97
```

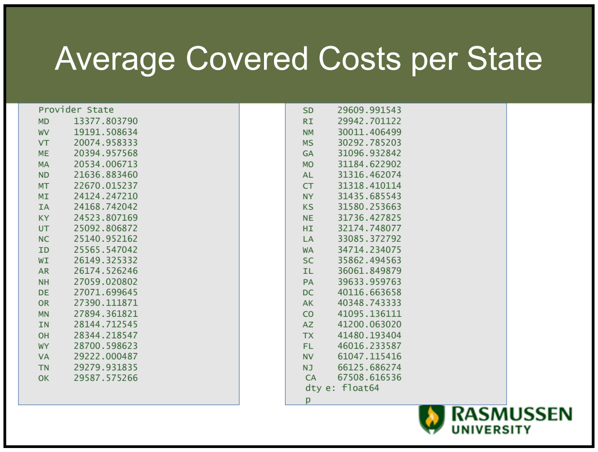
22



Group by State is More Useful

```
#!/usr/bin/env python
import numpy as np
from pandas import DataFrame, Series
import pandas as pd

def convert_acc(value):
    v2 = value.replace('$','')
    f = float(v2)
    return f

def pandas_example4():
    # source: data.gov
    df = pd.read_table("/home/rich/Data Analysis Python
Presentation/Inpatient/Inpatient Prospective Payment System IPPS_Provider_Summary
_for_the_Top_100_Diagnosis-Related_Groups_DRG_-_FY2011.csv",
                        sep=',',
                        converters= { u' Average Covered Charges ': convert_acc } )

    grouped = df.groupby(u'Provider State')
    get_weighted_average = lambda g: np.average(g[u' Average Covered Charges '])
    applied = grouped.apply(get_weighted_average)
    print applied.sort_values()
    return

if___name__== "_main__":
    pandas_example4()
```

23



Average Covered Costs per State

```
Provider State
MD    13377.803790
WV    19191.508634
VT    20074.958333
ME    20394.957568
MA    20534.006713
ND    21636.883460
MT    22670.015237
MI    24124.247210
IA    24168.742042
KY    24523.807169
UT    25092.806872
NC    25140.952162
ID    25565.547042
WI    26149.325332
AR    26174.526246
NH    27059.020802
DE    27071.699645
OR    27390.111871
MN    27894.361821
IN    28144.712545
OH    28344.218547
WY    28700.598623
VA    29222.000487
TN    29279.931835
OK    29587.575266
```

```
SD    29609.991543
RI    29942.701122
NM    30011.406499
MS    30292.785203
GA    31096.932842
MO    31184.622902
AL    31316.462074
CT    31318.410114
NY    31435.685543
KS    31580.253663
NE    31736.427825
HI    32174.748077
LA    33085.372792
WA    34714.234075
SC    35862.494563
IL    36061.849879
PA    39633.959763
DC    40116.663658
AK    40348.743333
CO    41095.136111
AZ    41200.063020
TX    41480.193404
FL    46016.233587
NV    61047.115416
NJ    66125.686274
 CA   67508.616536
 dty e: float64
 p
```

24

Some thoughts

- Where can I get inexpensive and quality medical care? Which states could benefit from promoting medical tourism?
- Which states have higher costs of living? Are procedures less expensive in low cost of living regions?
- Which states have higher average income? Do those states have a greater proportion of expensive procedure types?
- Which states have favorable or unfavorable regulatory environments? (LA Lottery)

25

Bucket and Quantile Analysis

```
#!/usr/bin/env python
import numpy as np
from pandas import DataFrame, Series
import pandas as pd

grades = np.random.randint(100, size=33)
print grades

frame = DataFrame({'grades' : grades } )  factor =

pd.cut(frame.grades, 4 )

print factor
```

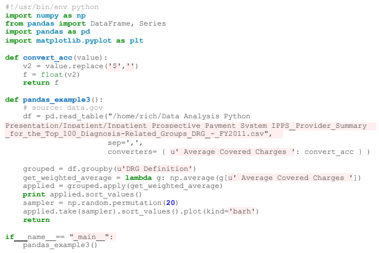
26

Bucket and Quantile Output

```
[34 37 19 79 90  7 58  5 77  9 88 18 10  0 89 16 58 59  0 89 27  5  6 71  3
 10 48 73 21 13 10 84 28]
0        (22.5, 45]
1        (22.5, 45]
2       (0.09, 22.5]
3         (67.5, 90]
4         (67.5, 90]
<SNIP>
28      (0.09, 22.5]
29      (0.09, 22.5]
30      (0.09, 22.5]
31        (67.5, 90]
32        (22.5, 45]
Name: grades, dtype: category
Categories (4, object): [(0.09, 22.5] < (22.5, 45] < (45, 67.5] < (67.5, 90]]
```

27

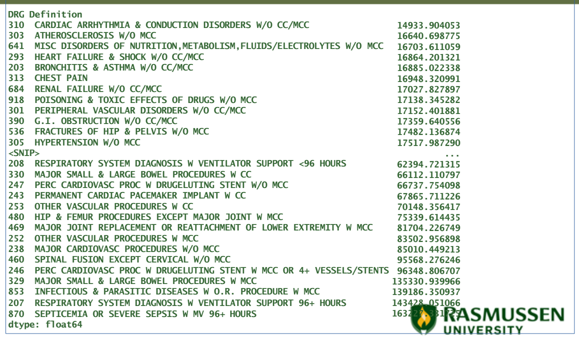
Grouping by Procedure



```
#!/usr/bin/env python
import numpy as np
from pandas import DataFrame, Series
import pandas as pd
import matplotlib.pyplot as plt

def convert_acc(value):
    v2 = value.replace('$','')
    f = float(v2)
    return f

def pandas_example3():
    # source: data.gov
    df = pd.read_table("/home/rich/Data Analysis Python
Presentation/Inpatient/Inpatient Prospective Payment System IPPS_Provider_Summary
_for_the_Top_100_Diagnosis-Related_Groups_DRG_-_FY2011.csv",
                       sep=',',
                       converters= { u' Average Covered Charges ': convert_acc } )

    grouped = df.groupby(u'DRG Definition')
    get_weighted_average = lambda g: np.average(g[u' Average Covered Charges '])
    applied = grouped.apply(get_weighted_average)
    print applied.sort_values()
    sampler = np.random.permutation(20)
    applied.take(sampler).sort_values().plot(kind='barh')
    return

if___name__== "_main__":
    pandas_example3()
```

28

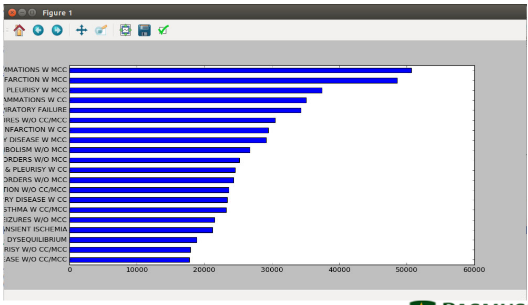
Group by Procedure Output



```
DRG Definition
310  CARDIAC ARRHYTHMIA & CONDUCTION DISORDERS W/O CC/MCC                  14933.904053
303  ATHEROSCLEROSIS W/O MCC                                               16640.698775
641  MISC DISORDERS OF NUTRITION,METABOLISM,FLUIDS/ELECTROLYTES W/O MCC    16703.611059
293  HEART FAILURE & SHOCK W/O CC/MCC                                      16864.201321
203  BRONCHITIS & ASTHMA W/O CC/MCC                                        16885.022338
313  CHEST PAIN                                                            16948.320991
684  RENAL FAILURE W/O CC/MCC                                              17027.827897
918  POISONING & TOXIC EFFECTS OF DRUGS W/O MCC                            17138.345282
301  PERIPHERAL VASCULAR DISORDERS W/O CC/MCC                              17152.401881
390  G.I. OBSTRUCTION W/O CC/MCC                                           17359.640556
536  FRACTURES OF HIP & PELVIS W/O MCC                                     17482.136874
305  HYPERTENSION W/O MCC                                                  17517.987290
<SNIP>                                                                              ...
208  RESPIRATORY SYSTEM DIAGNOSIS W VENTILATOR SUPPORT <96 HOURS           62394.721315
330  MAJOR SMALL & LARGE BOWEL PROCEDURES W CC                             66112.110797
247  PERC CARDIOVASC PROC W DRUGELUTING STENT W/O MCC                      66737.754098
243  PERMANENT CARDIAC PACEMAKER IMPLANT W CC                              67865.711226
253  OTHER VASCULAR PROCEDURES W CC                                        70148.356417
480  HIP & FEMUR PROCEDURES EXCEPT MAJOR JOINT W MCC                       75339.614435
469  MAJOR JOINT REPLACEMENT OR REATTACHMENT OF LOWER EXTREMITY W MCC      81704.226749
252  OTHER VASCULAR PROCEDURES W MCC                                       83502.956898
238  MAJOR CARDIOVASC PROCEDURES W/O MCC                                   85010.449213
460  SPINAL FUSION EXCEPT CERVICAL W/O MCC                                 95568.276246
246  PERC CARDIOVASC PROC W DRUGELUTING STENT W MCC OR 4+ VESSELS/STENTS  96348.806707
329  MAJOR SMALL & LARGE BOWEL PROCEDURES W MCC                           135330.939966
853  INFECTIOUS & PARASITIC DISEASES W O.R. PROCEDURE W MCC               139186.350937
207  RESPIRATORY SYSTEM DIAGNOSIS W VENTILATOR SUPPORT 96+ HOURS          143428.051066
870  SEPTICEMIA OR SEVERE SEPSIS W MV 96+ HOURS                           163227.331725
dtype: float64
```

29

And the Bar Chart



30

Wakario.io: iPython Online

Free with some limitations

Can install iPython locally instead if you

like: http://ipython.org/

https://www.pythonanywhere.com/try-ipython/

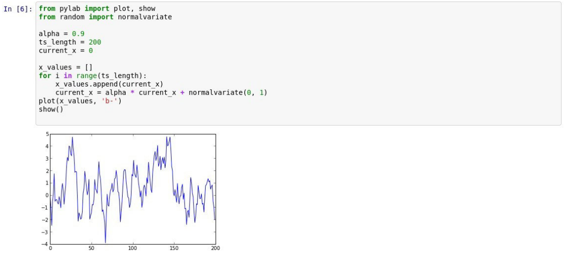
Like iPython, makes something like an academic paper

A little confusing, when you open and close you'll need to manually re-run prior values to set variables for later values

Excellent way to try data analysis in Python

31

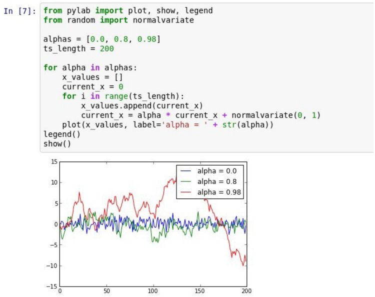
Wakari.io: Simple example



```
In [6]: from pylab import plot, show
        from random import normalvariate

        alpha = 0.9
        ts_length = 200
        current_x = 0

        x_values = []
        for i in range(ts_length):
            x_values.append(current_x)
            current_x = alpha * current_x + normalvariate(0, 1)
        plot(x_values, 'b-')
        show()
```

32

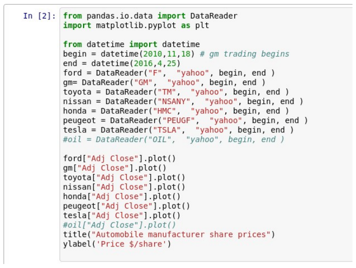
Wakari.io: A littlebit fancier now



```
In [7]: from pylab import plot, show, legend
        from random import normalvariate

        alphas = [0.0, 0.8, 0.98]
        ts_length = 200

        for alpha in alphas:
            x_values = []
            current_x = 0
            for i in range(ts_length):
                x_values.append(current_x)
                current_x = alpha * current_x + normalvariate(0, 1)
            plot(x_values, label='alpha = ' + str(alpha))
        legend()
        show()
```

33

Examining the Stocks of US Auto Manufacturers

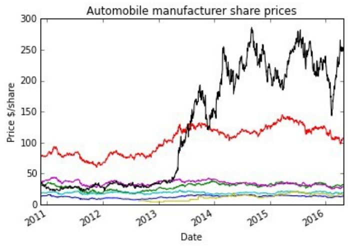


```
In [2]: from pandas.io.data import DataReader
        import matplotlib.pyplot as plt

        from datetime import datetime
        begin = datetime(2010,11,18) # gm trading begins
        end = datetime(2016,4,25)
        ford = DataReader("F",  "yahoo", begin, end )
        gm= DataReader("GM",  "yahoo", begin, end )
        toyota = DataReader("TM",  "yahoo", begin, end )
        nissan = DataReader("NSANY",  "yahoo", begin, end )
        honda = DataReader("HMC",  "yahoo", begin, end )
        peugeot = DataReader("PEUGF",  "yahoo", begin, end )
        tesla = DataReader("TSLA",  "yahoo", begin, end )
        #oil = DataReader("OIL",  "yahoo", begin, end )

        ford["Adj Close"].plot()
        gm["Adj Close"].plot()
        toyota["Adj Close"].plot()
        nissan["Adj Close"].plot()
        honda["Adj Close"].plot()
        peugeot["Adj Close"].plot()
        tesla["Adj Close"].plot()
        #oil["Adj Close"].plot()
        title("Automobile manufacturer share prices")
        ylabel('Price $/share')
```

34

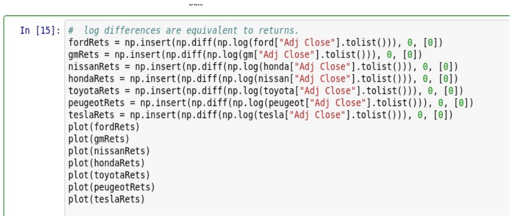
Examining the Stocks of US Auto Manufacturers

```
Out[2]: <matplotlib.text.Text at 0x7fa5b943d8d0>
```



35

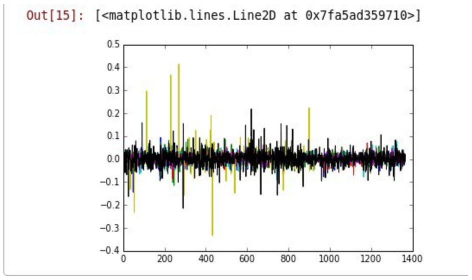
Log of differences are equivalent to stock returns



```
In [15]: #  log differences are equivalent to returns.
         fordRets = np.insert(np.diff(np.log(ford["Adj Close"].tolist())), 0, [0])
         gmRets = np.insert(np.diff(np.log(gm["Adj Close"].tolist())), 0, [0])
         nissanRets = np.insert(np.diff(np.log(honda["Adj Close"].tolist())), 0, [0])
         hondaRets = np.insert(np.diff(np.log(nissan["Adj Close"].tolist())), 0, [0])
         toyotaRets = np.insert(np.diff(np.log(toyota["Adj Close"].tolist())), 0, [0])
         peugeotRets = np.insert(np.diff(np.log(peugeot["Adj Close"].tolist())), 0, [0])
         teslaRets = np.insert(np.diff(np.log(tesla["Adj Close"].tolist())), 0, [0])
         plot(fordRets)
         plot(gmRets)
         plot(nissanRets)
         plot(hondaRets)
         plot(toyotaRets)
         plot(peugeotRets)
         plot(teslaRets)
```

36

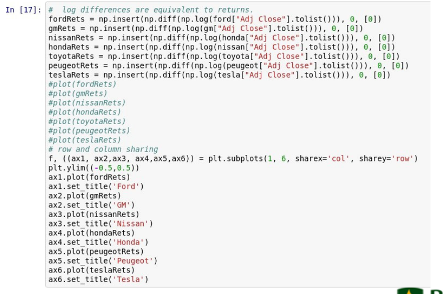
A very useful plot



```
Out[15]: [<matplotlib.lines.Line2D at 0x7fa5ad359710>]
```

37

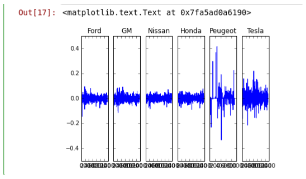
Subplots: Help us Fogg Nelson!



```
In [17]: #  log differences are equivalent to returns.
         fordRets = np.insert(np.diff(np.log(ford["Adj Close"].tolist())), 0, [0])
         gmRets = np.insert(np.diff(np.log(gm["Adj Close"].tolist())), 0, [0])
         nissanRets = np.insert(np.diff(np.log(honda["Adj Close"].tolist())), 0, [0])
         hondaRets = np.insert(np.diff(np.log(nissan["Adj Close"].tolist())), 0, [0])
         toyotaRets = np.insert(np.diff(np.log(toyota["Adj Close"].tolist())), 0, [0])
         peugeotRets = np.insert(np.diff(np.log(peugeot["Adj Close"].tolist())), 0, [0])
         teslaRets = np.insert(np.diff(np.log(tesla["Adj Close"].tolist())), 0, [0])
         #plot(fordRets)
         #plot(gmRets)
         #plot(nissanRets)
         #plot(hondaRets)
         #plot(toyotaRets)
         #plot(peugeotRets)
         #plot(teslaRets)
         # row and column sharing
         f, ((ax1, ax2,ax3, ax4,ax5,ax6)) = plt.subplots(1, 6, sharex='col', sharey='row')
         plt.ylim((-0.5,0.5))
         ax1.plot(fordRets)
         ax1.set_title('Ford')
         ax2.plot(gmRets)
         ax2.set_title('GM')
         ax3.plot(nissanRets)
         ax3.set_title('Nissan')
         ax4.plot(hondaRets)
         ax4.set_title('Honda')
         ax5.plot(peugeotRets)
         ax5.set_title('Peugeot')
         ax6.plot(teslaRets)
         ax6.set_title('Tesla')
```

38

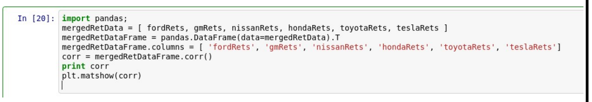
Daily gains or losses per US auto manufacturer stock



```
Out[17]: <matplotlib.text.Text at 0x7fa5ad0a6190>
```

39

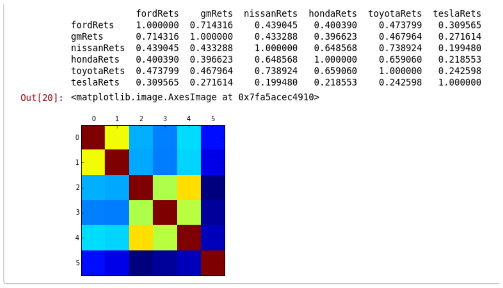
Should I just invest in them all?
Correlation coefficients and the ideal investment



```
In [20]: import pandas;
         mergedRetData = [ fordRets, gmRets, nissanRets, hondaRets, toyotaRets, teslaRets ]
         mergedRetDataFrame = pandas.DataFrame(data=mergedRetData).T
         mergedRetDataFrame.columns = [ 'fordRets', 'gmRets', 'nissanRets', 'hondaRets', 'toyotaRets', 'teslaRets']
         corr = mergedRetDataFrame.corr()
         print corr
         plt.matshow(corr)
```

40

Conclusion: Tesla is not an auto company



```
            fordRets    gmRets  nissanRets  hondaRets  toyotaRets  teslaRets
fordRets    1.000000  0.714316    0.439045   0.400390    0.473799   0.309565
gmRets      0.714316  1.000000    0.433288   0.396623    0.467964   0.271614
nissanRets  0.439045  0.433288    1.000000   0.648568    0.738924   0.199480
hondaRets   0.400390  0.396623    0.648568   1.000000    0.659060   0.218553
toyotaRets  0.473799  0.467964    0.738924   0.659060    1.000000   0.242598
teslaRets   0.309565  0.271614    0.199480   0.218553    0.242598   1.000000
Out[20]: <matplotlib.image.AxesImage at 0x7fa5acec4910>
```

41

MODULE 06 HELPS

- https://www.geeksforgeeks.org/python-introduction-matplotlib/
- https://realpython.com/python-matplotlib-guide/
- https://www.datacamp.com/community/tutorials/matplotlib-tutorial-python
- https://python-graph-gallery.com/matplotlib/

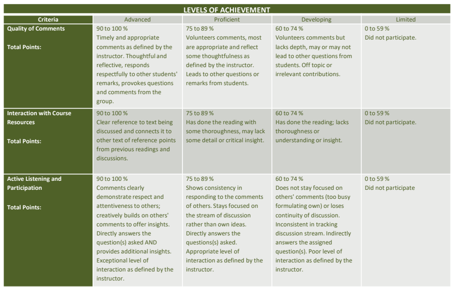
42

Wrapping it up ...

- Python has powerful data analysis tools
- Use them in Pycharm (or any IDE) or iPython (Vim also works)
- Analyzing data can help us make more informed decisions
- Libraries make most things easy
- Thank you!

43

Module 06
DISCUSSION FORUM



LEVELS OF ACHIEVEMENT				
Criteria	Advanced	Proficient	Developing	Limited
Quality of Comments **Total Points:**	90 to 100 % Timely and appropriate comments as defined by the instructor. Thoughtful and reflective, responds respectfully to other students' remarks, provokes questions and comments from the group.	75 to 89 % Volunteers comments, most are appropriate and reflect some thoughtfulness as defined by the instructor. Leads to other questions or remarks from students.	60 to 74 % Volunteers comments but lacks depth, may or may not lead to other questions from students. Off topic or irrelevant contributions.	0 to 59 % Did not participate.
Interaction with Course Resources **Total Points:**	90 to 100 % Clear reference to text being discussed and connects it to other text of reference points from previous readings and discussions.	75 to 89 % Has done the reading with some thoroughness, may lack some detail or critical insight.	60 to 74 % Has done the reading; lacks thoroughness or understanding or insight.	0 to 59 % Did not participate.
Active Listening and Participation **Total Points:**	90 to 100 % Comments clearly demonstrate respect and attentiveness to others; creatively builds on others' comments to offer insights. Directly answers the question(s) asked AND provides additional insights. Exceptional level of interaction as defined by the instructor.	75 to 89 % Shows consistency in responding to the comments of others. Stays focused on the stream of discussion rather than own ideas. Directly answers the questions(s) asked. Appropriate level of interaction as defined by the instructor.	60 to 74 % Does not stay focused on others' comments (too busy formulating own) or loses continuity of discussion. Inconsistent in tracking discussion stream. Indirectly answers the assigned question(s). Poor level of interaction as defined by the instructor.	0 to 59 % Did not participate

44

MODULE 06
DISCUSSION FORUM

- You are working as analytics developer in fortune 500 company. Your company wants to you to use Bag-Of-Words model.
- For this discussion, describe and explain the following in your initial post. Your initial post should be a minimum of at least two fully-formed, well-thought-out scholarly paragraphs (blocks of code examples and graphics are considered additional information beyond the two required paragraphs but are encouraged when appropriate).
- Describe at least one way to use Bag-Of-Words Model.
- What are the advantages and limitation of the Bag-Of-Words model?

45

MODULE 06 DISCUSSION FORUM

For your reply, choose two other student responses and provide additional insights to each of them that add value to their posting. The reply should be at least one fully-formed, well-thought-out scholarly paragraph (blocks of code examples are considered additional information beyond the required paragraph but are encouraged when appropriate). A simple "I agree" type of post is unacceptable.

Due dates for your initial and response posts can be found by checking the Course Syllabus and Course Calendar.

46

MODULE 06 COURSE PROJECT: GRADING RUBRIC

Criteria	Points
A correct Python script is attached.	50
Output screenshots are attached.	50
Total	100

47

MODULE 06 COURSE PROJECT FINAL SUBMISSION

You must submit a final copy of your course project and all the contents.

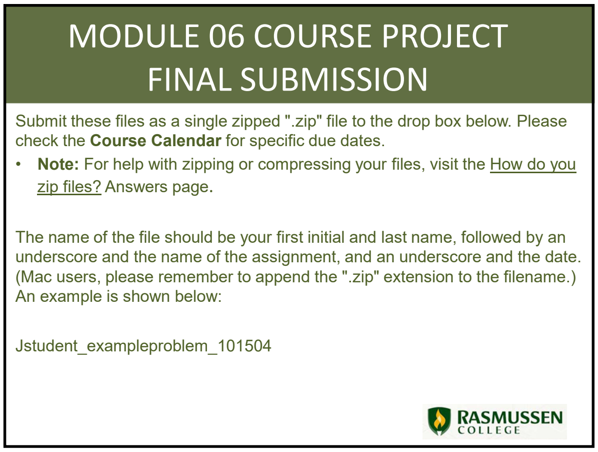
Verify the requirements below before you submit the final project.

- Final Python script.
- Overall architecture diagram.
- Apply scripting standard.
- It must be original work and include comments throughout your code.

You need to submit following things as a part of week two submission.

- Entire scripting solution
- Output screenshot for each week

48



MODULE 06 COURSE PROJECT FINAL SUBMISSION

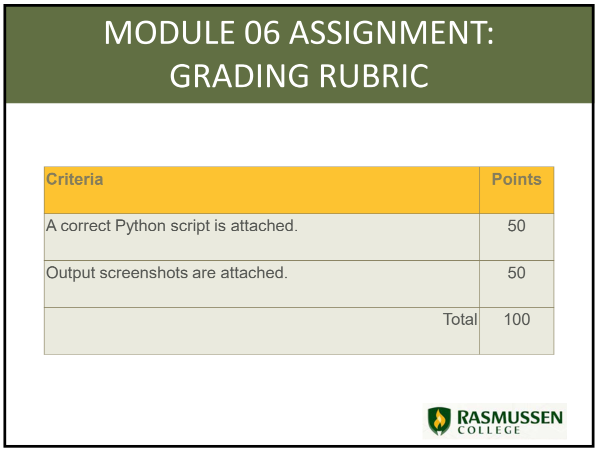
Submit these files as a single zipped ".zip" file to the drop box below. Please check the **Course Calendar** for specific due dates.

- **Note:** For help with zipping or compressing your files, visit the How do you zip files? Answers page.

The name of the file should be your first initial and last name, followed by an underscore and the name of the assignment, and an underscore and the date. (Mac users, please remember to append the ".zip" extension to the filename.) An example is shown below:

Jstudent_exampleproblem_101504

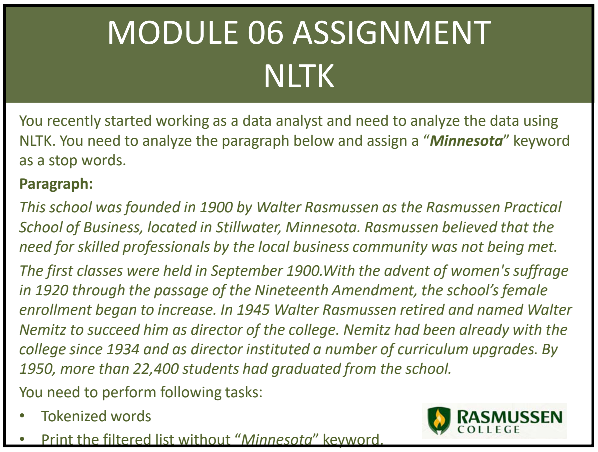
49



MODULE 06 ASSIGNMENT: GRADING RUBRIC

Criteria	Points
A correct Python script is attached.	50
Output screenshots are attached.	50
Total	100

50



MODULE 06 ASSIGNMENT NLTK

You recently started working as a data analyst and need to analyze the data using NLTK. You need to analyze the paragraph below and assign a "***Minnesota***" keyword as a stop words.

Paragraph:

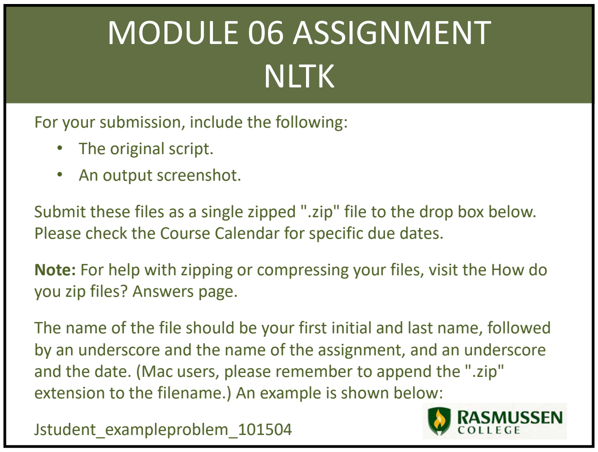
This school was founded in 1900 by Walter Rasmussen as the Rasmussen Practical School of Business, located in Stillwater, Minnesota. Rasmussen believed that the need for skilled professionals by the local business community was not being met.

The first classes were held in September 1900.With the advent of women's suffrage in 1920 through the passage of the Nineteenth Amendment, the school's female enrollment began to increase. In 1945 Walter Rasmussen retired and named Walter Nemitz to succeed him as director of the college. Nemitz had been already with the college since 1934 and as director instituted a number of curriculum upgrades. By 1950, more than 22,400 students had graduated from the school.

You need to perform following tasks:

- Tokenized words
- Print the filtered list without "*Minnesota*" keyword.

51



MODULE 06 ASSIGNMENT NLTK

For your submission, include the following:

- The original script.
- An output screenshot.

Submit these files as a single zipped ".zip" file to the drop box below. Please check the Course Calendar for specific due dates.

Note: For help with zipping or compressing your files, visit the How do you zip files? Answers page.

The name of the file should be your first initial and last name, followed by an underscore and the name of the assignment, and an underscore and the date. (Mac users, please remember to append the ".zip" extension to the filename.) An example is shown below:

Jstudent_exampleproblem_101504

RASMUSSEN COLLEGE

52