GENERAL ASSEMBLY

DATA SCIENCE

LECTURE 2: DATA FORMAT, ACCESS & TRANSFORMATION

FRANCESCO MOSCONI / ROB HALL / DAT-16

RECAP

LAST TIME:

I. DATA SCIENCE

II. DATA SCIENTIST

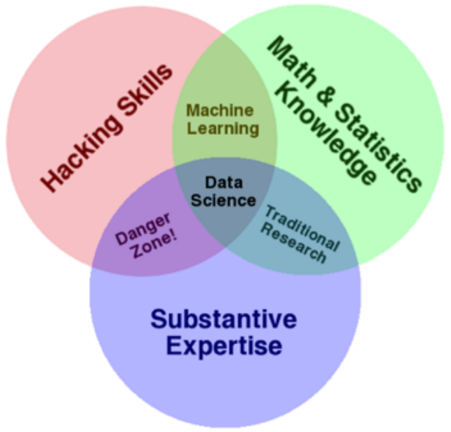
III. DATA MINING WORKFLOW

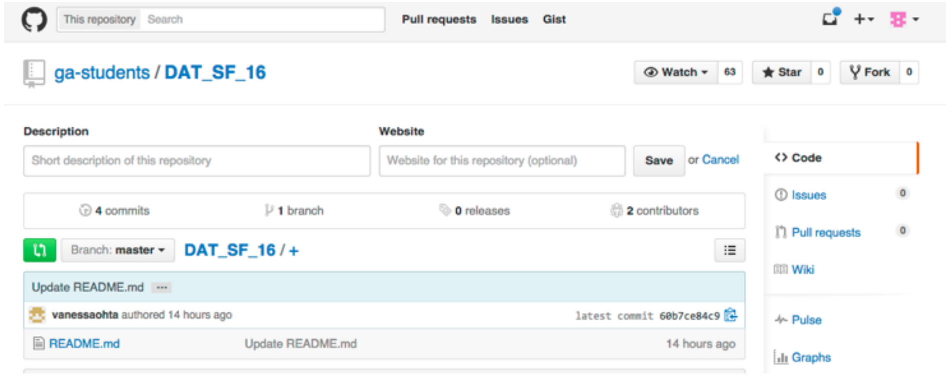
IV. GIT & GITHUB

EXERCISES:

V. I-PYTHON NOTEBOOK INTRO

QUESTIONS?





ga-students / DAT_SF_16

Description: Short description of this repository

Website: Website for this repository (optional) Save or Cancel

4 commits 1 branch 0 releases 2 contributors

Branch: master DAT_SF_16 / +

Update README.md ...

vanessaohta authored 14 hours ago latest commit 60b7ce84c9

README.md Update README.md 14 hours ago

AGENDA

I. DATA SOURCES
II. APIS
III. PYTHON QUICK REVIEW

EXERCISES:
IV. PANDAS
V. EXTRACTING DATA FROM API

WHERE DOES THE DATA COME FROM?

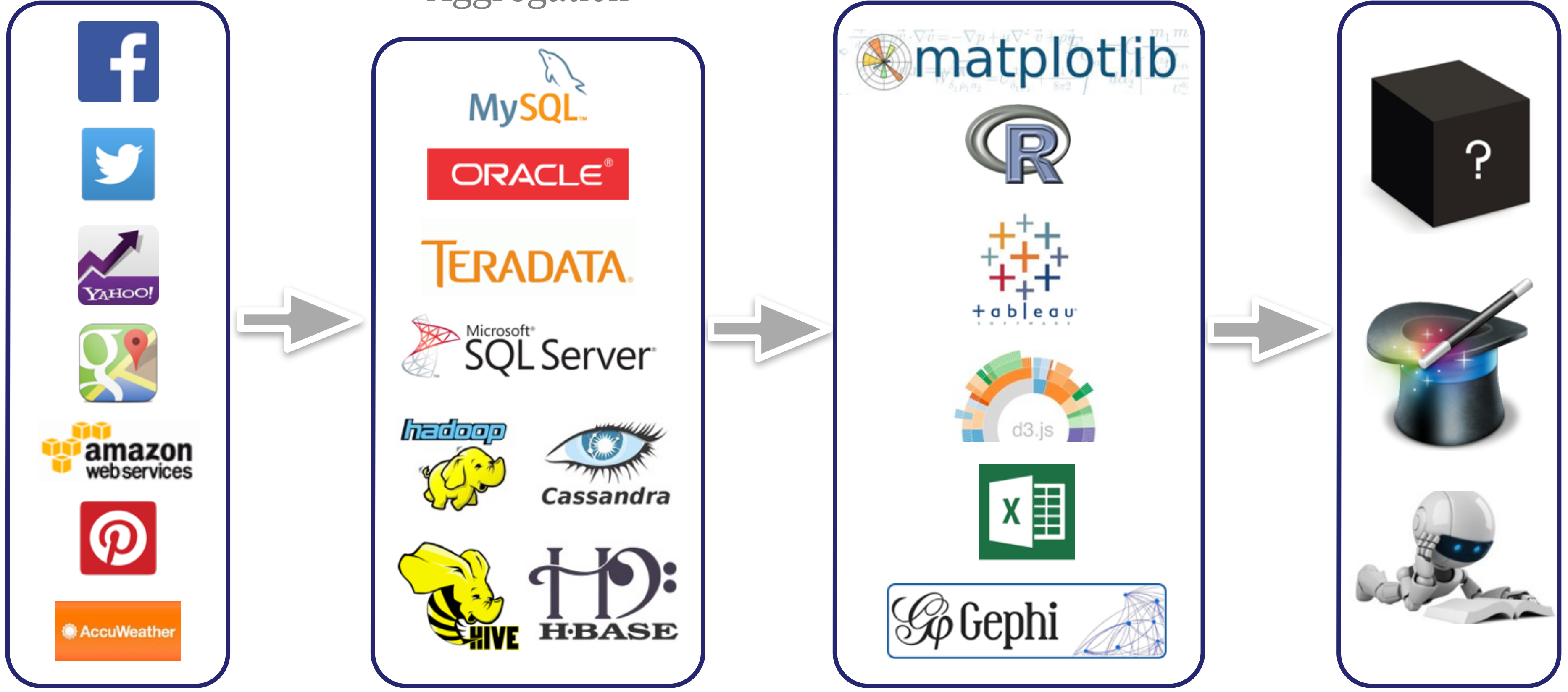
DATA FLOW

Data Retrieval

Data ETL and Aggregation

Data Visualization

Machine Learning



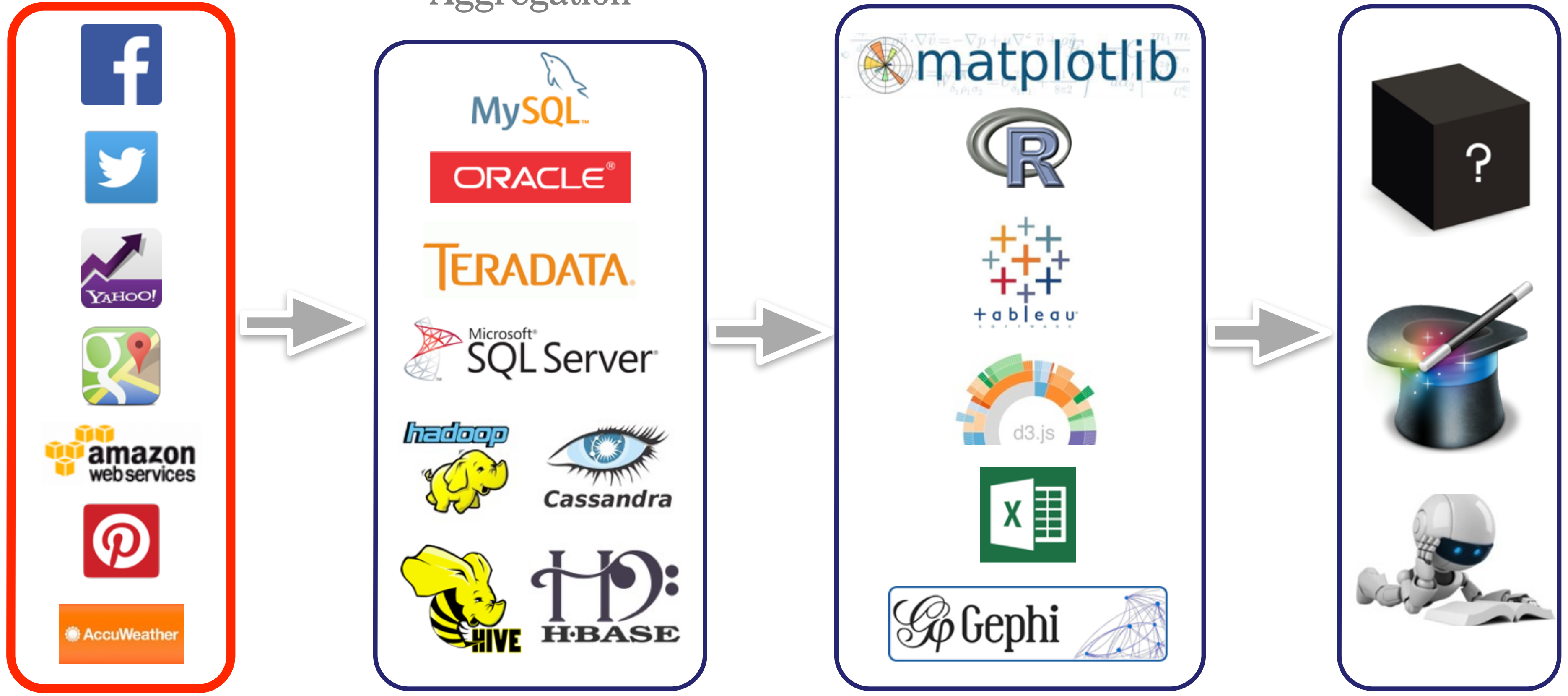
DATA FLOW

Data Retrieval

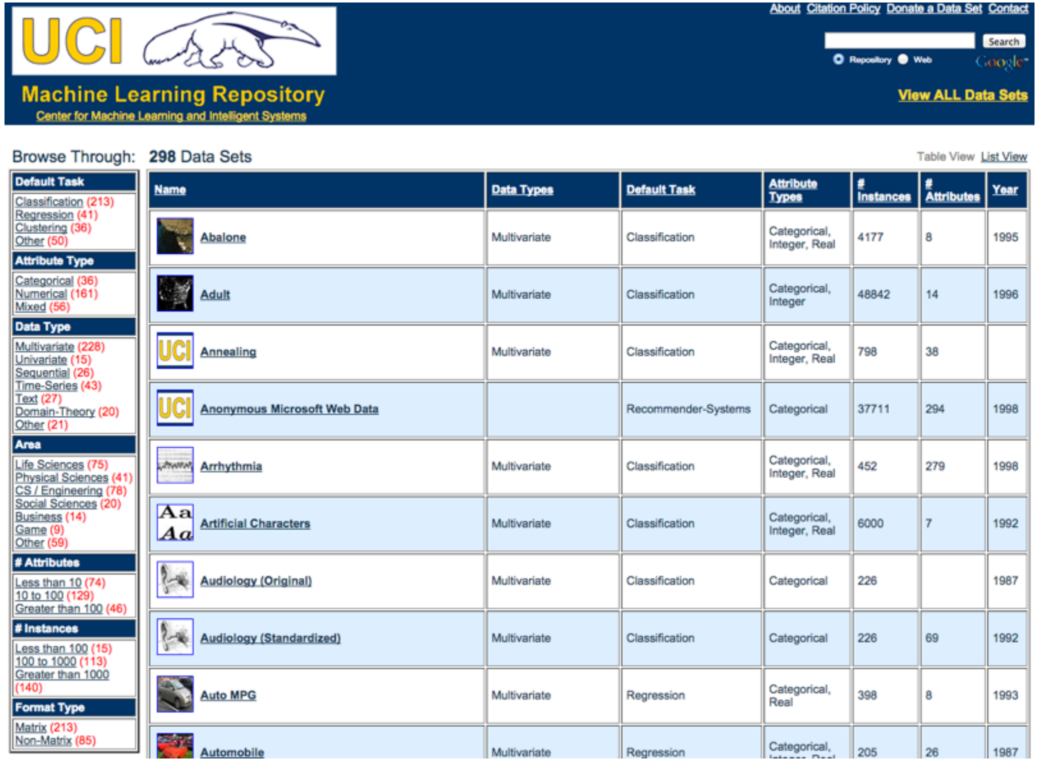
Data ETL and Aggregation

Data Visualization

Machine Learning



DATA SOURCES



About Citation Policy Donate a Data Set Contact

Search

Repository Web

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

View ALL Data Sets

Browse Through: 298 Data Sets

Table View List View

Default Task
- Classification (213)
- Regression (41)
- Clustering (36)
- Other (50)

Attribute Type
- Categorical (36)
- Numerical (161)
- Mixed (56)

Data Type
- Multivariate (228)
- Univariate (15)
- Sequential (26)
- Time-Series (43)
- Text (27)
- Domain-Theory (20)
- Other (21)

Area
- Life Sciences (75)
- Physical Sciences (41)
- CS / Engineering (78)
- Social Sciences (20)
- Business (14)
- Game (9)
- Other (59)

# Attributes
- Less than 10 (74)
- 10 to 100 (129)
- Greater than 100 (46)

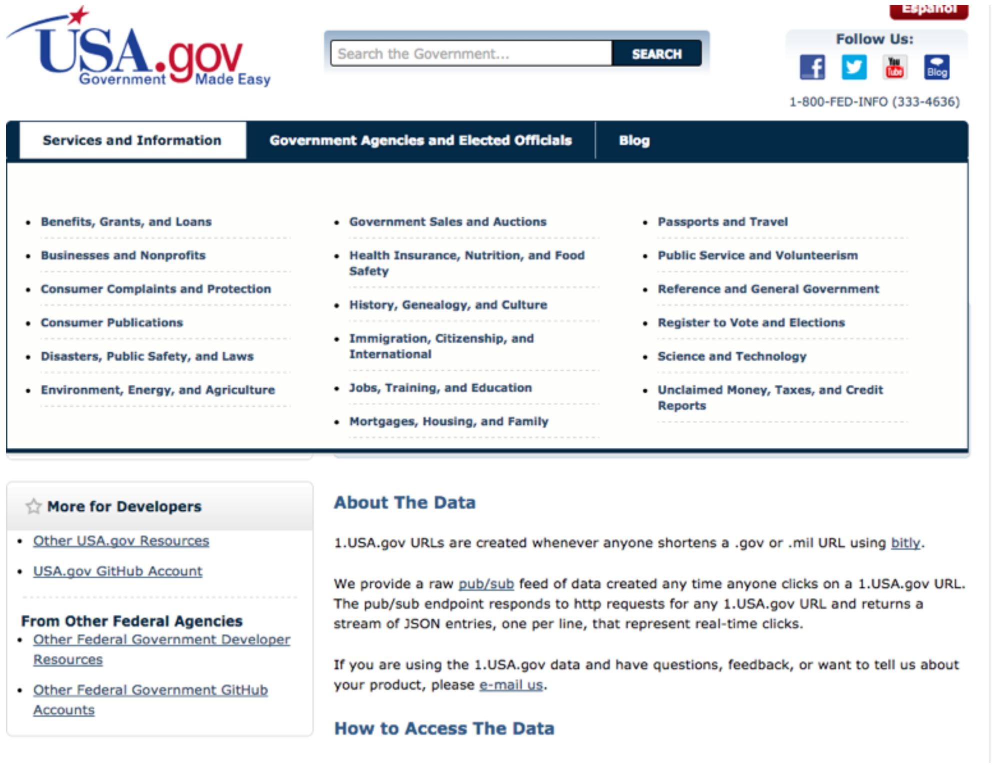
# Instances
- Less than 100 (15)
- 100 to 1000 (113)
- Greater than 1000 (140)

Format Type
- Matrix (213)
- Non-Matrix (85)

Name	Data Types	Default Task	Attribute Types	# Instances	# Attributes	Year
Abalone	Multivariate	Classification	Categorical, Integer, Real	4177	8	1995
Adult	Multivariate	Classification	Categorical, Integer	48842	14	1996
Annealing	Multivariate	Classification	Categorical, Integer, Real	798	38	
Anonymous Microsoft Web Data		Recommender-Systems	Categorical	37711	294	1998
Arrhythmia	Multivariate	Classification	Categorical, Integer, Real	452	279	1998
Artificial Characters	Multivariate	Classification	Categorical, Integer, Real	6000	7	1992
Audiology (Original)	Multivariate	Classification	Categorical	226		1987
Audiology (Standardized)	Multivariate	Classification	Categorical	226	69	1992
Auto MPG	Multivariate	Regression	Categorical, Real	398	8	1993
Automobile	Multivariate	Regression	Categorical,	205	26	1987

Source: http://archive.ics.uci.edu/ml/datasets.html

DATA SOURCES


USA.gov Government Made Easy
Español
Search the Government...
SEARCH
Follow Us:
Blog
1-800-FED-INFO (333-4636)
Services and Information
Government Agencies and Elected Officials
Blog

- Benefits, Grants, and Loans
- Businesses and Nonprofits
- Consumer Complaints and Protection
- Consumer Publications
- Disasters, Public Safety, and Laws
- Environment, Energy, and Agriculture
- Government Sales and Auctions
- Health Insurance, Nutrition, and Food Safety
- History, Genealogy, and Culture
- Immigration, Citizenship, and International
- Jobs, Training, and Education
- Mortgages, Housing, and Family
- Passports and Travel
- Public Service and Volunteerism
- Reference and General Government
- Register to Vote and Elections
- Science and Technology
- Unclaimed Money, Taxes, and Credit Reports

More for Developers

- Other USA.gov Resources
- USA.gov GitHub Account

From Other Federal Agencies

- Other Federal Government Developer Resources
- Other Federal Government GitHub Accounts

About The Data

1.USA.gov URLs are created whenever anyone shortens a .gov or .mil URL using bitly.

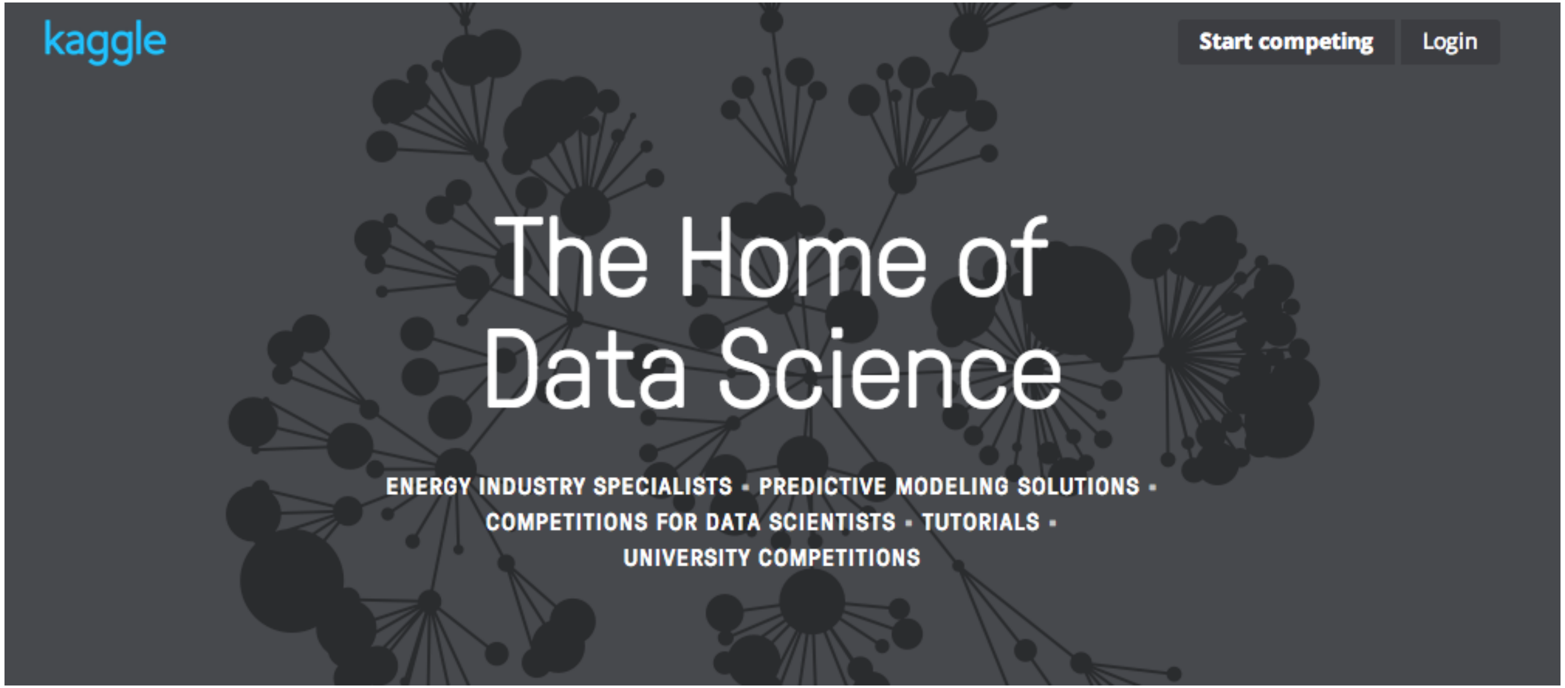
We provide a raw pub/sub feed of data created any time anyone clicks on a 1.USA.gov URL. The pub/sub endpoint responds to http requests for any 1.USA.gov URL and returns a stream of JSON entries, one per line, that represent real-time clicks.

If you are using the 1.USA.gov data and have questions, feedback, or want to tell us about your product, please e-mail us.

How to Access The Data

Source: http://www.usa.gov/About/developer-resources/1usagov.shtml

DATA SOURCES



The Home of Data Science

ENERGY INDUSTRY SPECIALISTS ▪ PREDICTIVE MODELING SOLUTIONS ▪ COMPETITIONS FOR DATA SCIENTISTS ▪ TUTORIALS ▪ UNIVERSITY COMPETITIONS

Source: *http://www.kaggle.com/*

DATA SOURCES

1) PETE SKOMOROCH (LINKEDIN) HTTPS://DELICIOUS.COM/PSKOMOROCH/DATASET

2) HILARY MASON (ACCEL PARTNERS, BITLY) HTTPS://BITLY.COM/BUNDLES/HMASON/1

3) KEVIN CHAI (U. OF NEW SOUTH WALES, SYDNEY) HTTP://KEVINCHAI.NET/DATASETS

4) JEFF HAMMERBACHER (CLOUDERA) HTTP://WWW.QUORA.COM/JEFF-HAMMERBACHER/INTRODUCTION-TO-DATA-SCIENCE-DATA-SETS

5) JERRY SMITH (3I-MIND) HTTP://DATASCIENTISTINSIGHTS.COM/2013/10/07/DATA-REPOSITORIES-MOTHERS-MILK-FOR-DATA-SCIENTISTS/

6) GREGORY PIATETSKY-SHAPIRO (KDD) HTTP://WWW.KDNUGGETS.COM/DATASETS/INDEX.HTML

7) HTTP://WWW.QUORA.COM/DATA/WHERE-CAN-I-FIND-LARGE-DATASETS-OPEN-TO-THE-PUBLIC

8) HTTPS://GITHUB.COM/CAESAR0301/AWESOME-PUBLIC-DATASETS

Source: http://blog.mortardata.com/post/67652898761/6-dataset-lists-curated-by-data-scientists

PAIR EXERCISE:

CHOOSE A DATA SOURCE AND LOOK AT WHAT DATA YOU CAN GET

DISCUSS HOW YOU WOULD USE THE DATA

QUESTIONS?

JSON, CSV, ETC…

JSON (JavaScript Object Notation) is:

a lightweight data-interchange format

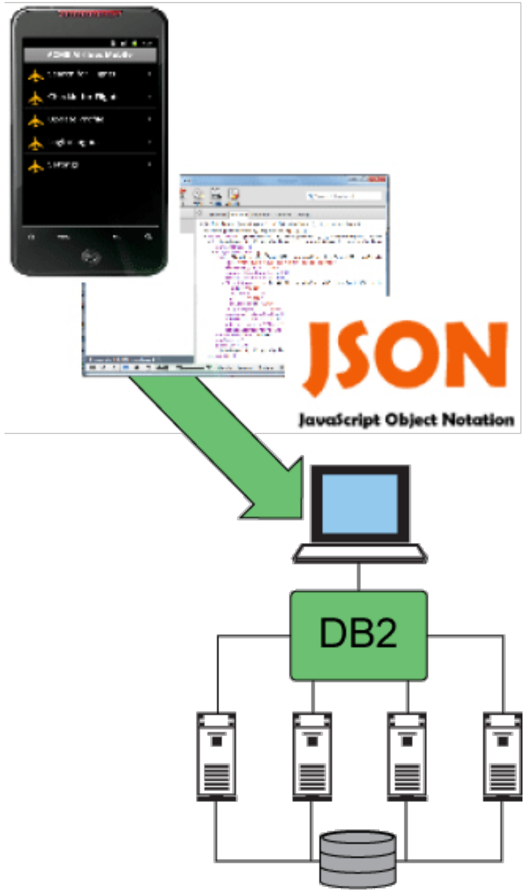
a string

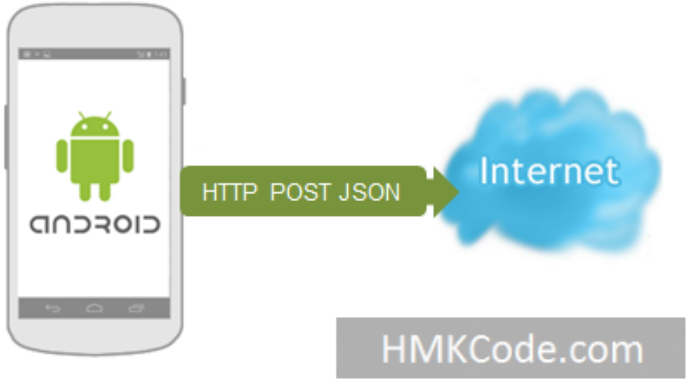
JSON

JSON can be passed

between applications

easy for machines to parse and generate





JSON are passed through applications

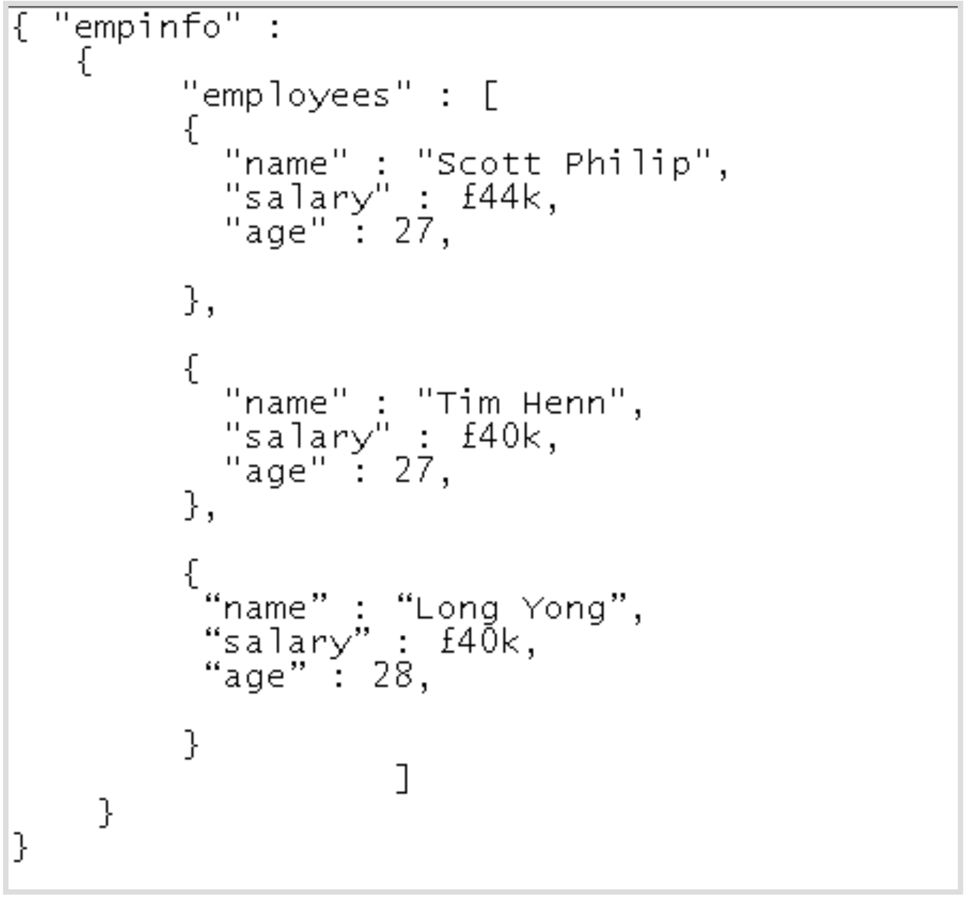
as strings

and converted into native objects per language.

JSON are passed through applications

as strings

and converted into native objects per language.



```
{ "empinfo" :
    {
           "employees" : [
           {
              "name" : "Scott Philip",
              "salary" : £44k,
              "age" : 27,

           },

           {
              "name" : "Tim Henn",
              "salary" : £40k,
              "age" : 27,
           },

           {
             “name” : “Long Yong”,
             “salary” : £40k,
             “age” : 28,

           }
                         ]
      }
}
```

JSON STRING ——> PYTHON OBJECT

```
import json
py_object = [ { 'a':'A', 'b':(2, 4), 'c':3.0 } ]
json_string = json.dumps(py_object)
print ‘JSON:’, json_string
```

```
JSON: [{"a": "A", "c": 3.0, "b": [2, 4]}]
```

PYTHON OBJECT ——> JSON STRING

```
decoded = json.loads(json_string)
```

PYTHON JSON LIBRARY

https://docs.python.org/2/library/json.html

CSV (Comma Separated Values):

```
name,game,points
John,basketball,3
Mary,volleyball,5
James,ping pong,2
…
```

CSV (Comma Separated Values):

- **easy to read and write**
- **structured like a table**
- **very common**
- **can export to/from MS Excel**

PYTHON CSV LIBRARY

https://docs.python.org/2/library/csv.html

OTHER DATA FORMATS

txt

tsv

xml

dat

images

binary

etc…

APIs

APIs (Application Programming Interface) allow people to interact with the structures of an application

- get
- put
- delete
- update
- …

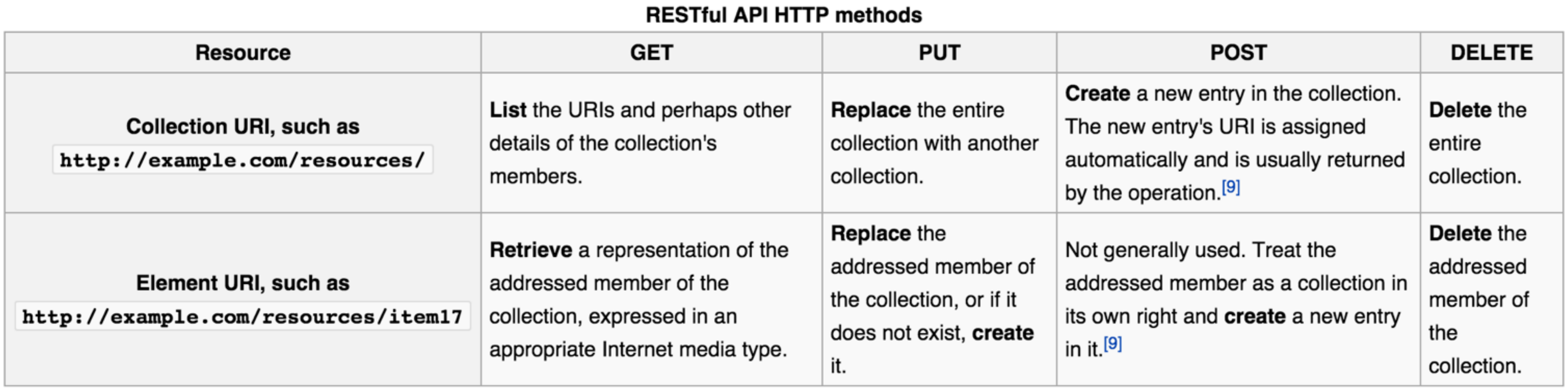
Best practices for APIs are to use RESTful principles.

Best practices for APIs are to use RESTful principles.

↓

Representational State Transfer (REST)

RESTFUL EXAMPLE



RESTful API HTTP methods

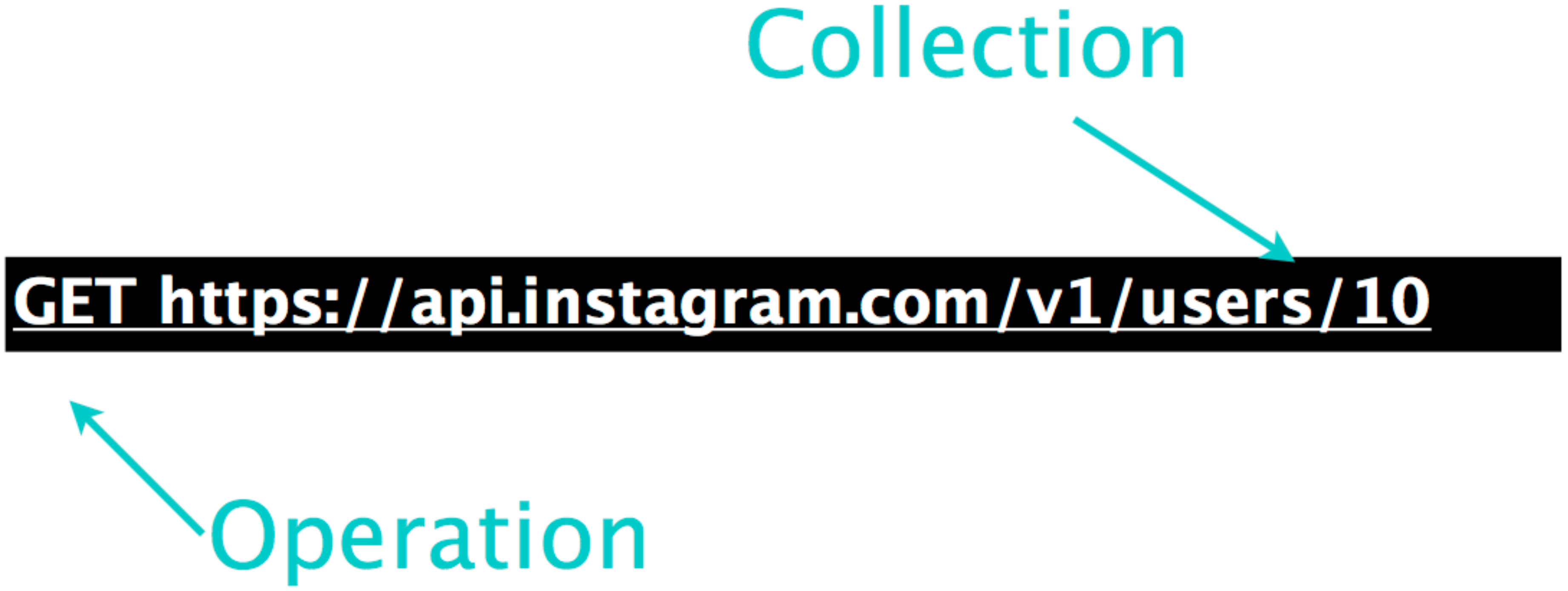
Resource	GET	PUT	POST	DELETE
Collection URI, such as `http://example.com/resources/`	**List** the URIs and perhaps other details of the collection's members.	**Replace** the entire collection with another collection.	**Create** a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation.[9]	**Delete** the entire collection.
Element URI, such as `http://example.com/resources/item17`	**Retrieve** a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	**Replace** the addressed member of the collection, or if it does not exist, **create** it.	Not generally used. Treat the addressed member as a collection in its own right and **create** a new entry in it.[9]	**Delete** the addressed member of the collection.

http://en.wikipedia.org/wiki/Representational_state_transfer

RESTFUL API

- The Base URL
- An interactive media type (usually JSON)
- Operations (GET, PUT, POST, DELETE)
- Driven by http requests

REST API EXAMPLE



```
GET https://api.instagram.com/v1/users/10
```

REST API EXAMPLE

```
GET https://api.instagram.com/v1/users/
search/?q=andy
```

Querystring

TWITTER REST API

https://dev.twitter.com/rest/public

LINKEDIN REST API

https://developer.linkedin.com/docs/signin-with-linkedin

LIST OF PYTHON APIS

http://www.pythonapi.com/

PAIR EXERCISE:

http://www.pythonapi.com/

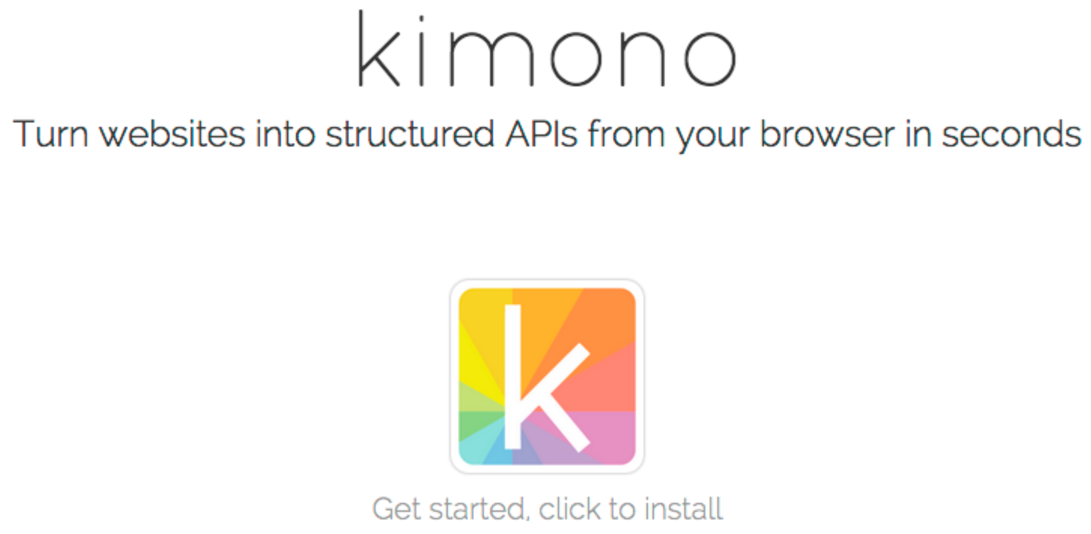
1) CHOOSE 1 API: WHAT DATA YOU CAN GET?

2) INSTALL PYTHON MODULE, TRY TO EXTRACT DATA

3) DISCUSS: HOW COULD YOU LEVERAGE THAT API? HOW COULD YOU USE THE DATA?

KIMONO LABS

www.kimonolabs.com



kimono

Turn websites into structured APIs from your browser in seconds

Get started, click to install

QUESTIONS?

PYTHON QUICK REVIEW

Q: What is Python?

A: An open source, high-level, dynamic scripting language.

- open source*: free! (both binaries and source files)*
- high-level*: interpreted (not compiled)*
- dynamic*: things that would typically happen at compile time happen at runtime instead (eg, dynamic typing)*
- scripting language*: "middle-weight"*

PEP 20: THE ZEN OF PYTHON

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
```

http://legacy.python.org/dev/peps/pep-0020/

Lets write a list of:

- *python data types*
- *python control flow statements*
- *misc python useful commands*

PYTHON DATA STRUCTURES

BASIC DATA STRUCTURES

The most basic data structure is the None *type. This is the equivalent of NULL in other languages.*

Basic numeric types:

1. int (< 2^{63}) / long (≥ 2^{63})*
   ** on 64-bit OS X/Linux, sys.maxint = 2**63-1*
2. float (a “decimal”)
3. bool (True/False) or (1/0)
4. complex (“imaginary”)

```
>>> type(None)
<type 'NoneType'>
>>> type(1)
<type ‘int’>
>>> type(2.5)
<type ‘float’>
>>> type(True)
<type ‘bool’>
>>> type(2+3j)
<type ‘complex’>
```

http://docs.python.org/2/library/stdtypes.html#numeric-types-int-float-long-complex

Array type, implemented in Python as a **list***.*

- *zero-base numbered, ordered collection of elements*
- *elements of arbitrary type.*
- mutable (*can be changed in-place)*

```
>>> a = [1,'b',True]
>>> a[2]
True
>>> a[1]='aa'
>>> a
[1,'aa',True]
```

http://docs.python.org/2/library/stdtypes.html#numeric-types-int-float-long-complex

Tuples*: immutable arrays of arbitrary elements.*

```
>>> x = (1,'a',2.5)
>>> x[0]
1
>>> x[0]='b'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> a,b = (1,2)
>>> a
1
```

Tuples are frequently used behind the scenes in a special type of variable assignment called tuple packing/unpacking*.*

http://docs.python.org/2/library/stdtypes.html#numeric-types-int-float-long-complex

The string *type*

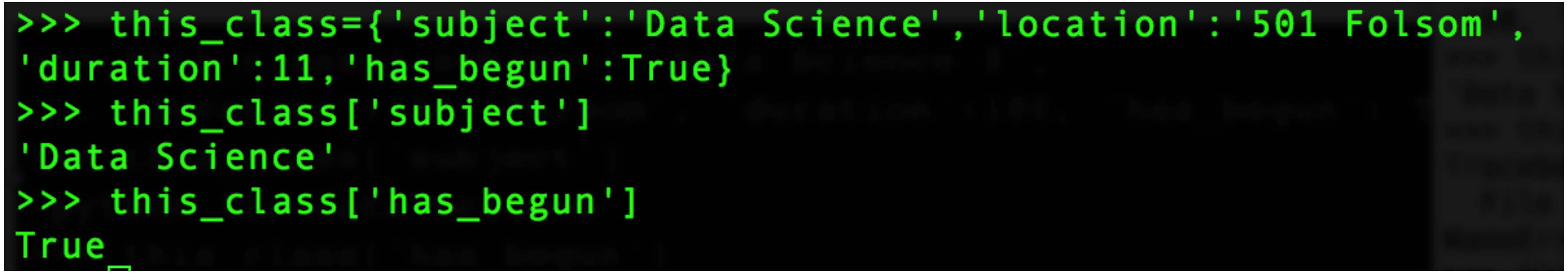
- *immutable ordered array of characters (note there is no char type).*
- *support slicing and indexing operations like arrays*
- *have many other string-specific functions as well*

String processing is one area where Python excels.

http://docs.python.org/2/library/stdtypes.html

dictionary *type*

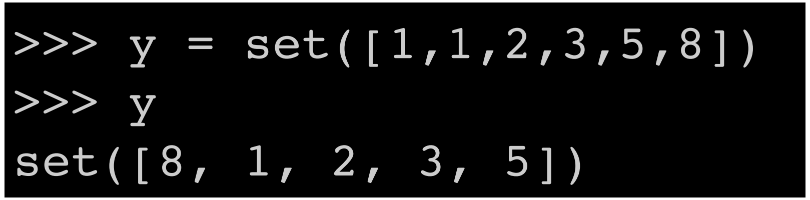
- *Associative arrays (or hash tables)*
- *unordered collections of* key-value pairs
- *keys must be immutable*



```
>>> this_class={'subject':'Data Science','location':'501 Folsom',
'duration':11,'has_begun':True}
>>> this_class['subject']
'Data Science'
>>> this_class['has_begun']
True
```

Sets

- *unordered mutable collections of distinct elements*
- *useful for checking membership of an element*
- *useful for ensuring element uniqueness*



```
>>> y = set([1,1,2,3,5,8])
>>> y
set([8, 1, 2, 3, 5])
```

http://docs.python.org/2/library/sets.html file object

e.g *open connection to a file*

```
>>> with open('output_file.txt','w') as f:
...     f.write('test')
```

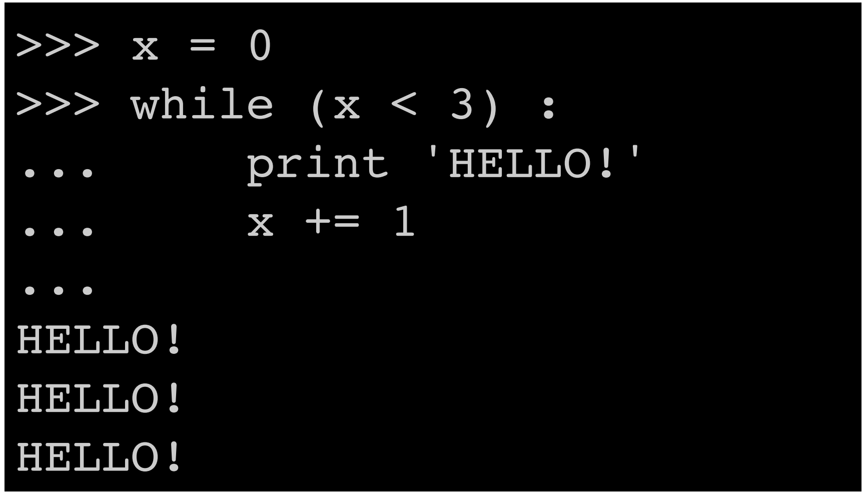
*note the "*with*" statement context manager, which automatically closes the file handle when it goes out of scope.*

PYTHON CONTROL FLOW

if-else allows to execute alternative statements based on conditions

```
>>> x, y = False, False
>>> if x :
...     Print 'x is True'
... elif y :
...     Print 'y is True'
... else :
...     Print 'Neither...'
...
Neither...
```

while loop *executes while a given condition evaluates to True*



```
>>> x = 0
>>> while (x < 3) :
...     print 'HELLO!'
...     x += 1
...
HELLO!
HELLO!
HELLO!
```

for loop *executes a block of code for a range of values*

```
>>> for k in range(4) :
...     print k**2
...
0
1
4
9
```

The object that a for loop iterates over is called (appropriately) an iterable.

try-except block

```
>>> try:
...     print undefined_variable
... except :
...     print 'An Exception has been caught'
...
An Exception has been caught
```

useful for catching and dealing with errors, also called exception handling.

custom functions

```
>>> def x_minus_3(x) :
...     return x - 3
...
>>> x_minus_3(12)
9
```

NOTE: Functions can optionally return a value with a return statement *(as this example does).*

Functions arguments *as inputs, and these arguments can be provided in two ways:*

1) as positional arguments*:*

```
>>> def f(x,y) :
...     return x - y
...
>>> f(4,2)
2
>>> f(2,4)
-2
```

2) as keyword arguments:

```
>>> def g(arg1=10, arg2=20) :
...     return arg1 / float(arg2)
...
>>> g()
0.5
>>> g(1,20)
0.05
>>> g(arg2=100)
0.1
```

Classes *with* ***member attributes*** *and* ***functions***:

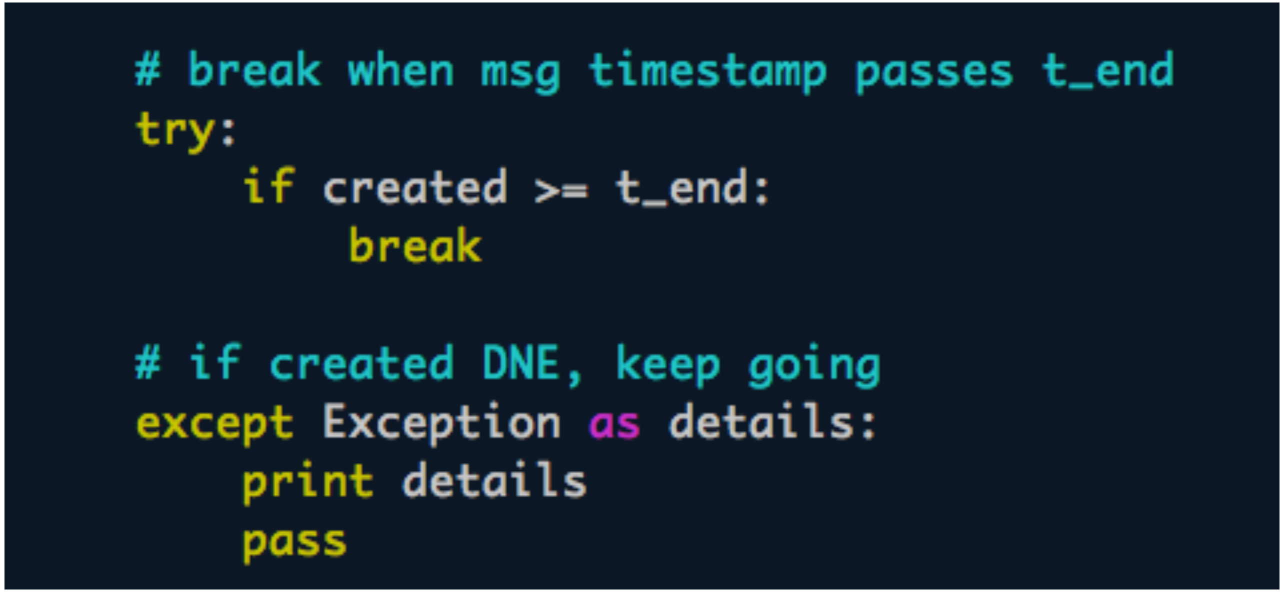
```
>>> from math import pi
>>>
>>> class Circle() :
...     def __init__(self, r=1) :
...         self.radius = r
...     def area(self) :
...         return pi * (self.radius ** 2)
...
>>> c=Circle(4)
>>> c.radius
4
>>> c.area()
50.26548245743669
>>> 3.141592653589793 * 4 * 4
50.26548245743669
```

***import** statement to load libraries and functions:*

```
>>> import math
>>> math.pi
3.141592653589793
>>> from math import sin
>>> sin(math.pi/2)
1.0
>>> from math import *
>>> print e, log10(1000), cos(pi)
2.71828182846 3.0 -1.0
```

The three methods differ with respect to the interaction with the local namespace.

Comments are very important to make your code readable to others



```
# break when msg timestamp passes t_end
try:
    if created >= t_end:
        break

# if created DNE, keep going
except Exception as details:
    print details
    pass
```

QUESTIONS?