Robert Dempsey

- 15+ years in tech with 8 years programming
- Built 3 businesses
- Currently specializing in data wrangling
- Learning more of the data science

Robert Dempsey

robertwdempsey.com

rdempsey

robertwdempsey

rdempsey

Course Overview

Data Acquisition & Wrangling

- **Part One: pulling and combining data from APIs**
- Part Two: web scraping

Pre-Requisites

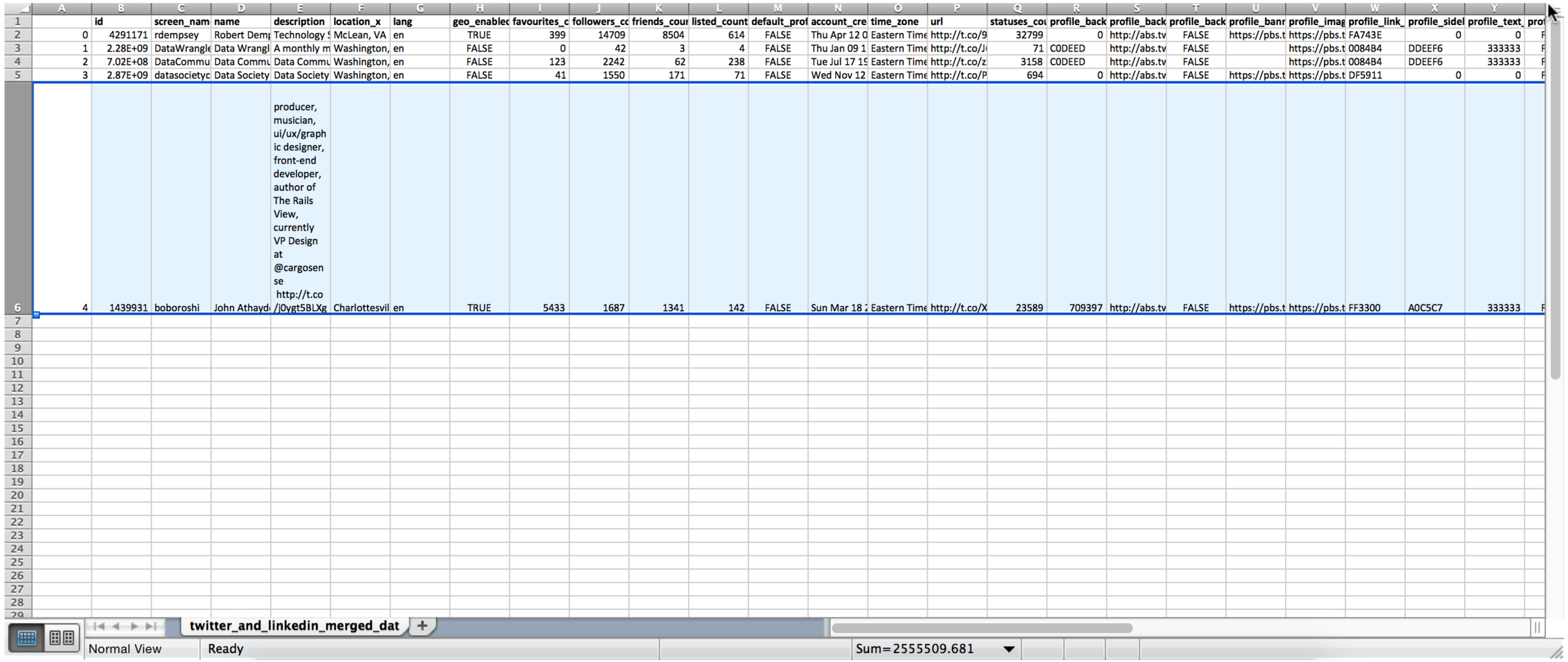
Module Pre-Requisites

- Some experience with Python
- Python 2.7.9 installed (Anaconda suggested)
- iPython Notebook
  - Install Command: pip install ipython

Goal

Understand more about your customers from their social profiles.

What We're Creating



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	
1		id	screen_nam	name	description	location_x	lang	geo_enablec	favourites_c	followers_cc	friends_cour	listed_count	default_prof	account_cre	time_zone	url	statuses_cou	profile_back	profile_back	profile_back	profile_banr	profile_imag	profile_link_	profile_sidel	profile_text_	prof
2	0	4291171	rdempsey	Robert Demp	Technology S	McLean, VA	en	TRUE	399	14709	8504	614	FALSE	Thu Apr 12 0	Eastern Time	http://t.co/9	32799	0	http://abs.tv	FALSE	https://pbs.t	https://pbs.t	FA743E	0	0	F
3	1	2.28E+09	DataWrangle	Data Wrangl	A monthly m	Washington,	en	FALSE	0	42	3	4	FALSE	Thu Jan 09 1	Eastern Time	http://t.co/J	71	C0DEED	http://abs.tv	FALSE		https://pbs.t	0084B4	DDEEF6	333333	F
4	2	7.02E+08	DataCommu	Data Commu	Data Commu	Washington,	en	FALSE	123	2242	62	238	FALSE	Tue Jul 17 19	Eastern Time	http://t.co/z	3158	C0DEED	http://abs.tv	FALSE		https://pbs.t	0084B4	DDEEF6	333333	F
5	3	2.87E+09	datasocietyc	Data Society	Data Society	Washington,	en	FALSE	41	1550	171	71	FALSE	Wed Nov 12	Eastern Time	http://t.co/P	694	0	http://abs.tv	FALSE	https://pbs.t	https://pbs.t	DF5911	0	0	F
6	4	1439931	boboroshi	John Athayd	producer, musician, ui/ux/graphic designer, front-end developer, author of The Rails View, currently VP Design at @cargosense http://t.co/j0ygt5BLXg	Charlottesvil	en	TRUE	5433	1687	1341	142	FALSE	Sun Mar 18 2	Eastern Time	http://t.co/X	23589	709397	http://abs.tv	FALSE	https://pbs.t	https://pbs.t	FF3300	A0C5C7	333333	F
7																										
8																										
9																										
10																										
11																										
12																										
13																										
14																										
15																										
16																										
17																										
18																										
19																										
20																										
21																										
22																										
23																										
24																										
25																										
26																										
27																										
28																										
29																										

twitter_and_linkedin_merged_dat

Normal View | Ready | Sum=2555509.681

Real-World Applications

- Personal Social Network Analysis
  - Compare Twitter followers to LinkedIn contacts
    - How much overlap is there?
    - Should you connect with more people on LinkedIn to create stronger contacts?
- Business Social Network Analysis
  - Compare your customer base to your Twitter followers
    - Are your best customers on Twitter?
    - Could you turn more customers into frequent customers if you connect with them on social media?

Introduction

Why This Is Useful

- An increasing amount of data is being made available via APIs
- Being able to obtain that data and combine it with other data provides more insights
- Creating a single profile of a customer, employee, or competitor provides a holistic view of that entity.

Use Case: miVEDiX

- Integrate disparate data sources
- Visualize complex information
- Social + location + sales data FTW!
- Much more

What You'll Learn

- Use Python to pull data from APIs
- Handle JSON data
- Creating a single record from multiple data sources

An Overview of Python

Overview

- “A programming language that lets you work quickly and integrate systems more effectively.”
- A scripting language – no compiling (unlike Java, C, C++)
- Open source
- Thousands of third-party modules for web and internet development, database access, desktop GUIs, scientific and numeric computing, and much more.
- Along with R it’s winning the data when it comes to all things data.
- Can be used to create individual scripts to desktop apps to enterprise-level applications.

For Loop

```
for x in range(0,3):
    print("We're on time {}".format(x))
```

```
We're on time 0
We're on time 1
We're on time 2
```

While Loop

```
x = 1
while x < 10:
    print("Now on {}".format(x))
    x += 1
```

```
Now on 1
Now on 2
Now on 3
Now on 4
Now on 5
Now on 6
Now on 7
Now on 8
Now on 9
```

If Statement

```
turkey = "tasty"

if turkey == "tasty":
    print("Turkey is tasty")
else:
    print("Turkey is not tasty")
```

```
Turkey is tasty
```

Defining And Calling A Function

```
def my_function(a,b):
  print(a,b)

my_function(1,2)
```

```
(1, 2)
```

Importing Libraries

```
import pandas as pd

import string

from fuzzywuzzy import fuzz
```

Python - Resources

- python.org – the main website
- learnpythonthehardway.org – beginner programming course (video/pdf)
- continuum.io - Anaconda installers and more
- pythonweekly.com - weekly Python newsletter chock full of what's new and awesome in the world of Python
- stackoverflow.com/questions/tagged/python - if you have a Python question it's probably already been answered

APIs Explained

What Is An API?

- Defines how two things are going to talk with each other
  - Software - Software
  - Hardware - Hardware
  - Software – Hardware
- Specifies the operations, inputs, outputs and underlying types
- Defines implementation-independent functionality
- Often come in the form of a library
- Can specify remote calls exposed to an API consumer
  - SOAP
  - REST

SOAP

- Simple Object Access Protocol
- Uses XML to communicate
- Less used today

SOAP - Request

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle=http://www.w3.org/2001/12/soap-encoding>
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

SOAP - Response

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle=http://www.w3.org/2001/12/soap-encoding>
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>
  </soap:Body>
</soap:Envelope>
```

REST

- Representational State Transfer
- Uses the HTTP verbs of GET, POST, PUT, DELETE, etc.
- Returns data in XML, JSON, CSV
- More used today

Example REST Calls

https://api.twitter.com/1.1/statuses/user_timeline.json

https://api.meetup.com/2/events

What is JSON?

- Stands for JavaScript Object Notation
- Commonly used data format we need to know in order to handle other APIs
- Programming language-independent data format
- Uses human-readable text to transmit data between a server and an application
- An alternative to XML
- Winning the day

JSON Example


```
{
  "breakfast_menu": {
    "items": [
      {
        "name": "Belgian Waffles",
        "price": "5.95",
        "description": "Two of our famous Belgian Waffles with plenty of real maple syrup",
        "calories": "650"
      },
      {
        "name": "Strawberry Belgian Waffles",
        "price": "7.95",
        "description": "Light Belgian waffles covered with strawberries and whipped cream",
        "calories": "900"
      }
    ]
  }
}
```

Other Types of Data

- XML
- CSV
- Excel
- PDF

XML

```
<breakfast_menu>
  <food>
    <name>Belgian Waffles</name>
    <price>$5.95/price>
    <description>Two of our famous Belgian Waffles with plenty of real maple syrup</description>
    <calories>650</calories>
  </food>
  <food>
    <name>Strawberry Belgian Waffles</name>
    <price>$7.95</price>
    <description>Light Belgian waffles covered with strawberries and whipped cream</description>
    <calories>900</calories>
  </food>
</breakfast_menu>
```

CSV

menu,type,name,price,description,calories

breakfast,food,”Belgian Waffles”,5.95,”Two of our famous Belgian Waffles with plenty of real maple syrup”,650

breakfast,food,”Strawberry Belgian Waffles”,7.95,”Light Belgian waffles covered with strawberries and whipped cream”,900

Excel

	D	E
	Price	**Description**
	$5.95	Two of our famous Belgian Waffles with plent
ian Waffles	$7.95	Light Belgian waffles covered with strawberrie

How To Pull Data From APIs with Python

The Real-World Issue

- Most of the data produced is outside of our company boundaries. We need to get that data.
- We're going to pull data from the LinkedIn and Twitter APIs

Pull Data From LinkedIn

Overview

- Use Python 2.7.9
- Create an application on LinkedIn
  - https://www.linkedin.com/secure/developer
- Install the python-linkedin library
  - https://github.com/ozgur/python-linkedin
- Write our code
- Save the output to a CSV for later

Create A LinkedIn Application

https://www.linkedin.com/secure/developer

Create A LinkedIn Application

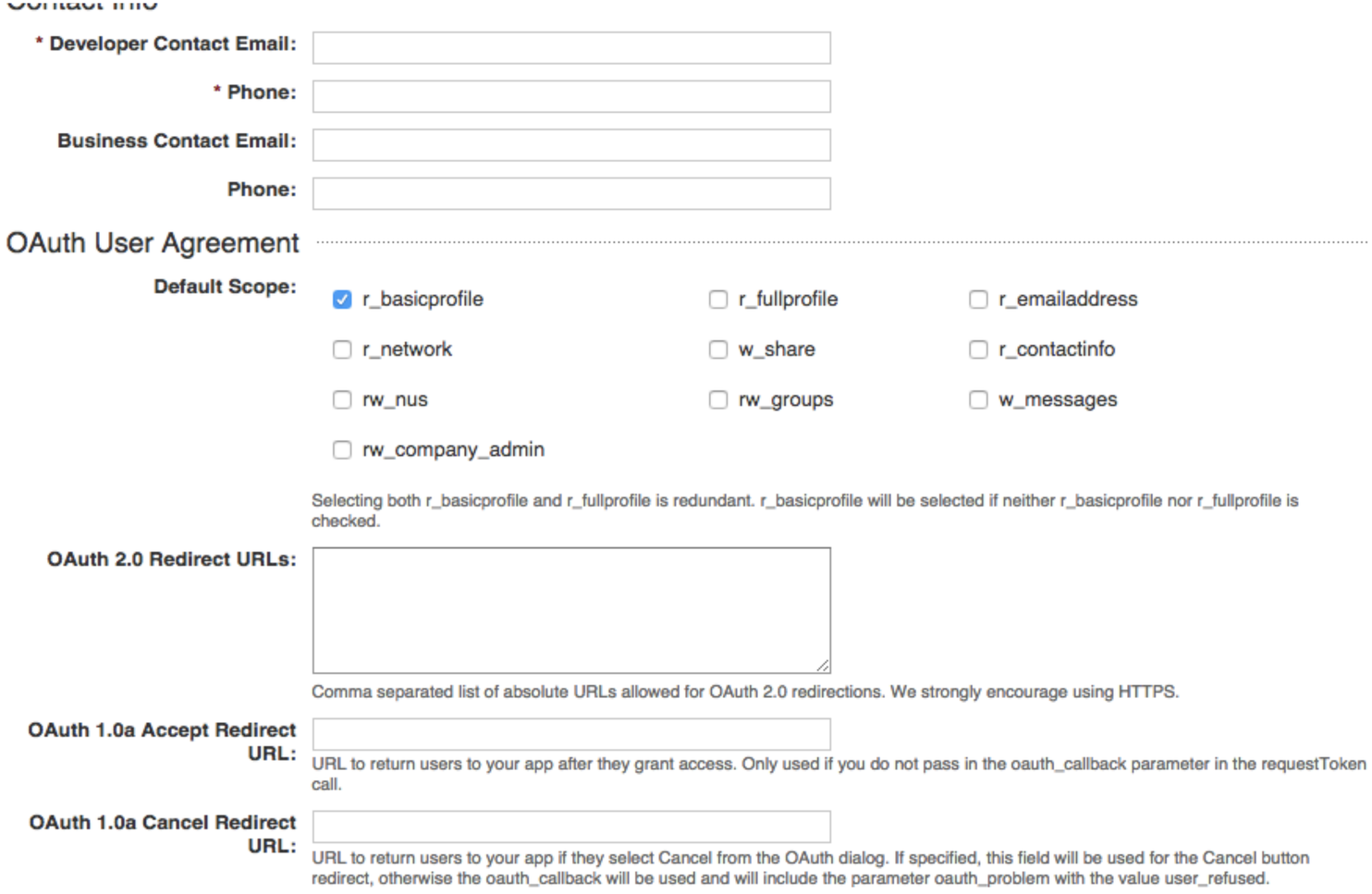
List of Applications

Company	Application Name
Dempsey Marketing	BKS 2012 Mastermind Dempsey Marketing CRM Dempsey Marketing Peeps
Robert Dempsey	Intridea Client Intelligence DS Course One APIs

Add New Application

« Back to LinkedIn Developer Network

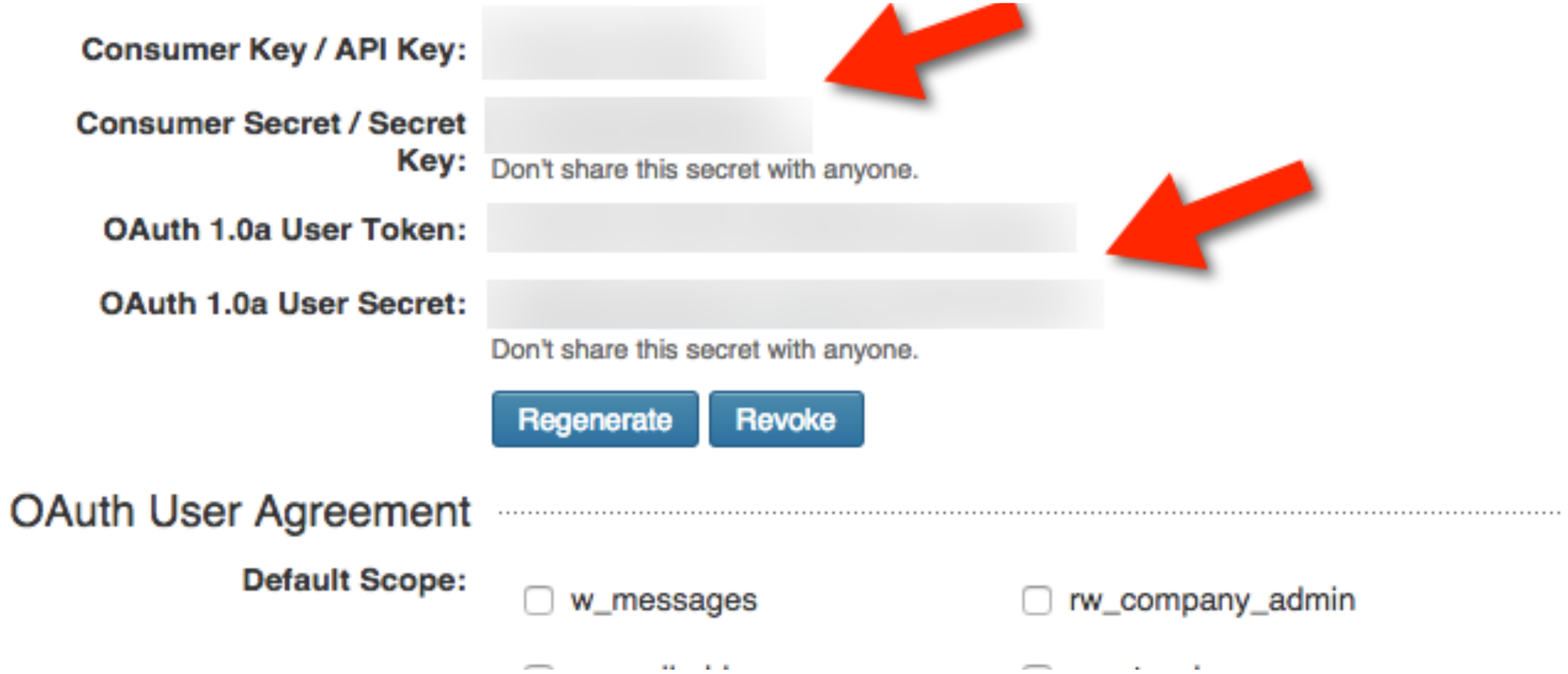
Create A LinkedIn Application



Create A LinkedIn Application

- Website_url
  - Put anything as long as it's a valid URL
- Live Status
  - Development
- Oauth section
  - Select the r_fullprofile checkbox
  - Everything else can be left blank

Create A LinkedIn Application



Install Everything We Need

pip install python-linkedin

pip install prettytable

Import What We Need

We first import our library and then declare variable for our user credentials

```
# Import everything we're going to need

import json
import csv
from linkedin import linkedin
from prettytable import PrettyTable
```

1. Import everything

2. Import only what's needed

Define Constants

Define the constants for your LinkedIn credentials.

```
# Use the LinkedIn OAuth credentials from the app you c

# Access credentials
CONSUMER_KEY = ''
CONSUMER_SECRET = ''
USER_TOKEN = ''
USER_SECRET = ''

# Return url: not needed for development but we'll keep
RETURN_URL = ''

# Create the authorization
authentication = linkedin.LinkedInDeveloperAuthenticati
```

Define constants to hold our LinkedIn credentials

Create a JSON File of Contacts

Retrieve your LinkedIn contacts and save them in a JSON file.

```
my_connections = li_app.get_connections()

# By default, this file will be stored in the same folder that your code r
# If using iPython Notebook as we are now, then it will be in the same fol
my_connections_file = 'my_linkedin_connections.json'

f = open(my_connections_file, 'w')
f.write(json.dumps(my_connections, indent=1))
f.close()

print("JSON file creation complete")
```

Use our app to get our connections

Create a json file

Let us know when the file has been created.

Looking at Pure JSON


```
{u'_total': 2051, u'_count': 1994, u'_start': 0, u'values': [{u'firs
e Your Social | Maximize Social Business | Social Media Center of Ex
e': u'Schaffer', u'industry': u'Marketing and Advertising', u'siteSt
nkedin.com/profile/view?id=235001&authType=name&authToken=AJbZ&trk=a
s://media.licdn.com/mpr/mprx/0_in1UQZKpi2nRlVIZTArXQR8Kfa3clxsZS-FXQ
tion': {u'country': {u'code': u'us'}, u'name': u'Orange County, Cali
{u'url': u'https://api.linkedin.com/v1/people/OCPLiDywUD', u'headers
uth-token', u'value': u'name:AJbZ'}]}}, u'id': u'OCPLiDywUD'}, {u'fi
angelist, ALPFA.org \u2605LinkedIn & Social Media Authority \u2605 A
\u2605Suite Branding', u'lastName': u'Ruff', u'industry': u'Nonprofi
eRequest': {u'url': u'https://www.linkedin.com/profile/view?id=31736
4*s4165914*'}, u'pictureUrl': u'https://media.licdn.com/mpr/mprx/0_-
3dA2AeOc-lDUknR1XRiS9GZiAjB', u'location': {u'country': {u'code': u'
```

Using prettytable

Use prettytable to make your work look nice, and more legible

```
pt = PrettyTable(field_names=['Name', 'Location'])
# Left justify everything
pt.align = 'l'

# If the person has a location, add a row for them
# If the person is keeping their information private, their name
# will show up as "private private" with no location
# The code below will exclude those entries
for c in connections['values']:
    if c.has_key('location'):
        pt.add_row([c['firstName'] + ' ' + c['lastName'], c['location']['name']])

print(pt)
```

Headers

Alignment

Adding data

Show the table

A Whole Lotta Pretty...table

Name	Location
Neal Schaffer	Orange County, California Area
Lori Ruff	Washington D.C. Metro Area
Patrick Schwerdtfeger	San Francisco Bay Area
Bill Vick	Dallas/Fort Worth Area
Howard Berg--World's Fastest Reader	Dallas/Fort Worth Area
Neil Patel	Orange County, California Area
Jessica Recco	Washington D.C. Metro Area
Milton B Taft, PhD	Greater New York City Area

Sorting a Prettytable

```
# By default the positions are sorted by the start date.
# Let's see a prettytable sorted by company name instead

print ct.get_string(sortby="Company")
```

Sort key

Convert JSON to CSV

```
for c in connections['values']:
    if c.has_key('location'):
        first_name = c['firstName'] if c.has_key('firstName') else ''
        last_name = c['lastName'] if c.has_key('lastName') else ''
        headline = c['headline'] if c.has_key('headline') else ''
        industry = c['industry'] if c.has_key('industry') else ''
        picture_url = c['pictureUrl'] if c.has_key('pictureUrl') else ''
        location = c['location']['name'] if c.has_key('location') else ''
        country = c['location']['country']['code'] if c.has_key('location') else ''
        profile_request_url = c['apiStandardProfileRequest']['url'] if c.has_key('ap
```

Pull Data From Twitter

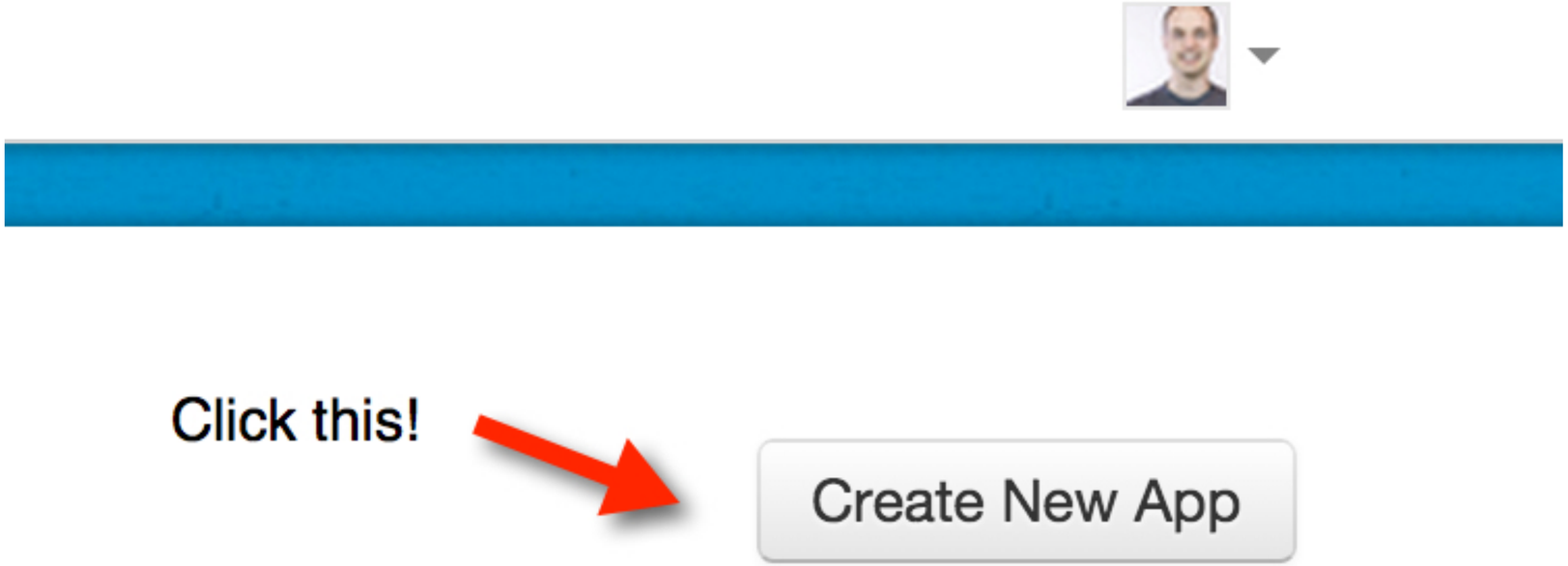
Overview

- Use Python 2.7.9
- Create an application on Twitter
  - https://apps.twitter.com
- Install the python-twitter library
  - https://pypi.python.org/pypi/python-twitter/
- Write our code
- Save the output to a CSV for later

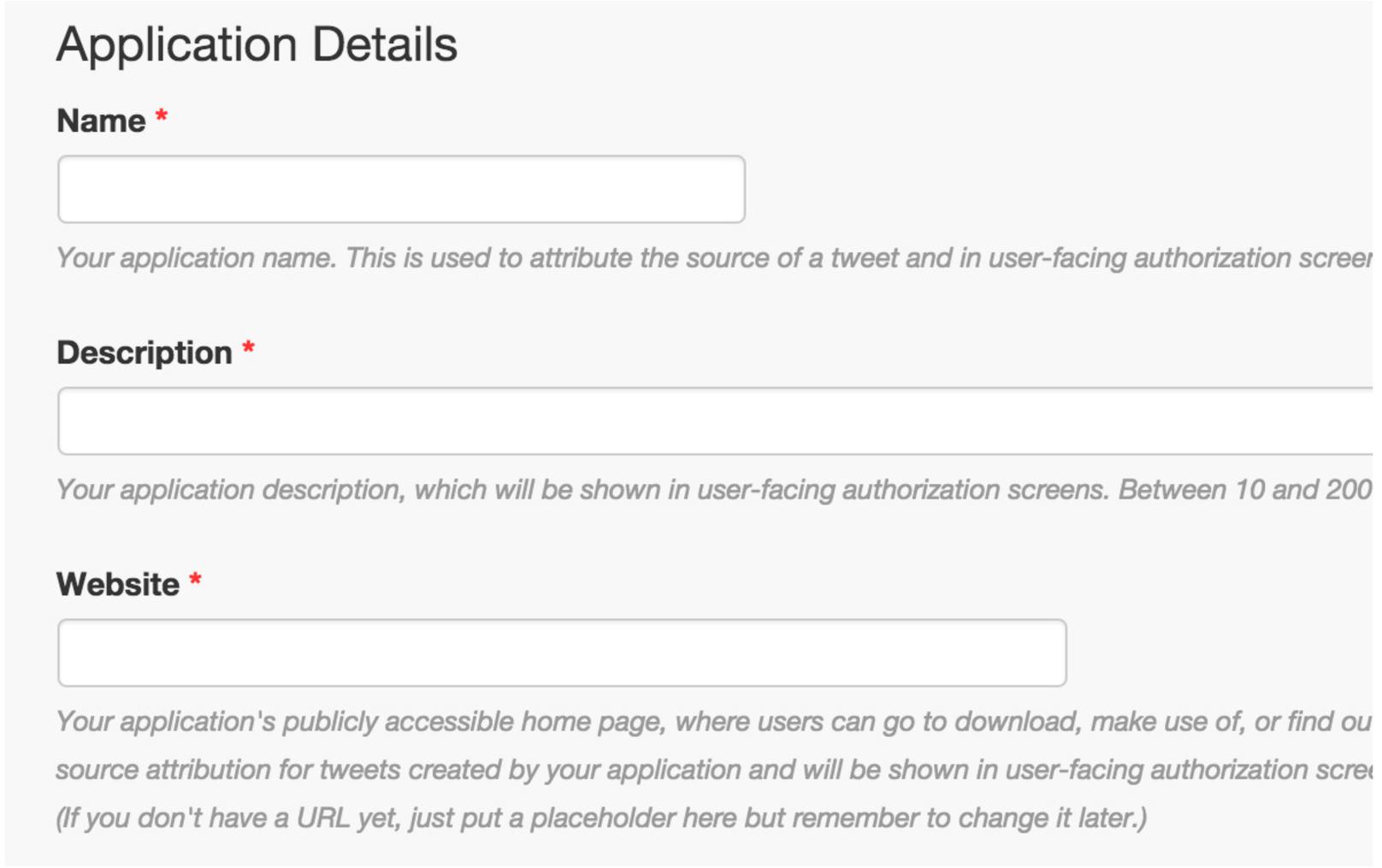
Create a Twitter App

https://apps.twitter.com/

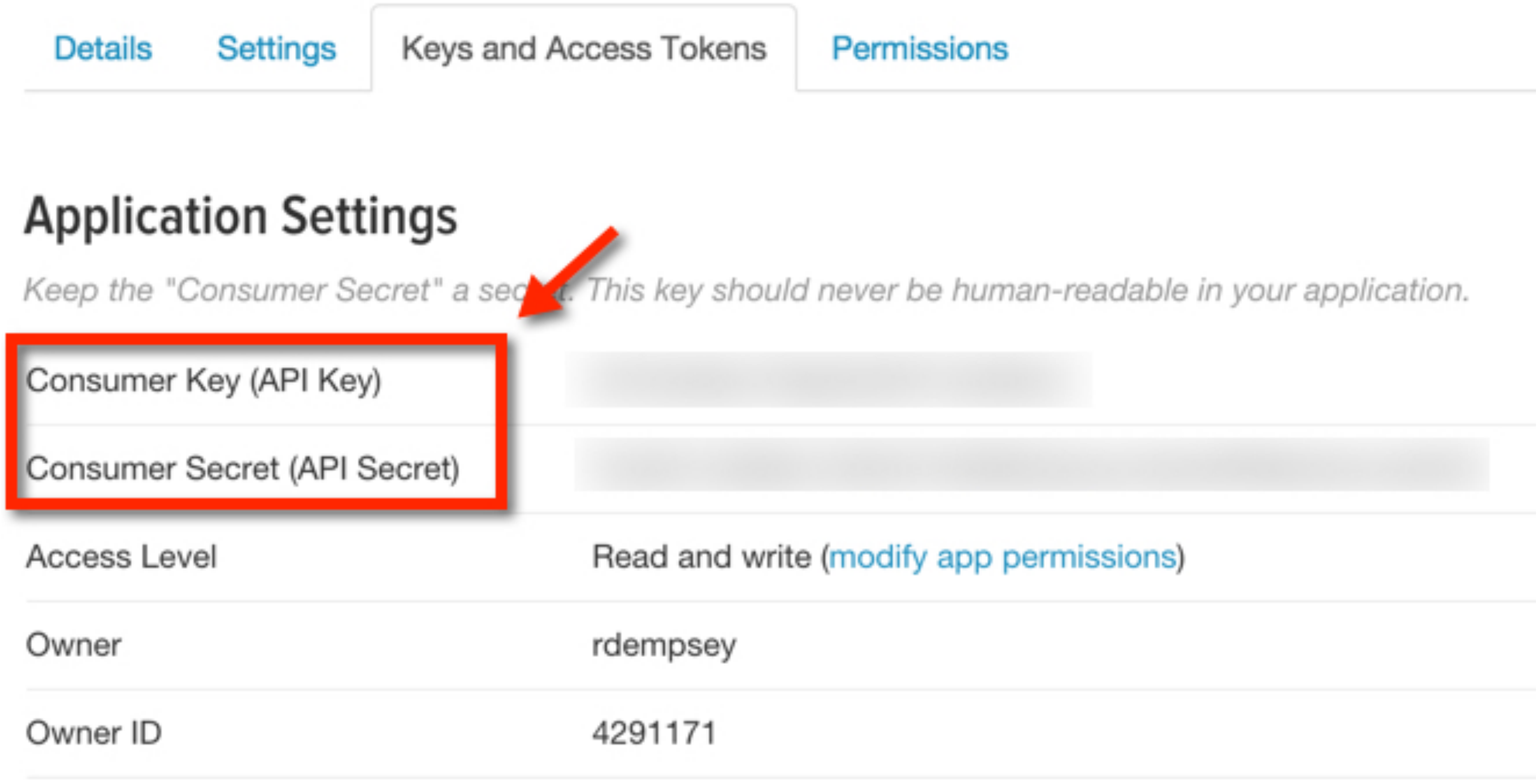
Create a Twitter App



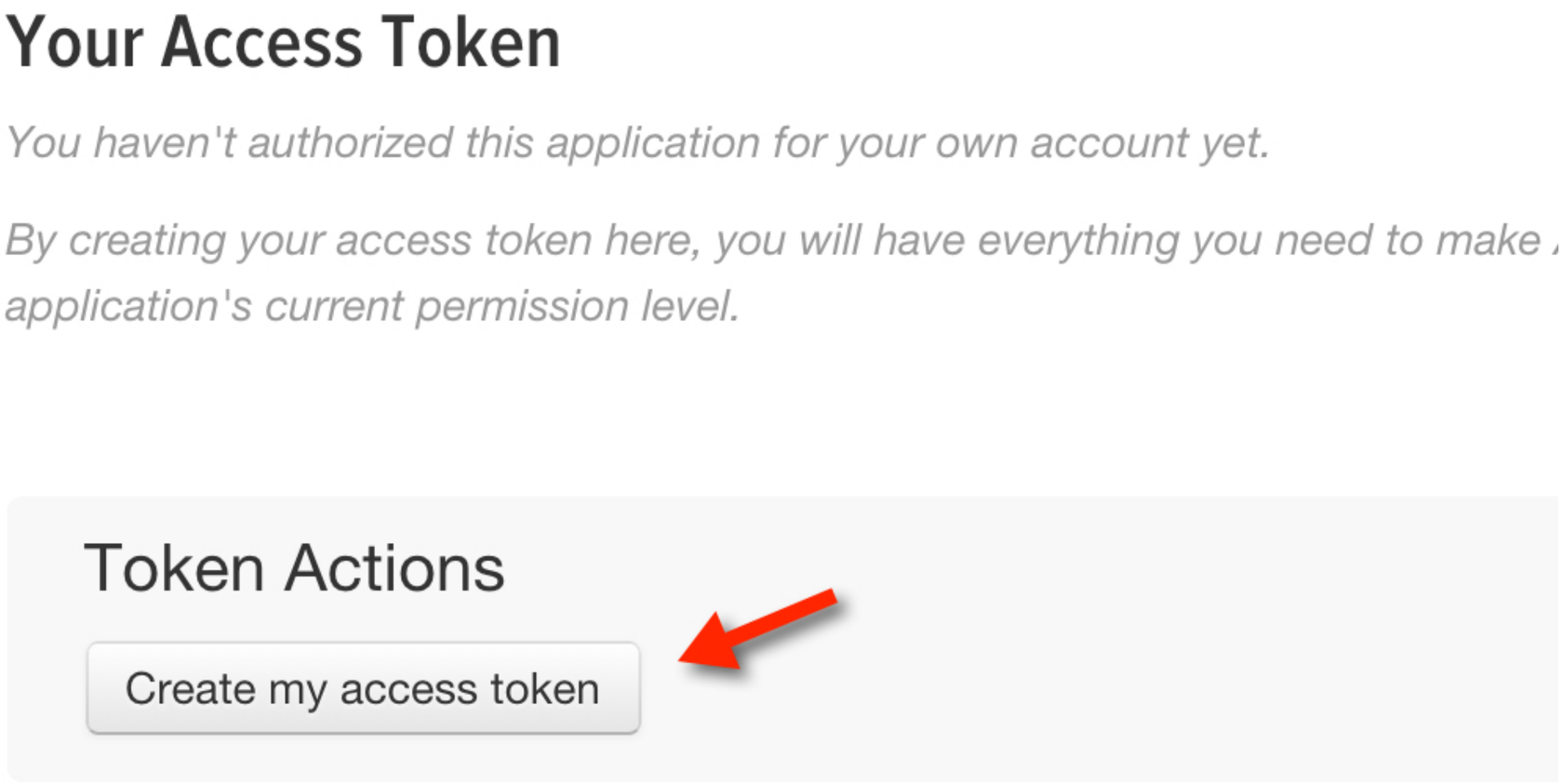
Create a Twitter App



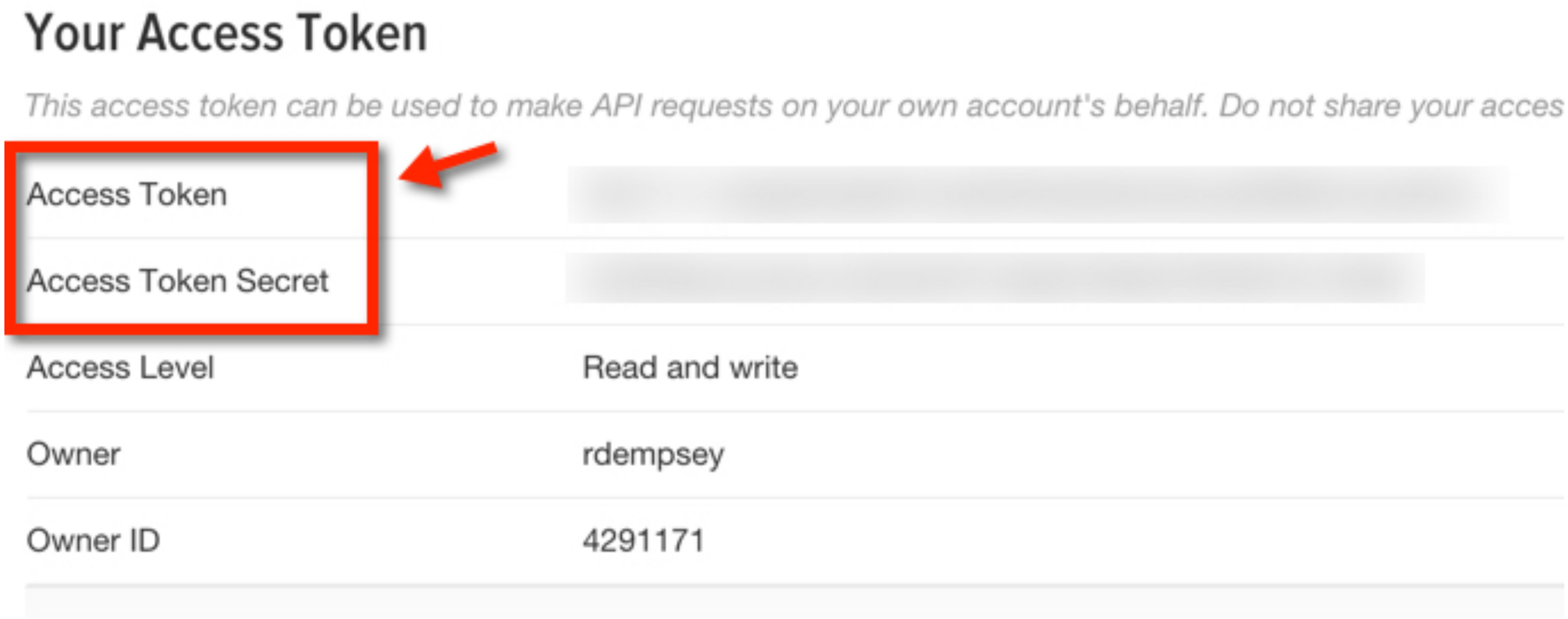
Create a Twitter App



Create a Twitter App



Create a Twitter App



Install Everything We Need

pip install python-twitter

Set Up Your API Credentials

```
api = twitter.Api(consumer_key='',
                  consumer_secret='',
                  access_token_key='',
                  access_token_secret='')

# Verify that your credentials are working
print(api.VerifyCredentials())
```

Could use constants here

Verify it's working

Fetch a User's Statuses

```
# Fetch a single user's public status
statuses = api.GetUserTimeline(screen_name='rdempsey')
for s in statuses:
    print(s.text)
```

Print out the statuses

Use any screen name

Manually Print Some Data

```
# Print out a few of the available attributes
# No prettytable here, yet
print("Name: {}".format(user.name))
print("Geo Enabled: {}".format(user.geo_enabled))
print("Description: {}".format(user.description))
print("Follower Count: {}".format(user.followers_count))
print("Friend Count: {}".format(user.friends_count))
print("Verified: {}".format(user.verified))
```

Leverage the user object to print out part of the data

Manual Printing Results

```
Name: Robert Dempsey
Geo Enabled: True
Description: Technology Strategist, Agile PM, Hacker of Things, Data Geek, Spartan
Follower Count: 14704
Friend Count: 8497
Verified: False
```

Iterate Over The User Object

Special method on the user object

```
# Make it look nicer
for key, value in user.AsDict().items():
    print(key, value)
```

Iteration Results

```
key,    value
('id', 4291171)
('profile_sidebar_fill_color', u'000000')
('profile_text_color', u'000000')
('followers_count', 14704)
('location', u'McLean, VA')
('profile_background_color', u'000000')
('listed_count', 620)
```

Create a CSV File

```
with open('twitter_user_data.csv', 'wb') as csvfile:
    writer = csv.writer(csvfile, delimiter=',', quotechar='"', quoting=csv.QUOTE_ALL)
    writer.writerow(['id', 'screen_name', 'name', 'description', 'location', 'lang', '
                     'favourites_count', 'followers_count', 'friends_count', 'listed_c
                     'account_created_at', 'time_zone', 'url', 'statuses_count', 'prof
                     'profile_background_image_url', 'profile_background_tile', 'profi
                     'profile link color', 'profile sidebar fill color', 'profile text
```

Write Multiple Profiles to CSV

```
for u in profiles_to_retrieve:
    user = api.GetUser(screen_name='{}'.format(u))
    writer.writerow([user.id, user.screen_name, user.n
                     user.favourites_count, user.follo
                     user.created_at, user.time_zone,
                     user.profile_background_image_url
                     user.profile_image_url, user.prof
                     user.profile_text_color, user.pro
```

What, No Library?

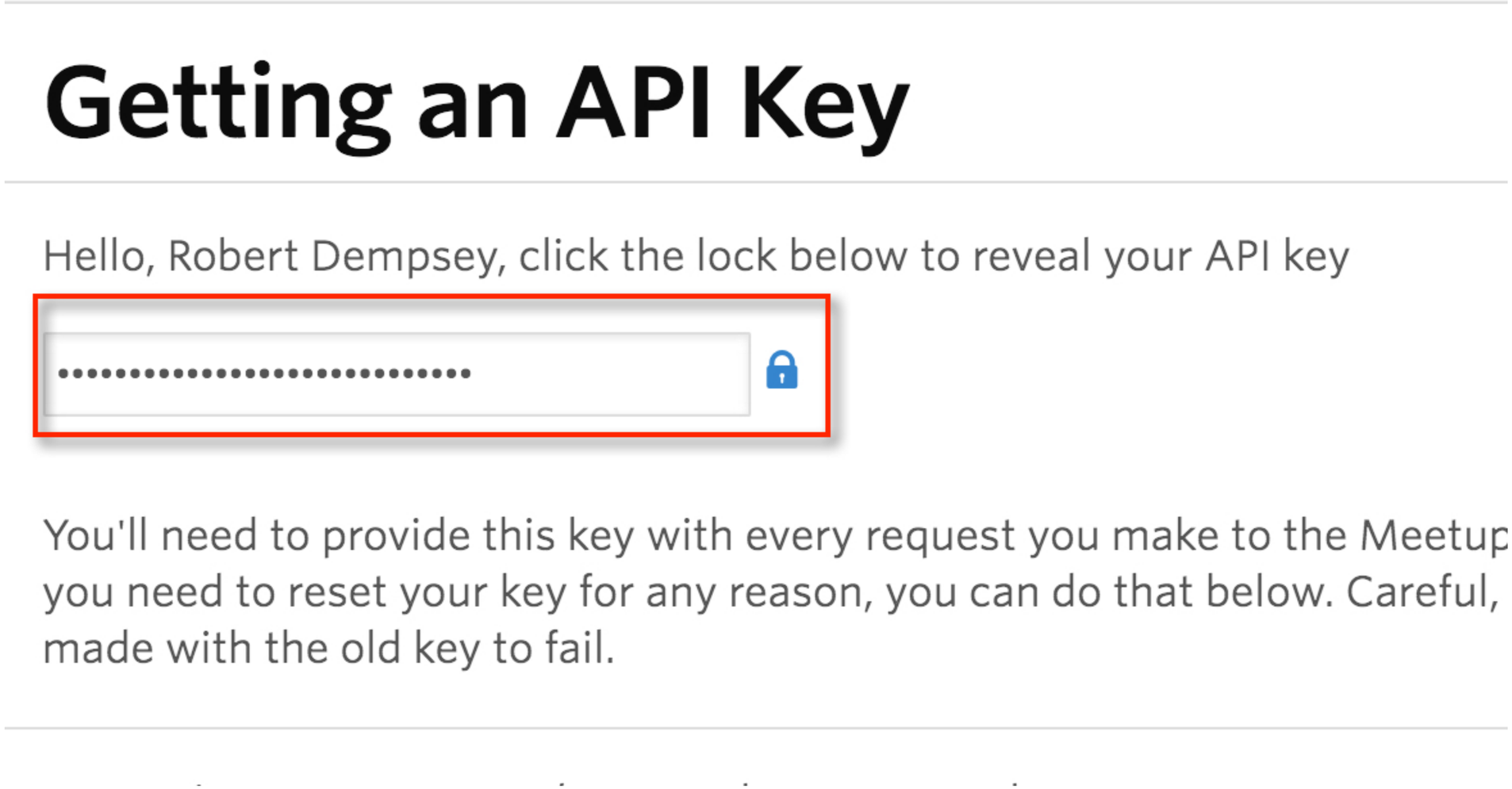
Overview

- Use Python 2.7.9
- Use a tool to test our REST calls and explore the JSON data
- Get a meetup API key
- Write our code
- Save the output to a CSV for later

Get a Meetup API Key

https://secure.meetup.com/meetup_api/key/

Meetup API Key



Build Our URL

```
# First we'll get a list of categories

# Your meetup API key. This will be used for all requests.
api_key = '                                '

# The base path for calling the api
base_path = "https://api.meetup.com"

# Meetup category path - returns a list of all the categories
category_path = '/2/categories'
```

Retrieve The Data

```
# Build the full url to call
url_to_call = base_path + category_path + '?&sign=true&key=' + api_key

# Check to be sure we have the right one
print(url_to_call)

# Load the JSON response as a python dict
category_list = json.load(urllib2.urlopen(url_to_call))
```

Print the Results

```
# Loop through the results and print them out
for result in category_list['results']:
    print(result)
```

Meetup API Results

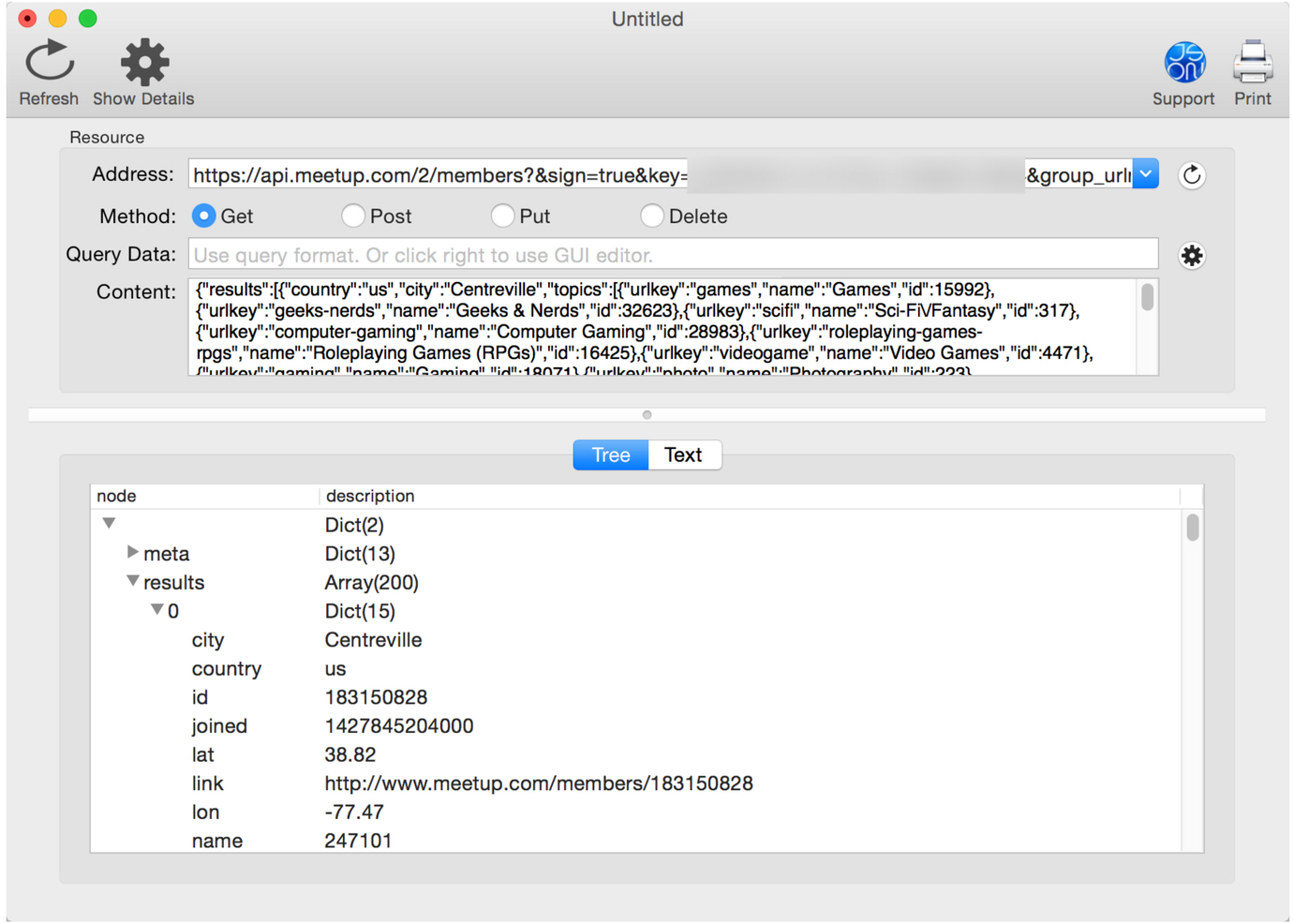
```
{u'shortname': u'Arts', u'name': u'Arts & Culture', u'id': 1}
{u'shortname': u'Business', u'name': u'Career & Business', u'id': 2}
{u'shortname': u'Auto', u'name': u'Cars & Motorcycles', u'id': 3}
{u'shortname': u'Community', u'name': u'Community & Environment', u'id': 4}
{u'shortname': u'Dancing', u'name': u'Dancing', u'id': 5}
{u'shortname': u'Education', u'name': u'Education & Learning', u'id': 6}
{u'shortname': u'Fashion', u'name': u'Fashion & Beauty', u'id': 8}
{u'shortname': u'Fitness', u'name': u'Fitness', u'id': 9}
{u'shortname': u'Food & Drink', u'name': u'Food & Drink', u'id': 10}
{u'shortname': u'Games', u'name': u'Games', u'id': 11}
```

Categories to CSV

```
# Save the categories to a CSV file
with open('meetup_categories.csv', 'wb') as csvfile:
    writer = csv.writer(csvfile, delimiter=',', quotechar='"', quoting=csv.QUOTE_ALL)
    writer.writerow(['id', 'shortname', 'name'])
    for result in category_list['results']:
        writer.writerow([result['id'],
                         result['shortname'],
                         result['name']])

                         # So we know that the CSV file was indeed created
print("CSV creation complete")
```

VisualJSON (Mac)



Aggregating Multi-Source Data

Why Aggregation Is Important

- In order to get a complete picture, we need to funnel all of the data into one data source
- Allows for faster processing
- Makes the data more accessible
- Makes analyzing a profile easier

Aggregation Steps

- Import each file into a pandas dataframe
- Create a “key” so that you can perform a database-like join
- Perform any formatting you need in order to help the merge
- Create a new dataframe and use the “merge” function to perform the merge
- Save to CSV with one line of code

Overview

- Use Python 2.7.9
- Install the titlecase library
  - https://pypi.python.org/pypi/titlecase
- Write our code
- Save the output to a CSV

CSV to Pandas Dataframe

```
# Import the LinkedIn and Twitter data into individual pandas dataframes

# Define the paths to the files
# If you named your files differently, change the names
linkedin_data_file = 'linkedin_connection_data.csv'
twitter_data_file = 'twitter_user_data.csv'

# Create the dataframes by reading from the CSV files
ldf = pd.read_csv(linkedin_data_file)
tdf = pd.read_csv(twitter_data_file)
```

Create a pandas data frame from a CSV file

Explore the Data

Command to run: print(ldf)

```
          first_name                        last_name  \
0               Neal                          Schaffer
1               Lori                             Ruff
2            Patrick                     Schwerdtfeger
3               Bill                             Vick
4             Howard  Berg--World's Fastest Reader
5               Neil                            Patel
6             Jessica                           Recco
7           Milton B                       Taft, PhD
8               Jure                          KLEPIC
9               Jeff                          Putnam
10             Maria                          Haynes
```

Define a Titlecase Function

```
# Titlecase anything
def titlecase_anything(thing):
    try:
        thing = titlecase(thing)
    except:
        pass
    return thing
```

Create a Full Name Function

```
# Create a full name from the first and last
def create_full_name(first_name, last_name):
    return (first_name + " " + last_name)
```

Apply The Functions

Note that we don't have to pass any arguments to our functions for them to work properly.

```
ldf.first_name = ldf.first_name.apply(titlecase_anything)
ldf.last_name = ldf.last_name.apply(titlecase_anything)
```

Create a New “Column”

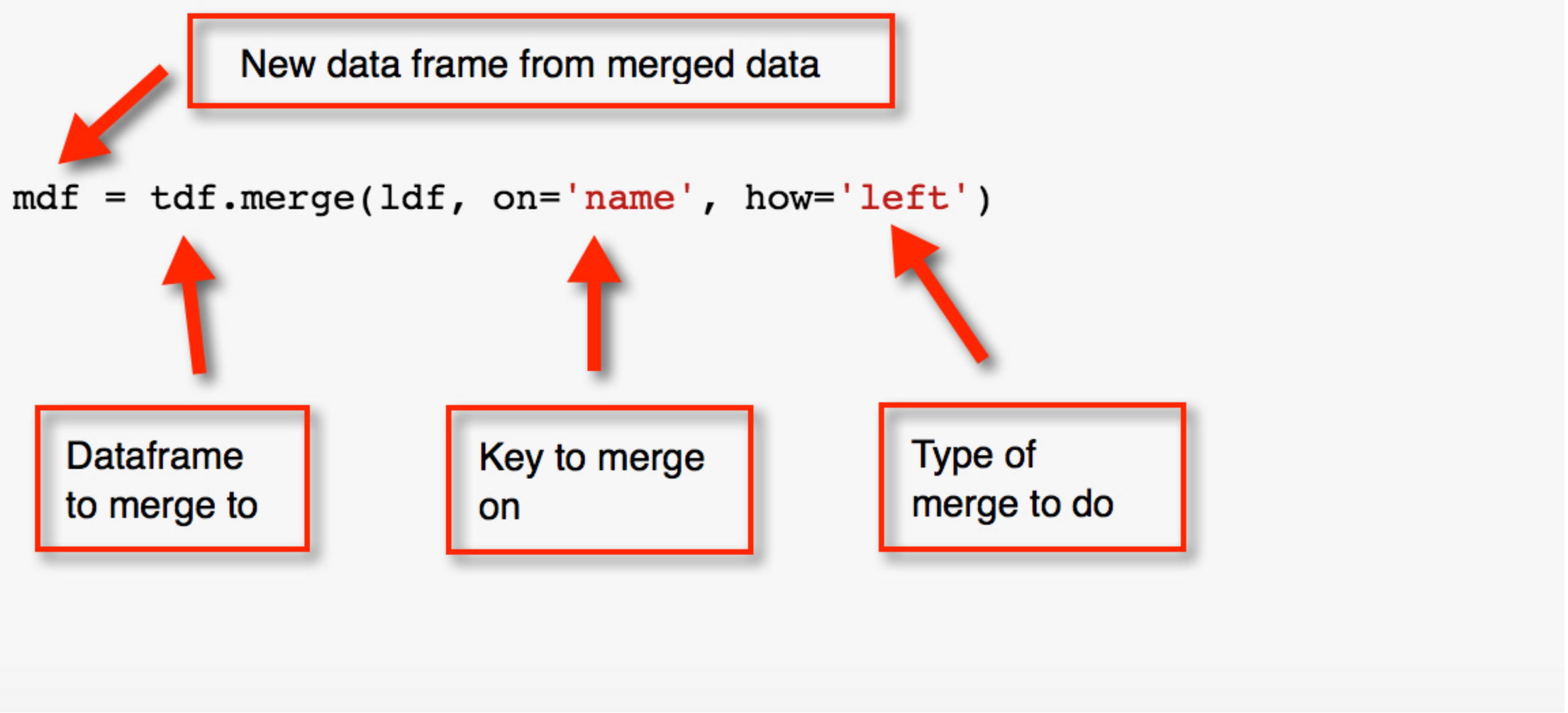
```
ldf['name'] = ldf.first_name + "  " + ldf.last_name
```

Creates a new “column” in our data frame using existing data

Result

	name
0	Neal Schaffer
1	Lori Ruff
2	Patrick Schwerdtfeger
3	Bill Vick
4	Howard Berg--World's Fastest Reader
5	Neil Patel
6	Jessica Recco
7	Milton B Taft, PhD
8	Jure Klepic
9	Jeff Putnam

Create a Merged Dataframe



One-Line Dataframe to CSV

```
mdf.to_csv('twitter_and_linkedin_merged_data.csv', sep=',', encoding='utf-8')
```

Use the built-in “to_csv” function of pandas to export the entire dataframe to a csv file

How To Use The Data

Basic Analysis You Can Do

- Questions we can ask of the data, without doing math
  - How many tweets?
  - Where are they tweeting from?
  - Which meetups do they go to?
  - Are they posting to LinkedIn?
- What are the implications of this?
- How can we interpret this data?

Turn Your Questions Into Code

Robert Dempsey

robertwdempsey.com

rdempsey

robertwdempsey

rdempsey