EEC1509 - Machine Learning

Lesson #2 - Platforms & Fundamentals of Python for ML

Ivanovitch Silva

August, 2018

Agenda

- Development platform
- Numpy vs Pandas

How to Become a **Data Scientist**

MODERN DATA SCIENTIST

Data Scientist, the sexiest job of 21th century requires a mixture of multidisciplinary skills ranging from an intersection of mathematics, statistics, computer science, communication and business. Finding a data scientist is hard. Finding people who understand who a data scientist is, is equally hard. So here is a little cheat sheet on who the modern data scientist really is.

MATH & STATISTICS

- Machine learning
- Statistical modeling
- Experiment design
- Bayesian inference
- Supervised learning: decision trees, random forests, logistic regression
- Unsupervised learning: clustering, dimensionality reduction
- Optimization: gradient descent and variants

PROGRAMMING & DATABASE

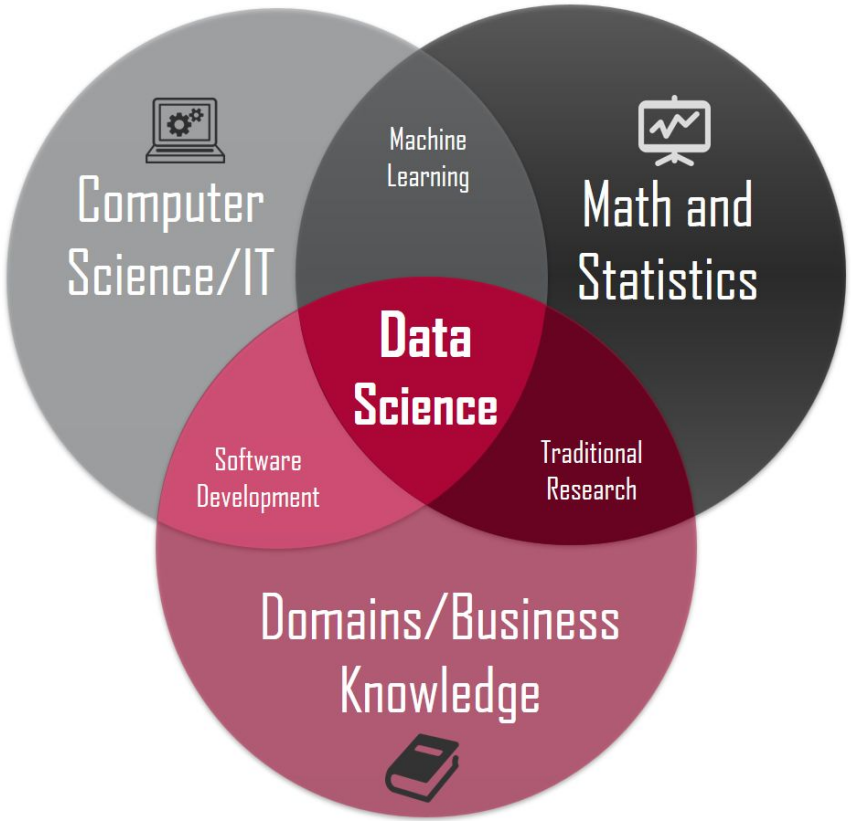
- Computer science fundamentals
- Scripting language e.g. Python
- Statistical computing package e.g. R
- Databases SQL and NoSQL
- Relational algebra
- Parallel databases and parallel query processing
- MapReduce concepts
- Hadoop and Hive/Pig
- Custom reducers
- Experience with xaaS like AWS

DOMAIN KNOWLEDGE & SOFT SKILLS

- Passionate about the business
- Curious about data
- Influence without authority
- Hacker mindset
- Problem solver
- Strategic, proactive, creative, innovative and collaborative

COMMUNICATION & VISUALIZATION

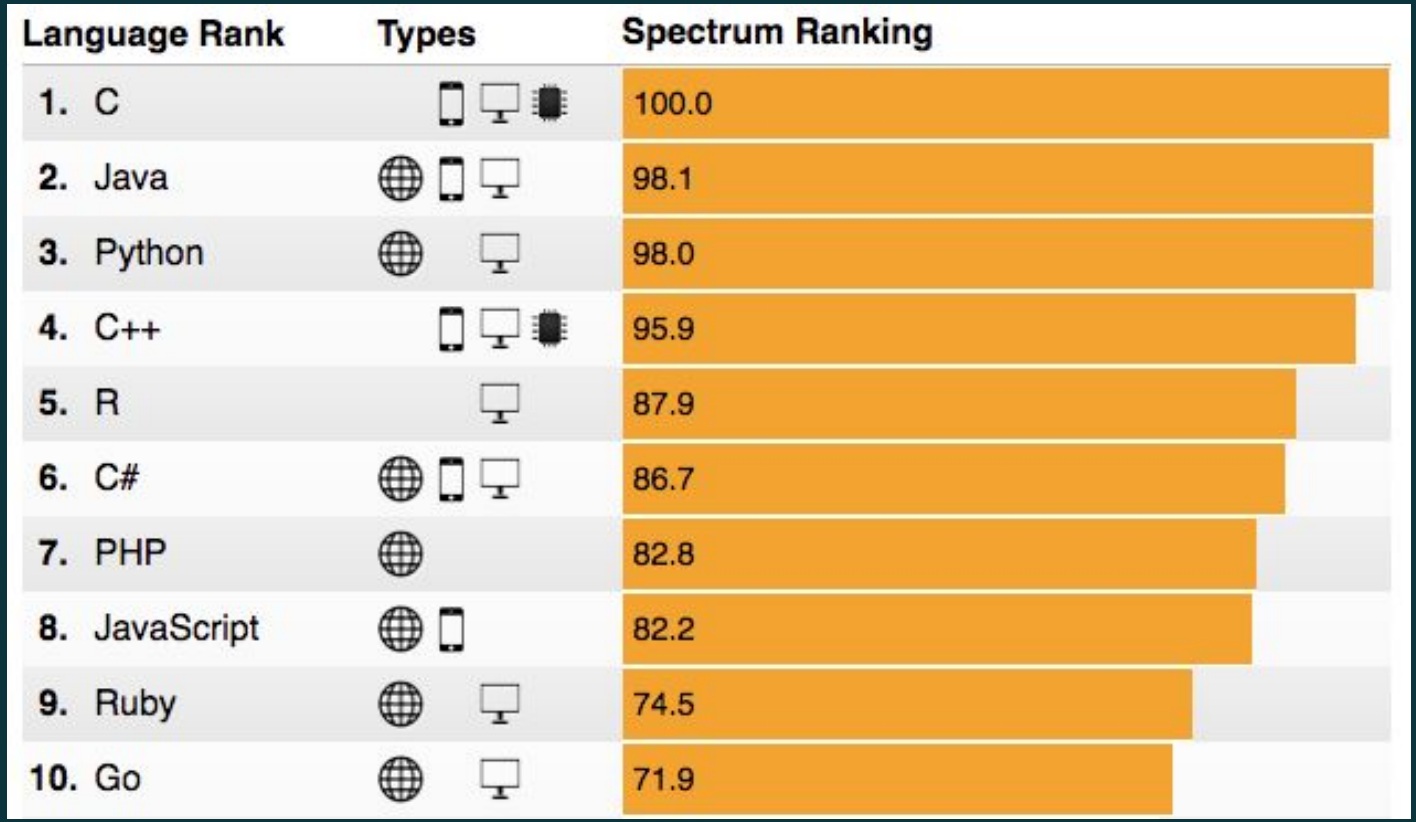
- Able to engage with senior management
- Story telling skills
- Translate data-driven insights into decisions and actions
- Visual art design
- R packages like ggplot or lattice
- Knowledge of any of visualization tools e.g. Flare, D3.js, Tableau



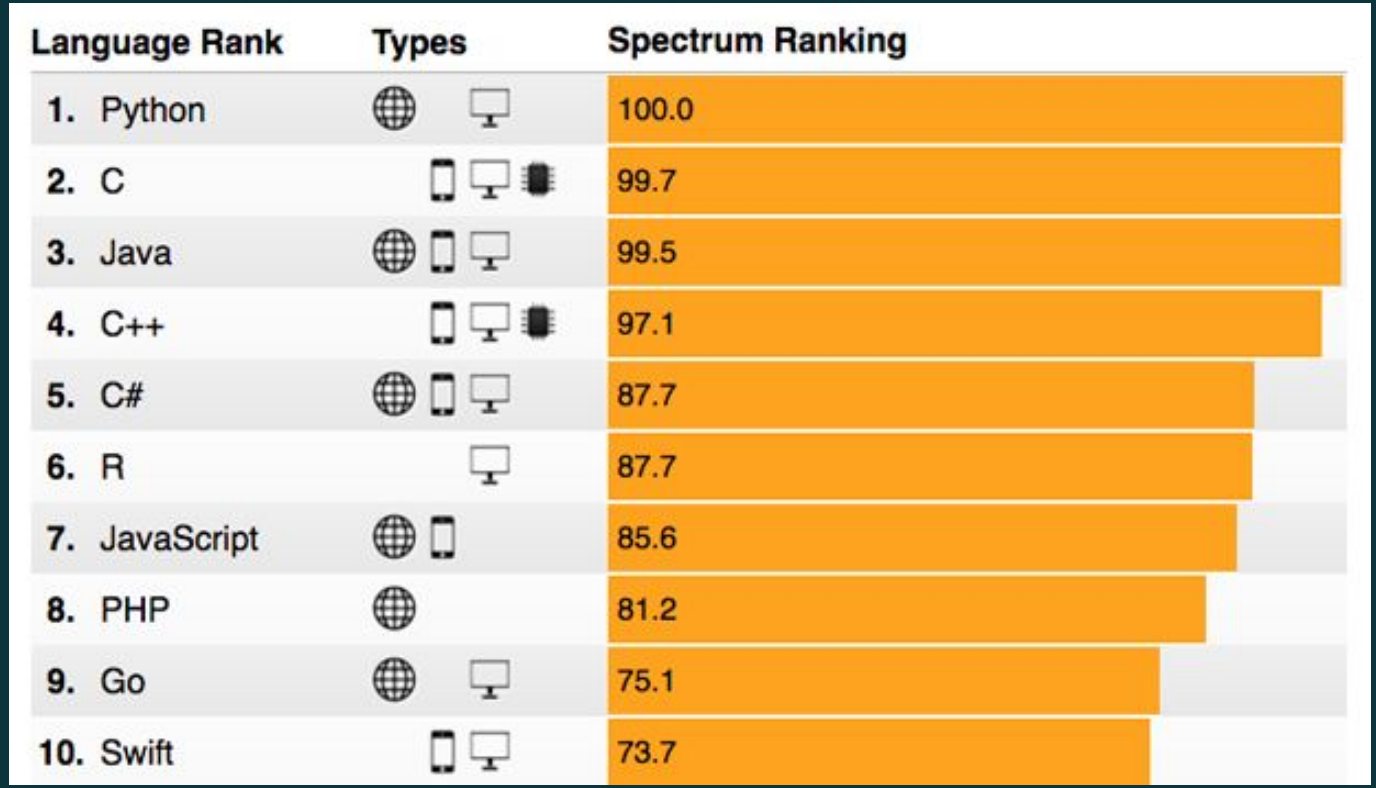
Pick ONE programming language and STICK to it. Don’t go back and constantly change your choice of language to study. If you do, you will slow your progress down.


which programming
language to learn first (ML)?

https://goo.gl/VKYfXn

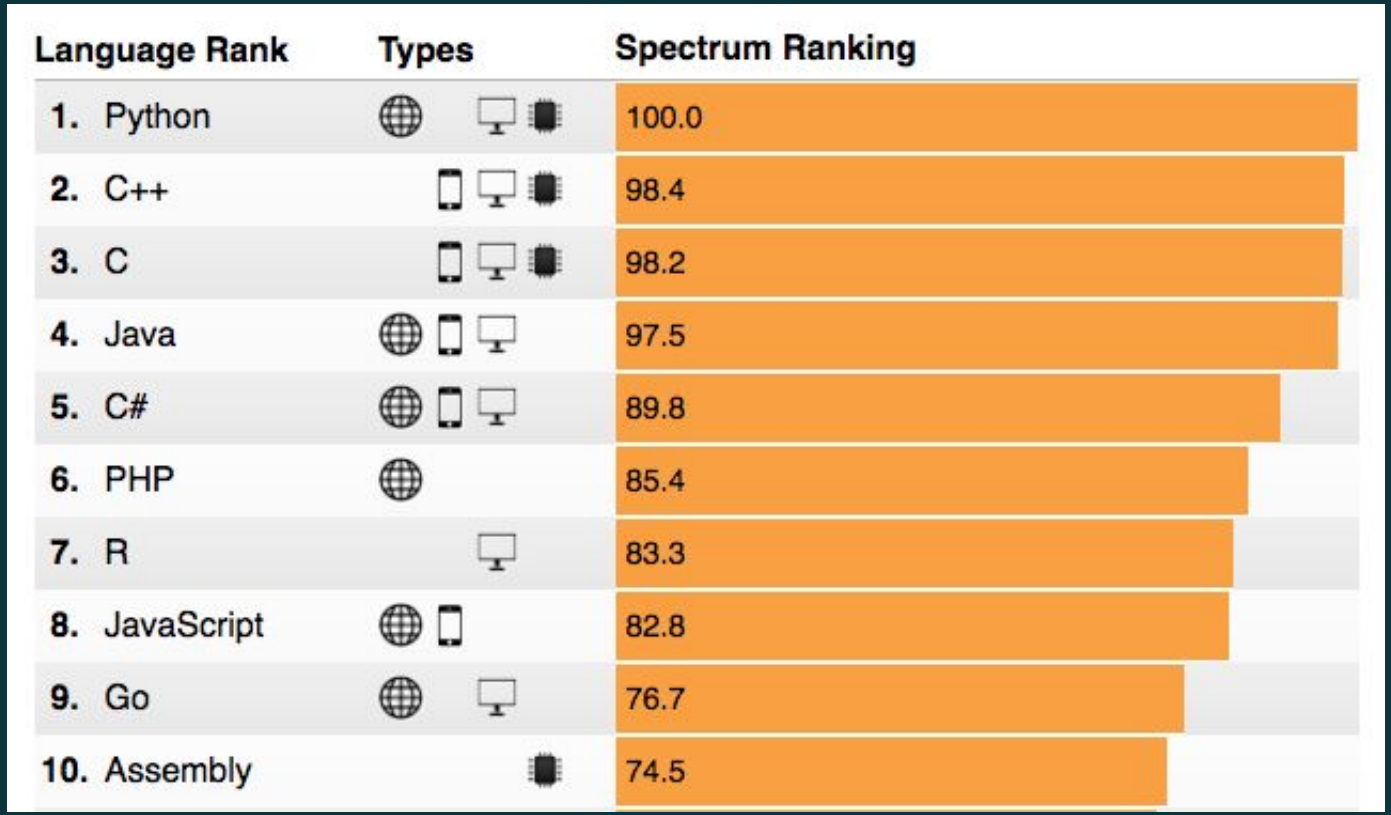


IEEE Spectrum - Jul 2016 https://goo.gl/BqrkDI



Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C		99.7
3. Java		99.5
4. C++		97.1
5. C#		87.7
6. R		87.7
7. JavaScript		85.6
8. PHP		81.2
9. Go		75.1
10. Swift		73.7

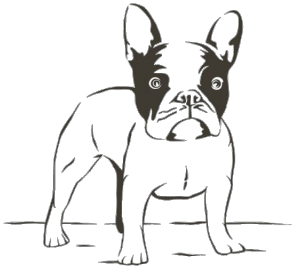
IEEE Spectrum - Jul 2017 https://goo.gl/HSPLWe



Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C++		98.4
3. C		98.2
4. Java		97.5
5. C#		89.8
6. PHP		85.4
7. R		83.3
8. JavaScript		82.8
9. Go		76.7
10. Assembly		74.5

IEEE Spectrum - Jul 2018
https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages

have
a pet
project

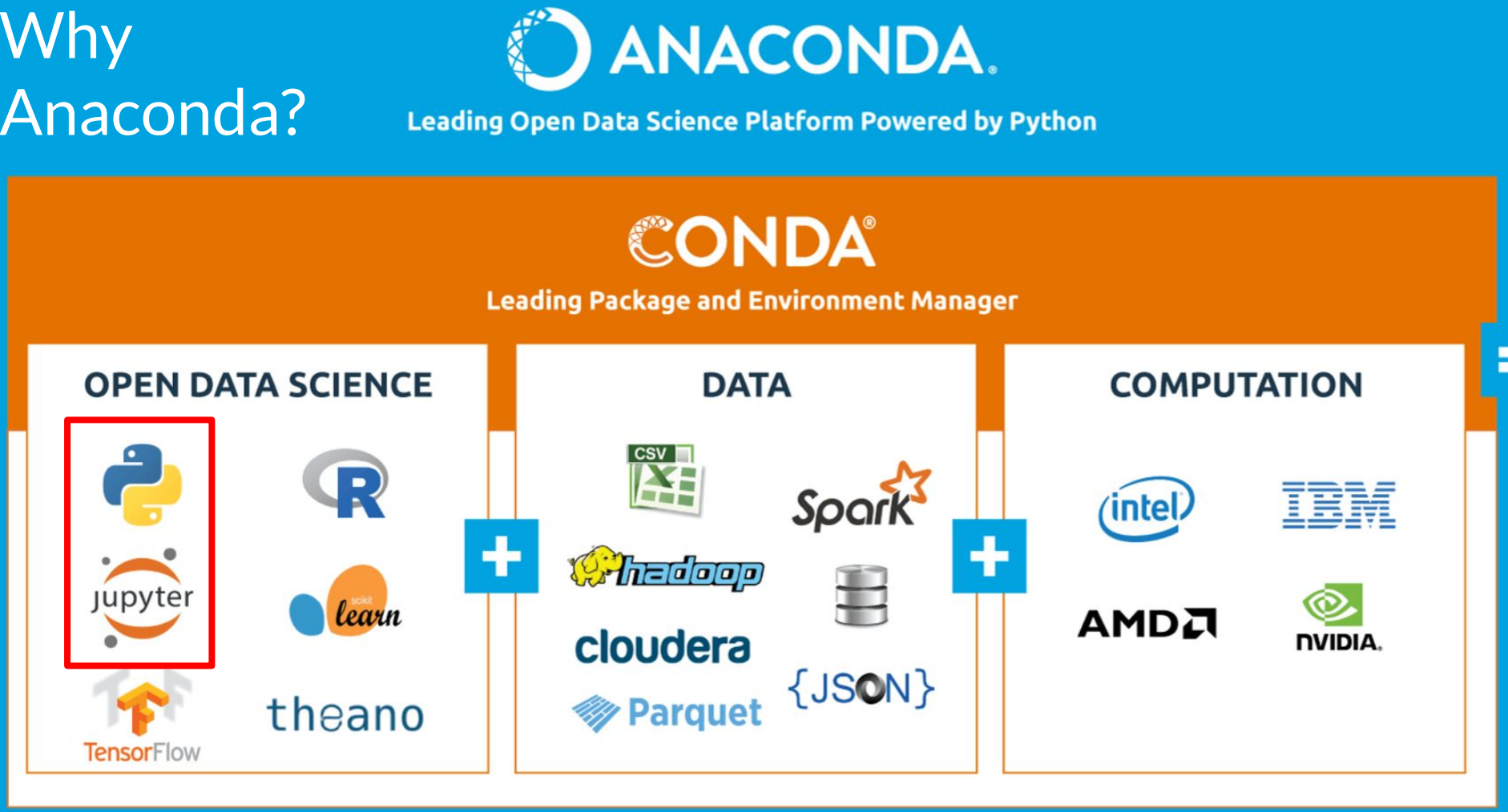


Be clear about your motivation. The reason this is important because learning Data Science is HARD. VERY HARD! So it's easy to lose motivation when on the journey.

Immerse yourself in the community (newsletters, articles, books, podcasts, youtube, hackathons and meetups)

https://www.continuum.io/downloads

Why Anaconda?



jupyter Untitled (unsaved changes)

File Edit View Insert Cell Kernel Help Python 2













Code CellToolbar

Simple Jupyter demo

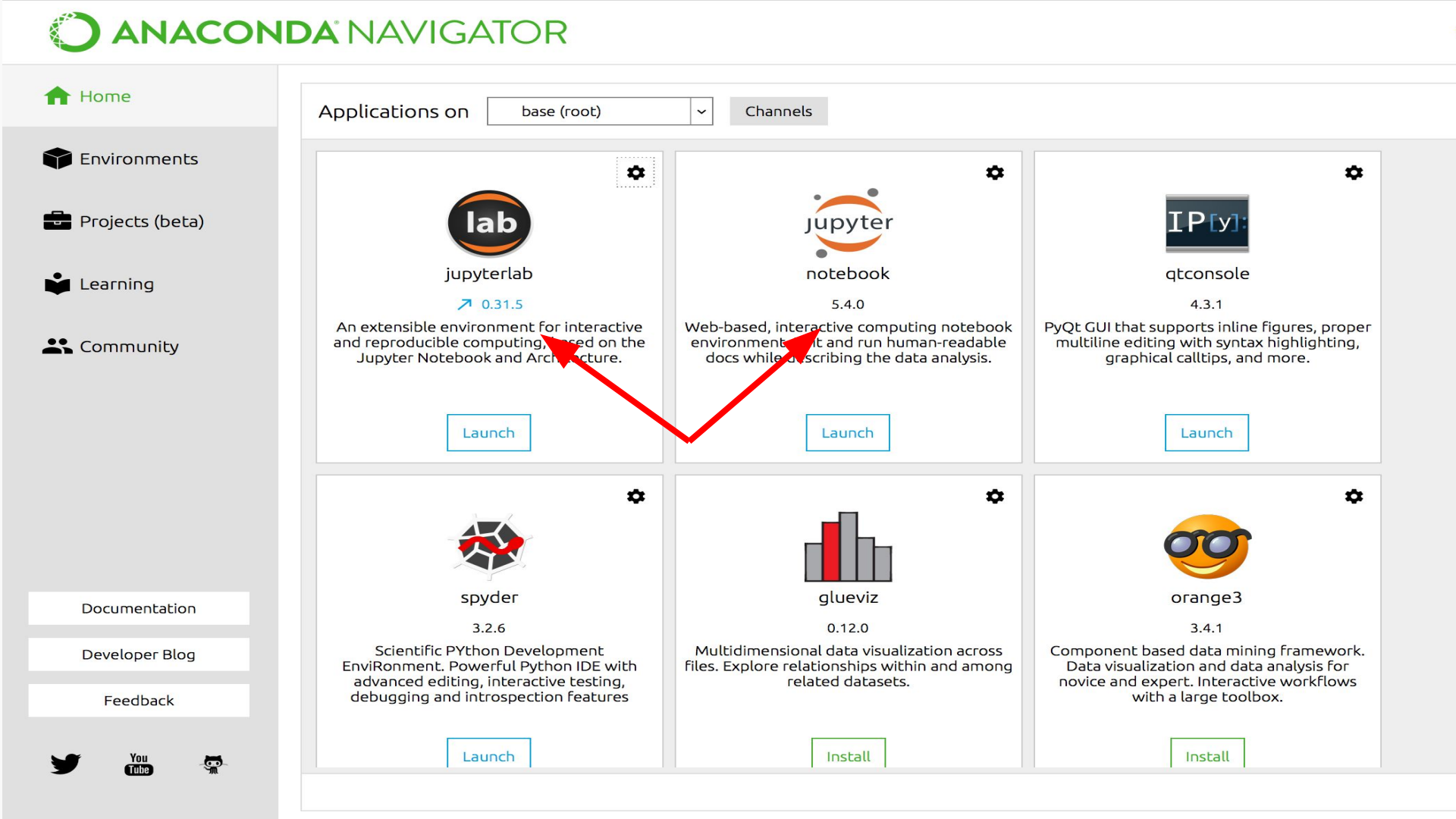
This cell has text formatted using the markdown language, which gets rendered like regular html. The next cell has some code:

In [57]:

```
import random
for i in range(3):
    print random.random()
x = 10
```

```
0.10564822904
0.153941700348
0.518503128416
```

Here is another text cell, with some *formatting*.


ANACONDA NAVIGATOR
Home
Environments
Projects (beta)
Learning
Community
Documentation
Developer Blog
Feedback
Applications on
base (root)
Channels
jupyterlab
0.31.5
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
notebook
5.4.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
qtconsole
4.3.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch
spyder
3.2.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch
glueviz
0.12.0
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install
orange3
3.4.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Install

Sponsors

Project Jupyter receives direct funding from the following sources:

THE LEONA M. AND HARRY B. HELMSLEY CHARITABLE TRUST

ALFRED P. SLOAN FOUNDATION

GORDON AND BETTY MOORE FOUNDATION

Google

rackspace
the #1 managed cloud company

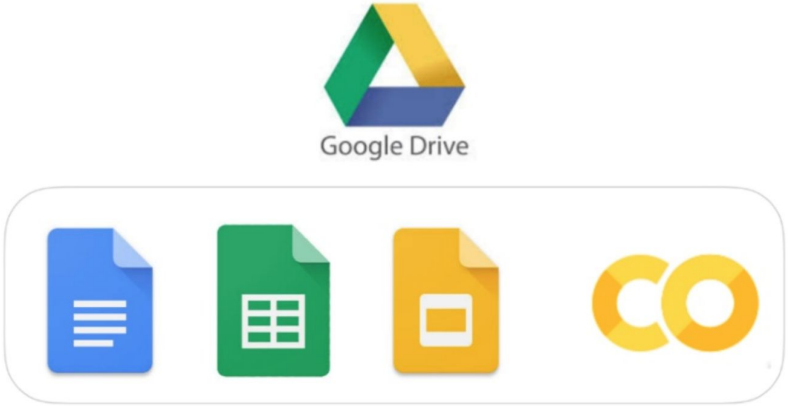
fastly

Horizon 2020
European Union funding
for Research & Innovation

Microsoft

Google Colaboratory

https://colab.research.google.com/



Colaboratory is a Google research project created to help disseminate machine learning education and research. It's a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud.

Colaboratory notebooks are stored in Google Drive and can be shared just as you would with Google Docs or Sheets. Colaboratory is free to use.

Installing Git

https://git-scm.com/downloads

Update repository

```
git clone https://github.com/ivanovitchm/EEC1509_MachineLearning.git
```

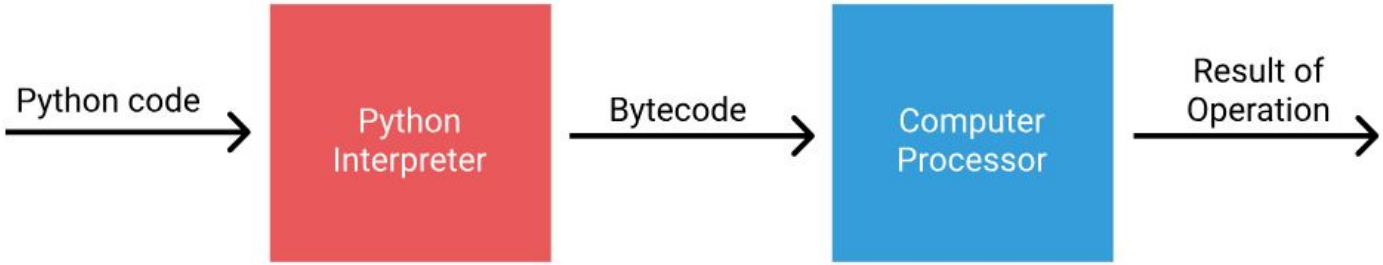
Ou ....

```
git pull
```

+

Pandas

Understanding Vectorization



Language Type	Example	Time taken to write program	Control over program performance
High-Level	Python	Low	Low
Low-Level	C	High	High

Unvectorized code using list of lists

6	5
1	3
5	6
1	4
3	7
5	8
3	5
8	4

Two columns of numbers

```
my_numbers = [
                [6, 5],
                [1, 3],
                [5, 6],
                [1, 4],
                [3, 7],
                [5, 8],
                [3, 5],
                [8, 4]
              ]
```

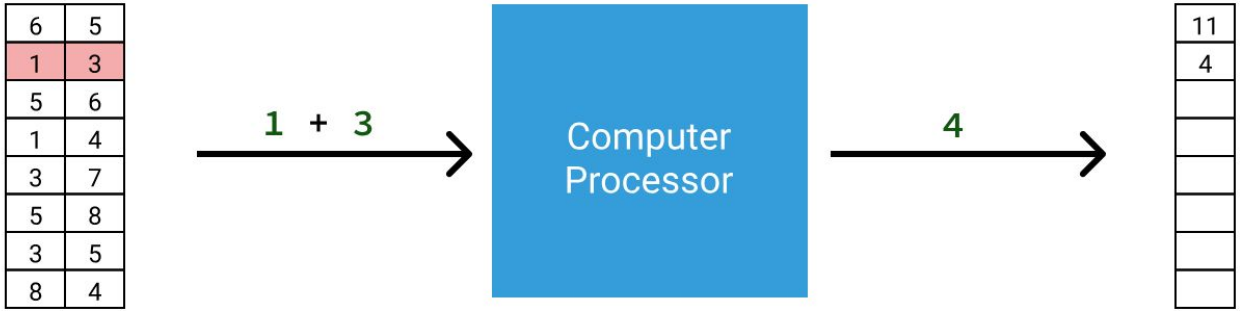
List of lists representation

```
sums = []

for row in my_numbers:
    row_sum = row[0] + row[1]
    sums.append(row_sum)
```

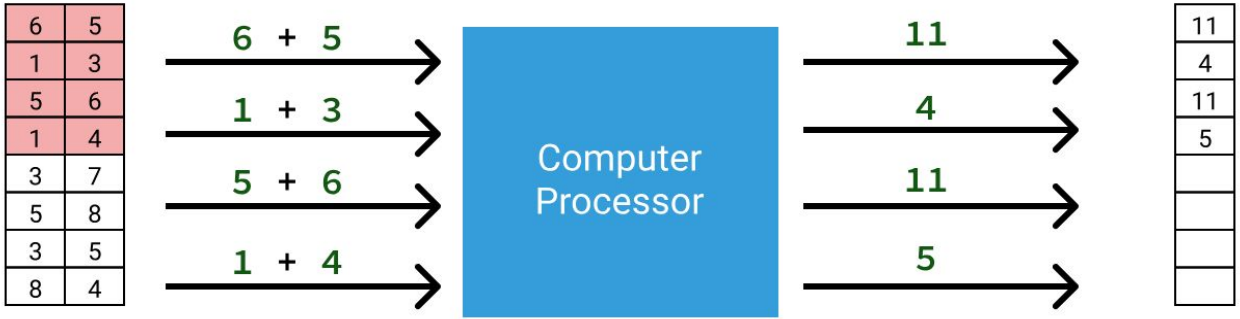
Python code to sum each row

How Vectorization Makes Code Faster

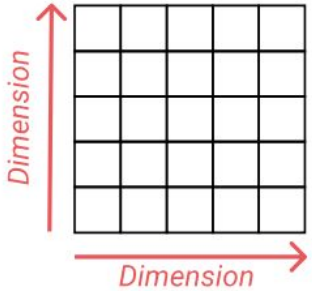
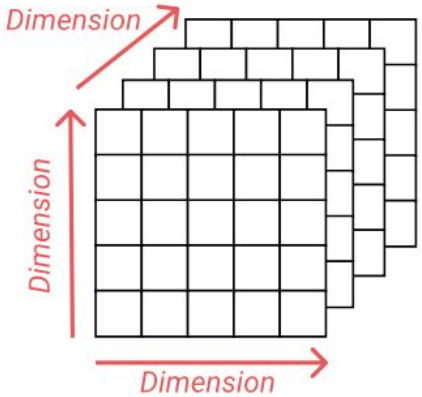
Single Instruction Multiple Data (SIMD)



Numpy
Pandas



Understanding Numpy ndarray

	Number of Dimensions	Known As
	One	One-dimensional array, array, list, vector, sequence
	Two	Two-dimensional array, matrix, table, list of lists, spreadsheet
	Three	Three-dimensional array, multi-dimensional array, panel

NYC Taxi-Airport Data

There is data on over 1.3 trillion individual trips, reaching back as far as 2009 and is regularly updated

nyc_taxis.csv

NYC Taxi-Airport Data

pickup_year	pickup_month	pickup_day	pickup_dayofweek	pickup_time	pickup_location_code	dropoff_location_code	trip_distance	trip_length	fare_amount	total_amount
2016	1	1	5	0	2	4	21.00	2037	52.0	69.99
2016	1	1	5	0	2	1	16.29	1520	45.0	54.30
2016	1	1	5	0	2	6	12.70	1462	36.5	37.80
2016	1	1	5	0	2	6	8.70	1210	26.0	32.76
2016	1	1	5	0	2	6	5.56	759	17.5	18.80
2016	1	1	5	0	4	2	21.45	2004	52.0	105.60
2016	1	1	5	0	2	6	8.45	927	24.5	32.25
2016	1	1	5	0	2	6	7.30	731	21.5	22.80
2016	1	1	5	0	2	5	36.30	2562	109.5	131.38
2016	1	1	5	0	6	2	12.46	1351	36.0	37.30

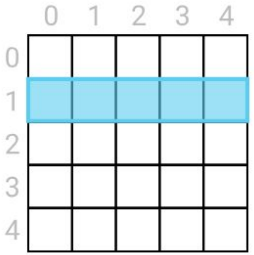
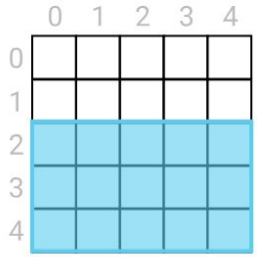
Introduction to Numpy

```
import csv
import numpy as np

# import nyc_taxi.csv as a list of lists
f = open("nyc_taxis.csv", "r")
taxi_list = list(csv.reader(f))

# remove the header row
taxi_list = taxi_list[1:]

# convert all values to floats
converted_taxi_list = []
for row in taxi_list:
    converted_row = []
    for item in row:
        converted_row.append(float(item))
    converted_taxi_list.append(converted_row)

# start writing your code below this comment
taxi = np.array(converted_taxi_list)
```

Introduction to Numpy

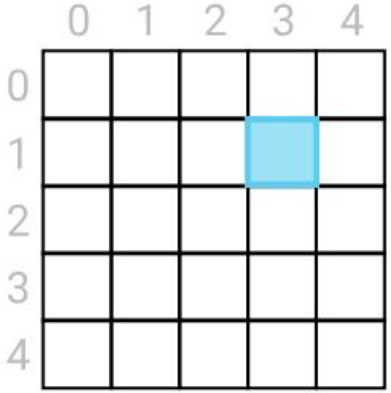
```
>>> print(taxi)

    [[ 2016.   1.    1.   ..., 11.65   69.99    1. ]
     [ 2016.   1.    1.   ...,  8.     54.3     1. ]
     [ 2016.   1.    1.   ...,  0.     37.8     2. ]
     ...,
     [ 2016.   6.   30.   ...,  5.     63.34    1. ]
     [ 2016.   6.   30.   ...,  8.95   44.75    1. ]
     [ 2016.   6.   30.   ...,  0.     54.84    2. ]]

 >>> taxi.shape
     (89560, 15)
```

Selecting and Slicing Rows and Items from ndarrrays

	List of lists method	*NumPy method*
Selecting a single row 	`sel_lol = data_lol[1]`	`sel_np = data_np[1]` *Same syntax as list of lists. Produces a 1D ndarray.*
Selecting multiple rows 	`sel_lol = data_lol[2:]`	`sel_np = data_np[2:]` *Same syntax as list of lists. Produces a 2D ndarray.*

Selecting and Slicing Rows and Items from ndarrrays

Selecting a single item



List of lists method

```
sel_lol = data_lol[1][3]
```

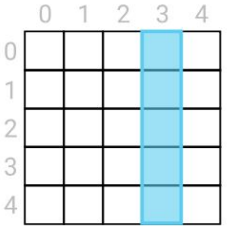
NumPy method

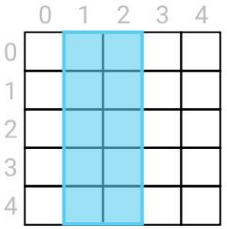
```
sel_np = data_np[1,3]
```

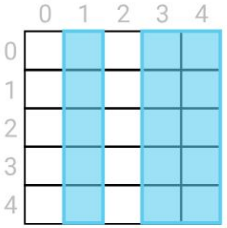
Comma separated row/column locations. Produces a single Python object.

Selecting Columns and Custom Slicing ndarrays

	List of lists method	*NumPy method*

Selecting a single column 	`sel_lol = []`
	
	`for row in data_lol:`
	`  col4 = row[3]`
	`  sel_lol.append(col4)`
		`sel_np = data_np[:,3]`
		
		Comma separated row wildcard and column location. Produces a 1D ndarray

Selecting multiple columns 	`sel_lol = []`
	
	`for row in data_lol:`
	`  col23 = row[1:3]`
	`  sel_lol.append(col23)`
		`sel_np = data_np[:,1:3]`
		
		Comma separated row wildcard and column slice location. Produces a 2D ndarray

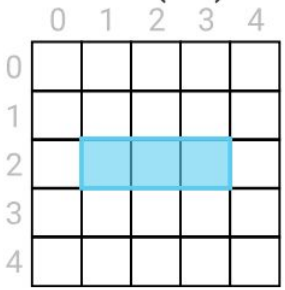
Selecting multiple, specific columns 	`sel_lol = []`
	
	`for row in data_lol:`
	`  cols = [row[1],`
	`  row[3],row[4]]`
	`  sel_lol.append(cols)`
		`cols = [1,3,4]`
		`sel_np = data_np[:,cols]`
		
		Comma separated row wildcard and list of column locations. Produces a 2D ndarray

Selecting Columns and Custom Slicing ndarrays

List of lists method

NumPy method

Selecting a 1D slice (row)

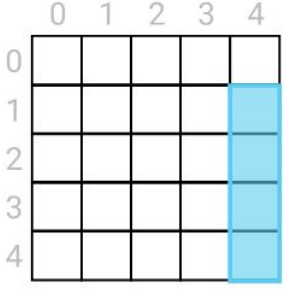



```
sel_lol = data_lol[2][1:4]
```

```
sel_np = data_np[2,1:4]
```

Comma separated row location and column slice. Produces a 1D ndarray

Selecting a 1D slice (column)

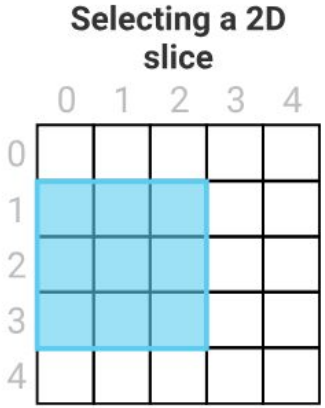



```
sel_lol = []

rows = data_lol[1:]
for r in rows:
    col5 = r[4]
    sel_lol.append(col5)
```

```
sel_np = data_np[1:,4]
```

Comma separated row slice and column location. Produces a 1D ndarray

Selecting Columns and Custom Slicing ndarrays



List of lists method

```
sel_lol = []

rows = data_lol[1:4]
for r in rows:
    new_row = r[:3]
    sel_lol.append(new_row)
```

NumPy method

```
sel_np = data_np[1:4,:3]
```

Comma separated row/column slice locations. Returns a 2D ndarray

Vector Math (list of lists vs numpy)

```
import numpy as np

# create random (500,5) numpy arrays and list of lists
np_array = np.random.rand(500,5)
list_array = np_array.tolist()

def python_subset():
  filtered_cols = []
  for row in list_array:
    filtered_cols.append([row[1],row[2]])
  return filtered_cols

def numpy_subset():
  return np_array[:,1:3]
```

Vector Math (list of lists vs numpy)

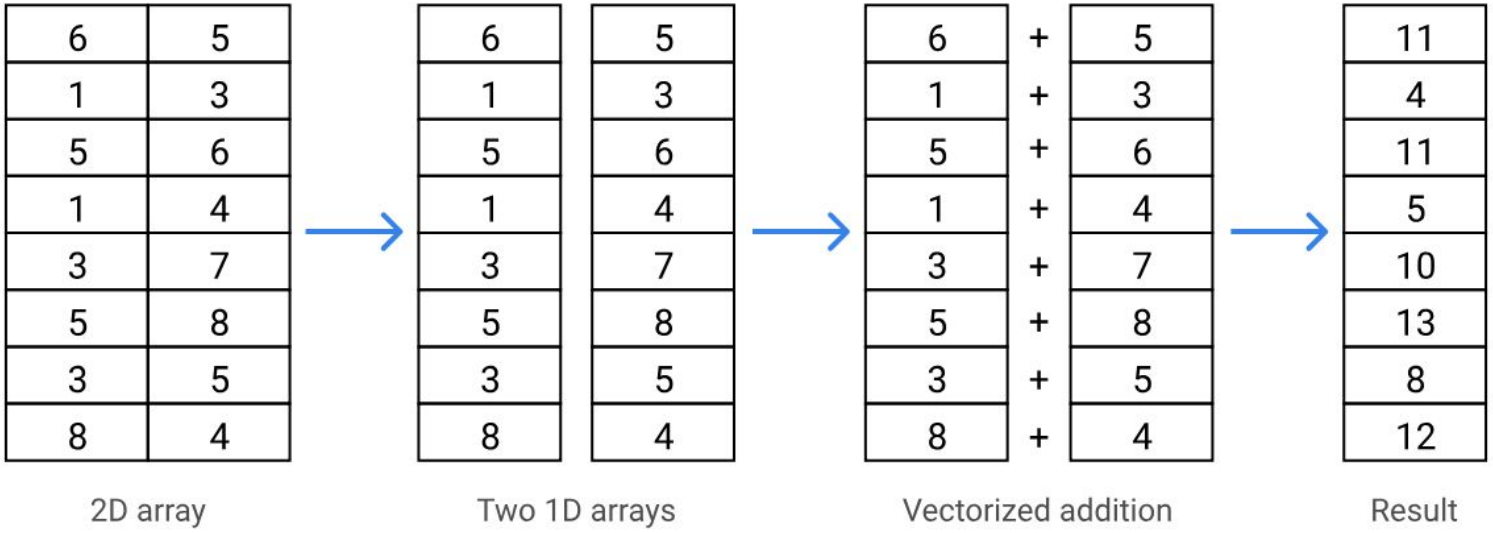
```
%%timeit -r 1 -n 1
list_of_list = python_subset()
```

```
1 loop, best of 1: 182 µs per loop
```

```
%%timeit -r 1 -n 1
numpy_array = numpy_subset()
```

```
1 loop, best of 1: 9.06 µs per loop
```

Vector Math (numpy)



Calculating Statistics for 1D ndarrays

Calculation	Function Representation	Method Representation
Calculate the minimum value of **trip_mph**	np.min(trip_mph)	trip_mph.min()
Calculate the maximum value of **trip_mph**	np.max(trip_mph)	trip_mph.max()
Calculate the mean average value of **trip_mph**	np.mean(trip_mph)	trip_mph.mean()
Calculate the median average value of **trip_mph**	np.median(trip_mph)	There is no ndarray median method

Calculating Statistics for 2D ndarrays

2D ndarray

1	0	1	1
0	1	4	3
0	1	0	2
3	0	1	3

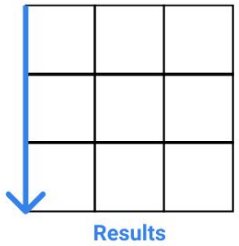
`ndarray.max(axis=0)`

1	0	1	1
0	1	4	3
0	1	0	2
3	0	1	3

3	1	4	3

Result

`ndarray.method(axis=0)`
Calculates along the **row** axis



Calculates result for each **column**.

2D ndarray

1	0	1	1
0	1	4	3
0	1	0	2
3	0	1	3

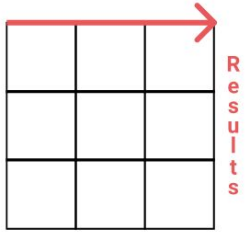
`ndarray.max(axis=1)`

1	0	1	1
0	1	4	3
0	1	0	2
3	0	1	3

Result

1
4
2
3

`ndarray.method(axis=1)`
Calculates along the **column** axis



Calculates result for each **row**.

Adding Rows and Columns to ndarrays (concatenate)

```
>>> print(ones)

    [[ 1  1  1]
     [ 1  1  1]]

>>> print(zeros)

    [ 0  0  0]
```

```
>>> print(ones.shape)

    (2, 3)

>>> print(zeros.shape)

    (3,)
```

Adding Rows and Columns to ndarrays (concatenate)

```
>>> zeros_2d = np.expand_dims(zeros,axis=0)

>>> print(zeros_2d)

    [[ 0  0  0]]

>>> print(zeros_2d.shape)

    (1, 3)
```

Adding Rows and Columns to ndarrays (concatenate)

```
>>> combined = np.concatenate([ones,zeros_2d],axis=0)

>>> print(combined)

    [[ 1  1  1]
     [ 1  1  1]
     [ 0  0  0]]
```

Sorting ndarrays

```
fruit = np.array(['orange', 'banana',
                  'apple', 'grape',
                  'cherry'])
```

orange	0
banana	1
apple	2
grape	3
cherry	4

```
sorted_fruit = fruit[sorted_order]
```

apple
banana
cherry
grape
orange

```
sorted_order = np.argsort(fruit)
```

2	1	4	3	0

Lesson #02.ipynb

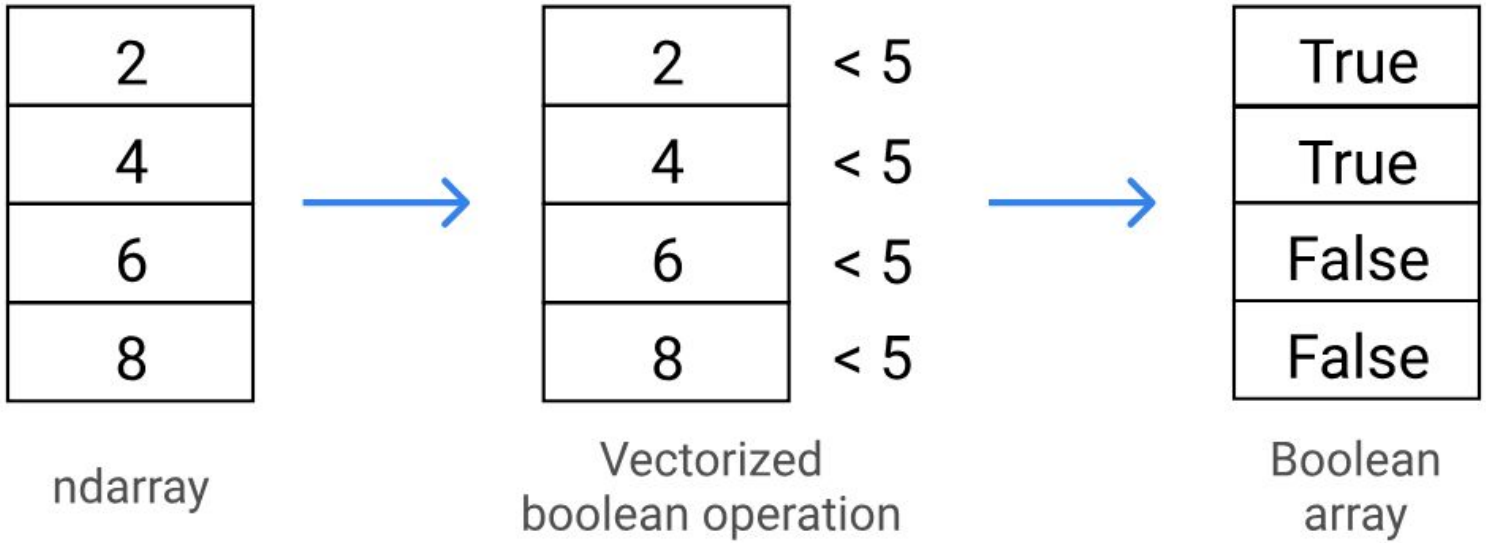
- Section 2

Reading CSV files from Numpy

```
taxi = np.genfromtxt('nyc_taxis.csv', delimiter=',')
print(taxi)
```

```
[[   nan    nan    nan ...,    nan    nan    nan]
 [  2016      1      1 ...,  11.65  69.99      1]
 [  2016      1      1 ...,      8   54.3      1]
 ...,
 [  2016      6     30 ...,      5  63.34      1]
 [  2016      6     30 ...,   8.95  44.75      1]
 [  2016      6     30 ...,      0  54.84      2]]
```

Slicing from boolean arrays



Boolean indexing with 1D ndarrays

```
c = np.array([80.0, 103.4,
              96.9, 200.3])
```

80.0
103.4
96.6
200.3

c

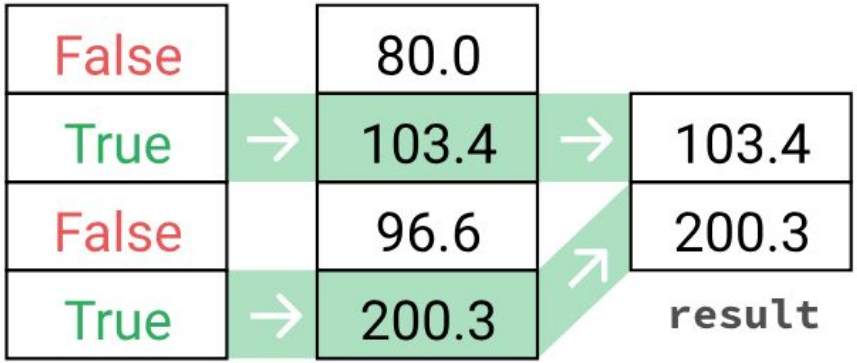
```
c_bool = c > 100
```

False
True
False
True

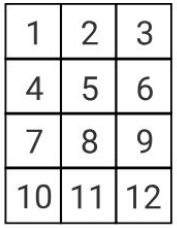
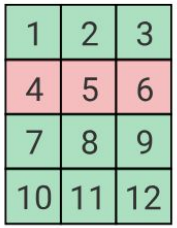
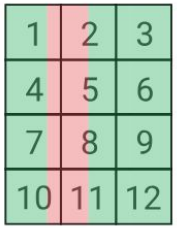
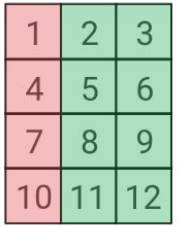
c_bool

Boolean indexing with 1D ndarrays

```
result = c[c_bool]
```



Boolean Indexing with 2D ndarrays

Code	Visualization	Explanation
`arr = np.array([ [ 1, 2, 3], [ 4, 5, 6], [ 7, 8, 9], [10, 11, 12] ]) print(arr)`		The original array
`bool_1 = [True, False, True, True] print(arr[bool_1])`		`bool_1`'s shape (4) is the same as the shape of `arr`'s first axis (4), so this selects the 1st, 3rd, and 4th rows.
`print(arr[:,bool_1])`		`bool_1`'s shape (4) is not the same as the shape of `arr`'s second axis (3), so it can't be used to index and produces an `error`
`bool_2 = [False, True, True] print(arr[:,bool_2])`		`bool_2`'s shape (3) is the same as the shape of `arr`'s second axis (3), so this selects the 2nd and 3rd columns.

Assigning values in 1D ndarray

```
a = np.array(['red','blue','black','blue','purple'])
a[0] = 'orange'
print(a)

['orange', 'blue', 'black', 'blue', 'purple']
```

```
a[3:] = 'pink'
print(a)

['orange', 'blue', 'black', 'pink', 'pink']
```

Assigning values in 2D ndarray

```
ones = np.array([[1, 1, 1, 1, 1],
                 [1, 1, 1, 1, 1],
                 [1, 1, 1, 1, 1]])
ones[1,2] = 99
print(ones)
```

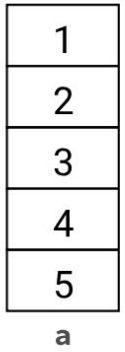
```
[[ 1,  1,  1,  1,  1],
 [ 1,  1, 99,  1,  1],
 [ 1,  1,  1,  1,  1]]
```

```
ones[0] = 42
print(ones)
```

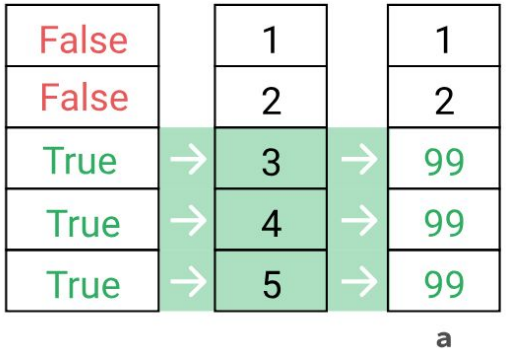
```
[[42, 42, 42, 42, 42],
 [ 1,  1, 99,  1,  1],
 [ 1,  1,  1,  1,  1]]
```

Assignment Using Boolean Arrays

```
a = np.array([1, 2, 3, 4, 5])
```




```
a[a > 2] = 99
```



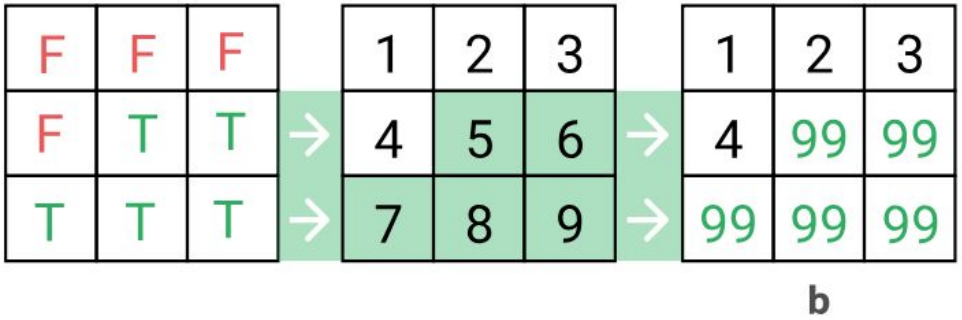
Assignment Using Boolean Arrays

```
b = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])
```

1	2	3
4	5	6
7	8	9

b

```
b[b > 4] = 99
```



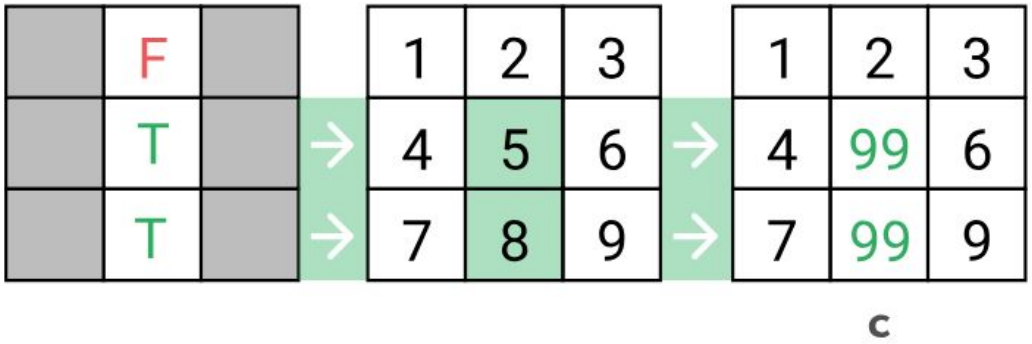
Assignment Using Boolean Arrays

```
c = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])
```

1	2	3
4	5	6
7	8	9

c

```
c[c[:, 1] > 2, 1] = 99
```



Challenges

Which is the most popular airport?
Calculating statistics for trip?

Lesson #02.ipynb

- Section 3

Understanding Pandas & Numpy

- Numpy
  a. Lack support for column names
  b. Support for only one data type per ndarray
  c. There are lots of low level methods, however there are many common analysis patterns that don't have pre-built methods.

The pandas library provides solutions to all of these pain points and more. Pandas is not so much a replacement for NumPy as an extension of NumPy.

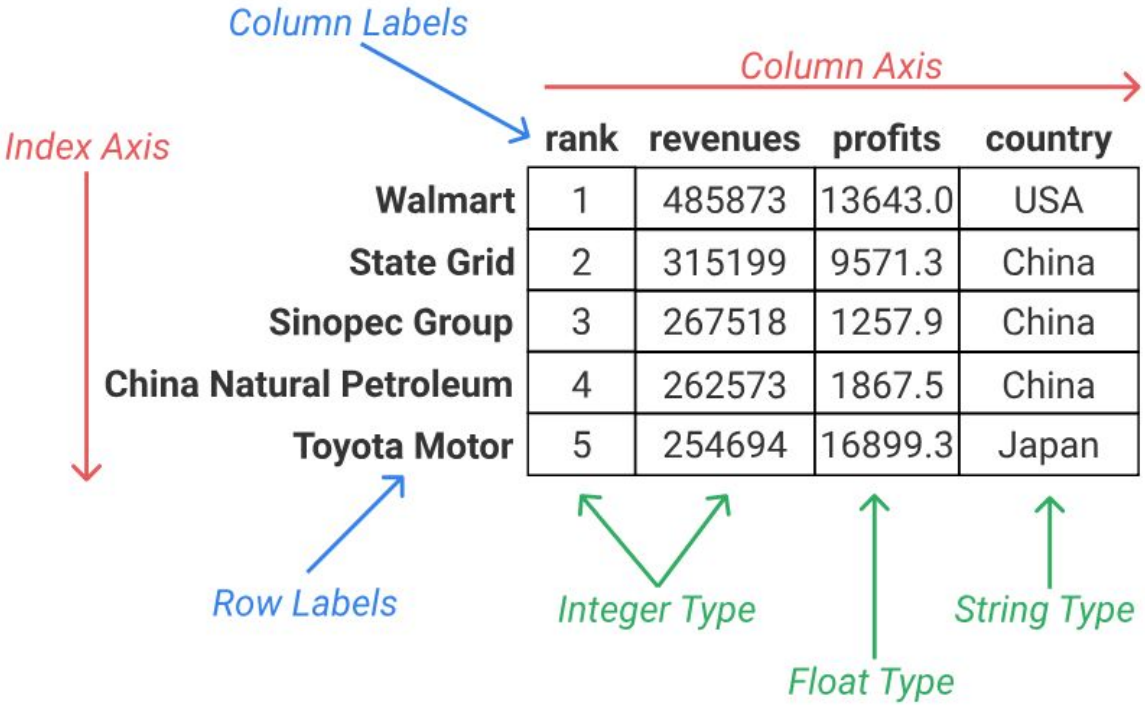

THE WORLD'S LARGEST CORPORATIONS
FORTUNE
GLOBAL
500

The dataset

	rank	revenues	revenue_change	profits	assets	profit_change	ceo	industry	sector	previous_rank
Walmart	1	485873	0.8	13643.0	198825	-7.2	C. Douglas McMillon	General Merchandisers	Retailing	1
State Grid	2	315199	-4.4	9571.3	489838	-6.2	Kou Wei	Utilities	Energy	2
Sinopec Group	3	267518	-9.1	1257.9	310726	-65.0	Wang Yupu	Petroleum Refining	Energy	4
China National Petroleum	4	262573	-12.3	1867.5	585619	-73.7	Zhang Jianhua	Petroleum Refining	Energy	3
Toyota Motor	5	254694	7.7	16899.3	437575	-12.3	Akio Toyoda	Motor Vehicles and Parts	Motor Vehicles & Parts	8

```
import pandas as pd
f500 = pd.read_csv("f500.csv", index_col=0)
f500.index.name = None
```

Introducing Dataframes



Introducing Dataframes

```
# put your code here
f500.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 500 entries, Walmart to AutoNation
Data columns (total 16 columns):
rank                          500 non-null int64
revenues                      500 non-null int64
revenue_change                498 non-null float64
profits                       499 non-null float64
assets                        500 non-null int64
profit_change                 436 non-null float64
ceo                           500 non-null object
industry                      500 non-null object
sector                        500 non-null object
previous_rank                 500 non-null int64
country                       500 non-null object
hq_location                   500 non-null object
website                       500 non-null object
years_on_global_500_list      500 non-null int64
employees                     500 non-null int64
total_stockholder_equity      500 non-null int64
dtypes: float64(3), int64(7), object(6)
memory usage: 66.4+ KB
```

f500.head()
f500.tail()

Selecting Columns From a Dataframe by label

```
f500_selection
```

	rank	revenues	profits	country
Walmart	1	485873	13643.0	USA
State Grid	2	315199	9571.3	China
Sinopec Group	3	267518	1257.9	China
China Natural Petroleum	4	262573	1867.5	China
Toyota Motor	5	254694	16899.3	Japan

```
f500_selection.loc[:,"rank"]
```

Walmart	1
State Grid	2
Sinopec Group	3
China Natural Petroleum	4
Toyota Motor	5

Selecting Columns From a Dataframe by label

```
f500_selection
```

	rank	revenues	profits	country
Walmart	1	485873	13643.0	USA
State Grid	2	315199	9571.3	China
Sinopec Group	3	267518	1257.9	China
China Natural Petroleum	4	262573	1867.5	China
Toyota Motor	5	254694	16899.3	Japan

```
f500_selection.loc[:,["country","rank"]]
```

	country	rank
Walmart	USA	1
State Grid	China	2
Sinopec Group	China	3
China Natural Petroleum	China	4
Toyota Motor	Japan	5

Selecting Columns From a Dataframe by label

```
f500_selection
```

	rank	revenues	profits	country
Walmart	1	485873	13643.0	USA
State Grid	2	315199	9571.3	China
Sinopec Group	3	267518	1257.9	China
China Natural Petroleum	4	262573	1867.5	China
Toyota Motor	5	254694	16899.3	Japan

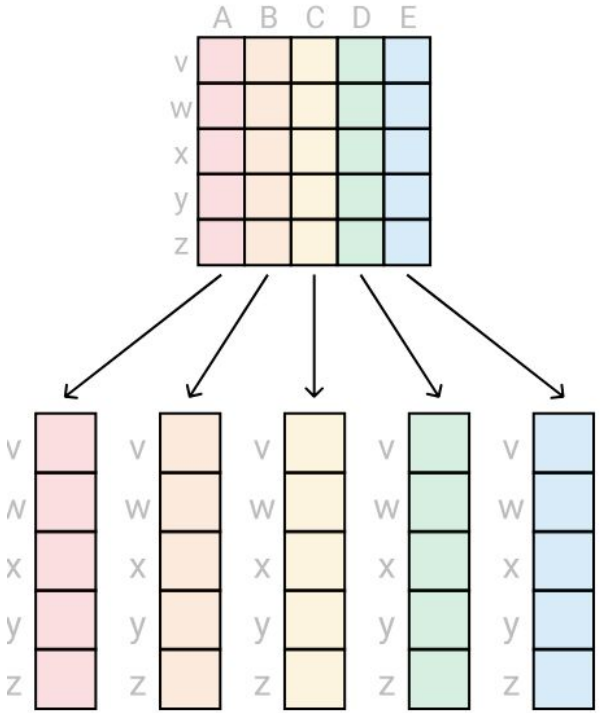
```
f500_selection.loc[:,"rank":"profits"]
```

	rank	revenues	profits
Walmart	1	485873	13643.0
State Grid	2	315199	9571.3
Sinopec Group	3	267518	1257.9
China Natural Petroleum	4	262573	1867.5
Toyota Motor	5	254694	16899.3

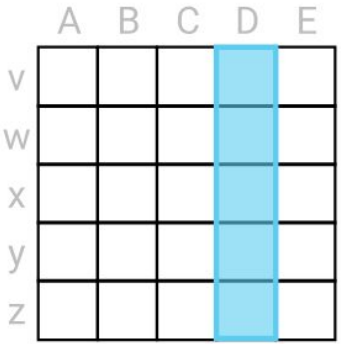
Column selection shortcuts

Select by Label	Explicit Syntax	Common Shorthand	Other Shorthand
Single column	`df.loc[:,"col1"]`	`df["col1"]`	`df.col1`
List of columns	`df.loc[:,["col1", "col7"]]`	`df[["col1", "col7"]]`	
Slice of columns	`df.loc[:,"col1":"col4"]`		

Selecting Items from a Series by Label



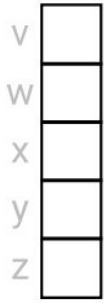
Original Dataframe



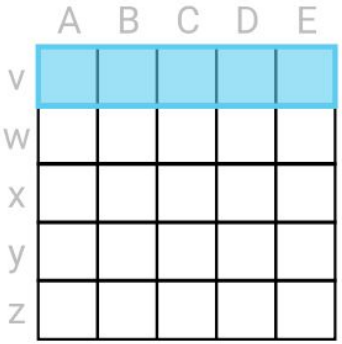
Code

```
single_col = df["D"]
```

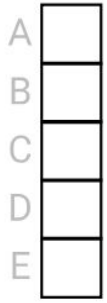
Result



single_col is a series object

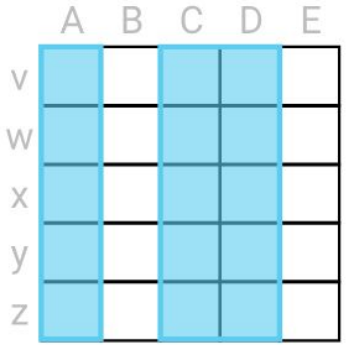



```
single_row = df.head(1)
```



single_row is a series object

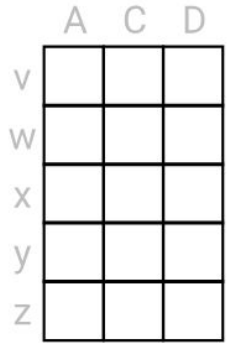
Original Dataframe



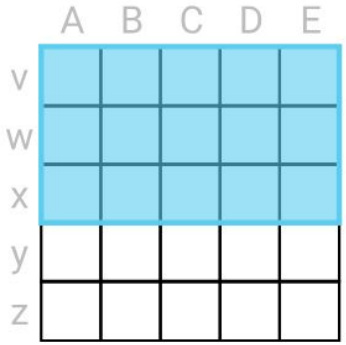
Code

```
multi_cols = df[["A", "C", "D"]]
```

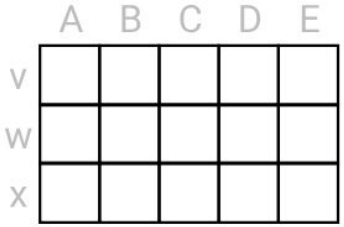
Result



multi_cols is a dataframe object




```
multi_rows = df.head(3)
```



multi_rows is a dataframe object

Dataframe vs Series

	Series	DataFrame
Dimensions	One	Two
Has 'index' axis	Yes	Yes
Has 'columns' axis	No	Yes
Number of dtypes	One	Many (one per column)

Series and Dataframe Describe Methods

```
revs = f500["revenues"]
print(revs.describe())
```

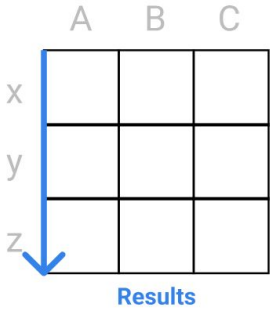
```
count       500.000000
mean      55416.358000
std       45725.478963
min       21609.000000
25%       29003.000000
50%       40236.000000
75%       63926.750000
max      485873.000000
Name: revenues, dtype: float64
```

```
print(f500["assets"].describe())
```

```
count    5.000000e+02
mean     2.436323e+05
std      4.851937e+05
min      3.717000e+03
25%      3.658850e+04
50%      7.326150e+04
75%      1.805640e+05
max      3.473238e+06
Name: assets, dtype: float64
```

`DataFrame.method(axis=0)`
or
`DataFrame.method(axis="index")`

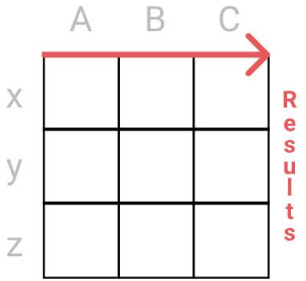
Calculates along the **row** axis



Calculates result for each **column**.

`DataFrame.method(axis=1)`
or
`DataFrame.method(axis="column")`

Calculates along the **column** axis



Calculates result for each **row**.

More data exploration methods

```
medians = f500[["revenues", "profits"]].median(axis=0)
# we could also use .median(axis="index")
print(medians)

revenues    40236.0
profits      1761.6
dtype: float64
```

```
>>> print(top5_rank_revenue)
                              rank  revenues
    Walmart                      1    485873
    State Grid                   2    315199
    Sinopec Group                3    267518
    China National Petroleum     4    262573
    Toyota Motor                 5    254694

>>> top5_rank_revenue["revenues"] = 0

>>> print(top5_rank_revenue)
                              rank  revenues
    Walmart                      1         0
    State Grid                   2         0
    Sinopec Group                3         0
    China National Petroleum     4         0
    Toyota Motor                 5         0
```

Assignment with Pandas

Assignment with Pandas

```
>>> top5_rank_revenue.loc["Sinopec Group", "revenues"] = 999

>>> print(top5_rank_revenue)
                              rank  revenues
    Walmart                      1         0
    State Grid                   2         0
    Sinopec Group                3       999
    China National Petroleum     4         0
    Toyota Motor                 5         0
```

Add a new column

```
>>> top5_rank_revenue["year_founded"] = 0

>>> print(top5_rank_revenue)
```

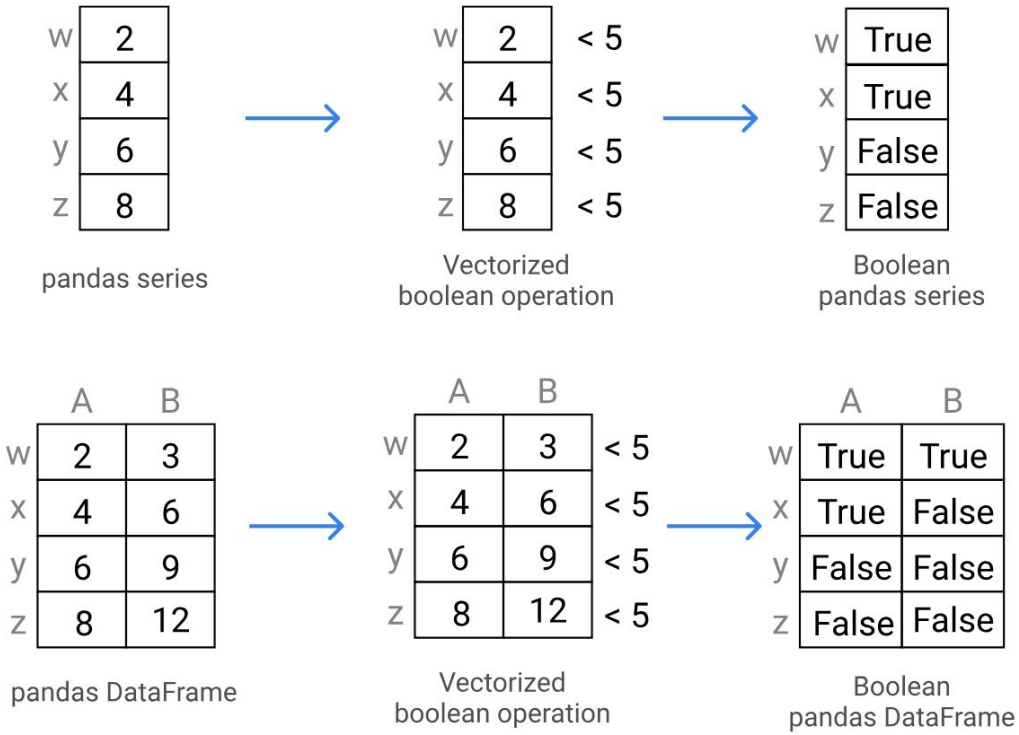
	rank	revenues	year_founded
Walmart	1	0	0
State Grid	2	0	0
Sinopec Group	3	999	0
China National Petroleum	4	0	0
Toyota Motor	5	0	0

Add a new row

```
>>> top5_rank_revenue.loc["My New Company"] = 555

>>> print(top5_rank_revenue)
```

	rank	revenues	year_founded
Walmart	1	0	0
State Grid	2	0	0
Sinopec Group	3	999	0
China National Petroleum	4	0	0
Toyota Motor	5	0	0
My New Company	555	555	555

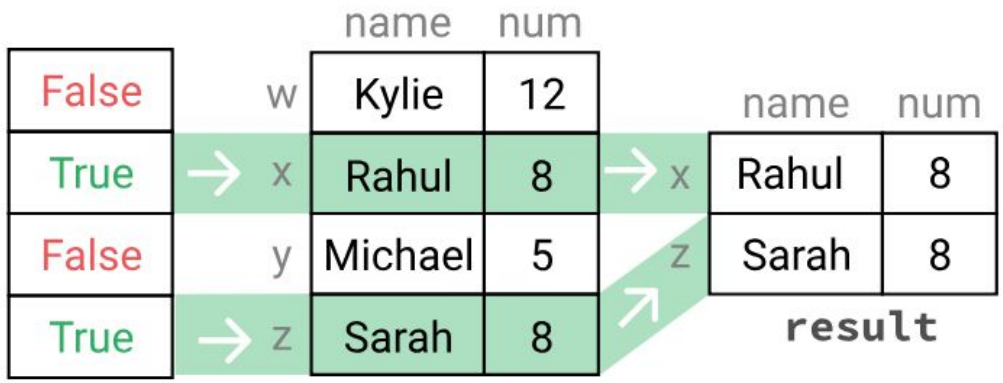
Using boolean indexing with pandas objects



Using boolean indexing with pandas objects

```
result = df.loc[num_bool, "name"]
```


```
result = df[num_bool]
```



Using boolean arrays to assign values

```
f500.loc[f500["sector"] == "Motor Vehicles & Parts","sector"] = "Motor Vehicles and Parts"
```

Challenge

Finding top performers by country

```
>>> top_3_countries = f500["country"].value_counts().head(3)

>>> print(top_3_countries)

USA       132
China     109
Japan      51
Name: country, dtype: int64
```