

A world map with a light blue background and tan landmasses. Three blue location pins are placed on the map: one in the western United States, one in the eastern United States, and one in the southeast of Australia. The map is centered on the Atlantic Ocean.

# Oh Hacker, Where art thou?

A hands-on Geolocation lab using Python

SANS free resources are aimed to provide the latest in research and technology available to help support awareness and growth across a wide range of IT and OT security considerations.



# SANS

A world map in light blue and beige tones serves as the background. Three blue location pins are placed on the map: one in North America, one in Europe, and one in Australia.

# **How to setup your machine**

- **Register and log into the SLACK Channel**
  - **<https://sansurl.com/balance-17may>**
- **Download the Class Virtual Machine and import it into VMWare**
  - **<https://tinyurl.com/locatehacker>**
  - **Download password is "whereRUhacker?"**
- **Login as "student" with password of "student"**
- **Double-Click "lab\_setup.sh" on the Desktop inside the VM**

A world map with a light blue background and white landmasses. Three blue location pins are placed on the map: one in North America, one in Europe, and one in Australia. The title 'The Backup Plan' is centered over the map.

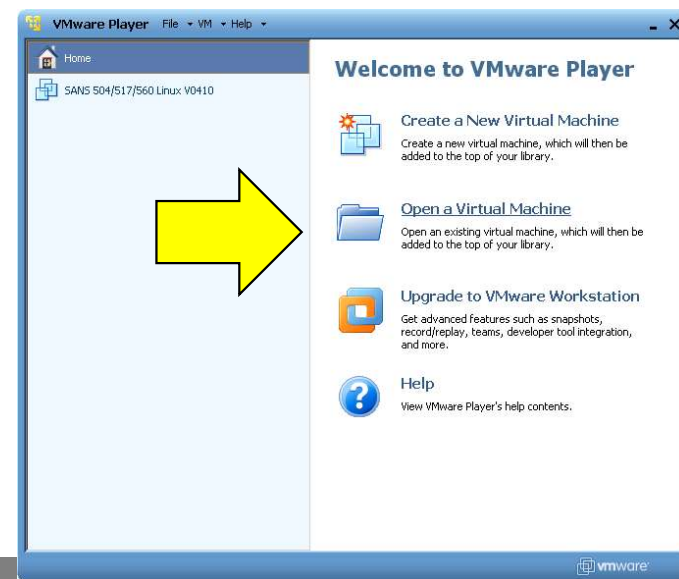
# **The Backup Plan**

**If you can not get the Virtual Machine working**

**<http://github.com/markbaggett/GeoLocationNotebook>**

## Start Configuring your Workshop VM Now

- You will need a virtualization product for this workshop
- VMWare Workstation offers a free 30 day trial:
  - <https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html>
- Import the VM Into VMWare or Virtualization Software:
  - select File > Import in VMware
  - Select "Open a Virtual Machine" and select the OVF file
- Run VMware, open the VM, and boot it
- Make sure your VM has access to the internet
- Once the Desktop appears you can click the lab\_setup script to download the material.



```
Get-ADUser -Filter "Mark Baggett" | fl -Properties *
```

- Mark Baggett
- Penetration Testing and Incident Response Consulting
- Senior SANS Instructor
- Author of SANS SEC573 Automating InfoSec with Python
- Masters in Information Security Engineering
- GSE #15
- DoD Advisor, Former CISO 18 years commercial

```
student@573:/opt/metasploit-framework$ grep -Ri "mark baggett" | wc -l  
7
```



## Today's Topic

- Today's topic assumes you know some Python
- SANS Sec573 Automating Information Security with Python does not assume prior knowledge
- OnDemand is now available!

## Course Overview


- Days 1 and 2: Essentials Workshop
  - Build the skills required to rapidly develop information security tools on your own
- Days 3–5: Continue Learning Coding Concepts
  - Day 3 - Defensive focused projects
  - Day 4 - Forensics focused projects
  - Day 5 - Offensive focused projects
- Day 6: Capstone "CTF" Event
- Days 1–5: pyWars Challenge and CTF
  - Master the nuances of Python programming



## Recommended New Coders' GPYC Self-Paced Study Plan

- **Listen to lecture** and expect to **finish 33%–50% of labs** (shown in green) during time allotted in class. **GREEN** = Completed In Class
- **Finish** all the **non-pyWars labs** during the time allotted **in class**
- **Complete the rest** of labs (shown in blue) on your local pyWars server before the exam; **evenings** after class **or** when you get **back home**. **BLUE** = At Home
- CTF Challenges (shown in red) are scattered throughout pyWars. They are for the veterans' CTF. **RED** = Ignore Them

SECTION 1														SECTION 2														3																
LAB 1				LAB 2					LAB 3					LAB1							LAB 2			CTF				LAB1																
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	42	43	44	45



- Alternate Approaches: It's self-paced! Some students choose to spend all week on Sections 1 and 2, but then you're on your own for 3–5.

## Common Veteran Coders' Self-Paced Study Plan

- Usually, veteran coders will ignore lectures and race through the first two days on Day 1 and then focus on Days 3–5
- Use the "Roadmap Slide" to determine when you need to pay closer attention
- After completing Days 3–5, there are 20+ CTF challenges that go far beyond GPYC and the course material
- The FIRST person to complete ALL the challenges gets a CTF coin
- Instructors will be vague when assisting students in CTF until someone finishes
- Finished? Most "Finishers" have skipped all non-pyWars labs, including Day 5!

Day 1		Day 2	Day 3	Day4	DAY 5	At Home	
Day 1 1-18	Day 2 19-35	Day 3 36-61	Day 4 51-66	Advanced pyWars CTF 67 - 90+	Non-pyWars Labs	Day 5	Hall of Fame Challenges

- The top 1% will complete all pyWars and only some of the non-pyWars labs

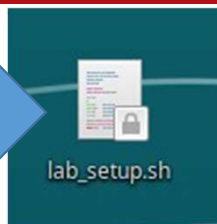
## Majority of Students' Self-Paced Study Plan

- The majority of you will be somewhere in between those two approaches
- Alternate back and forth between listening to lectures and self-paced labs
- You also have more work than you can finish! Choose where to focus your time
- You may have an incomplete lab or two that you can finish later, particularly on Day 3, when we provide more than double the labs than you can normally complete
- Keep in mind that the ONLY thing you don't have access to after class is the Veterans CTF. For this reason, some people choose to leave some non-pyWars labs or a few pyWars challenges unsolved and take a shot at those advanced CTF questions.
- You choose the learning model that is best for you. In other words, it is self-paced.

DAY 1	DAY 2	DAY 3	DAY 4	DAY 5		At Home
1-18 and Non- pyWars Labs	19-35 and Non- pyWars Labs	36-61 and Non- pyWars Labs	62-65 and Non- pyWars Labs	Day 5 is all Non- pyWars Labs	Advanced PyWars CTF 65 - 90+	Hall of Fame Challenges

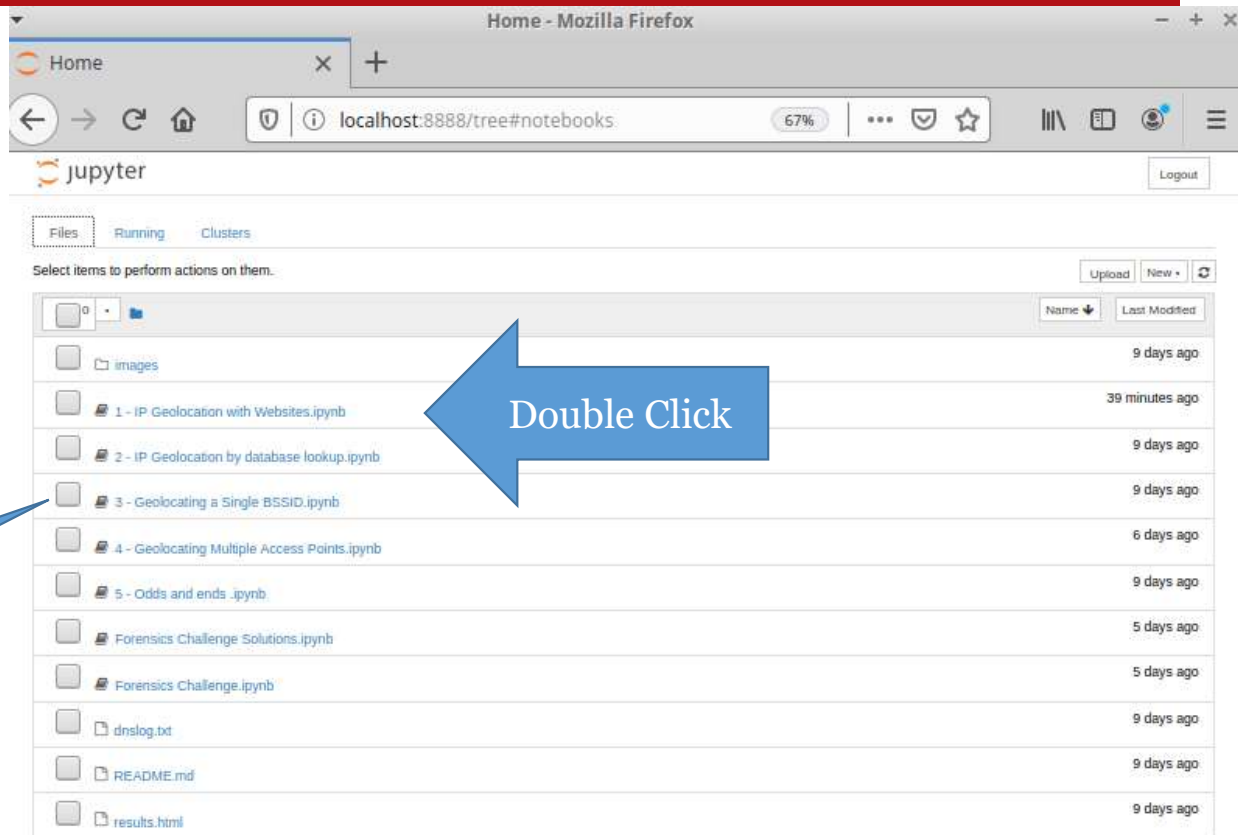
## Double Click lab\_setup.sh on the Desktop to Start

Double Click

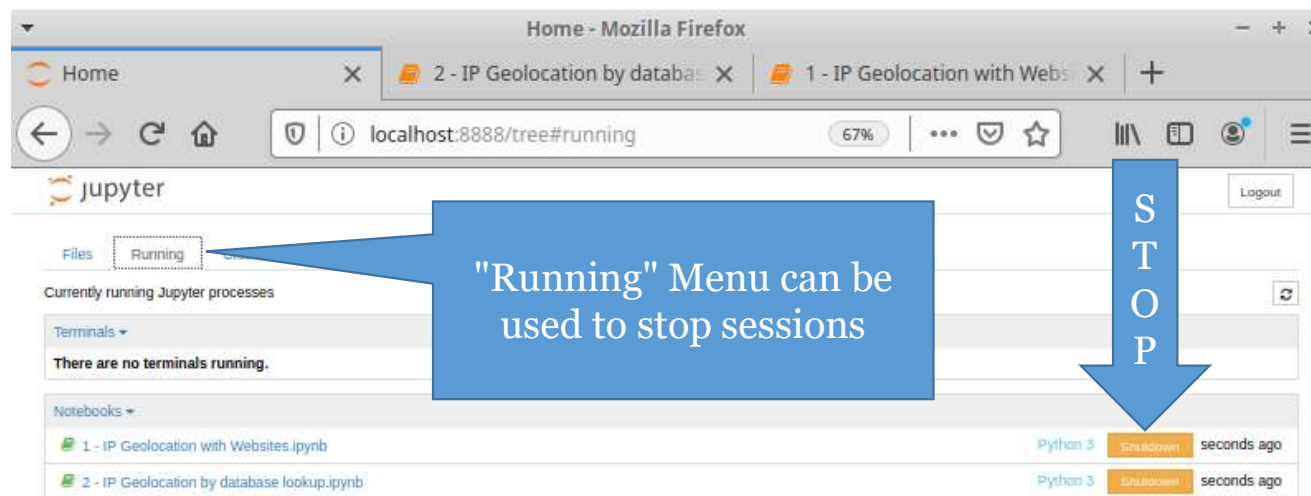


After running lab\_setup  
Jupyter notebooks will  
open!

Double Click a page  
to **launch a new  
session** and interact  
with that page



# Shutting Down a Session



## You Can Change These Pages!

- On each page there are "CELLS" that contain code
- Blue line is read mode
- Double click a cell to begin editing
- Green line is edit mode
- You can add new cells

```
In [ ]: import requests  
browser = requests.session()  
response = browser.post("https://iplocation.com/", {"ip":"8.8.8.8"})  
response.json()
```

```
In [ ]: you can edit a cell when the line is GREEN  
import requests  
browser = requests.session()  
response = browser.post("https://iplocation.com/", {"ip":"8.8.8.8"})  
response.json()
```



# Run a single cell by pressing CONTROL+ENTER

1 - IP Geolocation with Websites - Mozilla Firefox

Home X 1 - IP Geolocation with Webs X +

localhost:8888/notebooks/1 - IP Geolocatio 67%

jupyter 1 - IP Geolocation with Websites Last Checkpoint: 03/21/2020 (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run

**Welcome!**

Welcome to this special workshop on using Python for Geolocation of forensic artifacts! In this session we will explore a couple of online APIs that can be used to determine the physical location of a device or connection.

One easy way to determine a location is based upon the IP Address being used. IP Addresses are often found inside of logs on victim machines and forensics images. On victim machines these IP Addresses can be less than reliable because attackers will use various VPN services to hide their actual IP Address. We will discuss how to get around that problem later. For now lets assume we are on a forensics image of the attacker machine or we have captured an early connection before the attacker began obfuscating their location. To determine their location we can use an online website such as <https://iplocation.com/>

By viewing the source code on this website we can determine how to interact with the website and use it automatically.

Lets walk through that process together.

Once we understand the website form we can use Python to automatically retrieve information from the website. Websites have different terms of use so make sure you are limiting your interactions with the website to those terms.

```
In [ ]: import requests
browser = requests.session()
response = browser.post("https://iplocation.com/", {"ip": "8.8.8.8"})
response.json()
```

This process is known as "Web Scraping". We are using our script to automate the actions of a human being interacting with the website. The website author may have intended the website to be used exclusively by humans using a browser. Instead of scraping the website we could go through an API (Application Programming Interface).

CONTROL + ENTER to  
execute a "cell" in  
Python session

Turns black  
while code is  
running



## Reset your Python Session with Kernel Menu

The screenshot shows a Jupyter Notebook titled "1 - IP Geolocation with Websites" running in Mozilla Firefox. The browser address bar shows "localhost:8888/notebooks/1 - IP Geolocation with Websites". The Jupyter interface includes a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". The "Kernel" menu is open, displaying options: "Interrupt", "Restart", "Restart & Clear Output", "Restart & Run All", "Reconnect", "Shutdown", and "Change kernel". A blue arrow points from the "Restart & Clear Output" option in the menu to a larger, detailed view of the "Kernel" menu on the right. This detailed view shows the same options, with "Restart & Clear Output" highlighted in yellow. Below the menu, the notebook content is visible, showing two code cells. The first cell contains a loop that fetches geolocation data for a list of IP addresses. The second cell contains code to use a defaultdict to group the data by country. The status bar at the bottom indicates the file path: "localhost:8888/notebooks/1 - IP Geolocation with Websites.ipynb#".

```
In [ ]: list_of_addresses = ['144.12.97.175', '47.33.198.195', '54.213.136.220', '37.27.54.15', '91.107.252.94', '112.245.28.1']
for each_ip in list_of_addresses:
    url = f"https://freegeoip.app/json/{each_ip}"
    result = browser.get(url).json()
    print(f"The country for IP Address {result.get('ip')} is {result.get('country_name')}")
```

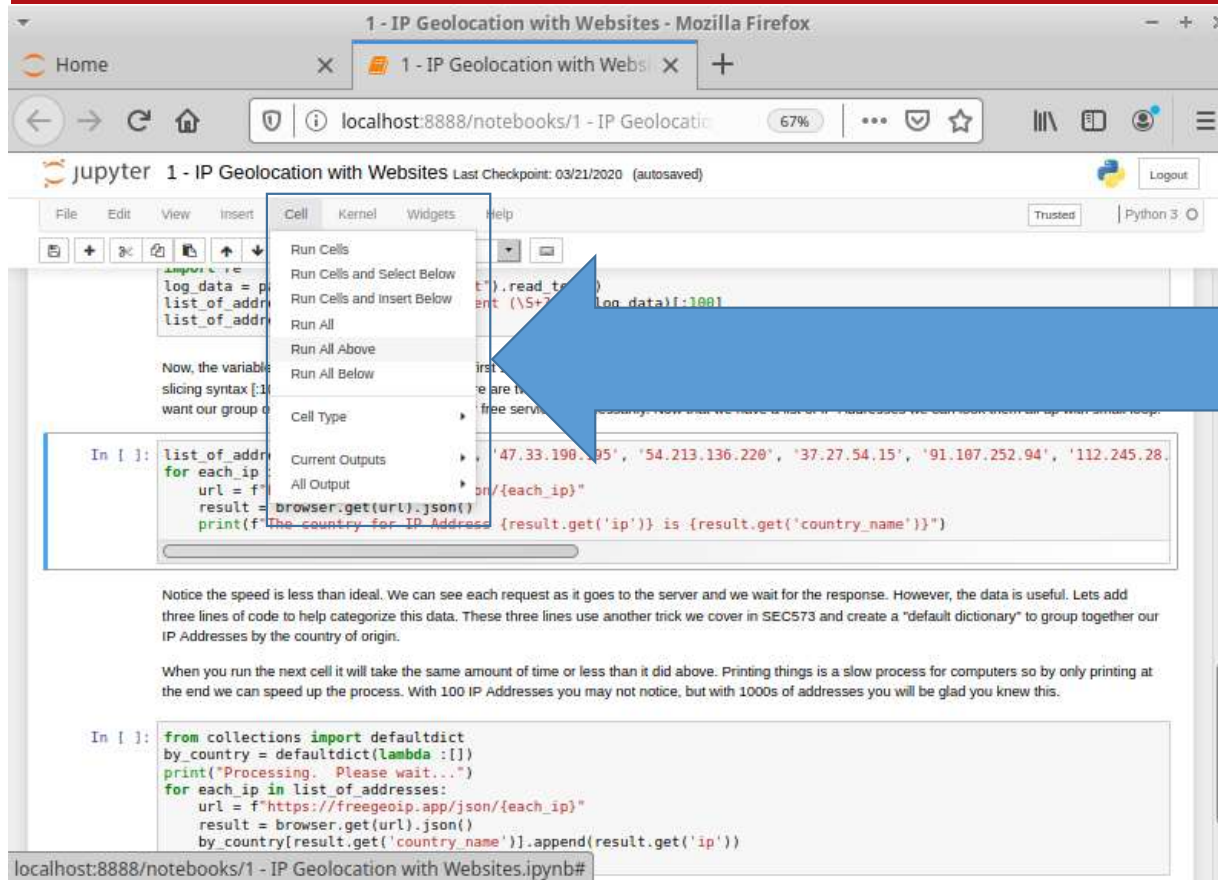
Notice the speed is less than ideal. We can see each request as it goes to the server and we wait for the response. However, the data is useful. Lets add three lines of code to help categorize this data. These three lines use another trick we cover in SEC573 and create a "default dictionary" to group together our IP Addresses by the country of origin.

When you run the next cell it will take the same amount of time or less than it did above. Printing things is a slow process for computers so by only printing at the end we can speed up the process. With 100 IP Addresses you may not notice, but with 1000s of addresses you will be glad you knew this.

```
In [ ]: from collections import defaultdict
by_country = defaultdict(lambda :[])
print("Processing. Please wait...")
for each_ip in list_of_addresses:
    url = f"https://freegeoip.app/json/{each_ip}"
    result = browser.get(url).json()
    by_country[result.get('country_name')].append(result.get('ip'))
```

localhost:8888/notebooks/1 - IP Geolocation with Websites.ipynb#

## Cell Menu allows you to Run Previous Cells



1 - IP Geolocation with Websites - Mozilla Firefox

Home X 1 - IP Geolocation with Webs X +

localhost:8888/notebooks/1 - IP Geolocation 67%

jupyter 1 - IP Geolocation with Websites Last Checkpoint: 03/21/2020 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help

Run Cells  
Run Cells and Select Below  
Run Cells and Insert Below  
Run All  
Run All Above  
Run All Below

Cell Type

Current Outputs  
All Output

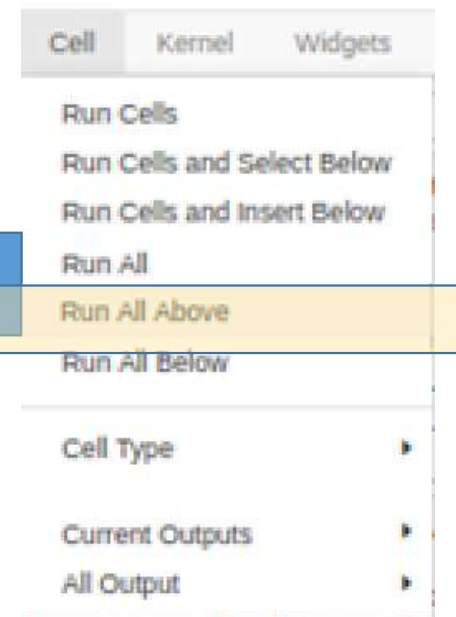
```
In [ ]: list_of_addresses = [...]
for each_ip in list_of_addresses:
    url = f"https://freegeoip.app/json/{each_ip}"
    result = browser.get(url).json()
    print(f"The country for IP Address {result.get('ip')} is {result.get('country_name')}")
```

Notice the speed is less than ideal. We can see each request as it goes to the server and we wait for the response. However, the data is useful. Lets add three lines of code to help categorize this data. These three lines use another trick we cover in SEC573 and create a "default dictionary" to group together our IP Addresses by the country of origin.

When you run the next cell it will take the same amount of time or less than it did above. Printing things is a slow process for computers so by only printing at the end we can speed up the process. With 100 IP Addresses you may not notice, but with 1000s of addresses you will be glad you knew this.

```
In [ ]: from collections import defaultdict
by_country = defaultdict(lambda: [])
print("Processing. Please wait...")
for each_ip in list_of_addresses:
    url = f"https://freegeoip.app/json/{each_ip}"
    result = browser.get(url).json()
    by_country[result.get('country_name')].append(result.get('ip'))
```

localhost:8888/notebooks/1 - IP Geolocation with Websites.ipynb#



Cell Kernel Widgets

Run Cells  
Run Cells and Select Below  
Run Cells and Insert Below  
Run All  
Run All Above  
Run All Below

Cell Type

Current Outputs  
All Output

## Challenge 1

### Forensics Challenge 1 - The Outlook Web Access Attack

The victim lost 1.2 million dollars in an Outlook Web Access Phishing attack. You have pulled all of the SUCCESSFUL LOGINS from the Outlook Administrator console. But there are THOUSANDS of IP addresses to look through. The original attack came from Iran. You decide to focus your attention on those requests that are from the same country of origin.

```
[ '144.12.97.175',  
  '47.33.190.195',  
  '54.213.136.220',  
  '37.27.54.15',  
  '91.107.252.94',  
  '112.245.28.187',  
  '77.198.208.109',  
  '79.195.246.215',  
  '109.219.196.231',  
  '111.14.43.156',  
  '102.111.174.6',  
  '176.65.253.46',  
  '102.155.114.82' ]
```

Which IP Addresses originated in Iran?

## Locating an Attacker by IP Address

- We find IP Addresses in Logs and Forensics Artifacts everywhere we look!
- Why turn those into locations?
  - Put bad guys in jail!
  - Determine relationships between IP Addresses in logs for more detailed investigations
- What are the Limitations?
  - UDP protocols connections can be spoofed
  - TCP Protocol can be spoofed by ISPs and any other "upstream" entities
  - Some TCP Protocol artifacts can be spoofed such as SMTP Headers
  - Criminals love to hide behind VPN connections that hide their real IP

## Using websites to lookup IP Addresses

- Python can use website to determine where the originator of a connection is physically location based on their IP Address
- For Python to interact with website we will import a module called "Requests"

## Using Requests Session()

- Think of this as creating a browser that remembers settings and headers like User-Agent and maintains state via cookies
- Call the browser objects `get()`, `post()`, and so on to use the settings

```
>>> import requests
>>> browser = requests.session()
>>> browser.headers
{'User-Agent': 'python-requests/2.21.0', 'Accept-Encoding': 'gzip, deflate', 'Accept': '*/*',
'Connection': 'keep-alive'}
>>> browser.headers['User-Agent']
'python-requests/2.21.0'
>>> browser.headers['User-Agent']='Mozilla FutureBrowser 145.9'
>>> browser.headers
{'User-Agent': 'Mozilla FutureBrowser 145.9', 'Accept-Encoding': 'gzip, deflate', 'Accept':
'*/*', 'Connection': 'keep-alive'}
>>> x = browser.get("http://isc.sans.edu")
>>> type(x)
<class 'requests.models.Response'>
```

Create a browser object that remembers cookies, settings, and headers

Outbound headers are stored in a dictionary that you can modify to meet your needs

## Response Objects

- All those methods return a response object with access to full details about the webpage's response

```
>>> resp = browser.get("http://isc.sans.edu")
>>> type(resp)
<class 'requests.models.Response'>
>>> dir(resp)
[ <dunders deleted>, '_content', '_content_consumed', 'apparent_encoding', 'close',
'connection', 'content', 'cookies', 'elapsed', 'encoding', 'headers', 'history',
'is_permanent_redirect', 'is_redirect', 'iter_content', 'iter_lines', 'json', 'links', 'ok',
'raise_for_status', 'raw', 'reason', 'request', 'status_code', 'text', 'url']
>>> resp.status_code, resp.reason
(200, 'OK')
>>> resp.headers
{'Content-Length': '29055', 'Content-Encoding': 'gzip', 'Set-Cookie':
'SRCHD=AF=NOFORM; domain=.bing.com; expires=Wed, 11-Apr-2019 12:48:57 GMT;
'Content-Type': 'text/html; charset=utf-8'}
>>> resp.content[:70]
b'<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://'
```

The response from the server

All the headers in the response

The content of the webpage!



## Browser GET/POST Requests

Create your browser object

```
>>> import requests
>>> browser = requests.session()
```

Make GET requests to retrieve content

```
>>> resp = browser.get("http://www.bing.com")
>>> resp.content[:60]
b'<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//E'
```

Make POST requests to submit data to forms

```
>>> postdata = {'username': 'markb', 'password': 'sec573'}
>>> resp = browser.post("http://web.page/login", postdata)
>>> print(resp.content)
b'Login Failed'
```

Cookies and other settings automatically persist across all actions that use the browser object

## Instead of a website we could use a database!

- MaxMind maintains and distributes several IP information databases
- The free location database product is referred to as "GeoLite2"
- Module is easily installed with pip `$ pip install geoip2`
- Databases can be directly downloaded from their website
  - <https://dev.maxmind.com/geoip/geoip2/geolite2/>
- Or a utility called `geoipupdate` can automatically download the monthly updates
  - First, add the MaxMind repository to your apt repository list
  - Then install their free `geoipupdate` utility and update your local database
  - Register for a free account to obtain an API key

```
$ sudo add-apt-repository ppa:maxmind/ppa
$ sudo apt update
$ sudo apt install geoipupdate
$ sudo geoipupdate
$ sudo python -m pip install geoip2
```

---

# **WALKTHROUGH WORKBOOK 1 and 2 TOGETHER**

---

## Challenge 2

# Forensics Challenge 2 - At the scene of the crime

After obtaining the suspects laptop you need to place him at the scene of the crime. After pulling their wireless history from the registry, srurn and event logs you come up with the following list of Wireless BSSIDs. What locations can you place the suspects laptop at from the following Wireless BSSID addresses?

```
[ '70-0F-6A-32-2C-0F' ,  
  '0C-F5-A4-96-CD-C7' ,  
  '74-3E-2B-36-57-E8' ,  
  '08-4F-A9-3C-F5-A0' ,  
  '70-70-8B-29-12-81' ,  
  '0C-F5-A4-96-CD-CB' ]
```

## Workbook 3 - Looking up a single BSSID

- There are limited resources available for looking up the location of 1 wireless network.
- Wigle.net - Free community based website
  - Fed by people collecting these from their phones and other devices and uploading that to wiggle
  - Free API is available for anyone to query the data
  - Administrators keep tight reigns on abuse
- A similar site to wigle focused on Europe
  - Much smaller database
  - "<http://api.mylnikov.org/geolocation/wifi>

## How to get a Wigle.net API key

- <http://wigle.net> and click TOOLS -> ACCOUNT or
- <https://wigle.net/login?destination=/account>



Username:

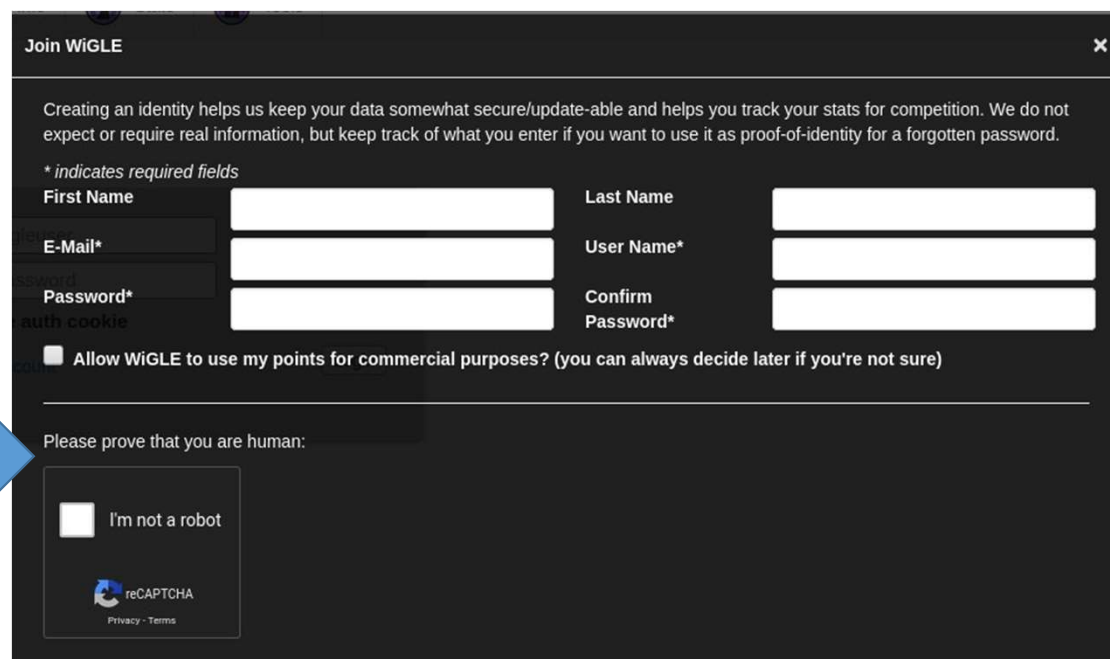
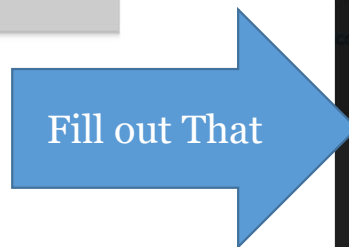
Password:

☐ Don't expire auth cookie

[register a new account](#)  
[reset password](#)

Login

Click That



Join WIGLE

Creating an identity helps us keep your data somewhat secure/update-able and helps you track your stats for competition. We do not expect or require real information, but keep track of what you enter if you want to use it as proof-of-identity for a forgotten password.

*\* indicates required fields*

First Name	<input type="text"/>	Last Name	<input type="text"/>
E-Mail*	<input type="text"/>	User Name*	<input type="text"/>
Password*	<input type="password"/>	Confirm Password*	<input type="password"/>

☐ Allow WIGLE to use my points for commercial purposes? (you can always decide later if you're not sure)

Please prove that you are human:

☐ I'm not a robot

reCAPTCHA  
[Privacy](#) - [Terms](#)

## Go back to your account page after logging in

- Login to wigle and go back to your ACCOUNT page

The screenshot shows the WIGLE API interface with several fields and a button. A blue arrow labeled "Click That" points to the "Show my token" button. Below this, there are four rows of information:

- Encoded for use:** A text input field containing a long blacked-out string.
- Your Header value should be:** A text input field containing "Authorization: Basic" followed by a long blacked-out string.
- To test with curl:** A text input field containing the command: `curl -i -H 'Accept:application/json' -u [redacted] --basic https://api.wigle.net/api/v2/profile/user`.
- API Name:** A text input field containing a blacked-out string.
- API Token:** A text input field containing a blacked-out string.

Two blue arrows point to the "API Name" and "API Token" fields: one labeled "Username" pointing to the API Name field, and another labeled "Password" pointing to the API Token field.



## Workbook 4 - Geolocating BSSIDs

- Google and several others provide WIFI triangulation services
  - You submit BSSID and the signal strength and it calculates where you are relative to those Wireless Access Points
  - Event 6100s are pretty scarce in forensics information
  - With a little creativity we can still use these APIs to find artifacts that are plentiful in our forensics investigations

## Google Geolocation API Pricing

- \$5.00 for 1000 API requests
- The first \$200.00 (40,000 request) a month is free

## Obtaining a Google Maps API key

- Google "Get Google Maps API key"
- <https://cloud.google.com/console/google/maps-apis/overview>

This is a three step process

- 1) Enable the GEOLOCATION API in your account
- 2) Provide a Credit Card under your billing information
- 3) Create an API key that can use that API

Maps APIs and Servi... - Google Maps - My Project - Google Cloud Platform - Mozilla Firefox

Get an API Key | Maps Java... | Maps APIs and Servi... - Goop... | Google Cloud Platform

https://console.cloud.google.com/google/maps-apis/new?project=my-project-1494086980873&folder=&organizationId=

Your free trial is waiting: activate now to get \$300 credit to explore Google Cloud products. [Learn more](#)

Google Cloud Platform My Project Search products and resources

Google Maps

- Overview
- APIs
- Quotas
- Credentials
- Metrics
- Support

Maps APIs and Services

Browse the [Marketplace](#) to find and use Maps APIs and services.

Geolocation API Google Location data from cell towers and WiFi nodes.

Click That

Maps Embed API Google Make places easily discoverable with interactive Google Maps.

Time Zone API Google Time zone data for any point in the world.

Then ENABLE the API

Directions API Google Directions between multiple locations.

Maps Static API Google Simple, embeddable map image with minimal code.

Geocoding API Google Convert between addresses and geographic coordinates.

Street View Static API Google Real-world imagery and panoramas.

Geolocation API Google Location data from cell towers and WiFi nodes. ENABLE

Free Trail

MISS ACTIVATE

Google Cloud Platform My Project

API Library

# Create the API key and check Restrictions

The screenshot illustrates the process of creating an API key and checking its restrictions in the Google Cloud Platform console. The interface is divided into two main sections: the left sidebar and the main content area.

**Left Sidebar:**

- Home
- Pins appear here
- Marketplace
- Billing
- APIs & Services
- Support
- IAM & Admin
- Getting started
- Security

**Main Content Area:**

- Google Cloud Platform
- TestProject
- APIs & Services
- Credentials
- + CREATE CREDENTIALS

**Application restrictions:**

An application restriction controls which websites, IP addresses, or applications can use your API key. You can set one application restriction per key.

- ☒ None
- ☐ HTTP referrers (web sites)
- ☐ IP addresses (web servers, cron jobs, etc.)
- ☐ Android apps
- ☐ iOS apps

**API restrictions:**

API restrictions specify the enabled APIs that this key can call

- ☐ Don't restrict key
- ☒ Restrict key

1 API

Selected APIs:

Geolocation API

Note: It may take up to 5 minutes for settings to take effect

SAVE CANCEL

**Annotations:**

- 1 - Click That (points to the 'APIs & Services' menu item)
- 2 - Click That (points to the 'Credentials' menu item)
- 3 - Click That (points to the 'CREATE CREDENTIALS' button)
- 4 - Click That (points to the 'CREATE CREDENTIALS' button)
- 5 - Good Idea (points to the 'Restrict key' radio button)

You have an API key!

Google Cloud Platform TestProject Search products and resources

Google Maps Credentials Geolocation API

Overview APIs Quotas Credentials Metrics Support

### Credentials compatible with this API

To view all credentials or create new credentials visit [Credentials in APIs & Services](#)

Remember to configure the OAuth consent screen with information about your application. [CONFIGURE CONSENT SCREEN](#)

#### API Keys

Name	Creation date	Restrictions ↓	Key		Usage with this service (last 30 days) ?	Usage with all services (last 30 days) ?	
✓ API key 1	Feb 20, 2020	Geolocation API	[REDACTED]		47	47	

---

# **WALKTHROUGH WORKBOOK 3 and 4 TOGETHER**

---



## A Tool that does all this for you?

- Werejugo does all of this for you.
  - BSSID in registry, event logs and other used to determine locations
    - Single BSSIDs via Wigle and BSSID pairs via google
  - Names of those wireless network in other artifacts used for date and ties
- It is Free!
- Download it here:  
<https://github.com/MarkBaggett/werejugo>
- You provide your own API keys
- "results.html" in your workbook is sample output

## NOW IT IS YOUR TURN

- Can you solve the two forensics challenges?
- Use code from Workbooks 1 and 2 to solve Challenge #1
- Use code from Workbooks 3 and 4 to solve Challenge #2
- You can copy and paste code from the workbooks into your solutions book.
- You can continue to use my personal API key as you work on Forensics Challenge 2 but please be considerate.

## LAB TIME!

- If you have questions or need help you can:
  - Post a question directly into SLACK
  - Reach out to one of our instructor via a Direct Message
  - Ask any of the following people for help!

Mark Baggett

Mike Pilkington

Mark Hallman

Don Williams

Bryan Scarbrough

Charlie Goldner