Module 3 - Web APIs

Table of Contents

Introduction

- Back to Table of Contents

In this module, we introduce two general ways that one can retrieve data from data sources on the Internet: APIs and web scraping. We've already covered web scraping, and given the messiness of that subject you may find yourself really appreciating the relative simplicity of using APIs - once you grasp the concepts.

API stands for "Application Programming Interface". An API is an agreed upon way for one computer program to interact with another computer program. There are many different kinds of APIs. Some facilitate interaction between computers over the Internet, some do not. In fact, the Python module SQL Alchemy that we used yesterday is a type of API for talking to databases - you'll see more on APIs for databases with SQL Alchemy later in this course.

For this session, we focus on web APIs over HTTP that let a user query and retrieve data over the Internet. This type of API documents an agreed-upon structure and content of requests and responses a program can use to interact with a system. As long as your code adheres to a system's API, it should be able to reliabily request and receive data from the system.

Below, we show how to make network API requests using HTTP(S).

Learning Objectives

- Back to Table of Contents

Learning objectives:

- **Become familiar with different types of APIs.** Includes GET- and POST- based HTTP APIs, different formats of request bodies for POST-based APIs (form inputs, arbitrary JSON and XML, and then formalized dialects of each like SOAP), and how to learn a given API.
- **Learn the tools used to interact with network-based APIs.** Understand the tools for talking directly with APIs over HTTP connection, introduce libraries that abstract the details of the API and present a simplified programmatic interface, and then understand how to choose a tool.

Topics

- Back to Table of Contents

Outline of topics covered in this notebook:

- Making raw HTTP API requests
- Using pre-packaged API client libraries
- Practical considerations - Knowing API rules and coding to follow them, and performance
- Example: OpenTripPlanner

Setup - Load Python packages

- back to Table of Contents

In [1]:

```
## import Python packages ##
import time # to convert time as needed and report how long some functio
ns take

# interacting with websites and web-APIs
import requests # easy way to interact with web sites and services
import json # read/write JavaScript Object Notation (JSON)
from bs4 import BeautifulSoup

# data manipulation
import pandas as pd # easy data manipulation
import geopandas as gpd # geographic data manipulation
# from geopandas.tools import sjoin, overlay # spatial join and overlay
 functions
from shapely.geometry import Point, LineString # to create lines from a
 list of points

# visualization
import matplotlib as mplib
import matplotlib.pyplot as plt # visualization package

# so images get plotted in the notebook
%matplotlib inline
```

In [2]:

```
print("Package versions")
print("requests: {}".format(requests.__version__))
print("json: {}".format(json.__version__))
print("pandas: {}".format(pd.__version__))
print("geopandas: {}".format(gpd.__version__)) # check that correct vers
ion of geopandas is installed, should be v0.2+
print("matplotlib: {}".format(mplib.__version__))
```

```
Package versions
requests: 2.13.0
json: 2.0.9
pandas: 0.19.2
geopandas: 0.3.0
matplotlib: 2.0.0
```

Using APIs

- Back to the Table of Contents

API overview

- In general: APIs (Application Programming Interfaces (https://en.wikipedia.org/wiki/Application_programming_interface)) are "set[s] of subroutine definitions, protocols, and tools for building software and applications. A good API makes it easier to develop a program by providing all the building blocks, which are then put together by the programmer."
- Here we're looking at a **web-API**, a specific type of API which makes it easier to interact with some aspect of a website. In this course, we'll be using APIs to gether data in an automated way - like grabbing a bunch of prior tweets from Twitter. More generally, APIs can also be used to interact with websites in any way the API is designed to. For instance, you can post and delete tweets with Twitter's API, too.

Geocoding

- Back to the Table of Contents

We have scraped addresses, but OTP works best with latitude and longitude coordinates. We can use the geocoder module (https://pypi.python.org/pypi/geocoder) to get latitude and longitude exactly just from the organization addresses. Note this is a combination of great tools - a simple Python module (`geocoder`) interacting with Google's wonderful geocoding API. The code below would work out in the wild, but since we are working in a restricted environment, we can't get to the Google API.

The geocoder module can speak to a wide range of external services, including ArcGIS, Bing, MapBox, OpenStreetMaps, and many others, in addition to Google. The API lets you geocode (addresses to latitue and longitude), reverse geocode (latitude and longitude to addresses), as well as get timezones and elevations on locations.

Google geocoder API

Use Google API (https://developers.google.com/maps/documentation/geocoding/intro) to geocode service locations

```
In [3]: centers_df = pd.read_csv('./data/chicago-workforce-centers.csv')
        centers_df.head()
```

Out[3]:

	Unnamed: 0	address	center_name	phone_number
0	0	5957 W 87th St. Oak Lawn IL 60453	A.E.R.O. Special Education	(708) 499-0181
1	1	6707 North Ave. Oak Park IL 60302	African American Christian Foundation	(708) 848-1700
2	2	1945 W Wilson Chicago IL 60640	Albany Park Community Center	(773) 539-3828
3	3	1807 W Sunnyside Suite 1D Chicago IL 60640	Alternative Schools Network	(773) 728-4030
4	4	723 W Algonquin Arlington Heights IL 60005	Arlington Heights Workforce CenterBusiness & C...	(847) 981-7400

```
In [4]: centers_df.shape

Out[4]: (55, 4)
```

In [5]:

```
import geocoder

lat = []
lon = []

for add in centers_df["address"]:
    g = geocoder.google(add)

    if g.status == 'OK':
        lon.append(g.latlng[1])
        lat.append(g.latlng[0])
    else:
        print('No result or over query limit for {}, adding empty value
 as placeholder'.format(add))
        lon.append(None)
        lat.append(None)
```

```
No result or over query limit for Prairie State CollegeATOC Building202
S. Halsted, Suite 148Chicago Heights, IL 60411, adding empty value as p
laceholder
No result or over query limit for 500 N. Sacramento Chicago IL 60612790
N MilwaukeeChicago IL 60642, adding empty value as placeholder
No result or over query limit for 216 W. JacksonChicago IL 60606300 Rev
ere Dr.Northbrook IL 60062, adding empty value as placeholder
No result or over query limit for 1010 Dixie Hwy #102 Chicago Heights I
L 60411200 W Adams Chicago IL 60606, adding empty value as placeholder
No result or over query limit for 571 West Jackson Blvd. Chicago IL 606
6115402 Center Harvey IL 60426, adding empty value as placeholder
No result or over query limit for 500 N Dearborn Chicago IL 60654, addi
ng empty value as placeholder
No result or over query limit for 191 N. Wacker Drive Suite 925 Chicago
IL 60606, adding empty value as placeholder
No result or over query limit for 4343 W Wrightwood Ave Chicago IL 6063
9936 N Ashland Ave Chicago IL 606226520 S Wood St Chicago IL 60636, add
ing empty value as placeholder
```

In [6]:

```
# add results to our data frame

centers_df["latitude"] = pd.Series(lat)
centers_df["longitude"] = pd.Series(lon)
```

In [7]:

```
# optional hardcoded values if needed

# lat = [41.733737,
#              41.9087846,
#              41.9647695,
#              41.9631174,
#              42.0457523,
#              41.9697109,
#              41.9252578,
#              41.4824241,
#              41.5081785,
#              41.8322347,
#              41.6944193,
#              41.8444394,
#              41.8849173,
#              41.5255653,
#              41.8543913,
#              41.8511856,
#              41.8931701,
#              41.8804296,
#              41.8905965,
#              42.0076194,
#              41.9647485,
#              41.8457521,
#              41.8409604,
#              41.9589605]

# lon = [-87.770246,
#               -87.7931388,
#               -87.6786497,
#               -87.6748518,
#               -87.9922059,
#               -87.6598793,
#               -87.7008122,
#               -87.6782855,
#               -87.6234975,
#               -87.5990999,
#               -87.5990999,
#               -87.7236882,
#               -87.6231249,
#               -87.6386009,
#               -87.6355797,
#               -87.7775432,
#               -87.6614166,
#               -87.7066519,
#               -87.702801,
#               -87.6689743,
#               -87.6570292,
#               -87.6858569,
#               -87.6862319,
#               -87.6747326]
```

In [8]:

```
# if use hardcoded:
# centers_lim = centers_df[0:24]
# centers_lim["latitude"] = pd.Series(lat)
# centers_lim["longitude"] = pd.Series(lon)

# if use geocoded values
centers_lim = centers_df[~centers_df['latitude'].isnull()]
print(centers_lim.shape)
centers_lim.head()
```

```
(47, 6)
```

Out[8]:

	Unnamed: 0	address	center_name	phone_number	latitude	longitude
0	0	5957 W 87th St. Oak Lawn IL 60453	A.E.R.O. Special Education	(708) 499-0181	41.733737	-87.770246
1	1	6707 North Ave. Oak Park IL 60302	African American Christian Foundation	(708) 848-1700	41.908789	-87.793126
2	2	1945 W Wilson Chicago IL 60640	Albany Park Community Center	(773) 539-3828	41.964770	-87.678650
3	3	1807 W Sunnyside Suite 1D Chicago IL 60640	Alternative Schools Network	(773) 728-4030	41.963117	-87.674852
4	4	723 W Algonquin Arlington Heights IL 60005	Arlington Heights Workforce CenterBusiness & C...	(847) 981-7400	42.045752	-87.992206

OpenTripPlanner

OpenTripPlanner (OTP) (http://docs.opentripplanner.org/en/latest/) is an open source routing software that provides a number of services, here we'll explore:

1. Index API - provides information about the data loaded into OTP, for instance what transportation agencies' data are included;
2. Routing API - creates a plan for how to get from one location to another, with a number of additional options such as:
   - Departure time (and date) - if you're curious about a specific departure time or date;
   - transit modes - default is to consider any public transportation option in the system, but it can also be set to "AUTO" to do vehicle routing or "WALK" for walking only directions.
3. Isochrone API - generates a polygon representing the area a traveler can reach if they start from a given location and travel for a specified amount of time (isochrone means 'equal time').

Data:

- OpenStreetMap (OSM) for Chicago from Mapzen's Metro Extracts (https://mapzen.com/data/metro-extracts/metro/chicago_illinois/)
- General Transit Feed Specification (GTFS) for Chicago from transit.land (https://transit.land/feed-registry/?metro=Chicago)

If we "build a graph" locally we can go to our browser here (http://localhost:8080/ (http://localhost:8080/)) and see what we have.

RESTful APIs

- Back to the Table of Contents

The OTP APIs are what is called "RESTful (https://en.wikipedia.org/wiki/Representational_state_transfer)" web services. REST stands for REpresentational State Transfer, but don't worry about the acronym so much as the idea. RESTful services adhere to a series of requirements (https://www.restapitutorial/whatisrest.html) that enable them to be consistent, scalable, reliable, and relatively simple. RESTful APIs allow you to access a pre-defined set of operations through HTTP(S) requests. REST is fantastic because, in part, if you can generate the right URL, you'll always get the right response (this was not always the case with SOAP - the predecessor to REST).

To use a RESTful API, we'll need to understand (1) how to properly format the request and (2) how to manage and make use of the response from the API. Below we will walk through these concepts while using some of OTP's web services.

In [9]:

```
### First, we need to set a few parameters. ###

# base URL where OTP is installed
base_url = "http://localhost:8080/otp/routers/"
```

In [10]:

```
# Router ID -
# OTP could have many different routers available for different cities or subsets of transportation agencies.
# in this example we only have one, unnamed router for Chicago
routerID = 'default/' #

# update base_url to include router name
base_url += routerID
print(base_url)
```

```
http://localhost:8080/otp/routers/default/
```

Index API

- Back to the Table of Contents

The Index API provides access to general information about the data loaded into a given OTP router (as specified by the 'routerID' variable set above). Full list of options are here (http://dev.opentripplanner.org/apidoc/1.0.0/resource_IndexAPI.html).

Below, we can make a request simply by taking our base URL and adding the `feeds` endpoint. Here, we use the term endpoint to refer to the completed URL that links to the most granular aspect of an API. The combination of the router id, index API, and feeds request make up our endpoint. This will provide a list of data feeds available for the router we've selected.

In [11]:

```
# Set up query URL
qry_url = '{}index/feeds'.format(base_url)

# Again, since we are still using HTTP, we can use the requests package's get
response = requests.get(qry_url)

# Convert response to text
response = response.text

# Our response is a JSON array:
print(response)
```

```
["1","2","3"]
```

JSON

- Back to the Table of Contents

JSON (www.json.org) is a common non-tabular data format often used by services and software on the internet. We'll introduce JSON slowly, but it's helpful to know that it is oriented around the idea of `key-value` pairs. The `keys` refer to information about the data while the `values` is the data itself. For instance, if you were to translate a tabular dataset into JSON, the column names (and possibly row names/numbers) would become `keys`, while the data in the cells would become `values`. Our first JSON response is a simple array, the equivalent of a Python list.

In [12]:

```
# Convert text to a Python object using the 'json' package
feeds = json.loads(response)

# And now we have a Python list:
print(type(feeds))
print(feeds)
```

```
<class 'list'>
['1', '2', '3']
```

This is just a list of feed IDs created by OTP - so there are three agencies providing data feeds to our version of the OTP. This is not particularly informative, but we did get a response from the web API. Let's check the 'agencies' endpoint to see what more it provides.

In [13]:

```
## We can use the /agences resource of the Index API to get more informa
tion.
## Below, we combine the previous steps into one line and ask for the ag
ency associated with the first feed:
print('{}index/agencies/{}'.format(base_url, '1'))
print(json.loads(requests.get('{}index/agencies/{}'.format(base_url,
'1')).text))
```

```
http://localhost:8080/otp/routers/default/index/agencies/1
[{'id': '1', 'name': 'Chicago Transit Authority', 'url': 'http://transi
tchicago.com', 'timezone': 'America/Chicago', 'lang': 'en', 'phone': '1
-888-YOURCTA', 'fareUrl': 'http://www.transitchicago.com/travel_informa
tion/fares/default.aspx'}]
```

This is more helpful - we now know the agency associated with the feed id, as well as its website and other information. Let's use a loop to repeat this for the other feeds.

In [14]:
```
## Let's do the same for each feed:
for feed in feeds:

    # print out which feed we're looking at on this pass of the loop
    print("feed {}".format(feed))

    # get agency information for this feed just as we did above, but using the feed from our list of feeds
    agency = json.loads(requests.get('{}index/agencies/{}'.format(base_url, feed)).text)

    print(agency)

    # add a blank line after each feed for legibility
    print('')
```

```
feed 1
[{'id': '1', 'name': 'Chicago Transit Authority', 'url': 'http://transi
tchicago.com', 'timezone': 'America/Chicago', 'lang': 'en', 'phone': '1
-888-YOURCTA', 'fareUrl': 'http://www.transitchicago.com/travel_informa
tion/fares/default.aspx'}]

feed 2
[{'id': 'METRA', 'name': 'Metra', 'url': 'http://www.metrarail.com/',
 'timezone': 'America/Chicago', 'lang': 'EN'}]

feed 3
[{'id': 'PACE', 'name': 'PACE', 'url': 'http://www.pacebus.com', 'timez
one': 'America/Chicago', 'lang': 'en'}]
```

API Documentation

- Back to the Table of Contents

If you want more information about what routes are included in a given feed, you can query the 'routes' resources as below. At this point, you may be wondering how you would know what resources and endpoints are available for a given API. This is where API documentation comes in. The OpenTripPlanner Index API documentation (dev.opentripplanner.org/apidoc/0.20.0/resource_indexAPI.html) includes a list of valid HTTP methods (mostly get and a few post) for the resources and specific endpoints within the Index API.

For instance, there is a valid HTTP get request for the URL `/routers/{routerid}/index/agencies/{feedId}` where routerid and feedId are changeable parameters. This is how you would have known that the above requests would be successful.

Below, we can examine all the routes of one agency in the format: `/routers/{router_id}/index/agencies/{feedID}/{agencyID}/routes`

In [15]:

```
# Using agency 'METRA' from first feed
routes = json.loads(requests.get('{0}index/agencies/{1}/{2}/routes'.form
at(base_url, '1', '1')).text)
print(routes)
```

```
[{'id': '1:1', 'shortName': '1', 'longName': 'Bronzeville/Union Statio
n', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id':
 '1:121', 'shortName': '121', 'longName': 'Union/Wacker Express', 'mod
e': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:124',
 'shortName': '124', 'longName': 'Navy Pier', 'mode': 'BUS', 'agencyNam
e': 'Chicago Transit Authority'}, {'id': '1:3', 'shortName': '3', 'long
Name': 'King Drive', 'mode': 'BUS', 'agencyName': 'Chicago Transit Auth
ority'}, {'id': '1:2', 'shortName': '2', 'longName': 'Hyde Park Expres
s', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id':
 '1:126', 'shortName': '126', 'longName': 'Jackson', 'mode': 'BUS', 'ag
encyName': 'Chicago Transit Authority'}, {'id': '1:5', 'shortName':
 '5', 'longName': 'South Shore Night Bus', 'mode': 'BUS', 'agencyName':
'Chicago Transit Authority'}, {'id': '1:4', 'shortName': '4', 'longNam
e': 'Cottage Grove', 'mode': 'BUS', 'agencyName': 'Chicago Transit Auth
ority'}, {'id': '1:125', 'shortName': '125', 'longName': 'Water Tower E
xpress', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'i
d': '1:7', 'shortName': '7', 'longName': 'Harrison', 'mode': 'BUS', 'ag
encyName': 'Chicago Transit Authority'}, {'id': '1:6', 'shortName':
 '6', 'longName': 'Jackson Park Express', 'mode': 'BUS', 'agencyName':
 'Chicago Transit Authority'}, {'id': '1:9', 'shortName': '9', 'longNam
e': 'Ashland', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authorit
y'}, {'id': '1:8', 'shortName': '8', 'longName': 'Halsted', 'mode': 'BU
S', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:21', 'shortNa
me': '21', 'longName': 'Cermak', 'mode': 'BUS', 'agencyName': 'Chicago
 Transit Authority'}, {'id': '1:20', 'shortName': '20', 'longName': 'Ma
dison', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'i
d': '1:12', 'shortName': '12', 'longName': 'Roosevelt', 'mode': 'BUS',
 'agencyName': 'Chicago Transit Authority'}, {'id': '1:11', 'shortNam
e': '11', 'longName': 'Lincoln', 'mode': 'BUS', 'agencyName': 'Chicago
 Transit Authority'}, {'id': '1:15', 'shortName': '15', 'longName': 'Je
ffery Local', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authorit
y'}, {'id': '1:18', 'shortName': '18', 'longName': '16th-18th', 'mode':
'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:19', 'shor
tName': '19', 'longName': 'United Center Express', 'mode': 'BUS', 'agen
cyName': 'Chicago Transit Authority'}, {'id': '1:120', 'shortName': '12
0', 'longName': 'Ogilvie/Wacker Express', 'mode': 'BUS', 'agencyName':
 'Chicago Transit Authority'}, {'id': '1:111', 'shortName': '111', 'lon
gName': '111th/King Drive', 'mode': 'BUS', 'agencyName': 'Chicago Trans
it Authority'}, {'id': '1:112', 'shortName': '112', 'longName': 'Vincen
nes/111th', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'},
 {'id': '1:115', 'shortName': '115', 'longName': 'Pullman/115th', 'mod
e': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:119',
 'shortName': '119', 'longName': 'Michigan/119th', 'mode': 'BUS', 'agen
cyName': 'Chicago Transit Authority'}, {'id': '1:Red', 'longName': 'Red
Line', 'mode': 'SUBWAY', 'color': 'C60C30', 'agencyName': 'Chicago Tran
sit Authority'}, {'id': '1:143', 'shortName': '143', 'longName': 'Stock
ton/Michigan Express', 'mode': 'BUS', 'agencyName': 'Chicago Transit Au
thority'}, {'id': '1:146', 'shortName': '146', 'longName': 'Inner Driv
e/Michigan Express', 'mode': 'BUS', 'agencyName': 'Chicago Transit Auth
ority'}, {'id': '1:148', 'shortName': '148', 'longName': 'Clarendon Mic
higan Express', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authorit
y'}, {'id': '1:147', 'shortName': '147', 'longName': 'Outer Drive Expre
ss', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id':
 '1:43', 'shortName': '43', 'longName': '43rd', 'mode': 'BUS', 'agencyN
ame': 'Chicago Transit Authority'}, {'id': '1:81W', 'shortName': '81W',
'longName': 'West Lawrence', 'mode': 'BUS', 'agencyName': 'Chicago Tran
sit Authority'}, {'id': '1:34', 'shortName': '34', 'longName': 'South M
```

```
ichigan', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'},
 {'id': '1:36', 'shortName': '36', 'longName': 'Broadway', 'mode': 'BU
S', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:35', 'shortNa
me': '35', 'longName': '35th', 'mode': 'BUS', 'agencyName': 'Chicago Tr
ansit Authority'}, {'id': '1:Blue', 'longName': 'Blue Line', 'mode': 'S
UBWAY', 'color': '00A1DE', 'agencyName': 'Chicago Transit Authority'},
 {'id': '1:37', 'shortName': '37', 'longName': 'Sedgwick', 'mode': 'BU
S', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:39', 'shortNa
me': '39', 'longName': 'Pershing', 'mode': 'BUS', 'agencyName': 'Chicag
o Transit Authority'}, {'id': '1:P', 'longName': 'Purple Line', 'mode':
'SUBWAY', 'color': '522398', 'agencyName': 'Chicago Transit Authorit
y'}, {'id': '1:132', 'shortName': '132', 'longName': 'Goose Island Expr
ess', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id':
'1:135', 'shortName': '135', 'longName': 'Clarendon/LaSalle Express',
 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:1
34', 'shortName': '134', 'longName': 'Stockton/LaSalle Express', 'mod
e': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:136',
 'shortName': '136', 'longName': 'Sheridan/LaSalle Express', 'mode': 'B
US', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:Y', 'longNam
e': 'Yellow Line', 'mode': 'SUBWAY', 'color': 'F9E300', 'agencyName':
 'Chicago Transit Authority'}, {'id': '1:49B', 'shortName': '49B', 'lon
gName': 'North Western', 'mode': 'BUS', 'agencyName': 'Chicago Transit
 Authority'}, {'id': '1:30', 'shortName': '30', 'longName': 'South Chic
ago', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id':
'1:31', 'shortName': '31', 'longName': '31st', 'mode': 'BUS', 'agencyNa
me': 'Chicago Transit Authority'}, {'id': '1:22', 'shortName': '22', 'l
ongName': 'Clark', 'mode': 'BUS', 'agencyName': 'Chicago Transit Author
ity'}, {'id': '1:24', 'shortName': '24', 'longName': 'Wentworth', 'mod
e': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:26',
 'shortName': '26', 'longName': 'South Shore Express', 'mode': 'BUS',
 'agencyName': 'Chicago Transit Authority'}, {'id': '1:111A', 'shortNam
e': '111A', 'longName': 'Pullman Shuttle', 'mode': 'BUS', 'agencyName':
'Chicago Transit Authority'}, {'id': '1:29', 'shortName': '29', 'longNa
me': 'State', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authorit
y'}, {'id': '1:G', 'longName': 'Green Line', 'mode': 'SUBWAY', 'color':
'009B3A', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:28', 's
hortName': '28', 'longName': 'Stony Island', 'mode': 'BUS', 'agencyNam
e': 'Chicago Transit Authority'}, {'id': '1:130', 'shortName': '130',
 'longName': 'Museum Campus', 'mode': 'BUS', 'agencyName': 'Chicago Tra
nsit Authority'}, {'id': '1:165', 'shortName': '165', 'longName': 'West
65th', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'i
d': '1:169', 'shortName': '169', 'longName': '69th-UPS Express', 'mod
e': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:60',
 'shortName': '60', 'longName': 'Blue Island/26th', 'mode': 'BUS', 'age
ncyName': 'Chicago Transit Authority'}, {'id': '1:63', 'shortName': '6
3', 'longName': '63rd', 'mode': 'BUS', 'agencyName': 'Chicago Transit A
uthority'}, {'id': '1:62', 'shortName': '62', 'longName': 'Archer', 'mo
de': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:65',
 'shortName': '65', 'longName': 'Grand', 'mode': 'BUS', 'agencyName':
 'Chicago Transit Authority'}, {'id': '1:56', 'shortName': '56', 'longN
ame': 'Milwaukee', 'mode': 'BUS', 'agencyName': 'Chicago Transit Author
ity'}, {'id': '1:55', 'shortName': '55', 'longName': 'Garfield', 'mod
e': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:57',
 'shortName': '57', 'longName': 'Laramie', 'mode': 'BUS', 'agencyName':
'Chicago Transit Authority'}, {'id': '1:59', 'shortName': '59', 'longNa
me': '59th/61st', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authori
ty'}, {'id': '1:155', 'shortName': '155', 'longName': 'Devon', 'mode':
 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:157', 'sh
ortName': '157', 'longName': 'Streeterville/Taylor', 'mode': 'BUS', 'ag
encyName': 'Chicago Transit Authority'}, {'id': '1:156', 'shortName':
 '156', 'longName': 'LaSalle', 'mode': 'BUS', 'agencyName': 'Chicago Tr
ansit Authority'}, {'id': '1:50', 'shortName': '50', 'longName': 'Dame
n', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id':
 '1:52', 'shortName': '52', 'longName': 'Kedzie/California', 'mode': 'B
US', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:51', 'shortN
ame': '51', 'longName': '51st', 'mode': 'BUS', 'agencyName': 'Chicago T
ransit Authority'}, {'id': '1:54', 'shortName': '54', 'longName': 'Cice
ro', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id':
 '1:53', 'shortName': '53', 'longName': 'Pulaski', 'mode': 'BUS', 'agen
cyName': 'Chicago Transit Authority'}, {'id': '1:44', 'shortName': '4
4', 'longName': 'Wallace-Racine', 'mode': 'BUS', 'agencyName': 'Chicago
Transit Authority'}, {'id': '1:47', 'shortName': '47', 'longName': '47t
h', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id':
 '1:49', 'shortName': '49', 'longName': 'Western', 'mode': 'BUS', 'agen
cyName': 'Chicago Transit Authority'}, {'id': '1:48', 'shortName': '4
8', 'longName': 'South Damen', 'mode': 'BUS', 'agencyName': 'Chicago Tr
ansit Authority'}, {'id': '1:151', 'shortName': '151', 'longName': 'She
ridan', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'i
d': '1:152', 'shortName': '152', 'longName': 'Addison', 'mode': 'BUS',
 'agencyName': 'Chicago Transit Authority'}, {'id': '1:81', 'shortNam
e': '81', 'longName': 'Lawrence', 'mode': 'BUS', 'agencyName': 'Chicago
Transit Authority'}, {'id': '1:80', 'shortName': '80', 'longName': 'Irv
ing Park', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'},
 {'id': '1:82', 'shortName': '82', 'longName': 'Kimball-Homan', 'mode':
'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:85', 'shor
tName': '85', 'longName': 'Central', 'mode': 'BUS', 'agencyName': 'Chic
ago Transit Authority'}, {'id': '1:84', 'shortName': '84', 'longName':
 'Peterson', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'},
{'id': '1:87', 'shortName': '87', 'longName': '87th', 'mode': 'BUS', 'a
gencyName': 'Chicago Transit Authority'}, {'id': '1:86', 'shortName':
 '86', 'longName': 'Narragansett/Ridgeland', 'mode': 'BUS', 'agencyNam
e': 'Chicago Transit Authority'}, {'id': '1:X9', 'shortName': 'X9', 'lo
ngName': 'Ashland Express', 'mode': 'BUS', 'agencyName': 'Chicago Trans
it Authority'}, {'id': '1:78', 'shortName': '78', 'longName': 'Montros
e', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id':
 '1:77', 'shortName': '77', 'longName': 'Belmont', 'mode': 'BUS', 'agen
cyName': 'Chicago Transit Authority'}, {'id': '1:79', 'shortName': '7
9', 'longName': '79th', 'mode': 'BUS', 'agencyName': 'Chicago Transit A
uthority'}, {'id': '1:85A', 'shortName': '85A', 'longName': 'North Cent
ral', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id':
'1:70', 'shortName': '70', 'longName': 'Division', 'mode': 'BUS', 'agen
cyName': 'Chicago Transit Authority'}, {'id': '1:72', 'shortName': '7
2', 'longName': 'North', 'mode': 'BUS', 'agencyName': 'Chicago Transit
 Authority'}, {'id': '1:71', 'shortName': '71', 'longName': '71st South
Shore', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'i
d': '1:74', 'shortName': '74', 'longName': 'Fullerton', 'mode': 'BUS',
 'agencyName': 'Chicago Transit Authority'}, {'id': '1:73', 'shortNam
e': '73', 'longName': 'Armitage', 'mode': 'BUS', 'agencyName': 'Chicago
Transit Authority'}, {'id': '1:X49', 'shortName': 'X49', 'longName': 'W
estern Express', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authorit
y'}, {'id': '1:76', 'shortName': '76', 'longName': 'Diversey', 'mode':
 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:75', 'sho
rtName': '75', 'longName': '74th-75th', 'mode': 'BUS', 'agencyName': 'C
hicago Transit Authority'}, {'id': '1:67', 'shortName': '67', 'longNam
```

```
e': '67th-69th-71st', 'mode': 'BUS', 'agencyName': 'Chicago Transit Aut
hority'}, {'id': '1:66', 'shortName': '66', 'longName': 'Chicago', 'mod
e': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:68',
 'shortName': '68', 'longName': 'Northwest Highway', 'mode': 'BUS', 'ag
encyName': 'Chicago Transit Authority'}, {'id': '1:Org', 'longName': 'O
range Line', 'mode': 'SUBWAY', 'color': 'F9461C', 'agencyName': 'Chicag
o Transit Authority'}, {'id': '1:171', 'shortName': '171', 'longName':
 'U. of Chicago/Hyde Park', 'mode': 'BUS', 'agencyName': 'Chicago Trans
it Authority'}, {'id': '1:172', 'shortName': '172', 'longName': 'U. of
 Chicago/Kenwood', 'mode': 'BUS', 'agencyName': 'Chicago Transit Author
ity'}, {'id': '1:63W', 'shortName': '63W', 'longName': 'West 63rd', 'mo
de': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:90',
 'shortName': '90', 'longName': 'Harlem', 'mode': 'BUS', 'agencyName':
 'Chicago Transit Authority'}, {'id': '1:92', 'shortName': '92', 'longN
ame': 'Foster', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authorit
y'}, {'id': '1:91', 'shortName': '91', 'longName': 'Austin', 'mode': 'B
US', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:94', 'shortN
ame': '94', 'longName': 'South California', 'mode': 'BUS', 'agencyNam
e': 'Chicago Transit Authority'}, {'id': '1:93', 'shortName': '93', 'lo
ngName': 'California/Dodge', 'mode': 'BUS', 'agencyName': 'Chicago Tran
sit Authority'}, {'id': '1:96', 'shortName': '96', 'longName': 'Lunt',
 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:9
5', 'shortName': '95', 'longName': '95th', 'mode': 'BUS', 'agencyName':
'Chicago Transit Authority'}, {'id': '1:97', 'shortName': '97', 'longNa
me': 'Skokie', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authorit
y'}, {'id': '1:88', 'shortName': '88', 'longName': 'Higgins', 'mode':
 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:8A', 'sho
rtName': '8A', 'longName': 'South Halsted', 'mode': 'BUS', 'agencyNam
e': 'Chicago Transit Authority'}, {'id': '1:192', 'shortName': '192',
 'longName': 'University of Chicago Hosp. Exp.', 'mode': 'BUS', 'agency
Name': 'Chicago Transit Authority'}, {'id': '1:62H', 'shortName': '62
H', 'longName': 'Archer/Harlem', 'mode': 'BUS', 'agencyName': 'Chicago
 Transit Authority'}, {'id': '1:X98', 'shortName': 'X98', 'longName':
 'Avon Express', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authorit
y'}, {'id': '1:53A', 'shortName': '53A', 'longName': 'South Pulaski',
 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:5
2A', 'shortName': '52A', 'longName': 'South Kedzie', 'mode': 'BUS', 'ag
encyName': 'Chicago Transit Authority'}, {'id': '1:Pink', 'longName':
 'Pink Line', 'mode': 'SUBWAY', 'color': 'E27EA6', 'agencyName': 'Chica
go Transit Authority'}, {'id': '1:201', 'shortName': '201', 'longName':
'Central/Ridge', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authorit
y'}, {'id': '1:205', 'shortName': '205', 'longName': 'Chicago/Golf', 'm
ode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:20
6', 'shortName': '206', 'longName': 'Evanston Circulator', 'mode': 'BU
S', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:55A', 'shortN
ame': '55A', 'longName': '55th/Austin', 'mode': 'BUS', 'agencyName': 'C
hicago Transit Authority'}, {'id': '1:55N', 'shortName': '55N', 'longNa
me': '55th/Narragansett', 'mode': 'BUS', 'agencyName': 'Chicago Transit
Authority'}, {'id': '1:54B', 'shortName': '54B', 'longName': 'South Cic
ero', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id':
'1:54A', 'shortName': '54A', 'longName': 'North Cicero/Skokie Blvd.',
 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:1
00', 'shortName': '100', 'longName': 'Jeffery Manor Express', 'mode':
 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:103', 'sh
ortName': '103', 'longName': 'West 103rd', 'mode': 'BUS', 'agencyName':
'Chicago Transit Authority'}, {'id': '1:106', 'shortName': '106', 'long
Name': 'East 103rd', 'mode': 'BUS', 'agencyName': 'Chicago Transit Auth
```

```
ority'}, {'id': '1:108', 'shortName': '108', 'longName': 'Halsted/95t
h', 'mode': 'BUS', 'agencyName': 'Chicago Transit Authority'}, {'id':
 '1:Brn', 'longName': 'Brown Line', 'mode': 'SUBWAY', 'color': '62361
B', 'agencyName': 'Chicago Transit Authority'}, {'id': '1:J14', 'shortN
ame': 'J14', 'longName': 'Jeffery Jump', 'mode': 'BUS', 'agencyName':
 'Chicago Transit Authority'}]
```

In [16]:

```
# Routes is a list - a data structure we are familiar with:
print(type(routes))

# However, the objects that makes up this list may be new to you, the py
thon dictionary:
print(type(routes[0]))
print(type(routes[1]))
print(type(routes[2]))
```

```
<class 'list'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
```

Python Dictionary

- Back to the Table of Contents

The dictionary (https://docs.python.org/3/tutorial/datastructures.html#dictionaries), or dict for short, is a common type of python object used to store sets of key-value pairs. Sound familiar? It should! Dictionaries are python's internal counterpart to JSON data.

Here, we'll learn to grab data from within a dict by using the key name, following this syntax: `dict['key']`

In [17]:

```
test_route = routes[3]
print(test_route)
print('------')
print(test_route['agencyName'])
print('------')
print(test_route['mode'])
```

```
{'id': '1:3', 'shortName': '3', 'longName': 'King Drive', 'mode': 'BU
S', 'agencyName': 'Chicago Transit Authority'}
------
Chicago Transit Authority
------
BUS
```

In [18]:

```
## Simple enough - let's use a loop and our new understanding of dicts to print out the mode and route name:
for route in routes:
    print('mode: {} | id: {} | route name: {}'.format(route['mode'], route['id'], route['longName']))
```

```
mode: BUS | id: 1:1 | route name: Bronzeville/Union Station
mode: BUS | id: 1:121 | route name: Union/Wacker Express
mode: BUS | id: 1:124 | route name: Navy Pier
mode: BUS | id: 1:3 | route name: King Drive
mode: BUS | id: 1:2 | route name: Hyde Park Express
mode: BUS | id: 1:126 | route name: Jackson
mode: BUS | id: 1:5 | route name: South Shore Night Bus
mode: BUS | id: 1:4 | route name: Cottage Grove
mode: BUS | id: 1:125 | route name: Water Tower Express
mode: BUS | id: 1:7 | route name: Harrison
mode: BUS | id: 1:6 | route name: Jackson Park Express
mode: BUS | id: 1:9 | route name: Ashland
mode: BUS | id: 1:8 | route name: Halsted
mode: BUS | id: 1:21 | route name: Cermak
mode: BUS | id: 1:20 | route name: Madison
mode: BUS | id: 1:12 | route name: Roosevelt
mode: BUS | id: 1:11 | route name: Lincoln
mode: BUS | id: 1:15 | route name: Jeffery Local
mode: BUS | id: 1:18 | route name: 16th-18th
mode: BUS | id: 1:19 | route name: United Center Express
mode: BUS | id: 1:120 | route name: Ogilvie/Wacker Express
mode: BUS | id: 1:111 | route name: 111th/King Drive
mode: BUS | id: 1:112 | route name: Vincennes/111th
mode: BUS | id: 1:115 | route name: Pullman/115th
mode: BUS | id: 1:119 | route name: Michigan/119th
mode: SUBWAY | id: 1:Red | route name: Red Line
mode: BUS | id: 1:143 | route name: Stockton/Michigan Express
mode: BUS | id: 1:146 | route name: Inner Drive/Michigan Express
mode: BUS | id: 1:148 | route name: Clarendon Michigan Express
mode: BUS | id: 1:147 | route name: Outer Drive Express
mode: BUS | id: 1:43 | route name: 43rd
mode: BUS | id: 1:81W | route name: West Lawrence
mode: BUS | id: 1:34 | route name: South Michigan
mode: BUS | id: 1:36 | route name: Broadway
mode: BUS | id: 1:35 | route name: 35th
mode: SUBWAY | id: 1:Blue | route name: Blue Line
mode: BUS | id: 1:37 | route name: Sedgwick
mode: BUS | id: 1:39 | route name: Pershing
mode: SUBWAY | id: 1:P | route name: Purple Line
mode: BUS | id: 1:132 | route name: Goose Island Express
mode: BUS | id: 1:135 | route name: Clarendon/LaSalle Express
mode: BUS | id: 1:134 | route name: Stockton/LaSalle Express
mode: BUS | id: 1:136 | route name: Sheridan/LaSalle Express
mode: SUBWAY | id: 1:Y | route name: Yellow Line
mode: BUS | id: 1:49B | route name: North Western
mode: BUS | id: 1:30 | route name: South Chicago
mode: BUS | id: 1:31 | route name: 31st
mode: BUS | id: 1:22 | route name: Clark
mode: BUS | id: 1:24 | route name: Wentworth
mode: BUS | id: 1:26 | route name: South Shore Express
mode: BUS | id: 1:111A | route name: Pullman Shuttle
mode: BUS | id: 1:29 | route name: State
mode: SUBWAY | id: 1:G | route name: Green Line
mode: BUS | id: 1:28 | route name: Stony Island
mode: BUS | id: 1:130 | route name: Museum Campus
mode: BUS | id: 1:165 | route name: West 65th
mode: BUS | id: 1:169 | route name: 69th-UPS Express
```

```
mode: BUS | id: 1:60 | route name: Blue Island/26th
mode: BUS | id: 1:63 | route name: 63rd
mode: BUS | id: 1:62 | route name: Archer
mode: BUS | id: 1:65 | route name: Grand
mode: BUS | id: 1:56 | route name: Milwaukee
mode: BUS | id: 1:55 | route name: Garfield
mode: BUS | id: 1:57 | route name: Laramie
mode: BUS | id: 1:59 | route name: 59th/61st
mode: BUS | id: 1:155 | route name: Devon
mode: BUS | id: 1:157 | route name: Streeterville/Taylor
mode: BUS | id: 1:156 | route name: LaSalle
mode: BUS | id: 1:50 | route name: Damen
mode: BUS | id: 1:52 | route name: Kedzie/California
mode: BUS | id: 1:51 | route name: 51st
mode: BUS | id: 1:54 | route name: Cicero
mode: BUS | id: 1:53 | route name: Pulaski
mode: BUS | id: 1:44 | route name: Wallace-Racine
mode: BUS | id: 1:47 | route name: 47th
mode: BUS | id: 1:49 | route name: Western
mode: BUS | id: 1:48 | route name: South Damen
mode: BUS | id: 1:151 | route name: Sheridan
mode: BUS | id: 1:152 | route name: Addison
mode: BUS | id: 1:81 | route name: Lawrence
mode: BUS | id: 1:80 | route name: Irving Park
mode: BUS | id: 1:82 | route name: Kimball-Homan
mode: BUS | id: 1:85 | route name: Central
mode: BUS | id: 1:84 | route name: Peterson
mode: BUS | id: 1:87 | route name: 87th
mode: BUS | id: 1:86 | route name: Narragansett/Ridgeland
mode: BUS | id: 1:X9 | route name: Ashland Express
mode: BUS | id: 1:78 | route name: Montrose
mode: BUS | id: 1:77 | route name: Belmont
mode: BUS | id: 1:79 | route name: 79th
mode: BUS | id: 1:85A | route name: North Central
mode: BUS | id: 1:70 | route name: Division
mode: BUS | id: 1:72 | route name: North
mode: BUS | id: 1:71 | route name: 71st South Shore
mode: BUS | id: 1:74 | route name: Fullerton
mode: BUS | id: 1:73 | route name: Armitage
mode: BUS | id: 1:X49 | route name: Western Express
mode: BUS | id: 1:76 | route name: Diversey
mode: BUS | id: 1:75 | route name: 74th-75th
mode: BUS | id: 1:67 | route name: 67th-69th-71st
mode: BUS | id: 1:66 | route name: Chicago
mode: BUS | id: 1:68 | route name: Northwest Highway
mode: SUBWAY | id: 1:Org | route name: Orange Line
mode: BUS | id: 1:171 | route name: U. of Chicago/Hyde Park
mode: BUS | id: 1:172 | route name: U. of Chicago/Kenwood
mode: BUS | id: 1:63W | route name: West 63rd
mode: BUS | id: 1:90 | route name: Harlem
mode: BUS | id: 1:92 | route name: Foster
mode: BUS | id: 1:91 | route name: Austin
mode: BUS | id: 1:94 | route name: South California
mode: BUS | id: 1:93 | route name: California/Dodge
mode: BUS | id: 1:96 | route name: Lunt
mode: BUS | id: 1:95 | route name: 95th
mode: BUS | id: 1:97 | route name: Skokie
```

```
mode: BUS | id: 1:88 | route name: Higgins
mode: BUS | id: 1:8A | route name: South Halsted
mode: BUS | id: 1:192 | route name: University of Chicago Hosp. Exp.
mode: BUS | id: 1:62H | route name: Archer/Harlem
mode: BUS | id: 1:X98 | route name: Avon Express
mode: BUS | id: 1:53A | route name: South Pulaski
mode: BUS | id: 1:52A | route name: South Kedzie
mode: SUBWAY | id: 1:Pink | route name: Pink Line
mode: BUS | id: 1:201 | route name: Central/Ridge
mode: BUS | id: 1:205 | route name: Chicago/Golf
mode: BUS | id: 1:206 | route name: Evanston Circulator
mode: BUS | id: 1:55A | route name: 55th/Austin
mode: BUS | id: 1:55N | route name: 55th/Narragansett
mode: BUS | id: 1:54B | route name: South Cicero
mode: BUS | id: 1:54A | route name: North Cicero/Skokie Blvd.
mode: BUS | id: 1:100 | route name: Jeffery Manor Express
mode: BUS | id: 1:103 | route name: West 103rd
mode: BUS | id: 1:106 | route name: East 103rd
mode: BUS | id: 1:108 | route name: Halsted/95th
mode: SUBWAY | id: 1:Brn | route name: Brown Line
mode: BUS | id: 1:J14 | route name: Jeffery Jump
```

Routing API

- Back to the Table of Contents

Now that we've tested we can access OTP from Jupyter, let's do something a bit more interesting: get a route plan between some locations. This will allow us to answer "How long will it take to get from *here* to *there*?"

Similar to the Index API, the Routing API documentation (http://dev.opentripplanner.org/apidoc/1.0.0/resource_PlannerResource.html) tells us what features are available and how to access those feastures.

Planner Resource Syntax

- Back to the Table of Contents

The Planner Resource API does trip planning based on a large number of customizable parameters. To give you a sense of all the options available, OTP's planner resource allows users to set the additional time it will take baord a vehicle (like a bus) with a bike, as opposed to boarding on foot. There are a lot of available options. This is great for us, since once we understand the simple syntax of this API, we can avail ourselves of this granular customization if we want to.

This resource is located at `/OTP/routers/{routerID}/plan` and when setting options within a URL, they follow a single question mark. So first, let's plan a trip with just the required options, `fromPlace`, `toPlace`, and `datae` (which takes an option in the form YYY-MM-DD) which you can see we set after `/plan?` and separated by ampersands `&`:

```
In [19]: centers_lim.shape

Out[19]: (47, 6)
```

In [20]:

```
origin_lat = centers_lim["latitude"][1]
origin_lon = centers_lim["longitude"][1]

destination_lat = centers_lim["latitude"].values[10]
destination_lon = centers_lim["longitude"].values[10]

qry_url = '{}plan?fromPlace={},{}&toPlace={},{}'\
.format(base_url, origin_lat, origin_lon, destination_lat, destination_lon)

print(qry_url)

response = requests.get(qry_url)
response = response.text
plan = json.loads(response)

# Examine the response, which is a routing plan:
print(plan)
```

```
http://localhost:8080/otp/routers/default/plan?fromPlace=41.9087889,-8
7.7931255&toPlace=41.6944193,-87.5990999
```

```
{'requestParameters': {'fromPlace': '41.9087889,-87.7931255', 'toPlac
e': '41.6944193,-87.5990999'}, 'plan': {'date': 1511852867000, 'from':
 {'name': 'Origin', 'lon': -87.7931255, 'lat': 41.9087889, 'orig': '',
 'vertexType': 'NORMAL'}, 'to': {'name': 'Destination', 'lon': -87.5990
999, 'lat': 41.6944193, 'orig': '', 'vertexType': 'NORMAL'}, 'itinerari
es': [{'duration': 8459, 'startTime': 1511863780000, 'endTime': 1511872
239000, 'walkTime': 1411, 'transitTime': 6355, 'waitingTime': 693, 'wal
kDistance': 1829.7204661610099, 'walkLimitExceeded': False, 'elevationL
ost': 0.0, 'elevationGained': 0.0, 'transfers': 3, 'legs': [{'startTim
e': 1511863780000, 'endTime': 1511863845000, 'departureDelay': 0, 'arri
valDelay': 0, 'realTime': False, 'distance': 78.965, 'pathway': False,
 'mode': 'WALK', 'route': '', 'agencyTimeZoneOffset': -21600000, 'inter
lineWithPreviousLeg': False, 'from': {'name': 'Origin', 'lon': -87.7931
255, 'lat': 41.9087889, 'departure': 1511863780000, 'orig': '', 'vertex
Type': 'NORMAL'}, 'to': {'name': 'North Ave & Linden', 'stopId': '1:84
9', 'stopCode': '849', 'lon': -87.7924506, 'lat': 41.90899595, 'arriva
l': 1511863845000, 'departure': 1511863846000, 'stopIndex': 4, 'stopSeq
uence': 5, 'vertexType': 'TRANSIT'}, 'legGeometry': {'points': '}hx~Fra
zvOW?Y@AwA?a@', 'length': 5}, 'rentedBike': False, 'duration': 65.0, 't
ransitLeg': False, 'steps': [{'distance': 28.494, 'relativeDirection':
 'DEPART', 'streetName': 'service road', 'absoluteDirection': 'NORTH',
 'stayOn': False, 'area': False, 'bogusName': False, 'lon': -87.7930527
6267166, 'lat': 41.90879023607601, 'elevation': []}, {'distance': 50.47
1000000000004, 'relativeDirection': 'RIGHT', 'streetName': 'West North
 Avenue', 'absoluteDirection': 'EAST', 'stayOn': False, 'area': False,
 'bogusName': False, 'lon': -87.7930618, 'lat': 41.9090464, 'elevatio
n': []}]}, {'startTime': 1511863846000, 'endTime': 1511865631000, 'depa
rtureDelay': 0, 'arrivalDelay': 0, 'realTime': False, 'distance': 1031
1.042312088763, 'pathway': False, 'mode': 'BUS', 'route': '72', 'agency
Name': 'Chicago Transit Authority', 'agencyUrl': 'http://transitchicag
o.com', 'agencyTimeZoneOffset': -21600000, 'routeType': 3, 'routeId':
 '1:72', 'interlineWithPreviousLeg': False, 'tripBlockId': '48400000143
7', 'headsign': 'Clark', 'agencyId': '1', 'tripId': '1:484119443477',
 'serviceDate': '20171128', 'from': {'name': 'North Ave & Linden', 'sto
pId': '1:849', 'stopCode': '849', 'lon': -87.7924506, 'lat': 41.9089959
5, 'arrival': 1511863845000, 'departure': 1511863846000, 'stopIndex':
 4, 'stopSequence': 5, 'vertexType': 'TRANSIT'}, 'to': {'name': 'North
 Ave & Ashland', 'stopId': '1:902', 'stopCode': '902', 'lon': -87.66787
773, 'lat': 41.91055657, 'arrival': 1511865631000, 'departure': 1511865
631000, 'stopIndex': 55, 'stopSequence': 56, 'vertexType': 'TRANSIT'},
 'legGeometry': {'points': 'ojx~F|_zvOAi@Aw@?e@Am@?a@?i@?g@?m@?e@Ae@?g@
@IAs@Cq@?i@?e@Ce@?i@?g@?o@?i@?o@?e@?e@?k@?U?q@Cs@Ae@?o@Ak@?k@@i@?k@?e@?
i@?O?e@?c@?w@Ak@?m@?g@?i@?a@NYAy@Aq@?s@?q@?{@?s@B{@@o@?g@?C[q@?m@?e@?e
@?g@Ci@?k@?i@?i@?k@?i@?g@?]Lk@Ai@Cm@?}@AaA?aAA_A@_A?q@?m@@a@?i@A_@AsI??
Ca@CeMKm@Ao@Ae@Au@?u@?e@?g@?e@?i@?i@?e@?q@?o@Ai@?EL_@?e@Ao@?g@?g@?m@?c@
Ak@@e@Ao@Ac@Cg@@MCe@?m@?i@Ao@@y@?{@@g@@o@A_@IkA?y@Aa@?o@?m@?k@?g@Am@?_
@?g@?c@Ak@Ae@?k@?i@?i@?e@Ak@?a@?e@?i@?e@?o@?i@?KAe@Ao@Ai@?g@Ai@?e@?s@?e
@?e@@KBe@Au@Ck@?s@Aq@Cs@?u@?o@?q@Co@Ae@?o@?a@Dc@Eo@Cu@?e@?m@?i@?i@?k@BU
Ds@?k@?q@?o@?s@?g@?k@?o@?o@?o@?OCo@Ak@C{@B_A?y@?{@?k@?s@?WGu@Co@Ei@?u@A
e@?e@?o@Cg@Bm@?o@?m@?_@?o@Co@Ai@?]Ag@Au@Ce@?e@?e@?e@?e@?q@Bm@Bm@?g@?a@?
K?i@?o@?e@Aq@?i@?e@Ao@?[?o@Ai@Ao@?m@?u@?o@?o@Ae@?i@?a@?u@Au@Ai@?i@?e@?i
@?q@A_@Cg@Ai@?o@?o@Ae@?i@?k@?k@?k@?m@Ag@?m@?e@@k@@YCk@Ak@Co@?y@?w@?y@?w
@Cm@Aq@?i@?o@?UHm@?i@?s@?m@As@Ai@?e@?g@?e@?m@?m@?m@?O?g@Es@?u@?e@?u@@k
@?i@?IQm@Ae@Ai@?o@Ck@?s@?q@?q@Am@@k@Ba@Em@Cg@?o@Ao@?y@?w@?s@?o@Be@?g@?
C?k@Cc@?q@Ak@?u@?s@Aq@?e@Ao@Bc@@e@?A?i@Ck@Ci@?g@?s@?o@?o@Cg@Ai@Bo@?e@@?
```

```
R]GwPAg@@{EIeEAg@?s@?q@?s@?k@Ag@?s@Ak@Ae@?i@?i@?e@?k@@Q?m@Aq@?e@@i@@g@?
g@?m@?c@O[Ee@?i@Ao@?o@@q@?q@?m@@g@?c@?q@?e@?A?k@?m@Cu@?m@Aq@?s@?w@?s@?k
@Ao@?g@Bm@?g@@CAe@Ae@?o@Ak@?i@@e@?i@?e@?K@i@?a@Ci@?q@?c@?k@?e@Ck@?e@As
@?k@Di@?E?i@Ee@?m@Am@?q@?s@?q@?k@?k@?g@?q@?o@?UAq@Ao@Ae@?i@?k@?e@?m@?a
@?i@?G?o@Cy@?k@?o@Aq@Ao@Am@?k@Ba@@e@?SEk@Ae@?k@Co@?s@?q@Cs@Ak@?e@Bi@@a
@?KIe@Ci@Co@?i@Aa@?y@?q@?e@Ai@@o@Be@?IEm@Ag@Ag@?k@Ai@Ao@?e@?i@Ae@?q@Bc
@?QCo@Ak@?y@?q@Ao@?u@Bk@@i@Ak@?e@?CEi@Cq@?k@Ei@?o@?k@?k@?o@?k@?g@Bg@@YE
o@?q@?e@?e@?c@?u@Ae@?i@?e@?q@@k@?M?k@Ak@Ae@?i@Ai@?m@?i@Ak@?o@@s@?e@BACi
@Ce@?q@Cy@?s@Aq@?i@?g@@e@Fe@?c@', 'length': 554}, 'routeShortName': '7
2', 'routeLongName': 'North', 'rentedBike': False, 'duration': 1785.0,
 'transitLeg': True, 'steps': []}, {'startTime': 1511865631000, 'endTim
e': 1511865650000, 'departureDelay': 0, 'arrivalDelay': 0, 'realTime':
 False, 'distance': 21.444, 'pathway': False, 'mode': 'WALK', 'route':
 '', 'agencyTimeZoneOffset': -21600000, 'interlineWithPreviousLeg': Fal
se, 'from': {'name': 'North Ave & Ashland', 'stopId': '1:902', 'stopCod
e': '902', 'lon': -87.66787773, 'lat': 41.91055657, 'arrival': 15118656
31000, 'departure': 1511865631000, 'stopIndex': 55, 'stopSequence': 56,
'vertexType': 'TRANSIT'}, 'to': {'name': 'Ashland & North Ave', 'stopI
d': '1:6019', 'stopCode': '6019', 'lon': -87.66784143, 'lat': 41.910734
26, 'arrival': 1511865650000, 'departure': 1511866184000, 'stopIndex':
 15, 'stopSequence': 16, 'vertexType': 'TRANSIT'}, 'legGeometry': {'poi
nts': 'mtx~FhsavO?YS@', 'length': 3}, 'rentedBike': False, 'duration':
 19.0, 'transitLeg': False, 'steps': [{'distance': 10.072, 'relativeDir
ection': 'DEPART', 'streetName': 'West North Avenue', 'absoluteDirectio
n': 'EAST', 'stayOn': False, 'area': False, 'bogusName': False, 'lon':
 -87.66788008986359, 'lat': 41.910631065090115, 'elevation': []}, {'dis
tance': 11.372, 'relativeDirection': 'LEFT', 'streetName': 'North Ashla
nd Avenue', 'absoluteDirection': 'NORTH', 'stayOn': False, 'area': Fals
e, 'bogusName': False, 'lon': -87.66775840000001, 'lat': 41.9106332, 'e
levation': []}]}, {'startTime': 1511866184000, 'endTime': 151186984000
0, 'departureDelay': 0, 'arrivalDelay': 0, 'realTime': False, 'distanc
e': 22770.311642453416, 'pathway': False, 'mode': 'BUS', 'route': '9',
 'agencyName': 'Chicago Transit Authority', 'agencyUrl': 'http://transi
tchicago.com', 'agencyTimeZoneOffset': -21600000, 'routeType': 3, 'rout
eId': '1:9', 'interlineWithPreviousLeg': False, 'tripBlockId': '4840000
03378', 'headsign': '104th/Vincennes', 'agencyId': '1', 'tripId': '1:48
4119717237', 'serviceDate': '20171128', 'from': {'name': 'Ashland & Nor
th Ave', 'stopId': '1:6019', 'stopCode': '6019', 'lon': -87.66784143,
 'lat': 41.91073426, 'arrival': 1511865650000, 'departure': 15118661840
00, 'stopIndex': 15, 'stopSequence': 16, 'vertexType': 'TRANSIT'}, 't
o': {'name': 'Beverly & 103rd/Vincennes', 'stopId': '1:6138', 'stopCod
e': '6138', 'lon': -87.65616492, 'lat': 41.70695661, 'arrival': 1511869
840000, 'departure': 1511869840000, 'stopIndex': 87, 'stopSequence': 8
8, 'vertexType': 'TRANSIT'}, 'legGeometry': {'points': 'mux~FnravOj@FTC
D?TCNATANATAPCT?N?T?J?T?D?T?T?P?XCT?L?T?N?TANCT?L?T?J?TCJAT?N?T?T?T?P?
T?R?TBH@T?L?F?TCDAT?H?T?H?T?N?T?R?T?J?T?F?T?H?@?T?X?T?H?T?J?T?J?T?D?T?
J?T?L?T?N?T?H?T?PMDCJJT?TAT?T?T?TAT?T?T?TAT?T?T?TAT?T?T?TAT?TAT?T?TAT?T
AR?T?TAT?T?TAT?TAT?T?TAL?T?TAT?T?T?TAT?T?T?TAT?T?T?TAR?T?T?TAT?T?T?TAT?
T?T?T?TAT?T?T?T?T?P?TAT?TAT?TAT?T?TAT?T?RAT?TAT?T?TAT?TAT?T?TAD?T?T?TA
T?T?T?TAT?T?T?TAT?T?T?TAT?TAT?TAT?J?PAT?T?T?TAT?T?TAT?R?T?TAT?T?T?TAT?
T?TAT?TAT?T?TAT?T?TAT?TAT?T?TAT?TAR?T?TAL?l@S\\G^?`@?d@?f@Ed@A^E^?X?d@?
`@?\\?B?c@LpIGdKMVBzLITAhHIZ?D?TAT?T?T?TAT?T?T?TAT?T?TAT?TAT?X?TAT?T?R?
TAT?T?T?TAT?T?T?TAF?PGHD|MMLBxMKZ?|IIl@ATCRAT?H?T?L?TAHARCHAT?L?TANAT?
L?T?N?T?J?T?L?T?F?T?TATAT?TATAT?TATAT?TATAT?TAT?LAT@LANVd@?\\?`@C^?^EX?
^HRC`@UT@X?XAT?J?T?R?TATCT?V?TATAT?T?T?L?T?JCRCT?HAT?T?L?T?T@N?L?RAT?T?
TAT?TAT?T?JAN?`@Bb@?\\C`@Ab@?\\?d@?\\?Z@b@Af@?d@?Z?P@d@QT?T@T?T@T?T?T@
T?TARAT?F?T?F?T?F?T?H?T?H?T?J?T?H?TFLCLHLITCTAT?T?T?R?T?T?T?T?T?T?T?T?
```

```
T?T?T?T?H?T?T?T?RAZ?TETET?R?T?T?TAT?N?T?T?T?TAT?T?TAR?T?TAT?TAT?H?F?RH
T?T?T?TAT?T?T?TAT?T?T?T?T?T?TAT?T?T?T?T?T?TAT?T?R?T?T?T?T?TAT?T?T?
T?T?T?XAT?T?T?T?TAT?T?T?T?T?T?TAR?T?T?T?T?T?TAT?T?T?T?T?T?TAT?T?
H?TAT?TADATAJ?VGLS?OBS?UHQVMRMBCRMDAD_@G[UM[DSFMDQLIFK\\CVJZRPl@dANDR?r
CB^EJCT?T?T?TAT?T?T?T?TAT?T?T?T?T?T?PAT?R?T?TAT?T?T?TAT?T?T?T?R?N?PA
T?R?T?TAT?T?TAT?T?TAT?TAT?X?T?T?T?T?T?TAT?T?T?T?T?T?RAT?N?F?J?V?TA
T?TAJ?TAT?TAT?@?T?TAT?T?B?TAR?T?T?BAT?T?T?TA??T?T?TAT???T?TAT?P?TAT?T?
N?T?F@R?T@T?R?T?TAT?T?F?TAT?T?RAD?T?T?T?L?T?T?T?T?T?T?T?T???T?T?TAT?
T?T?JATAT?TA@?T?TAT?@?TGJMX^xGk@tSK^E`@?f@Aj@?l@Gh@?j@?b@?\\FP?TAT?T?T?
TAT?R?T?TAT?T?T?TAT?T?TAT?T?TAT?TAT?T?TAT?T?TAT?TAT?T?TAR?T?TAT?TAT@TB
L?T?TAT?T?@AT?T?T?RAT?P?l@E`@?Z?\\EZ?^?b@?b@?Z?\\C`@?^?\\?^BHCQAhJG\\?~
FIIHT?T?TAT?T?T?TAT?T?T?TAT?T?TAT?T?T?TAT?T?T?RAT?T?T?TAT?T?T?T?TAT?T?
L?T?TAR?TATAT?TAT?TAT?TAT?TAT?TAT?TAT?TAVAU?U?U?K?f@Bd@Cn@An@?l@?j@Cb@?
ZB\\@h@?^AJ?T?P?TAHAT?J?T?H?TALAT?L?T?J?TATCHAF?TAT?T?T?T?TAT?T?T?R?T?
T?T?J?T?J?T?L?R?T?F?T?F?T?J?V?\\?T?TAF?T?T?TAT?L?TAT?TAT?N?TAT?TAT?N?TA
T?TAT?P?TAT?N@XAPAT?TAT?T?NAT?T?TAT?N?T?T?RAT?N?T?T?TAT?N?T?T?T?T?P?T?
T?T?T?PAT?P?T?F?JYVPT?T?TAT?TAT?T?TAR?T?TAT?TAT?T?TAT?T?T?TAT?T?T?TAT?
R?T?T?T?T?T?J?T?T?T?VAH@AIPFT?RAT?T?T?TAT?T?T?TAP?TAF?F?T?F?T?H?T?L?T?
T?TCTCT?TAT?T?T?R?T?L?R?F?T?F?T?TALAT?PAT?F?TAJAT?J?T?L?T?H?T?T?T?T?T?
T?TAT?T?R?T?T?T?T?TAT?T?D?T?TAT?TAT?TAT?TAT?T?RAT?TAT?T?TAT?H?TAT?T?T?
T?T?T?T?TAJ?R?T?T?T?T?T?TAT?T?L?TAT?TAT?TAT?TAT?TAT?TAT?TAT?T?TAT?TAT?R
AT?T?TAT?T?TAT?TAT?T?TAT?T?VATADATCLAT?R?T?Z?T?X?T?P?T?F?@?R?T?T?T?TAT?
T?T?TAT?T?TAT?TAT?T?TAT?TAT?H?TAT?T?TAV?T?TAT?T?T?TAT?T?TAT?T?TAT?TAT?
T?TAT?T?TAT?TAT?T?VAT?T?T?P?T?T?T?R?T?R?T?R?T?T?TAT?B?T?TAT?T?FAT?L?T?
T?TAL?T?T?T?TAL?T?T?TAT?L?T?TAT?T?L?TAT?T?T?LAT?T?T?T?T?J?TAT?T?T?PAT?
T?T?TAN?T?T?T?TAN?T?R?\\CT?T?T?TAT?T?T?TAT?T?TAT?R?TAT?T?TAT?T?T?T?T?T?
T?T?R?T?T?T?TAX?T?P?T?T?VAT?TAT?TAT?NAT?TAT?TAT?RATAF?AA\\?^Cb@?d@Cb@Ab
@?f@?f@C^B^@J?h@Af@Cn@?n@?n@?l@?d@?\\?V?b@Eh@Ar@?Z?XCt@?h@?b@B^@L?TAHA
T?T?TAT?T?T?TAT?VAXAT?P?T?T?F?T?L?J?R?P?TAT?TAT?TATAT?TAT?T?L?T?F?N?LDR
CLAT?TAT?RAH?Z?XAT?F?T?TAT?T?T?T?TAT?T?T?T?TAT?T?T?TAT?R?T?T?T?T?T?T?T?
TAT?H?{AFTAT?TAT?R?TAT?F?TAT?TAT?BAT?TAT?TAB?T?T?TAP?T?TAT?P?T?TAT?P?T?
TAR?P?TAR?TAT?TAT?TAT?H?TAT?TAT?TAT?TAD@T?TAT?T?RAT?T???TAT?T?T?TAT?T??
AT?T?T?T?T?N?`@CTAT?T?T?TAT?T?T?TAT?T?TAT?J?TAT?T?TAT?T?TAT?TAT?P?T?T?
T?T?T?T?T?H?T?R?TAPAT?P?T?P?T?L?T?L?TAL?T?J?T?F?Z?F?T?F?T?J?T?N?T?T?TAP
AT?N?TAJAT?H?T?H?T?F?F?T?T?R?TATAT?TAN?T?F?VAFAHAJELIRGTKRIRIRK@?RIRIJE
VA\\Of@Qd@Sd@Q^S\\OXGHGZO`@Mp@YXMZMZOXMXKl@[d@S\\K\\IJEVMRITIRIRITIRIRI
RKTIRIRIRITIRKNGRGTIRIRIFCRIRITKRIRIRIRKTIRIRIRKTIRIRIRKTIRIRKRIRKTIHEJ
Ef@]f@Uj@Uh@Qb@OXMZGD?f@Yf@On@YTQn@]n@Un@Uj@[l@Sj@Ul@Qb@U`@I^O^OZM@A',
 'length': 1737}, 'routeShortName': '9', 'routeLongName': 'Ashland', 'r
entedBike': False, 'duration': 3656.0, 'transitLeg': True, 'steps':
 []}, {'startTime': 1511869840000, 'endTime': 1511869877000, 'departure
Delay': 0, 'arrivalDelay': 0, 'realTime': False, 'distance': 45.249, 'p
athway': False, 'mode': 'WALK', 'route': '', 'agencyTimeZoneOffset': -2
1600000, 'interlineWithPreviousLeg': False, 'from': {'name': 'Beverly &
103rd/Vincennes', 'stopId': '1:6138', 'stopCode': '6138', 'lon': -87.65
616492, 'lat': 41.70695661, 'arrival': 1511869840000, 'departure': 1511
869840000, 'stopIndex': 87, 'stopSequence': 88, 'vertexType': 'TRANSI
T'}, 'to': {'name': '103rd Street & Vincennes Ave', 'stopId': '1:1215
2', 'stopCode': '12152', 'lon': -87.65625638, 'lat': 41.70669321, 'arri
val': 1511869877000, 'departure': 1511869945000, 'stopIndex': 30, 'stop
Sequence': 31, 'vertexType': 'TRANSIT'}, 'legGeometry': {'points': 'u
{p}Fji_vOVMPD@n@', 'length': 4}, 'rentedBike': False, 'duration': 37.0,
'transitLeg': False, 'steps': [{'distance': 14.662, 'relativeDirectio
n': 'DEPART', 'streetName': 'South Beverly Avenue', 'absoluteDirectio
n': 'SOUTHEAST', 'stayOn': False, 'area': False, 'bogusName': False, 'l
on': -87.6560523024807, 'lat': 41.70699339896925, 'elevation': []}, {'d
istance': 10.328, 'relativeDirection': 'SLIGHTLY_RIGHT', 'streetName':
 'South Vincennes Avenue', 'absoluteDirection': 'SOUTH', 'stayOn': Fals
e, 'area': False, 'bogusName': False, 'lon': -87.65598150000001, 'lat':
41.706872600000004, 'elevation': []}, {'distance': 20.259, 'relativeDir
ection': 'RIGHT', 'streetName': 'West 103rd Street', 'absoluteDirectio
n': 'WEST', 'stayOn': False, 'area': False, 'bogusName': False, 'lon':
 -87.65601430000001, 'lat': 41.706783, 'elevation': []}]}, {'startTim
e': 1511869945000, 'endTime': 1511870558000, 'departureDelay': 0, 'arri
valDelay': 0, 'realTime': False, 'distance': 3174.127660079045, 'pathwa
y': False, 'mode': 'BUS', 'route': '103', 'agencyName': 'Chicago Transi
t Authority', 'agencyUrl': 'http://transitchicago.com', 'agencyTimeZone
Offset': -21600000, 'routeType': 3, 'routeId': '1:103', 'interlineWithP
reviousLeg': False, 'tripBlockId': '484000003202', 'headsign': '95th Re
d Line', 'agencyId': '1', 'tripId': '1:484119619830', 'serviceDate': '2
0171128', 'from': {'name': '103rd Street & Vincennes Ave', 'stopId':
 '1:12152', 'stopCode': '12152', 'lon': -87.65625638, 'lat': 41.7066932
1, 'arrival': 1511869877000, 'departure': 1511869945000, 'stopIndex': 3
0, 'stopSequence': 31, 'vertexType': 'TRANSIT'}, 'to': {'name': 'Michig
an & 102nd Street', 'stopId': '1:7617', 'stopCode': '7617', 'lon': -87.
61985695, 'lat': 41.70898175, 'arrival': 1511870558000, 'departure': 15
11870558000, 'stopIndex': 46, 'stopSequence': 47, 'vertexType': 'TRANSI
T'}, 'legGeometry': {'points': '}yp}Frk_vOBy@Ae@?i@?k@?i@?W?s@Aq@?i@?q
@?o@?o@?k@?o@?o@Ak@?m@?k@?g@?IAs@Ag@?o@?o@@o@?o@?k@@a@?GCq@?c@?q@?o@?q
@?s@?o@?m@?s@?k@?OAo@?u@?i@?u@?m@?o@?o@?k@?i@?k@@IAk@?c@?k@?o@?o@?o@?k
@?k@?o@@k@?[Ae@?y@?i@?k@Ck@Ci@?k@?k@Bi@?e@Bo@?O?u@?a@Am@@o@?g@?c@?k@?i
@?k@?k@?i@?K?k@?c@?k@?s@?q@?q@?s@?o@?k@?q@?SAo@Ak@?k@?u@?m@?m@Cu@?e@?i@
Bo@?ICe@?k@?i@?k@?k@?o@?k@?o@?k@?g@?q@?O?i@?q@?q@Cs@Bo@?u@Ck@?o@?e@?e@?
O?e@?y@?g@?i@?k@?o@Ai@?k@?i@?e@?g@@OAm@?u@?k@?k@?k@?m@?m@Am@@g@?i@AICa
@?s@?g@?i@Co@?o@C{@A_A?{@?{@C{@Ao@Ak@Gi@WU_@C_@AE?]@]G_@C]E[E_@I[?
[E??', 'length': 182}, 'routeShortName': '103', 'routeLongName': 'West
 103rd', 'rentedBike': False, 'duration': 613.0, 'transitLeg': True, 's
teps': []}, {'startTime': 1511870558000, 'endTime': 1511870574000, 'dep
artureDelay': 0, 'arrivalDelay': 0, 'realTime': False, 'distance': 18.5
63, 'pathway': False, 'mode': 'WALK', 'route': '', 'agencyTimeZoneOffse
t': -21600000, 'interlineWithPreviousLeg': False, 'from': {'name': 'Mic
higan & 102nd Street', 'stopId': '1:7617', 'stopCode': '7617', 'lon': -
87.61985695, 'lat': 41.70898175, 'arrival': 1511870558000, 'departure':
1511870558000, 'stopIndex': 46, 'stopSequence': 47, 'vertexType': 'TRAN
SIT'}, 'to': {'name': 'Michigan & 102nd Street', 'stopId': '1:7535', 's
topCode': '7535', 'lon': -87.61998228, 'lat': 41.70915553, 'arrival': 1
511870574000, 'departure': 1511870663000, 'stopIndex': 8, 'stopSequenc
e': 9, 'vertexType': 'TRANSIT'}, 'legGeometry': {'points': 'chq}FrgxuOK
AUA', 'length': 3}, 'rentedBike': False, 'duration': 16.0, 'transitLe
g': False, 'steps': [{'distance': 18.563, 'relativeDirection': 'DEPAR
T', 'streetName': 'South Michigan Avenue', 'absoluteDirection': 'NORT
H', 'stayOn': False, 'area': False, 'bogusName': False, 'lon': -87.6199
3094426605, 'lat': 41.708987715182616, 'elevation': []}]}, {'startTim
e': 1511870663000, 'endTime': 1511870964000, 'departureDelay': 0, 'arri
valDelay': 0, 'realTime': False, 'distance': 2332.1770375220744, 'pathw
ay': False, 'mode': 'BUS', 'route': '106', 'agencyName': 'Chicago Trans
it Authority', 'agencyUrl': 'http://transitchicago.com', 'agencyTimeZon
eOffset': -21600000, 'routeType': 3, 'routeId': '1:106', 'interlineWith
PreviousLeg': False, 'tripBlockId': '484000003242', 'headsign': 'Stony
 Island/103rd', 'agencyId': '1', 'tripId': '1:484119620253', 'serviceDa
te': '20171128', 'from': {'name': 'Michigan & 102nd Street', 'stopId':
 '1:7535', 'stopCode': '7535', 'lon': -87.61998228, 'lat': 41.70915553,
'arrival': 1511870574000, 'departure': 1511870663000, 'stopIndex': 8,
 'stopSequence': 9, 'vertexType': 'TRANSIT'}, 'to': {'name': '103rd Str
```

```
eet & Woodlawn', 'stopId': '1:12227', 'stopCode': '12227', 'lon': -87.5
9455305, 'lat': 41.70754289, 'arrival': 1511870964000, 'departure': 151
1870965000, 'stopIndex': 22, 'stopSequence': 23, 'vertexType': 'TRANSI
T'}, 'legGeometry': {'points': 'qiq}FxgxuOf@CZBb@D^?\\B\\D`@@\\F`@@\\?Z
D\\BTOBa@?AGg@Ae@?e@@c@?q@?OAo@?k@Am@?k@?o@?OCm@?e@Ao@?i@?i@?[Ae@?y@?k@
Ak@?k@?M@o@Cm@?i@?s@@k@@k@@o@AOBq@Ac@?o@Aq@?u@?o@Ao@?g@?GAu@?m@Ay@Ay@?a
AAq@?y@?k@?c@Bu@Cu@Ae@Ak@Ak@@o@?i@?e@@o@Ag@?o@?m@Am@Ac@Ae@Ak@Ai@?k@Ai@?
k@?i@?e@Ak@?i@?g@Am@?e@?GOi@?i@Ak@?{@?y@C{@?{@?y@?k@?c@?W?y@Ce@?o@?q@?e
@?e@?M?e@?a@Ao@?u@Ay@A{@?{@A{@As@?e@?e@?a@?u@Ai@?e@?u@Ai@?i@?g@@e@B]',
 'length': 133}, 'routeShortName': '106', 'routeLongName': 'East 103r
d', 'rentedBike': False, 'duration': 301.0, 'transitLeg': True, 'step
s': []}, {'startTime': 1511870965000, 'endTime': 1511872239000, 'depart
ureDelay': 0, 'arrivalDelay': 0, 'realTime': False, 'distance': 1665.36
3, 'pathway': False, 'mode': 'WALK', 'route': '', 'agencyTimeZoneOffse
t': -21600000, 'interlineWithPreviousLeg': False, 'from': {'name': '103
rd Street & Woodlawn', 'stopId': '1:12227', 'stopCode': '12227', 'lon':
-87.59455305, 'lat': 41.70754289, 'arrival': 1511870964000, 'departur
e': 1511870965000, 'stopIndex': 22, 'stopSequence': 23, 'vertexType':
 'TRANSIT'}, 'to': {'name': 'Destination', 'lon': -87.5990999, 'lat': 4
1.6944193, 'arrival': 1511872239000, 'orig': '', 'vertexType': 'NORMA
L'}, 'legGeometry': {'points': 'u_q}F~hsuO?iA`BJtAFrAAbAA|\\[h@?^@b@N`@
P`@NXJj@TpF~BhFxB|BbA|CrA`DtAJLFXDNJf@@b@?|B?b@fAZrAj@', 'length': 28},
'rentedBike': False, 'duration': 1274.0, 'transitLeg': False, 'steps':
 [{'distance': 30.625, 'relativeDirection': 'DEPART', 'streetName': 'Ea
st 103rd Street', 'absoluteDirection': 'EAST', 'stayOn': False, 'area':
False, 'bogusName': False, 'lon': -87.59455553753253, 'lat': 41.7076307
8200987, 'elevation': []}, {'distance': 761.119, 'relativeDirection':
 'RIGHT', 'streetName': 'South Woodlawn Avenue', 'absoluteDirection':
 'SOUTH', 'stayOn': False, 'area': False, 'bogusName': False, 'lon': -8
7.59418670000001, 'lat': 41.7076366, 'elevation': []}, {'distance': 78
1.525, 'relativeDirection': 'CONTINUE', 'streetName': 'South Doty Avenu
e', 'absoluteDirection': 'SOUTH', 'stayOn': False, 'area': False, 'bogu
sName': False, 'lon': -87.5941344, 'lat': 41.700796100000005, 'elevatio
n': []}, {'distance': 92.094, 'relativeDirection': 'LEFT', 'streetNam
e': 'service road', 'absoluteDirection': 'SOUTH', 'stayOn': False, 'are
a': False, 'bogusName': True, 'lon': -87.59846160000001, 'lat': 41.6951
19600000005, 'elevation': []}]}], 'tooSloped': False}, {'duration': 834
7, 'startTime': 1511867192000, 'endTime': 1511875539000, 'walkTime': 97
6, 'transitTime': 5503, 'waitingTime': 1868, 'walkDistance': 1254.27436
08975973, 'walkLimitExceeded': False, 'elevationLost': 0.0, 'elevationG
ained': 0.0, 'transfers': 4, 'legs': [{'startTime': 1511867192000, 'end
Time': 1511867339000, 'departureDelay': 0, 'arrivalDelay': 0, 'realTim
e': False, 'distance': 173.717, 'pathway': False, 'mode': 'WALK', 'rout
e': '', 'agencyTimeZoneOffset': -21600000, 'interlineWithPreviousLeg':
 False, 'from': {'name': 'Origin', 'lon': -87.7931255, 'lat': 41.908788
9, 'departure': 1511867192000, 'orig': '', 'vertexType': 'NORMAL'}, 't
o': {'name': 'North/Natoma/Columbian', 'stopId': '3:311s0373', 'lon': -
87.79148, 'lat': 41.9091744, 'arrival': 1511867339000, 'departure': 151
1867340000, 'stopIndex': 4, 'stopSequence': 5, 'vertexType': 'TRANSI
T'}, 'legGeometry': {'points': '}hx~FrazvOW?Y@AwA?a@?W?Y?[AiBY??I', 'le
ngth': 11}, 'rentedBike': False, 'duration': 147.0, 'transitLeg': Fals
e, 'steps': [{'distance': 28.494, 'relativeDirection': 'DEPART', 'stree
tName': 'service road', 'absoluteDirection': 'NORTH', 'stayOn': False,
 'area': False, 'bogusName': False, 'lon': -87.79305276267166, 'lat': 4
1.90879023607601, 'elevation': []}, {'distance': 126.49400000000001, 'r
elativeDirection': 'RIGHT', 'streetName': 'West North Avenue', 'absolut
eDirection': 'EAST', 'stayOn': False, 'area': False, 'bogusName': Fals
```

```
e, 'lon': -87.7930618, 'lat': 41.9090464, 'elevation': []}, {'distanc
e': 13.906, 'relativeDirection': 'LEFT', 'streetName': 'path', 'absolut
eDirection': 'NORTH', 'stayOn': False, 'area': False, 'bogusName': Tru
e, 'lon': -87.7915334, 'lat': 41.909066100000004, 'elevation': []}, {'d
istance': 4.823, 'relativeDirection': 'RIGHT', 'streetName': 'sidewal
k', 'absoluteDirection': 'EAST', 'stayOn': True, 'area': False, 'bogusN
ame': True, 'lon': -87.7915387, 'lat': 41.9091911, 'elevation': []}]},
 {'startTime': 1511867340000, 'endTime': 1511867760000, 'departureDela
y': 0, 'arrivalDelay': 0, 'realTime': False, 'distance': 2752.756023090
8825, 'pathway': False, 'mode': 'BUS', 'route': '311', 'agencyName': 'P
ACE', 'agencyUrl': 'http://www.pacebus.com', 'agencyTimeZoneOffset': -2
1600000, 'routeType': 3, 'routeId': '3:311-251', 'interlineWithPrevious
Leg': False, 'tripBlockId': '1570338', 'agencyId': 'PACE', 'tripId':
 '3:7814734-WES817-wk-Weekday-01', 'serviceDate': '20171128', 'from':
 {'name': 'North/Natoma/Columbian', 'stopId': '3:311s0373', 'lon': -87.
79148, 'lat': 41.9091744, 'arrival': 1511867339000, 'departure': 151186
7340000, 'stopIndex': 4, 'stopSequence': 5, 'vertexType': 'TRANSIT'},
 'to': {'name': 'Oak Park/South Blvd./CTA Green Line', 'stopId': '3:311
s0300', 'lon': -87.7944636, 'lat': 41.8866921, 'arrival': 151186776000
0, 'departure': 1511867760000, 'stopIndex': 19, 'stopSequence': 20, 've
rtexType': 'TRANSIT'}, 'legGeometry': {'points': 'ikx~FvwyvObBzUrGQjJCl
JQjJIjJ?hJGvJKbJQpHMxHFfFQnGDtHUdCE', 'length': 16}, 'routeShortName':
 '311', 'routeLongName': 'Oak Park Avenue', 'rentedBike': False, 'durat
ion': 420.0, 'transitLeg': True, 'steps': []}, {'startTime': 1511867760
000, 'endTime': 1511867821000, 'departureDelay': 0, 'arrivalDelay': 0,
 'realTime': False, 'distance': 78.294, 'pathway': False, 'mode': 'WAL
K', 'route': '', 'agencyTimeZoneOffset': -21600000, 'interlineWithPrevi
ousLeg': False, 'from': {'name': 'Oak Park/South Blvd./CTA Green Line',
 'stopId': '3:311s0300', 'lon': -87.7944636, 'lat': 41.8866921, 'arriva
l': 1511867760000, 'departure': 1511867760000, 'stopIndex': 19, 'stopSe
quence': 20, 'vertexType': 'TRANSIT'}, 'to': {'name': 'Oak Park-Green',
 'stopId': '1:30263', 'lon': -87.793783, 'lat': 41.886988, 'arrival': 15
11867821000, 'departure': 1511868720000, 'stopIndex': 1, 'stopSequenc
e': 2, 'vertexType': 'TRANSIT'}, 'legGeometry': {'points': 'y~s~FjjzvOg
@@?E?aC', 'length': 4}, 'rentedBike': False, 'duration': 61.0, 'transit
Leg': False, 'steps': [{'distance': 22.016, 'relativeDirection': 'DEPAR
T', 'streetName': 'South Oak Park Avenue', 'absoluteDirection': 'NORT
H', 'stayOn': False, 'area': False, 'bogusName': False, 'lon': -87.7944
5466527417, 'lat': 41.88669225597201, 'elevation': []}, {'distance': 5
6.278, 'relativeDirection': 'RIGHT', 'streetName': 'South Boulevard',
 'absoluteDirection': 'EAST', 'stayOn': False, 'area': False, 'bogusNam
e': False, 'lon': -87.7944609, 'lat': 41.8868902, 'elevation': []}]},
 {'startTime': 1511868720000, 'endTime': 1511871780000, 'departureDela
y': 0, 'arrivalDelay': 0, 'realTime': False, 'distance': 25638.45122118
314, 'pathway': False, 'mode': 'SUBWAY', 'route': 'Green Line', 'agency
Name': 'Chicago Transit Authority', 'agencyUrl': 'http://transitchicag
o.com', 'agencyTimeZoneOffset': -21600000, 'routeColor': '009B3A', 'rou
teType': 1, 'routeId': '1:G', 'routeTextColor': 'FFFFFF', 'interlineWit
hPreviousLeg': False, 'tripBlockId': '54011290554', 'headsign': 'Cottag
e Grove', 'agencyId': '1', 'tripId': '1:54111524202', 'serviceDate': '2
0171128', 'from': {'name': 'Oak Park-Green', 'stopId': '1:30263', 'lo
n': -87.793783, 'lat': 41.886988, 'arrival': 1511867821000, 'departur
e': 1511868720000, 'stopIndex': 1, 'stopSequence': 2, 'vertexType': 'TR
ANSIT'}, 'to': {'name': 'King Drive', 'stopId': '1:30217', 'lon': -87.6
15546, 'lat': 41.78013, 'arrival': 1511871780000, 'departure': 15118717
80000, 'stopIndex': 26, 'stopSequence': 27, 'vertexType': 'TRANSIT'},
 'legGeometry': {'points': 's`t~FdfzvOEiGCqFEoIA_ECeCE{ICoIAiDGmFCeHAqF
```

```
AuFCmFAwFEkE?cEAcEE{FCwJAgE?cEC{B?wD?iD?eBEcFCwGAkJEoMEaC@o@B{@Dq@Do@Fu
@LgALqAJsA@aAVwIPiIJ{FRiIZaMJeENyC??RyIRmIFuCZqNZmNL}FDgBFgCJuEHuDLmFFi
C\\wNBaBDgBB_AH}CPiIDkBHmDFiCHuDb@wRBu@DeBFoCHsCBqANeHDwBJiEPwIFyCLaFFw
CDiB@GBwA@MN{GD}AJ{DPcH?CAk@?yACmACmCGqCCyDOeIG_FKoHG}DCaEEwDI{DSoLGyHQ
wKOyLMwGGcEIsII{EEkDIkHIqFEkDMyGGuGMmNGeJCkEAkEIiIIqRCcIEcFEoNAcEAq@CgE
CeFCsFCeFAkEC}FA_F?eF?kE?kQAgI?wKAwFAuE?aD?eF?iC?O?e@?U?cAAwB?oE@iE?i@?
K?I@OFODGFGFAH?b@?fLMdVQpFGjGC|FK|@BVB^NTLRJTRPNVTTLZRTHVFRBR@TA`@?vUO`
OYlIC`WMvICxGKhAF~@D~HCxECzKGf@?~dCgBjCCpRObROnMG~@EPILGNML_@H]BW@o@AkA
EqDC}KE_HBw@D]FWNWPQVMXEf@?b@AdMKzLKtIKxKIdLMtGEjNKr@KVE\\GXCx@AtJClJOj
BA|LIhMOnGEbBAvJI~II|LObADjHEzOY~DAtMIzECXENQLYBc@EsAGsH', 'length': 25
7}, 'routeLongName': 'Green Line', 'rentedBike': False, 'duration': 306
0.0, 'transitLeg': True, 'steps': []}, {'startTime': 1511871780000, 'en
dTime': 1511871797000, 'departureDelay': 0, 'arrivalDelay': 0, 'realTim
e': False, 'distance': 18.97, 'pathway': False, 'mode': 'WALK', 'rout
e': '', 'agencyTimeZoneOffset': -21600000, 'interlineWithPreviousLeg':
 False, 'from': {'name': 'King Drive', 'stopId': '1:30217', 'lon': -87.
615546, 'lat': 41.78013, 'arrival': 1511871780000, 'departure': 1511871
780000, 'stopIndex': 26, 'stopSequence': 27, 'vertexType': 'TRANSIT'},
 'to': {'name': 'King Drive & 63rd Street', 'stopId': '1:2231', 'stopCo
de': '2231', 'lon': -87.61569366, 'lat': 41.77995847, 'arrival': 151187
1797000, 'departure': 1511872027000, 'stopIndex': 68, 'stopSequence': 6
9, 'vertexType': 'TRANSIT'}, 'legGeometry': {'points': 'wd_~FrlwuO`@?',
'length': 2}, 'rentedBike': False, 'duration': 17.0, 'transitLeg': Fals
e, 'steps': [{'distance': 18.97, 'relativeDirection': 'DEPART', 'street
Name': 'South Doctor Martin Luther King Junior Drive', 'absoluteDirecti
on': 'SOUTH', 'stayOn': False, 'area': False, 'bogusName': False, 'lo
n': -87.61561409176393, 'lat': 41.78012957843273, 'elevation': []}]},
 {'startTime': 1511872027000, 'endTime': 1511872490000, 'departureDela
y': 0, 'arrivalDelay': 0, 'realTime': False, 'distance': 2611.992484411
7218, 'pathway': False, 'mode': 'BUS', 'route': '3', 'agencyName': 'Chi
cago Transit Authority', 'agencyUrl': 'http://transitchicago.com', 'age
ncyTimeZoneOffset': -21600000, 'routeType': 3, 'routeId': '1:3', 'inter
lineWithPreviousLeg': False, 'tripBlockId': '484000002493', 'headsign':
'95th/CSU', 'agencyId': '1', 'tripId': '1:484119546923', 'serviceDate':
'20171128', 'from': {'name': 'King Drive & 63rd Street', 'stopId': '1:2
231', 'stopCode': '2231', 'lon': -87.61569366, 'lat': 41.77995847, 'arr
ival': 1511871797000, 'departure': 1511872027000, 'stopIndex': 68, 'sto
pSequence': 69, 'vertexType': 'TRANSIT'}, 'to': {'name': 'King Drive &
 76th Street', 'stopId': '1:2245', 'stopCode': '2245', 'lon': -87.61517
512, 'lat': 41.75647224, 'arrival': 1511872490000, 'departure': 1511872
490000, 'stopIndex': 82, 'stopSequence': 83, 'vertexType': 'TRANSIT'},
 'legGeometry': {'points': 'wc_~F|lwuOT?J?T?T?T?T?R?T?T?T?T?T?T@T?Z?l@K
Z?^?XA^?XA^?^?h@A^?\\?\\CN?V?V?h@?`@A\\?ZA^?Z?Z?\\?`@AR@^Af@?h@Ar@?f@?n
@Ad@AN?L?PAN?P?NA\\?`@Ah@?l@Ah@An@?r@A`@?j@A`@?R?^A`@?b@A\\A\\?d@?P?TA
V?\\?f@A\\A\\A^?\\?`@Af@?\\?ZAT?\\?p@Cb@?l@?l@Ah@A`@?\\Ap@?P?X?h@?b@Ab@
@`@?d@?b@?\\?`@?XAR?`@?h@Al@?n@Cn@?f@?b@A`@?V?T?\\?f@An@A`@?f@@d@C^@b@A
f@?NC^?p@?x@At@?j@Al@?^AV?h@AL@XA^?`@Ad@?d@Ab@?\\?`@?X?ZA', 'length': 1
46}, 'routeShortName': '3', 'routeLongName': 'King Drive', 'rentedBik
e': False, 'duration': 463.0, 'transitLeg': True, 'steps': []}, {'start
Time': 1511872490000, 'endTime': 1511873102000, 'departureDelay': 0, 'a
rrivalDelay': 0, 'realTime': False, 'distance': 810.1869999999999, 'pat
hway': False, 'mode': 'WALK', 'route': '', 'agencyTimeZoneOffset': -216
00000, 'interlineWithPreviousLeg': False, 'from': {'name': 'King Drive
 & 76th Street', 'stopId': '1:2245', 'stopCode': '2245', 'lon': -87.615
17512, 'lat': 41.75647224, 'arrival': 1511872490000, 'departure': 15118
72490000, 'stopIndex': 82, 'stopSequence': 83, 'vertexType': 'TRANSI
T'}, 'to': {'name': 'Cottage Grove & 76th Street', 'stopId': '1:2579',
 'stopCode': '2579', 'lon': -87.60546971, 'lat': 41.75684024, 'arriva
l': 1511873102000, 'departure': 1511873824000, 'stopIndex': 79, 'stopSe
quence': 80, 'vertexType': 'TRANSIT'}, 'legGeometry': {'points': '}pz}F
fiwuOU??]A}AA}BAuBCwBAwBA{BAwBAwBAyBAsBA}BAyBAwBAyBAwBAeB', 'length': 1
9}, 'rentedBike': False, 'duration': 612.0, 'transitLeg': False, 'step
s': [{'distance': 12.611, 'relativeDirection': 'DEPART', 'streetName':
 'South Doctor Martin Luther King Junior Drive', 'absoluteDirection':
 'NORTH', 'stayOn': False, 'area': False, 'bogusName': False, 'lon': -8
7.61507757935362, 'lat': 41.75647339865906, 'elevation': []}, {'distanc
e': 797.5759999999997, 'relativeDirection': 'RIGHT', 'streetName': 'Eas
t 76th Street', 'absoluteDirection': 'EAST', 'stayOn': False, 'area': F
alse, 'bogusName': False, 'lon': -87.61508, 'lat': 41.7565868, 'elevati
on': []}]}, {'startTime': 1511873824000, 'endTime': 1511875175000, 'dep
artureDelay': 0, 'arrivalDelay': 0, 'realTime': False, 'distance': 720
5.6973388606875, 'pathway': False, 'mode': 'BUS', 'route': '4', 'agency
Name': 'Chicago Transit Authority', 'agencyUrl': 'http://transitchicag
o.com', 'agencyTimeZoneOffset': -21600000, 'routeType': 3, 'routeId':
 '1:4', 'interlineWithPreviousLeg': False, 'tripBlockId': '48400000277
9', 'headsign': '115th Street', 'agencyId': '1', 'tripId': '1:484119549
162', 'serviceDate': '20171128', 'from': {'name': 'Cottage Grove & 76th
Street', 'stopId': '1:2579', 'stopCode': '2579', 'lon': -87.60546971,
 'lat': 41.75684024, 'arrival': 1511873102000, 'departure': 15118738240
00, 'stopIndex': 79, 'stopSequence': 80, 'vertexType': 'TRANSIT'}, 't
o': {'name': 'Cottage Grove & 111th Street', 'stopId': '1:2837', 'stopC
ode': '2837', 'lon': -87.60992757, 'lat': 41.69289738, 'arrival': 15118
75175000, 'departure': 1511875175000, 'stopIndex': 115, 'stopSequence':
116, 'vertexType': 'TRANSIT'}, 'legGeometry': {'points': 'osz}FtluuOd@A
d@?d@En@Gr@?n@Ah@@`@?ZFB?d@C\\Cf@?j@?l@Aj@?j@?`@?b@?F?`@CZA`@?f@Cb@?f@?
`@C^A\\?^@^B^?ZBTD^E^?f@Cl@?f@?b@?X?V?d@E\\Cd@Aj@?j@?l@?f@@^?f@?L?Z?h@?
^A`@Cd@?b@?`@?^?\\?\\?XAD?T?f@?^?b@?f@Cl@?f@Cd@?\\?X?Z?b@?Z?J?\\Ah@E`@?
`@?`@?\\?d@?Z?D?\\G^?h@?l@An@?n@Ch@?^?\\?N?`@A`@?f@?j@Gj@?h@?f@?`@?f@?
\\?^Ab@?F?YAlC?lCI^L`@?h@?f@A\\?XEb@?h@?h@?\\?V?f@O`@?h@?l@?n@?l@?j@?^?
\\E@?^?`@?h@Cf@?n@?n@?l@?d@E\\?F?^Nh@?p@EZ?Z?Z?Z?n@?f@?\\?b@CZ?L?^?d@?
X?`@Ef@Ch@?^?^?J?j@K`@Cb@?h@?j@Ah@?f@?b@?\\?\\?XCZB\\?YCdFCl@DHCT?T?T?
T?T?TAN?TAT?RAT?TAP?XAJ@TARAJAVCFGJGFUBc@A[?]?]A[?]?YBIFANBR?ABzAVpEl@T
DdJpAXDnC`@bDh@X@pJvAZDnHdARFbKrAZ?vGbAl@H~HlAb@DhJnA@?d@@xGbAd@HhIfAb@
DvHjAt@R~IfA^DhH`A^DvJvA^DhIrA', 'length': 252}, 'routeShortName': '4',
'routeLongName': 'Cottage Grove', 'rentedBike': False, 'duration': 135
1.0, 'transitLeg': True, 'steps': []}, {'startTime': 1511875175000, 'en
dTime': 1511875224000, 'departureDelay': 0, 'arrivalDelay': 0, 'realTim
e': False, 'distance': 61.22200000000001, 'pathway': False, 'mode': 'WA
LK', 'route': '', 'agencyTimeZoneOffset': -21600000, 'interlineWithPrev
iousLeg': False, 'from': {'name': 'Cottage Grove & 111th Street', 'stop
Id': '1:2837', 'stopCode': '2837', 'lon': -87.60992757, 'lat': 41.69289
738, 'arrival': 1511875175000, 'departure': 1511875175000, 'stopIndex':
115, 'stopSequence': 116, 'vertexType': 'TRANSIT'}, 'to': {'name': '111
th Street & Cottage Grove', 'stopId': '1:2895', 'stopCode': '2895', 'lo
n': -87.60929577, 'lat': 41.69269946, 'arrival': 1511875224000, 'depart
ure': 1511875239000, 'stopIndex': 7, 'stopSequence': 8, 'vertexType':
 'TRANSIT'}, 'legGeometry': {'points': 'ocn}FnhvuOVD?Y?U?aA', 'length':
5}, 'rentedBike': False, 'duration': 49.0, 'transitLeg': False, 'step
s': [{'distance': 14.057, 'relativeDirection': 'DEPART', 'streetName':
 'South Cottage Grove Avenue', 'absoluteDirection': 'SOUTH', 'stayOn':
 False, 'area': False, 'bogusName': False, 'lon': -87.60983620811567,
 'lat': 41.69288556417431, 'elevation': []}, {'distance': 47.1650000000
00006, 'relativeDirection': 'LEFT', 'streetName': 'East 111th Street',
 'absoluteDirection': 'EAST', 'stayOn': False, 'area': False, 'bogusNam
```

```
e': False, 'lon': -87.60986510000001, 'lat': 41.692761000000004, 'eleva
tion': []}]}, {'startTime': 1511875239000, 'endTime': 1511875448000, 'd
epartureDelay': 0, 'arrivalDelay': 0, 'realTime': False, 'distance': 94
0.5745872643523, 'pathway': False, 'mode': 'BUS', 'route': '111A', 'age
ncyName': 'Chicago Transit Authority', 'agencyUrl': 'http://transitchic
ago.com', 'agencyTimeZoneOffset': -21600000, 'routeType': 3, 'routeId':
'1:111A', 'interlineWithPreviousLeg': False, 'tripBlockId': '4840000030
06', 'headsign': 'Pullman Park', 'agencyId': '1', 'tripId': '1:48411961
8361', 'serviceDate': '20171128', 'from': {'name': '111th Street & Cott
age Grove', 'stopId': '1:2895', 'stopCode': '2895', 'lon': -87.6092957
7, 'lat': 41.69269946, 'arrival': 1511875224000, 'departure': 151187523
9000, 'stopIndex': 7, 'stopSequence': 8, 'vertexType': 'TRANSIT'}, 't
o': {'name': 'Doty & Walmart', 'stopId': '1:17857', 'stopCode': '1785
7', 'lon': -87.59869833, 'lat': 41.69509167, 'arrival': 1511875448000,
 'departure': 1511875449000, 'stopIndex': 10, 'stopSequence': 11, 'vert
exType': 'TRANSIT'}, 'legGeometry': {'points': 'kbn}F|ivuOEe@Dg@@g@?e@?
i@A{@?k@Ay@C{@As@?q@?IGq@?e@?s@Ae@Aq@Ao@?k@?m@?e@?k@?i@?Q?eAAe@?k@?u@Ae
AAcA?kAA_@?g@?c@EeA?eA?u@?k@?o@Oe@_@Ei@Ds@B[?]?Y?[?q@?o@?i@Y_@e@Oy@?aA?
{@?}@?k@?_@', 'length': 58}, 'routeShortName': '111A', 'routeLongName':
'Pullman Shuttle', 'rentedBike': False, 'duration': 209.0, 'transitLe
g': True, 'steps': []}, {'startTime': 1511875449000, 'endTime': 1511875
539000, 'departureDelay': 0, 'arrivalDelay': 0, 'realTime': False, 'dis
tance': 111.74000000000001, 'pathway': False, 'mode': 'WALK', 'route':
 '', 'agencyTimeZoneOffset': -21600000, 'interlineWithPreviousLeg': Fal
se, 'from': {'name': 'Doty & Walmart', 'stopId': '1:17857', 'stopCode':
'17857', 'lon': -87.59869833, 'lat': 41.69509167, 'arrival': 1511875448
000, 'departure': 1511875449000, 'stopIndex': 10, 'stopSequence': 11,
 'vertexType': 'TRANSIT'}, 'to': {'name': 'Destination', 'lon': -87.599
0999, 'lat': 41.6944193, 'arrival': 1511875539000, 'orig': '', 'vertexT
ype': 'NORMAL'}, 'legGeometry': {'points': 'oqn}FzbtuO@m@fAZrAj@', 'len
gth': 4}, 'rentedBike': False, 'duration': 90.0, 'transitLeg': False,
 'steps': [{'distance': 19.646, 'relativeDirection': 'DEPART', 'streetN
ame': 'South Doty Avenue', 'absoluteDirection': 'EAST', 'stayOn': Fals
e, 'area': False, 'bogusName': False, 'lon': -87.59869821208605, 'lat':
41.69512014626801, 'elevation': []}, {'distance': 92.094, 'relativeDire
ction': 'RIGHT', 'streetName': 'service road', 'absoluteDirection': 'SO
UTH', 'stayOn': False, 'area': False, 'bogusName': True, 'lon': -87.598
46160000001, 'lat': 41.695119600000005, 'elevation': []}]}], 'tooSlope
d': False}]}, 'debugOutput': {'precalculationTime': 39, 'pathCalculatio
nTime': 2752, 'pathTimes': [1681, 1070], 'renderingTime': 5, 'totalTim
e': 2796, 'timedOut': False}, 'elevationMetadata': {'ellipsoidToGeoidDi
fference': -34.39816959422139, 'geoidElevation': False}}
```

In [21]:
```
# So again our JSON object was transformed into a Python dict.
print(type(plan))
```

```
<class 'dict'>
```

In [22]:
```
# We can look at the available keys:
print(plan.keys())
```

```
dict_keys(['requestParameters', 'plan', 'debugOutput', 'elevationMetada
ta'])
```

In [23]:

```
# And use those keys to see the dict'svalues.
# For instance, let's print out the requestParameters:
print(plan['requestParameters'])
```

```
{'fromPlace': '41.9087889,-87.7931255', 'toPlace': '41.6944193,-87.5990
999'}
```

In [24]:

```
# Dicts can contain other dicts, like in the case of the plan:
print(type(plan["plan"]))
print(plan['plan'].keys())
```

```
<class 'dict'>
dict_keys(['date', 'from', 'to', 'itineraries'])
```

In [25]:

```
## We can use a similar syntax torfer to the keys of a dict within a dic
t:
print(plan["plan"]["to"])
```

```
{'name': 'Destination', 'lon': -87.5990999, 'lat': 41.6944193, 'orig':
 '', 'vertexType': 'NORMAL'}
```

In [26]:

```
# Time is stored in a raw computer format
print('raw time value: {}'.format(plan['plan']['date']))

# But we can convert it to a datetime object so it's comprehensible.
# note OTP returns raw time value with three extra zeros, divide by 1000
 to get rid of them
print('datetime formatted: {}'.format(time.strftime('%Y-%m-%d %H:%M:%S',
 time.localtime(plan['plan']['date']/1000))))
```

```
raw time value: 1511852867000
datetime formatted: 2017-11-28 02:07:47
```

In [27]:

```
# 'itineraries' holds a lot more information, let's start with how many
 itineraries were returned
print(len(plan['plan']['itineraries']))

# and list what keys exit for the first itinerary
print(plan['plan']['itineraries'][0].keys())
```

```
2
dict_keys(['duration', 'startTime', 'endTime', 'walkTime', 'transitTim
e', 'waitingTime', 'walkDistance', 'walkLimitExceeded', 'elevationLos
t', 'elevationGained', 'transfers', 'legs', 'tooSloped'])
```

In [28]:

```
# compare the three itineraries across some pieces
for i in plan['plan']['itineraries']:
    print('duration (minutes) = {0:.2f} | transfers = {1:} | walkDist = 
{2:.2f} | \
legs = {4:} | startTime = {5:} | endTime = {6:}'\
.format(i['duration']/60., i['transfers'], i['walkDistance'], i['walkTim
e'], len(i['legs']),
        time.strftime('%H:%M:%S', time.localtime(i['startTime']/1000)),
        time.strftime('%H:%M:%S', time.localtime(i['endTime']/1000))))
```

```
duration (minutes) = 140.98 | transfers = 3 | walkDist = 1829.72 | legs
= 9 | startTime = 05:09:40 | endTime = 07:30:39
duration (minutes) = 139.12 | transfers = 4 | walkDist = 1254.27 | legs
= 11 | startTime = 06:06:32 | endTime = 08:25:39
```

In [29]:

```
# note we just counted the length of the "legs" output, it contains the
 details of the actual route
# here is what is included in a "leg"
print(plan['plan']['itineraries'][0]['legs'][0].keys())
```

```
dict_keys(['startTime', 'endTime', 'departureDelay', 'arrivalDelay', 'r
ealTime', 'distance', 'pathway', 'mode', 'route', 'agencyTimeZoneOffse
t', 'interlineWithPreviousLeg', 'from', 'to', 'legGeometry', 'rentedBik
e', 'duration', 'transitLeg', 'steps'])
```

In [30]:

```
# let's compare the three legs of the first itinerary, similarly as we c
ompared the itineraries
for leg in plan['plan']['itineraries'][0]['legs']:
    print('distance = {:,.2f} | duration = {:.0f} | mode = {} | route = 
{} | steps = {}'.\
format(leg['distance'], leg['duration'], leg['mode'], leg['route'], 
len(leg['steps'])))
```

```
distance = 78.97 | duration = 65 | mode = WALK | route =  | steps = 2
distance = 10,311.04 | duration = 1785 | mode = BUS | route = 72 | step
s = 0
distance = 21.44 | duration = 19 | mode = WALK | route =  | steps = 2
distance = 22,770.31 | duration = 3656 | mode = BUS | route = 9 | steps
= 0
distance = 45.25 | duration = 37 | mode = WALK | route =  | steps = 3
distance = 3,174.13 | duration = 613 | mode = BUS | route = 103 | steps
= 0
distance = 18.56 | duration = 16 | mode = WALK | route =  | steps = 1
distance = 2,332.18 | duration = 301 | mode = BUS | route = 106 | steps
= 0
distance = 1,665.36 | duration = 1274 | mode = WALK | route =  | steps
 = 4
```

So, if mode is 'WALK' then route is blank and steps is a list. what is included in one of those 'steps'?

In [31]:

```
print(plan['plan']['itineraries'][0]['legs'][0]['steps'][0].keys())
```

```
dict_keys(['distance', 'relativeDirection', 'streetName', 'absoluteDire
ction', 'stayOn', 'area', 'bogusName', 'lon', 'lat', 'elevation'])
```

In [32]:
```
# so, what streets does this first route call for a person to walk on?
for leg in plan['plan']['itineraries'][0]['legs']:
    print('leg sends person on following streets:')
    if leg['mode']=='WALK':
        for step in leg['steps']:
            print(step['streetName'])
    else:
        print('N/A - not a walking leg.')
```

```
leg sends person on following streets:
service road
West North Avenue
leg sends person on following streets:
N/A - not a walking leg.
leg sends person on following streets:
West North Avenue
North Ashland Avenue
leg sends person on following streets:
N/A - not a walking leg.
leg sends person on following streets:
South Beverly Avenue
South Vincennes Avenue
West 103rd Street
leg sends person on following streets:
N/A - not a walking leg.
leg sends person on following streets:
South Michigan Avenue
leg sends person on following streets:
N/A - not a walking leg.
leg sends person on following streets:
East 103rd Street
South Woodlawn Avenue
South Doty Avenue
service road
```

Isochrone API

- Back to the Table of Contents

The Isochrone (meaning same-time) tool gives the area (as a polygon) a traveler can reach from a specified point within a travel time. Like the other APIs, the Isochrone API has many other query parameters the user can set if so desired, description here (http://dev.opentripplanner.org/apidoc/1.0.0/resource_LIsochrone.html). It requires that we define a starting location, a mode of transportation, a date, and an amount of travel time.

Below, we start in downtown Chicago, allowing use of foot and public transit, on a certain date, and with 30 minutes of travel time allowed.

In [33]:
```
# set start location
start_point = [41.846698, -87.621385] # Mercy Hospital & Medical Center

travel_time = 60 * 30 # time in seconds, so this is 30 minutes
mode = "WALK,TRANSIT"

url = ("{}isochrone?fromPlace={},{}&mode={}&cutoffSec={}").format(
    base_url,start_point[0],start_point[1],mode,travel_time)
print(url)

iso_response = requests.get(url)
print(iso_response.text)
```

```
http://localhost:8080/otp/routers/default/isochrone?fromPlace=41.84669
8,-87.621385&mode=WALK,TRANSIT&cutoffSec=1800
```

```
{"type":"FeatureCollection","features":[{"type":"Feature","geometry":
{"type":"MultiPolygon","coordinates":[[[[-87.6522,41.8467],[-87.6522,4
1.8453],[-87.652,41.8449],[-87.6517,41.8439],[-87.6504,41.8431],[-87.65
03,41.8431],[-87.6479,41.8448],[-87.6462,41.8444],[-87.6455,41.8439],[-
87.645,41.8435],[-87.6444,41.8431],[-87.6451,41.8416],[-87.6431,41.84
2],[-87.6427,41.8416],[-87.6422,41.8413],[-87.6413,41.8409],[-87.6408,4
1.8395],[-87.6418,41.8387],[-87.6418,41.8377],[-87.6413,41.8372],[-87.6
407,41.8366],[-87.6401,41.8364],[-87.64,41.8359],[-87.6395,41.835],[-8
7.6395,41.8341],[-87.6392,41.8335],[-87.6383,41.8326],[-87.6379,41.832
6],[-87.6379,41.8323],[-87.6374,41.8311],[-87.637,41.8305],[-87.6375,4
1.8293],[-87.6359,41.8294],[-87.6353,41.8291],[-87.6343,41.8287],[-87.6
344,41.828],[-87.6335,41.8272],[-87.6334,41.827],[-87.6333,41.8269],[-8
7.6323,41.8259],[-87.631,41.8257],[-87.6308,41.8253],[-87.6303,41.825
1],[-87.6298,41.8242],[-87.6295,41.8233],[-87.6301,41.8222],[-87.6302,4
1.8215],[-87.6302,41.8204],[-87.6286,41.8198],[-87.6286,41.8198],[-87.6
285,41.8197],[-87.627,41.8191],[-87.6262,41.8182],[-87.6261,41.818],[-8
7.6261,41.8179],[-87.6262,41.8161],[-87.6262,41.8161],[-87.6252,41.815
1],[-87.6241,41.8143],[-87.6254,41.8131],[-87.6238,41.8142],[-87.6227,4
1.8133],[-87.6218,41.8125],[-87.6218,41.8122],[-87.6221,41.8107],[-87.6
235,41.8091],[-87.6237,41.8089],[-87.6224,41.8082],[-87.6214,41.8073],
[-87.6213,41.8072],[-87.621,41.8071],[-87.6198,41.8065],[-87.619,41.805
9],[-87.6175,41.8064],[-87.6166,41.8067],[-87.6153,41.8071],[-87.6141,4
1.8075],[-87.614,41.8072],[-87.6141,41.8071],[-87.6134,41.8059],[-87.61
34,41.8053],[-87.6138,41.8038],[-87.6136,41.8035],[-87.6139,41.8019],[-
87.6137,41.8017],[-87.6137,41.8003],[-87.6117,41.8003],[-87.6113,41.800
2],[-87.6113,41.7999],[-87.6106,41.799],[-87.61,41.7981],[-87.6098,41.7
978],[-87.6094,41.7963],[-87.6094,41.7963],[-87.6097,41.7945],[-87.609
7,41.7942],[-87.6098,41.7927],[-87.6098,41.7924],[-87.6099,41.7909],[-8
7.6096,41.7907],[-87.6093,41.7904],[-87.6081,41.7901],[-87.6079,41.789
1],[-87.6084,41.788],[-87.6086,41.7873],[-87.609,41.7858],[-87.609,41.7
855],[-87.6093,41.7853],[-87.6097,41.7853],[-87.6113,41.7855],[-87.611
5,41.7857],[-87.6117,41.7856],[-87.612,41.7855],[-87.6141,41.7852],[-8
7.6147,41.7851],[-87.6166,41.7851],[-87.6178,41.7837],[-87.6176,41.782
9],[-87.6175,41.7819],[-87.6179,41.781],[-87.6166,41.7807],[-87.6152,4
1.7812],[-87.6141,41.7813],[-87.6129,41.7819],[-87.6117,41.7826],[-87.6
108,41.7826],[-87.6093,41.7827],[-87.6083,41.7827],[-87.6069,41.7827],
[-87.6056,41.7829],[-87.6045,41.7831],[-87.6036,41.7837],[-87.603,41.78
48],[-87.6028,41.7855],[-87.6032,41.7865],[-87.6033,41.7873],[-87.6032,
41.7883],[-87.6032,41.7891],[-87.6028,41.7904],[-87.6028,41.7909],[-87.
6021,41.7927],[-87.6021,41.7927],[-87.6021,41.7927],[-87.602,41.7945],
[-87.602,41.7946],[-87.6021,41.7951],[-87.6022,41.7962],[-87.6022,41.79
63],[-87.6021,41.7981],[-87.6021,41.7981],[-87.6021,41.7982],[-87.6017,
41.7999],[-87.6006,41.801],[-87.6001,41.8017],[-87.6006,41.8028],[-87.6
016,41.8035],[-87.6013,41.8041],[-87.6001,41.8053],[-87.6005,41.8065],
[-87.6009,41.8071],[-87.6002,41.8085],[-87.6,41.8089],[-87.5997,41.809
5],[-87.5992,41.8107],[-87.5985,41.8116],[-87.5981,41.8125],[-87.5983,4
1.8135],[-87.5988,41.8143],[-87.5972,41.8157],[-87.597,41.8161],[-87.59
7,41.8163],[-87.5972,41.8179],[-87.5973,41.8179],[-87.5973,41.8179],[-8
7.5982,41.819],[-87.5983,41.8197],[-87.5984,41.8207],[-87.5986,41.821
5],[-87.5976,41.823],[-87.5985,41.8233],[-87.5972,41.8243],[-87.5971,4
1.8251],[-87.5971,41.8252],[-87.5972,41.8254],[-87.5983,41.8261],[-87.5
997,41.8268],[-87.5998,41.8268],[-87.5998,41.8269],[-87.6012,41.8276],
[-87.601,41.8287],[-87.6007,41.8298],[-87.6004,41.8305],[-87.6013,41.83
1],[-87.6021,41.8315],[-87.6031,41.8315],[-87.6032,41.8323],[-87.6034,4
1.8331],[-87.6036,41.8341],[-87.6036,41.8347],[-87.6045,41.8357],[-87.6
```

```
047,41.8358],[-87.6049,41.8359],[-87.6045,41.8377],[-87.6044,41.8377],
[-87.6039,41.8382],[-87.6025,41.8395],[-87.603,41.8406],[-87.6045,41.84
11],[-87.6048,41.8411],[-87.6054,41.8413],[-87.6066,41.8415],[-87.6069,
41.842],[-87.6074,41.8427],[-87.6093,41.8422],[-87.6095,41.8429],[-87.6
095,41.8431],[-87.6098,41.8446],[-87.6099,41.8449],[-87.6095,41.8466],
[-87.6102,41.8467],[-87.6093,41.847],[-87.608,41.8485],[-87.608,41.849
4],[-87.6081,41.8503],[-87.6083,41.851],[-87.6086,41.8521],[-87.6087,4
1.8525],[-87.6093,41.8539],[-87.6093,41.8539],[-87.6093,41.8539],[-87.6
101,41.8551],[-87.6103,41.8557],[-87.6108,41.8564],[-87.6117,41.857],[-
87.6123,41.857],[-87.6131,41.8575],[-87.6117,41.8585],[-87.6106,41.859
3],[-87.6107,41.86],[-87.6108,41.8611],[-87.6108,41.8618],[-87.6117,41.
8626],[-87.6119,41.8627],[-87.612,41.8629],[-87.612,41.8645],[-87.6119,
41.8647],[-87.6117,41.8651],[-87.6098,41.8662],[-87.6093,41.8664],[-87.
6071,41.8665],[-87.6079,41.8675],[-87.6093,41.868],[-87.6097,41.868],[-
87.6099,41.8683],[-87.6111,41.8687],[-87.6117,41.8692],[-87.6125,41.869
5],[-87.6141,41.8697],[-87.6146,41.8697],[-87.6146,41.8701],[-87.6146,4
1.8715],[-87.6146,41.8719],[-87.6146,41.8733],[-87.6147,41.8737],[-87.6
146,41.8751],[-87.6145,41.8755],[-87.6146,41.8769],[-87.6147,41.8773],
[-87.6148,41.8786],[-87.6148,41.8791],[-87.6141,41.8806],[-87.6141,41.8
809],[-87.6117,41.8825],[-87.6115,41.8827],[-87.6114,41.8829],[-87.611,
41.8845],[-87.611,41.885],[-87.6117,41.8857],[-87.6121,41.886],[-87.612
5,41.8863],[-87.6119,41.8879],[-87.6141,41.8872],[-87.6146,41.8877],[-8
7.6166,41.8881],[-87.6166,41.8881],[-87.6166,41.8881],[-87.6167,41.889
8],[-87.6167,41.8899],[-87.6179,41.8906],[-87.619,41.89],[-87.6209,41.8
903],[-87.6214,41.8902],[-87.6224,41.8909],[-87.6232,41.8917],[-87.622,
41.893],[-87.6238,41.892],[-87.6245,41.8917],[-87.6262,41.8907],[-87.62
84,41.8899],[-87.6286,41.8898],[-87.6304,41.8881],[-87.631,41.8871],[-8
7.6314,41.8863],[-87.6314,41.886],[-87.6313,41.8845],[-87.6321,41.883
7],[-87.6332,41.8827],[-87.6335,41.8824],[-87.6356,41.8809],[-87.6356,4
1.8792],[-87.6357,41.8791],[-87.6359,41.8787],[-87.6373,41.8773],[-87.6
367,41.8766],[-87.6363,41.8755],[-87.6368,41.8748],[-87.6375,41.8737],
[-87.6369,41.8729],[-87.6366,41.8719],[-87.6375,41.8707],[-87.6375,41.8
701],[-87.6383,41.8692],[-87.6395,41.8683],[-87.6395,41.8674],[-87.638
3,41.8669],[-87.638,41.8667],[-87.6381,41.8665],[-87.6369,41.8658],[-8
7.6359,41.8659],[-87.6342,41.8659],[-87.6335,41.8658],[-87.6323,41.865
6],[-87.6322,41.8647],[-87.6321,41.8639],[-87.632,41.8629],[-87.6335,4
1.8614],[-87.634,41.8611],[-87.6359,41.8595],[-87.637,41.8603],[-87.638
3,41.861],[-87.6392,41.8604],[-87.6407,41.8596],[-87.6412,41.8593],[-8
7.6431,41.8581],[-87.6436,41.8575],[-87.6435,41.8572],[-87.6431,41.856
6],[-87.6427,41.856],[-87.6418,41.8557],[-87.6417,41.8549],[-87.6426,4
1.8539],[-87.6431,41.8533],[-87.6442,41.8521],[-87.6442,41.8513],[-87.6
431,41.8513],[-87.6425,41.8508],[-87.6421,41.8503],[-87.6431,41.849],[-
87.6443,41.8494],[-87.6455,41.8495],[-87.6464,41.8496],[-87.6479,41.849
7],[-87.6489,41.8485],[-87.6504,41.848],[-87.6522,41.8467]]],[[[-87.621
4,41.8002],[-87.6212,41.8001],[-87.619,41.8002],[-87.6183,41.8004],[-8
7.6166,41.8007],[-87.6146,41.8017],[-87.6149,41.8029],[-87.6153,41.803
5],[-87.6159,41.804],[-87.6166,41.8044],[-87.6175,41.8047],[-87.619,41.
805],[-87.6204,41.8043],[-87.6214,41.8038],[-87.6218,41.8035],[-87.623
6,41.8018],[-87.6238,41.8017],[-87.6234,41.8002],[-87.6214,41.8002]]],
[[[-87.6315,41.8086],[-87.631,41.8084],[-87.6294,41.8083],[-87.6286,41.
8073],[-87.628,41.8089],[-87.6269,41.8102],[-87.627,41.8107],[-87.627,4
1.8119],[-87.6286,41.8119],[-87.6297,41.8117],[-87.631,41.8116],[-87.63
12,41.8107],[-87.6316,41.8103],[-87.6325,41.8089],[-87.6315,41.8086]]],
[[[-87.6683,41.8387],[-87.6673,41.8381],[-87.6669,41.838],[-87.6662,41.
8377],[-87.6652,41.8374],[-87.6648,41.8373],[-87.6644,41.8377],[-87.663
6,41.8387],[-87.6628,41.8395],[-87.6625,41.8413],[-87.6648,41.8405],[-8
7.6662,41.8403],[-87.6673,41.8406],[-87.6682,41.8395],[-87.6683,41.838
```

```
7]]],[[[-87.66,41.8395],[-87.66,41.8395],[-87.6576,41.8409],[-87.6568,4
1.8413],[-87.6566,41.842],[-87.6576,41.842],[-87.6594,41.8417],[-87.66,
41.8415],[-87.6623,41.8413],[-87.6609,41.8406],[-87.66,41.8395],[-87.6
6,41.8395],[-87.66,41.8395]]],[[[-87.6195,41.7941],[-87.619,41.7939],[-
87.6172,41.7945],[-87.6173,41.7958],[-87.617,41.7963],[-87.6181,41.79
7],[-87.619,41.7971],[-87.6206,41.7963],[-87.6206,41.7951],[-87.6202,4
1.7945],[-87.6195,41.7941]]],[[[-87.6833,41.8276],[-87.6817,41.8282],[-
87.6801,41.8287],[-87.6803,41.8298],[-87.6817,41.8297],[-87.6833,41.828
7],[-87.6833,41.8276]]],[[[-87.6132,41.779],[-87.6117,41.7796],[-87.611
4,41.7801],[-87.6111,41.7806],[-87.6117,41.7807],[-87.6139,41.7801],[-8
7.6132,41.779]]]]},"properties":{"time":1800},"id":"fid-120ec34a_160017
03008_-7ffe"}]}
```

In [34]:

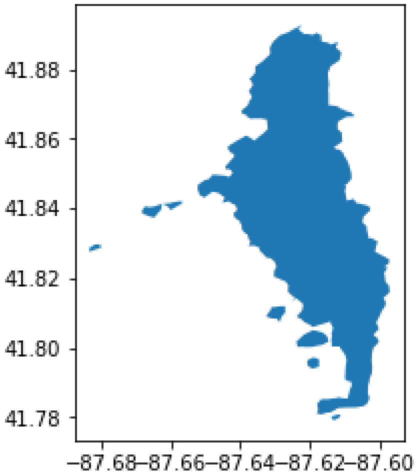
```python
iso_json = json.loads(iso_response.text)

## load isochrone into a geopandas dataframe
iso_gdf = gpd.GeoDataFrame.from_features(iso_json['features'])
iso_gdf[:]
```

Out[34]:

	geometry	time
0	(POLYGON ((-87.65219999999999 41.8467, -87.652...	1800

In [35]:

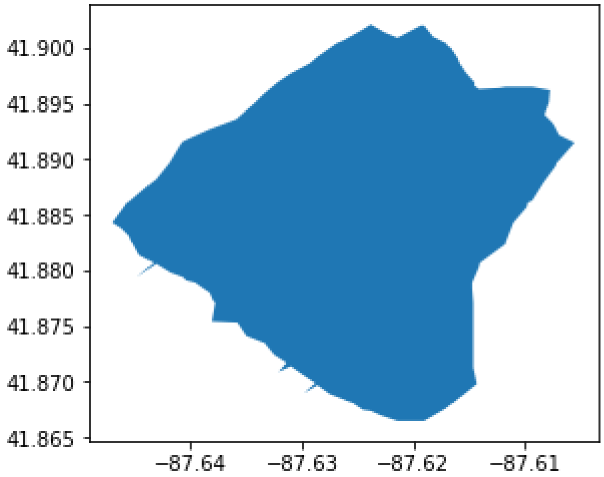
```python
# view the resulting isochrone shape (can you guess why there are separa
ted geographies?)
iso_gdf.plot();
```



One potential use case for this functionality: can people at two locations reach some common location within a specified travel time?

In [36]:
```
# 2nd location
start_point_2 = [41.884260, -87.630344] # Traffic Court in Richard J. Da
ley center

url_2 = ("{}isochrone?fromPlace={},{}&mode={}&date=2016-06-01&cutoffSec=
{}").format(
    base_url,start_point_2[0],start_point[1],mode,travel_time)

# get json request
iso_json_2 = json.loads(requests.get(url_2).text)

## load isochrone into a geopandas dataframe
iso_gdf_2 = gpd.GeoDataFrame.from_features(iso_json_2['features'])
```

In [37]:
```
# view the second isochrone
iso_gdf_2.plot();
```



In [38]:
```
# do the two isochrones intersect?
iso_gdf.intersects(iso_gdf_2)
```

Out[38]:
```
0    True
dtype: bool
```

In [39]:
```
# they do intersect, so create an overlay with a 'union'
iso_join = gpd.overlay(iso_gdf, iso_gdf_2, how='union')
```

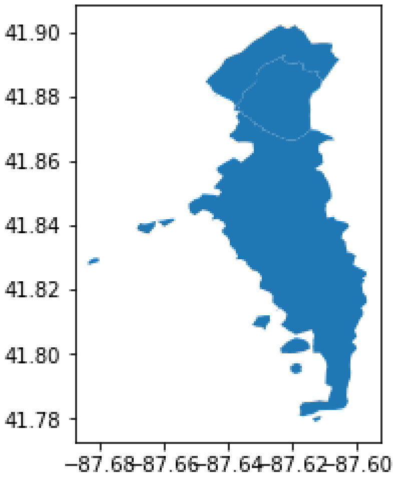
In [40]:
```
# what does the dataframe look like now?
iso_join.head()
```

Out[40]:

	time	time_2	geometry
0	1800.0	NaN	POLYGON ((-87.6833 41.8276, -87.6833 41.8287, ...
1	1800.0	NaN	POLYGON ((-87.6683 41.8387, -87.6682 41.8395, ...
2	NaN	1800.0	POLYGON ((-87.61442352941177 41.8697, -87.6146...
3	1800.0	1800.0	POLYGON ((-87.6146 41.8713, -87.6146 41.8701, ...
4	NaN	1800.0	POLYGON ((-87.6146 41.8733, -87.6147 41.8737, ...

In [41]:

```
# and visually?
iso_join.plot();
```



- Back to the Table of Contents

> A bit annoyingly this is difficult to tell where the two overlap. To fix this we can group based on the "time" and "time_2" columns to end with just 3 combinations:
>
> 1. accessible from our first location only,
> 2. accessible our second location only, and
> 3. accessible from either location

We'll do this by using the "dissolve" function (http://geopandas.org/aggregation_with_dissolve.html) from geopandas. However first we need to replace the "NaN" so those rows are not ignored

In [42]:

```
# replace NaN with placeholder value, let's say 99999
iso_join.fillna(99999, inplace=True)
```

In [43]:

```
iso_join = iso_join.dissolve(by=['time', 'time_2']).reset_index()

# Note: used reset_index() here so it's easier to use the 'time' and 'ti
me_2' columns if needed
```

In [44]:

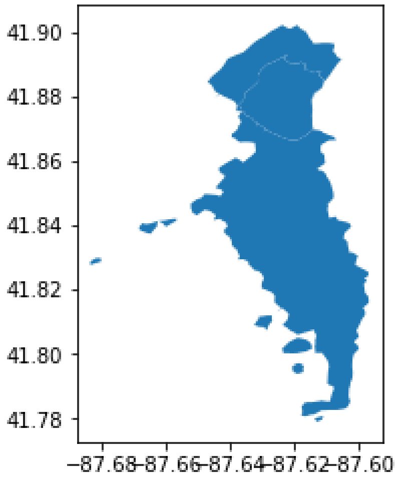
```
# now what does it look like?
iso_join
```

Out[44]:

	time	time_2	geometry
0	1800.0	1800.0	POLYGON ((-87.6146 41.8713, -87.6146 41.8701, ...
1	1800.0	99999.0	(POLYGON ((-87.61320000000001 41.779, -87.6139...
2	99999.0	1800.0	(POLYGON ((-87.61442352941177 41.8697, -87.614...

In [45]:

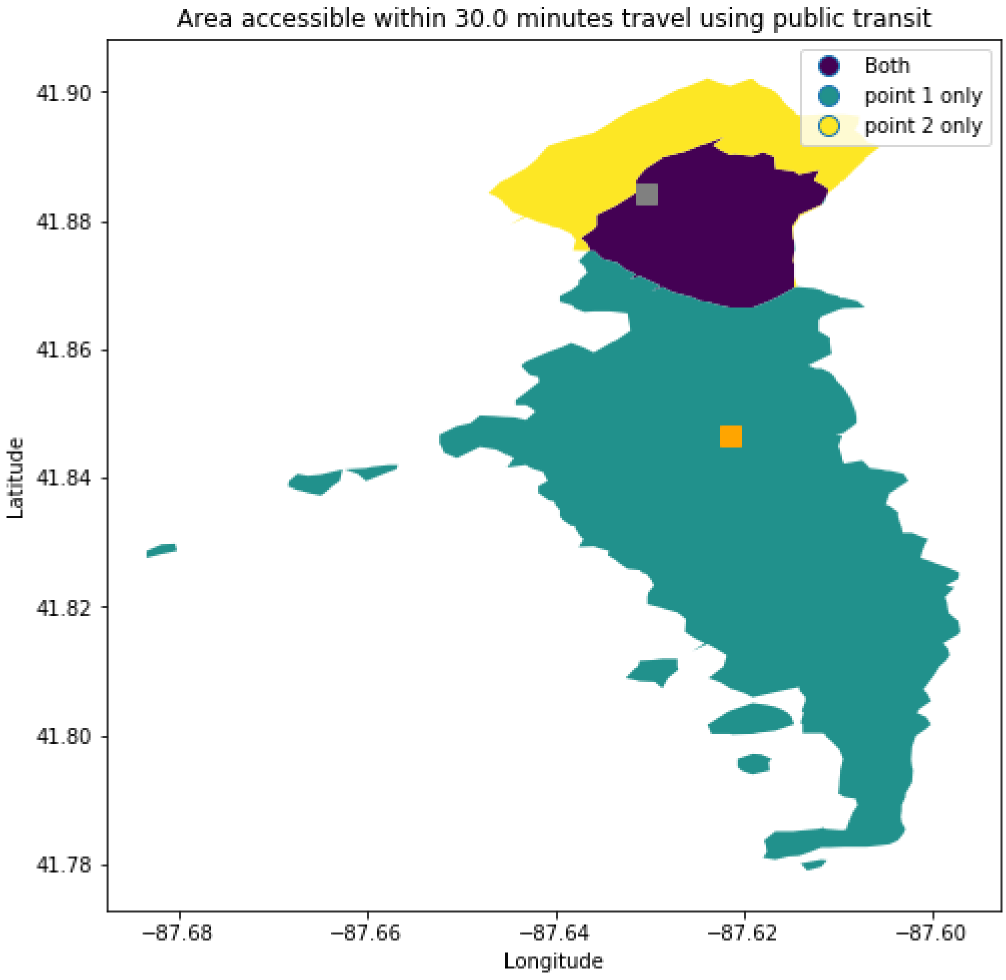
```
# and visually?
iso_join.plot();
```



In [46]:

```
# add a label column to use so we can include a legend
iso_join['label'] = ''

# use index slicing function '.loc' of dataframes to update each value o
f label appropriately
iso_join.loc[0,'label'] = 'Both'
iso_join.loc[1,'label'] = 'point 1 only'
iso_join.loc[2,'label'] = 'point 2 only'
```

```
In [47]: # set up a nicer visualization with labels
         f,ax = plt.subplots(figsize=(8,8))

         # use geopandas to specify label column and adding a legend to the matpl
         otlib object 'ax'
         iso_join.plot(column='label', ax=ax, legend=True, cmap='viridis');

         # also plot start and stop points on the same map, note matplotlib takes
          [x,y] coordinates
         ax.plot(start_point[1], start_point[0], 's', color='orange',
         markersize=10, label='point 1')
         ax.plot(start_point_2[1], start_point_2[0], 's', color='grey', markersiz
         e=10, label='point 2')

         # add some other labels
         ax.set_xlabel('Longitude')
         ax.set_ylabel('Latitude')

         # title
         ax.set_title('Area accessible within {} minutes travel using public tran
         sit'.format(travel_time/60));
```


Area accessible within 30.0 minutes travel using public transit
Both
point 1 only
point 2 only
41.90
41.88
41.86
41.84
41.82
41.80
41.78
Latitude
−87.68
−87.66
−87.64
−87.62
−87.60
Longitude