CS 205 **Lecture Handout**
November 5, 2015

An **API** (**A**pplication **P**rogramming **I**nterface) is any interface that allows you to get data in a format that is designed for another application to use.

The CUMTD API gives us data about the CUMTD bus system (where are stops located, what routes to buses take) and about the current state of the CUMTD buses.
…CUMTD documentation: https://developer.cumtd.com/

The first prat of working with an API is to find out how the APIs will communicate data with you. There are two major types of APIs:

- **RESTful** (**Re**presentational **S**tate **T**ransfer): For an API to be considered a RESTful API, it must have the following properties:
  1. **Client-Server Architecture**: API requests from a client must be directed to a server (a single endpoint, like a URL)
  2. **Stateless**: Every request must be entirely self-contained. The server must not "save" any information about one request and have it impact the next request.
  3. **Catchable**: Data must have a valid-until or expiration timestamp, or must be defined as part of the protocol.

- **Publish/Subtribe**: For an API to be considered a Publish/Subscribe API, a client must connect to a server as a subscriber and receive updates as they are published by the server.

Example: An API to return the current temperature at a given location:

RESTful	Publish/Subscribe
Client makes a request to get the current temperature and the current temperature is returned. If the client wants to know if the temperature has changed, it must make another API request.	Client subscribes to the temperature "Pub-Sub" server to get temperature updates. When the temperature changes, all subscribes receive a broadcast of the new temperature.

The CUMTD is a **RESTful** API, so we will be making a request every time we want to get updated information. We will not get notified if the information changes until we make another request.

CUMTD Request Format

To send a CUMTD request, their documentation tells us:

> *The RESTful endpoints use the HTTP GET verb with query string parameters.*

What is HTTP GET?

HTTP is the protocol used to request web pages and a GET request is an HTTP request that contains key-value pairs encoded as part of the URL.

```
http://cs.illinois.edu/?a=3&b=5&s=Hello
```

URL:	
Parameters:	

API Method: GetDeparturesByStop

Get a list of real-time departures for a specific stop_id.
URL: https://developer.cumtd.com/api/v2.2/json/GetDeparturesByStop

Parameters:

Key	Value
`key`	Your API key, **required**
`stop_id`	CUMTD identifier for a stop, **required** (`IU` is Illini Union, `IT` is Illinois Terminal; see GFTS data)
`route_id`	Semi-colon separated list of CUMTD route identifies, optional
`pt`	Preview time in minutes between 0 and 60, defaults to 30
`count`	Maximum number of departures to return, optional

Puzzle #1:

Make an HTML GET request by hand for getting all of the buses that will be stopping at the Illini Union in the next 30 minutes. Test that URL by entering it into a web browser.

Making API Requests in Python

Making a request by hand is good, but having a computer do it for you is even better! For this, there are three things we need to accomplish:

1. Load our CUMTD API key from …**`/static/keys/cumtd.txt`**
2. Construct the HTTP GET request
3. Read the HTTP response by CUMTD's API server

Part 1: Loading the CUMTD API key

You can load this file the same way you have loaded all the other files to date in Python. The path to this file is the tricky bit:

- Your code is running in: …/demo_cumtd/
- The cumtd.txt file is in: …/static/keys/
- This means you must navigate up to get out of demo_cumtd and then into the static/keys/ folder. *What path is that?*

Part 2: Construction the HTTP GET request

Since HTTP GET is a widely used standard, the Python language contains functions to help us make our HTTP GET query string:

```
# Load the urllib library
import urllib
...

# Construct a Python dictionary
parameters = {
  "key": myKey,
  "stop_id": "IU"
}

# Construct the full URL
url = "https://developer.cumtd.com/api/v2.2/json/" +
      "GetDeparturesByStop?" +
      urllib.urlencode( parameters )
```

Part 3: Read the response

Using the same library, we are able to make the HTTP GET request and get back the response into Python as a String:

```
# Read the response
response = urllib.urlopen( url )
```

Puzzle #2:

Complete the **`do_compute()`** function to make an API request to CUMTD. For now, **`print`** out the response to see what CUMTD gives us!

Building a Timetable Application

The timetable screens shown at the CUMTD bus stops shows the upcoming buses and the amount of time they're away from the current stop:

5E Green:	**2 mins**
12W Teal:	**5 mins**
22S Illini:	**6 mins**
13N Silver:	**10 mins**

In order to build this, we need to accomplish only three things:

1. Convert the response string into a Python dictionary
2. Loop through all the upcoming departures
3. Add relevant information about each departure into an new list

Part 1: Converting a string into a Python dictionary

Just like HTTP GET, JSON is a widely used standard. Python provides a way for us to convert a JSON string into a Python dictionary:

```
# Load the json library
import json
...

# Parse a JSON string into a Python dictionary
dictionary = json.load( response )
```

Part 2: Loop through the departures

Look at the data you received – what data do you want to loop through?

Part 3: Construct the results list

As your looping through the data, you should construct a new list of dictionaries. Specifically, your output should follow the following format:

```
[
  { "route": "5E Green", "expected": 3 },
  { "route": "12W Teal", "expected": 5 },
  { "route": "22S Illini", "expected": 6 },
  { "route": "13N Silver", "expected": 10 },
]
```

Puzzle #3:

Complete the **`do_compute()`** and save the new list as **timetable.json**. The HTML/JavaScript has already been created to display your data for you!