ICDSS
Imperial College Data Science Society

Python 101

Understanding the nuts and bolts of Python

Kane Wu
Email: kcw115@ic.ac.uk

Agenda

- Introduction to Data Analytics
- Python: An Overview
- Python Packages
- Basic Python Tutorial
- Reading Data into Python

1

Introduction to Data Analytics

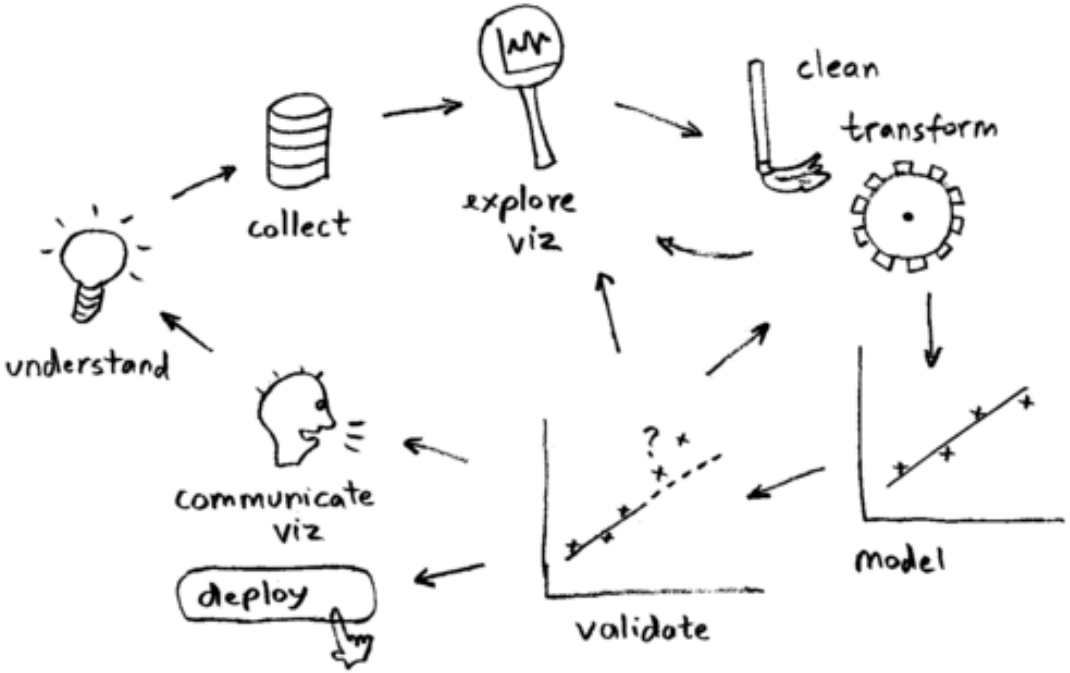
“Without big data analytics, companies are blind and deaf, wandering out onto the Web like deer on a freeway.”

Geoffrey Moore, Author of Crossing the Chasm & Inside the Tornado

“I never guess. It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts.”

Sir Arthur Conan Doyle, Author of Sherlock Holmes stories

Introduction to Data Analytics



http://datascience.la/wp-content/uploads/2014/09/data-science-workflow-szilard.png

Introduction to Data Analytics

Predictive analytics
What is likely to happen?

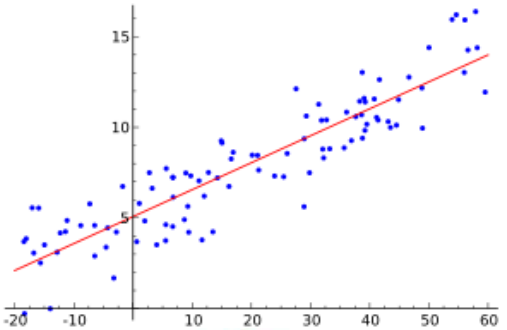
Diagnostic analytics
Why did it happen?

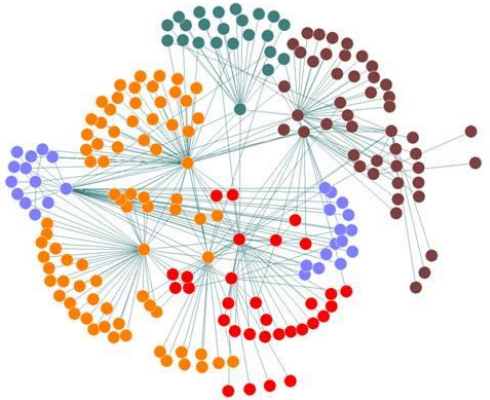
Descriptive analytics
what happened? where? when?

Complexity

Descriptive statistics include:

- Mean, mode and median values
- To understand the central tendency of the data
- Range and percentiles of the data
- To understand the distribution of the data
- Variance and standard deviations
- To understand the spread of the data
- Correlation coefficients
- To understand relationships between data or variables





Diagnostic and predictive analytics include:

- Regression analysis
- Pattern recognition
- Network analysis
- Cluster analysis

2

Python: An Overview

Python

General purpose high level programming language
Code Readability
Fewer Lines of Codes
Minimal Setup
Easy to Learn

Jan 2016	Jan 2015	Change	Programming Language	Ratings	Change
1	2	^	Java	21.465%	+5.94%
2	1	v	C	16.036%	-0.67%
3	4	^	C++	6.914%	+0.21%
4	5	^	C#	4.707%	-0.34%
5	8	^	Python	3.854%	+1.24%
6	6		PHP	2.706%	-1.08%
7	16	^^	Visual Basic .NET	2.582%	+1.51%
8	7	v	JavaScript	2.565%	-0.71%
9	14	^^	Assembly language	2.095%	+0.92%
10	15	^^	Ruby	2.047%	+0.92%
11	9	v	Perl	1.841%	-0.42%
12	20	^^	Delphi/Object Pascal	1.786%	+0.95%
13	17	^^	Visual Basic	1.684%	+0.61%
14	25	^^	Swift	1.363%	+0.62%

Tiobe Index for 2016

Python: An Overview

- What is special about python (not exhaustive)
  - Do not need to declare types of arguments or variables (dynamically typed)
  - No explicit begin or end, no curly braces to mark where the function code starts and stops. Uses indentation for code readability.
  - Uses : as a control character for loops and if statements
  - Uses the .py extension

Python: An Overview (Installation)

ANACONDA

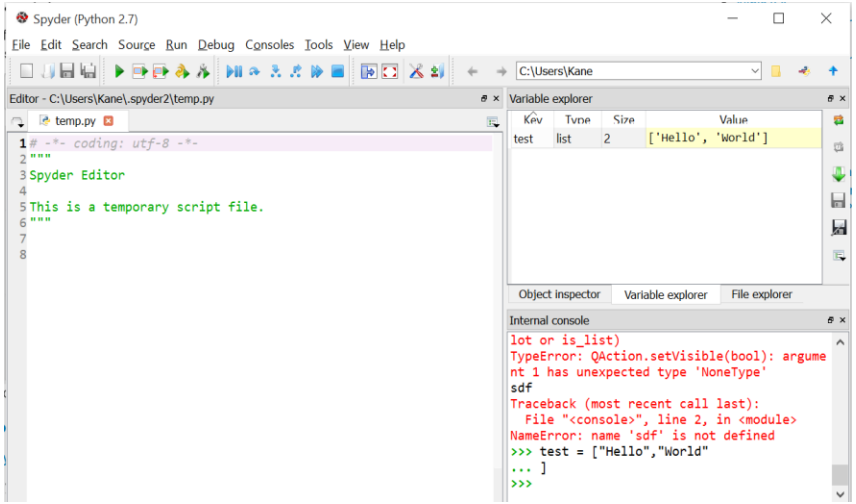
Anaconda for Windows

PYTHON 2.7	PYTHON 3.5
Windows 64-bit Graphical Installer 367M	Windows 64-bit Graphical Installer 392M
Windows 32-bit Graphical Installer 321M	Windows 32-bit Graphical Installer 316M

Behind a firewall? Use these zipped Windows installers.

- Anaconda is a completely free Python distribution (including for commercial use and redistribution).
- It includes more than 300 of the most popular Python packages for science, math, engineering, and data analysis.
- https://www.continuum.io/downloads

Python: An Overview (Installation)



- Spyder - Scientific PYthon Development EnviRonment
- Free interactive development environment (IDE) included with Anaconda
- Includes editing, interactive testing, debugging and introspection features
- Type spyder in cmd or terminal!

3 Python Packages

Python Packages

Numpy

Mathematical operations
Arrays
Matrices
Shape Manipulation
Sorting
Algebra
Statistical operations

Pandas

Data Structures
Data Analysis
Data Munging
Data Reading
Data Writing
Handling Missing Data
Merging and Joining Data

Scikit-learn

Machine Learning
Classification
Regression
Clustering
Dimensionality Reduction
Model Selection

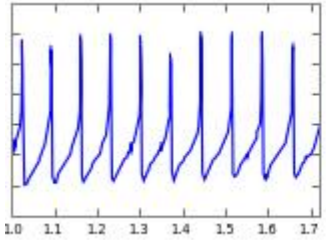
Python Packages

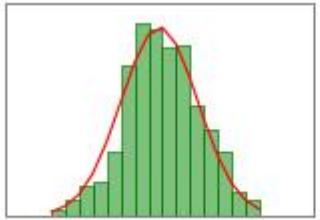
Matplotlib

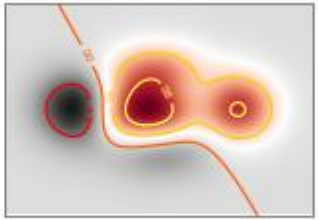
Grandfather of python visualization packages
Powerful but complex

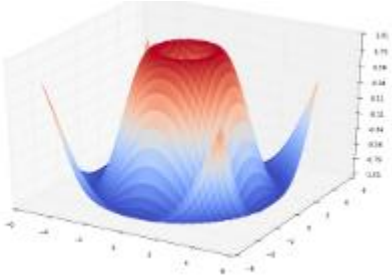
Seaborn

Visualization library based on matplotlib
Visualization more appealing
Complicated plots simple to create
Integrate well with pandas









“Data are just summaries of thousands of stories – tell a few of those stories to help make the data meaningful.”

Chip & Dan Heath, Authors of Made to Stick, Switch

3 Basic Python Tutorial

Basic Python Tutorial

Numbers, Strings and Boolean

Number data types stores numerical values: 1,2,3

String data types stores a line of characters: “Hello World”

Booleans are binary variable representing true or false

Lists

A list of items that does not need to be of the same type:

list3 = [1,"abc",3.5]

Dictionary

A list of items that uses a key to reference a value.

Associate one thing to another.

stuff **= {**'name'**:** ‘John'**,** 'age'**: 30,** 'height'**: 6 * 12 + 2}**

Basic Python Tutorial

- Numbers are created when you assign a numerical value to them.
- Strings are created when you assign a string to them with quotations.
- Booleans are created when you assign True or False to them
- You can print numbers and strings by typing print(variable)
- You can also print(type(variable)) to print the type

IPython console

Console 1/A

```
In [27]: var1 = 15

In [28]: var2 = 10.3

In [29]: print var1
15

In [30]: print var2
10.3

In [31]: string1 = "Hello World"

In [32]: print string1
Hello World
```

Console | History log | IPython console

Inclass Exercises

- Number and Strings
  - Assign the value 7 to a variable a
  - Assign the value “Hello World” to the variable b
  - Print a and b
  - Print the value 1000
  - Print the type of a and b
  - Using variable a and (+,-,*,/) print out -10

Basic Python Tutorial (List)

- Most versatile datatype in Python
- Written as a list of comma-seperated values
- List indices starts at 0
- Lists can be sliced, concatenated, iterated and so on…
- A list of items does not need to be the same type
- Example : list3 = [1,”abc”,3.5]

IPython console

Console 1/A

```
In [1]: list1 = [1, 2, 3, 4, 5 ];

In [2]: list2 = ["a", "b", "c", "d"];

In [3]: print list1
[1, 2, 3, 4, 5]

In [4]: print list2[2]
c

In [5]: print list2[1:3]
['b', 'c']

In [6]:
```

Console | History log | IPython console

Basic Python Tutorial (List)

Python Expression	Results	Description
list2[1] = “z”	['a', 'z', 'c', 'd']	Updating
del list2[1]	['a', 'c', 'd']	Deleting
list2.append("e")	['a', 'c', 'd', 'e']	Appending
list2 = list2[2:]	['d', 'e']	Slicing
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1, 2, 3]: print x,	1 2 3	Iteration

Inclass Exercises

- List
  - Append value “three” and “four” to l
  - Print l
  - Create a new list with the values 1,”two”,3,”four” in it and assign it to the variable n
  - Update “two” in n with 2
  - Print out the first two variables of n using slicing
  - Print out the length of n

IPython console

Console 1/A

```
In [53]: l = []

In [54]: l.append("one")

In [55]: l.append("two")

In [56]: print l
['one', 'two']

In [57]: m = ["one","two","three","four"]

In [58]: print m
['one', 'two', 'three', 'four']
```

Console | History log | IPython console

Basic Python Tutorial (Dict)

- A list of items that uses a key to reference a value. Associate one thing to another.
- Each key is separated from its value by a colon (:)
- The items are separated by commas, and the whole thing is enclosed in curly braces.

IPython console

Console 1/A

```
In [35]: stuff = {'name': 'John', 'age': 30, 'height': 6 * 12 + 2}

In [36]: print stuff
{'age': 30, 'name': 'John', 'height': 74}

In [37]: print stuff['name']
John

In [38]: stuff['age'] = 31

In [39]: print stuff['age']
31

In [40]:
```

Console | History log | IPython console

nissions: RW | End-of-lines: CRLF | Encoding: UTF-8 | Line: 1 | Column: 1

Inclass Exercises

- Dictionary
  - Create a Dictionary with “one” : 1, “two”: 2, “three”: “three”, “four”: 4.0 and assign it to dict
  - Print dict
  - Print out the value of “one”
  - Update “three” in dict with 3
  - Print out the type of the value of “four”

IPython console

Console 1/A

```
In [81]: dict = {}

In [82]: dict = {'name': 'John', 'age': 30, 'height': 6 * 12 + 2}

In [83]: print dict
{'age': 30, 'name': 'John', 'height': 74}

In [84]: dict['name'] = 'Kane'

In [85]: print dict['name']
Kane

In [86]:
```

Console | History log | IPython console

missions: RW | End-of-lines: CRLF | Encoding: UTF-8 | Line: 5 | Column: 33

Basic Python Tutorial

Functions

Define functions that we can call later. It will do what we tell it to do.

```
In [88]: def func(a,b):
    ...:     return a + b
    ...:

In [89]: print(func(5,6))
11
```

If Statement

Performs different actions depending on whether the boolean we pass is true

```
In [106]: def func(a):
     ...:     if a == True:
     ...:         print "True"
     ...:

In [107]: a = True

In [108]: func(a)
True
```

For Loops

Used to repeat a code n number of times

Iterate over the items of any sequence, such as a list or a string.

```
In [109]: fruits = ['banana', 'apple',  'mango']
     ...: for fruit in fruits:
     ...:    print 'Current fruit :', fruit
     ...:
Current fruit : banana
Current fruit : apple
Current fruit : mango
```

Basic Python Tutorial

- Functions are created through the key word def followed by the name of the functions and parentheses
- Functions are called by writing the function name.
- Print() is an example of a function
- You can pass parameters into the function by putting it in the parentheses

IPython console

Console 1/A

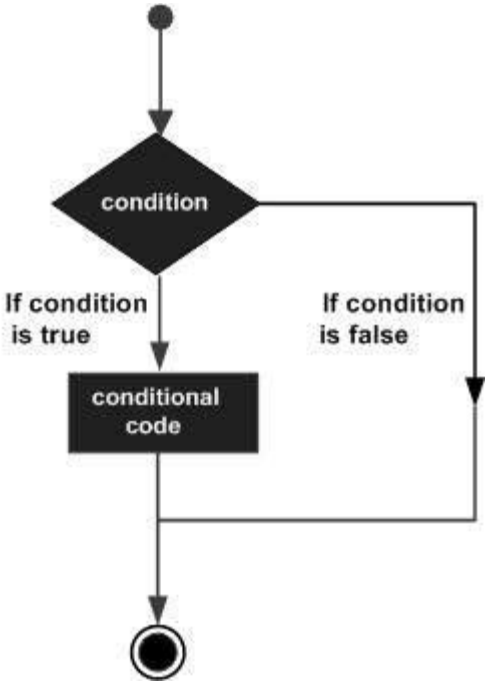
```
In [1]: def func(a,b):
   ...:     return a + b
   ...:

In [2]: func(1,2)
Out[2]: 3

In [3]: def func2(a,b):
   ...:     print a
   ...:     print b
   ...:

In [4]: func2("one","two")
one
two
```

Basic Python Tutorial




```
In [106]: def func(a):
     ...:     if a == True:
     ...:         print "True"
     ...:

In [107]: a = True

In [108]: func(a)
True

In [14]: def func(a):
    ...:     if a == True:
    ...:         print "true!"
    ...:     else:
    ...:         print "false"
    ...:

In [15]: a = "hi"

In [16]: func(a)
false

In [17]:
```

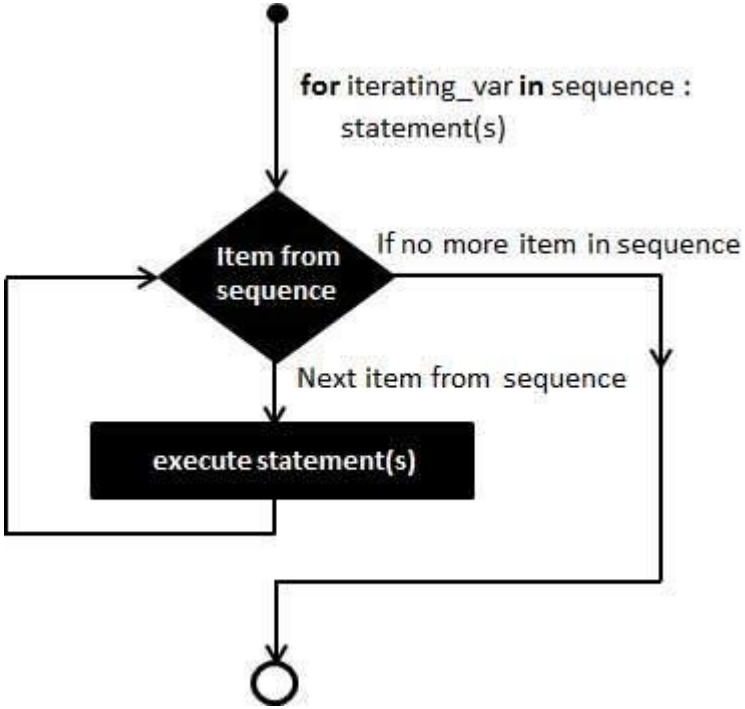
Basic Python Tutorial

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	
<>	If values of two operands are not equal, then condition becomes true.	(a <> b) is true. This is similar to != operator.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.

Inclass Exercises

- Functions
  - Create a function that prints “hello world”
  - Create a function that multiply two variables
- If Statements
  - Create a function that check if the sum of the parameters are more than 10. It will return true if it is, return false otherwise

Basic Python Tutorial




```
In [109]: fruits = ['banana', 'apple',  'mango']
     ...: for fruit in fruits:
     ...:     print 'Current fruit :', fruit
     ...:
Current fruit : banana
Current fruit : apple
Current fruit : mango
```

Inclass Exercises

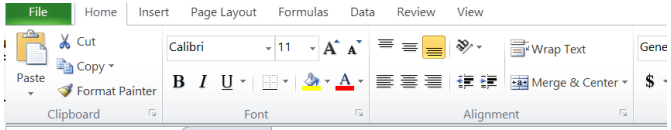
- For Statement
  - Print out all items in the list [“hi”,”how”,”are”,”you”] using a for loop.
  - Using the list [1,2,3,4,5,6,7,8,9,10], a for loop and a if statement return a list of only odd numbers. (Hint: 3%2 = 1, 5%2 = 1. Also add to a list using append())

5

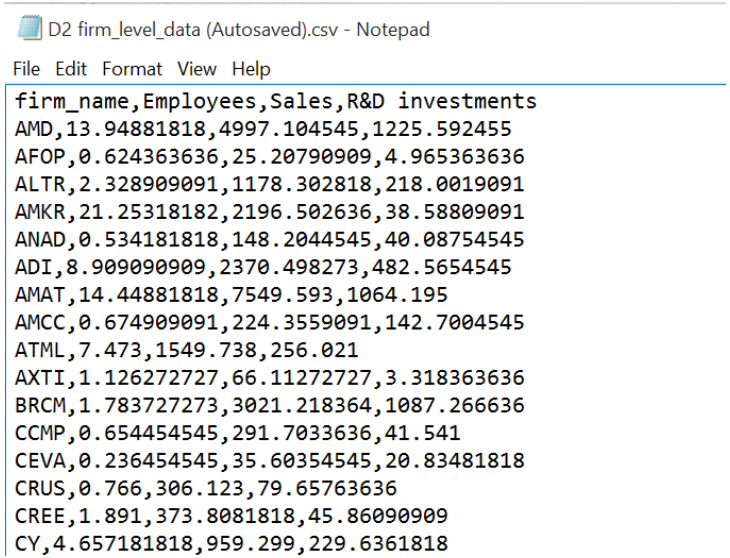
Reading Data Into Python

Reading Data Into Python

- What is a csv file?? (comma separated values **file)**



	A	B	C	D	E	F
1	firm_name	Employees	Sales	R&D investments		
2	AMD	13.94881818	4997.104545	1225.592455		
3	AFOP	0.624363636	25.20790909	4.965363636		
4	ALTR	2.328909091	1178.302818	218.0019091		
5	AMKR	21.25318182	2196.502636	38.58809091		
6	ANAD	0.534181818	148.2044545	40.08754545		
7	ADI	8.909090909	2370.498273	482.5654545		
8	AMAT	14.44881818	7549.593	1064.195		
9	AMCC	0.674909091	224.3559091	142.7004545		
10	ATML	7.473	1549.738	256.021		
11	AXTI	1.126272727	66.11272727	3.318363636		
12	BRCM	1.783727273	3021.218364	1087.266636		
13	CCMP	0.654454545	291.7033636	41.541		
14	CEVA	0.236454545	35.60354545	20.83481818		
15	CRUS	0.766	306.123	79.65763636		
16	CREE	1.891	373.8081818	45.86090909		
17	CY	4.657181818	959.299	229.6361818		




```
firm_name,Employees,Sales,R&D investments
AMD,13.94881818,4997.104545,1225.592455
AFOP,0.624363636,25.20790909,4.965363636
ALTR,2.328909091,1178.302818,218.0019091
AMKR,21.25318182,2196.502636,38.58809091
ANAD,0.534181818,148.2044545,40.08754545
ADI,8.909090909,2370.498273,482.5654545
AMAT,14.44881818,7549.593,1064.195
AMCC,0.674909091,224.3559091,142.7004545
ATML,7.473,1549.738,256.021
AXTI,1.126272727,66.11272727,3.318363636
BRCM,1.783727273,3021.218364,1087.266636
CCMP,0.654454545,291.7033636,41.541
CEVA,0.236454545,35.60354545,20.83481818
CRUS,0.766,306.123,79.65763636
CREE,1.891,373.8081818,45.86090909
CY,4.657181818,959.299,229.6361818
```

Reading Data Into Python

- We will be using pandas, the python package to read the file.
- import pandas as pd
- d2 = pd.read_csv("D2_data.csv")
- print d2['Sales'].sum()

IPython console

Console 1/A

```
In [41]: runfile('C:/Users/Kane/Desktop/pandas.py',
wdir='C:/Users/Kane/Desktop')

In [42]: d2.head(5)
Out[42]:
  firm_name  Employees        Sales  R&D investments
0       AMD  13.948818  4997.104545      1225.592455
1      AFOP   0.624364    25.207909         4.965364
2      ALTR   2.328909  1178.302818       218.001909
3      AMKR  21.253182  2196.502636        38.588091
4      ANAD   0.534182   148.204454        40.087545

In [43]: d2.describe()
Out[43]:
       Employees         Sales  R&D investments
count  59.000000     59.000000        59.000000
mean    5.224265   1645.697560       268.233482
std    12.247230   4786.165217       722.069926
min     0.061455     10.807182         3.318364
25%     0.550545    156.881909        26.156636
50%     1.741455    340.114273        45.860909
75%     3.688000   1297.167772       154.515045
max    85.400000  34568.818180      5083.727273
```

Numpy

- Support for arrays matrices manipulation
- Mathematical functions
- Import numpy as np
- print np.random.randn(5)

```
In [77]: print np.random.randn(5)
[-0.16258244 -0.16271272  0.80275495 -1.2129952  -0.3160987 ]
```

DataFrames

- loc allows selection of a row

```
In [85]: print d2.head()
  firm_name  Employees        Sales  R&D investments
0       AMD  13.948818  4997.104545      1225.592455
1      AFOP   0.624364    25.207909         4.965364
2      ALTR   2.328909  1178.302818       218.001909
3      AMKR  21.253182  2196.502636        38.588091
4      ANAD   0.534182   148.204454        40.087545

In [86]: print d2.loc[3]
firm_name              AMKR
Employees           21.2532
Sales                2196.5
R&D investments     38.5881
Name: 3, dtype: object

In [87]: print d2.loc[3]['Sales']
2196.502636
```

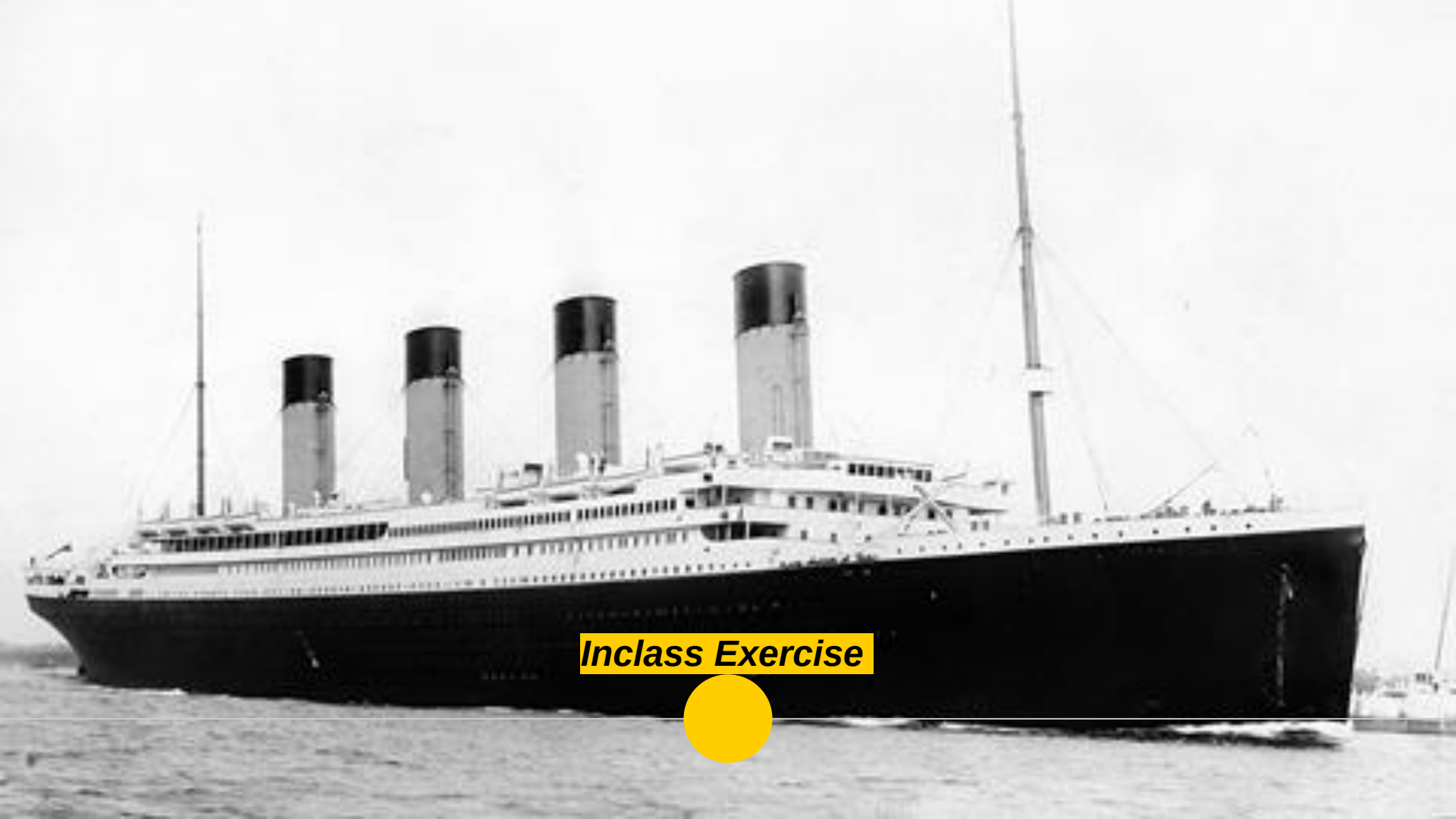
DataFrames

- Creates a table with rows and columns
- Belongs to the class Pandas
- Each row is a Series

```
In [77]: print np.random.randn(5)
[-0.16258244 -0.16271272  0.80275495 -1.2129952  -0.3160987 ]

In [78]: s = pd.Series(np.random.randn(5), index=['a', 'b', 'c', 'd', 'e'])

In [79]: print s
a    0.033969
b    1.093153
c    0.147675
d   -1.496880
e   -0.988708
dtype: float64
```



Inclass Exercise

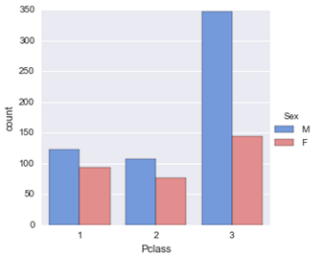
Inclass Exercises

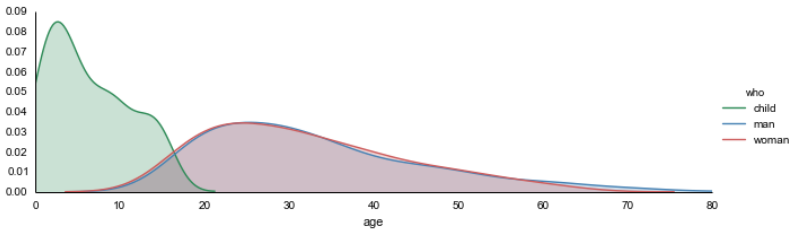
- Reading CSV
  - Read the titanic CSV file.
  - Run the command head(5) on the file
  - Run the command describe() on the file
  - What do you notice about the data??

Preview

- Manipulation of Titanic Data (Pandas)
  Data Structures, Data Analysis, Data Munging, Data Reading, Data Writing, Handling Missing Data

- Visualization of Titanic Data (Seaborn) *Installation!!*





Thanks!

Any questions ?

You can find me at

- kcw115@ic.ac.uk

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash and Vinsionaire