BATCH GEOCODING

Written By: MATINGO KUMBIRAI N

In this tutorial, we are going to be using the Python API to geocode a batch of addresses within Zimbabwe. I have also provided a link to the datasets which you can use just in case you do not have datasets for this tutorial.

Hint: Make sure you have gone through the previous 2 tutorials in order to get through this one. We won't be explaining some of the keywords and functions which we have already discussed in the other previous tutorials

Geocoding - the process of taking a text-based description of a location, such as an address or the name of a place, and returning geographic coordinates, frequently latitude/longitude pair, to identify a location on the Earth's surface. Source

In this tutorial, we have a **CSV** file which has names of places in Zimbabwe and these are names of the different Districts that are within the country. Using this CSV file, we are going to be assigning coordinates to each and every name in each row that exists within the file and also plot these on a map.

We are also going to be introducing you to basic map pop-ups.

Just a summary of this tutorial:

- geocoding
- adding popups to maps
- data alteration
- writing new data to a new file

Let's get started!

Import Library

Call the ArcGIS Python API that you installed within your notebook.

This time we have added a new library (`batch_geocode`). This is the Python GIS library which is going to help us during the geocoding process which we are going to be carrying out in this tutorial.

Import the library as shown below

Please Note: This function uses credits. Refer to the Understanding Credits Documentation provided on the ESRI platform to understand more about how credits work and how you can use them and be able to preserve them

In [1]:

```
import csv
import pandas as pd
from arcgis.gis import GIS
from arcgis.geocoding import batch_geocode # the geocoding library based on the ESRI Geocoding ser
vice
from getpass import getpass
```

Login to You ArcGIS Account

- replace *africansurveyors* with your ArcGIS Online Account `username`
- enter your account password in the prompt input box that appears below

In [2]:

```
gis = GIS("http://arcgis.com", "africansurveyors")
```

```
Enter password: ········
```

Retrieve Help on a function

Here, we just want to see what the `batch_geocode` function can do for us and how we can use it.

After you see the help information, just remove the help bar and continue

In [3]:

```
batch_geocode?
```

Define variables for file resources

We have define two variables, `input_file` which contains the intial data that we want to geocode in this tutorial and `output_file` which is going to be the output file with the Lat & Lon fields of this process.

In [4]:

```
# the source file
input_file = r"C:\Users\Surveyor Jr\Desktop\African Surveyors
Academy\tutorials_data\editable_zwe_data\geocoding_data_set.csv"

# This file does not exist yet.
output_file = csv1 = r"C:\Users\Surveyor Jr\Desktop\African Surveyors
Academy\tutorials_data\editable_zwe_data\final_geocoded_dataset.csv"
```

Specify column for addresses

Within our CSV file, the **District** column contains the address which we want to geocode. Specify the column name as per your dataset.

In [5]:

```
address_column_name = "District"
```

Read the data

- call the **pandas** function to read the CSV file and display this data so that we are certain its exactly what we want to work with

At times calling this function to view data is not neccessary, but I just like to be sure most of the time and avoid wasting precious time.

In [6]:

```
data = pd.read_csv(input_file, encoding='utf8')
data
```

Out[6]:

	ID_0	ISO	NAME_0	ID_1	Province	ID_2	District	TYPE_2	ENGTYPE_2	NL_NAME_2	VARNAME_2
0	256	ZWE	Zimbabwe	1	Bulawayo	1	Bulawayo	District	District	NaN	NaN
1	256	ZWE	Zimbabwe	2	Harare	2	Harare	District	District	NaN	Salisbury|Harare Urban
2	256	ZWE	Zimbabwe	3	Manicaland	3	Buhera	District	District	NaN	NaN
3	256	ZWE	Zimbabwe	3	Manicaland	4	Chimanimani	District	District	NaN	NaN
4	256	ZWE	Zimbabwe	3	Manicaland	5	Chipinge	District	District	NaN	NaN
5	256	ZWE	Zimbabwe	3	Manicaland	6	Makoni	District	District	NaN	NaN
6	256	ZWE	Zimbabwe	3	Manicaland	7	Mutare	District	District	NaN	Umtali
7	256	ZWE	Zimbabwe	3	Manicaland	8	Mutasa	District	District	NaN	NaN
8	256	ZWE	Zimbabwe	3	Manicaland	9	Nyanga	District	District	NaN	NaN
9	256	ZWE	Zimbabwe	4	Mashonaland Central	10	Bindura	District	District	NaN	NaN
10	256	ZWE	Zimbabwe	4	Mashonaland Central	11	Centenary	District	District	NaN	Muzarambani
11	256	ZWE	Zimbabwe	4	Mashonaland Central	12	Guruve	District	District	NaN	Sipolilo
12	256	ZWE	Zimbabwe	4	Mashonaland	13	[illegible]	District	District	NaN	NaN

	ID_0	ISO	NAME_0	ID_1	Province	ID_2	District	TYPE_2	ENGTYPE_2	NL_NAME_2	VARNAME_2
12	256	ZWE	Zimbabwe	4	Central	13	Mazowe	District	District	NaN	NaN
13	256	ZWE	Zimbabwe	4	Mashonaland Central	14	Mount Darwin	District	District	NaN	Mt Darwin
14	256	ZWE	Zimbabwe	4	Mashonaland Central	15	Rushinga	District	District	NaN	NaN
15	256	ZWE	Zimbabwe	4	Mashonaland Central	16	Shamva	District	District	NaN	NaN
16	256	ZWE	Zimbabwe	5	Mashonaland East	17	Chikomba	District	District	NaN	NaN
17	256	ZWE	Zimbabwe	5	Mashonaland East	18	Goromonzi	District	District	NaN	NaN
18	256	ZWE	Zimbabwe	5	Mashonaland East	19	Marondera	District	District	NaN	Marandellas
19	256	ZWE	Zimbabwe	5	Mashonaland East	20	Mudzi	District	District	NaN	NaN
20	256	ZWE	Zimbabwe	5	Mashonaland East	21	Murehwa	District	District	NaN	NaN
21	256	ZWE	Zimbabwe	5	Mashonaland East	22	Mutoko	District	District	NaN	NaN
22	256	ZWE	Zimbabwe	5	Mashonaland East	23	Seke	District	District	NaN	NaN
23	256	ZWE	Zimbabwe	5	Mashonaland East	24	UMP	District	District	NaN	Murehwa U.M.P.
24	256	ZWE	Zimbabwe	5	Mashonaland East	25	Wedza	District	District	NaN	Hwedza
25	256	ZWE	Zimbabwe	6	Mashonaland West	26	Chegutu	District	District	NaN	NaN
26	256	ZWE	Zimbabwe	6	Mashonaland West	27	Hurungwe	District	District	NaN	Karoi|Urungwe
27	256	ZWE	Zimbabwe	6	Mashonaland West	28	Kadoma	District	District	NaN	Gatooma
28	256	ZWE	Zimbabwe	6	Mashonaland West	29	Kariba	District	District	NaN	NaN
29	256	ZWE	Zimbabwe	6	Mashonaland West	30	Makonde	District	District	NaN	Lomagundi
30	256	ZWE	Zimbabwe	6	Mashonaland West	31	Zvimba	District	District	NaN	NaN
31	256	ZWE	Zimbabwe	7	Masvingo	32	Bikita	District	District	NaN	NaN
32	256	ZWE	Zimbabwe	7	Masvingo	33	Chiredzi	District	District	NaN	Hartley
33	256	ZWE	Zimbabwe	7	Masvingo	34	Chivi	District	District	NaN	NaN
34	256	ZWE	Zimbabwe	7	Masvingo	35	Gutu	District	District	NaN	NaN
35	256	ZWE	Zimbabwe	7	Masvingo	36	Masvingo	District	District	NaN	Victoria
36	256	ZWE	Zimbabwe	7	Masvingo	37	Mwenezi	District	District	NaN	NaN
37	256	ZWE	Zimbabwe	7	Masvingo	38	Zaka	District	District	NaN	NaN
38	256	ZWE	Zimbabwe	8	Matabeleland North	39	Binga	District	District	NaN	NaN
39	256	ZWE	Zimbabwe	8	Matabeleland North	40	Bubi	District	District	NaN	NaN
40	256	ZWE	Zimbabwe	8	Matabeleland North	41	Hwange	District	District	NaN	Wankie
41	256	ZWE	Zimbabwe	8	Matabeleland North	42	Lupane	District	District	NaN	NaN
42	256	ZWE	Zimbabwe	8	Matabeleland North	43	Nkayi	District	District	NaN	NaN
43	256	ZWE	Zimbabwe	8	Matabeleland North	44	Tsholotsho	District	District	NaN	NaN
44	256	ZWE	Zimbabwe	8	Matabeleland North	45	Umguza	District	District	NaN	NaN
45	256	ZWE	Zimbabwe	9	Matabeleland South	46	Beitbridge	District	District	NaN	NaN
46	256	ZWE	Zimbabwe	9	Matabeleland South	47	Bulilima (North)	District	District	NaN	Bulalimamangwe|Bulilima
47	256	ZWE	Zimbabwe	9	Matabeleland South	48	Gwanda	District	District	NaN	NaN
48	256	ZWE	Zimbabwe	9	Matabeleland South	49	Insiza	District	District	NaN	NaN
49	256	ZWE	Zimbabwe	9	Matabeleland South	50	Mangwe (South)	District	District	NaN	Bulalimamangwe South
50	256	ZWE	Zimbabwe	9	Matabeleland South	51	Matobo	District	District	NaN	NaN
51	256	ZWE	Zimbabwe	9	Matabeleland South	52	Umzingwane	District	District	NaN	NaN
52	256	ZWE	Zimbabwe	10	Midlands	53	Chirumhanzu	District	District	NaN	Chirumanzu
53	256	ZWE	Zimbabwe	10	Midlands	54	Gokwe North	District	District	NaN	Gokwe
54	256	ZWE	Zimbabwe	10	Midlands	55	Gokwe South	District	District	NaN	Gokwe
55	256	ZWE	Zimbabwe	10	Midlands	56	Gweru	District	District	NaN	Gwelo
56	256	ZWE	Zimbabwe	10	Midlands	57	Kwekwe	District	District	NaN	Que Que
57	256	ZWE	Zimbabwe	10	Midlands	58	Mberengwa	District	District	NaN	Mberingwe
58	256	ZWE	Zimbabwe	10	Midlands	59	Shurugwi	District	District	NaN	Selukwe
59	256	ZWE	Zimbabwe	10	Midlands	60	Zvishavane	District	District	NaN	Shabani, Shavani

Fill in Missing the Data

In the event you have missing data within your column, you can always find a default value to place in that same column. In my scenario I didn't have any missing data but the code below is for illustration purposes.

Explanation: If I had a missing value within my CSV file, that space would have been filled with the word "Zimbabwe" since I have placed it as the parameter in the `fillna()` function.

In [7]:

```
data['District'] = data['District'].fillna('Zimbabwe')
data
```

Out[7]:

	ID_0	ISO	NAME_0	ID_1	Province	ID_2	District	TYPE_2	ENGTYPE_2	NL_NAME_2	VARNAME_2
0	256	ZWE	Zimbabwe	1	Bulawayo	1	Bulawayo	District	District	NaN	NaN
1	256	ZWE	Zimbabwe	2	Harare	2	Harare	District	District	NaN	Salisbury|Harare Urban
2	256	ZWE	Zimbabwe	3	Manicaland	3	Buhera	District	District	NaN	NaN
3	256	ZWE	Zimbabwe	3	Manicaland	4	Chimanimani	District	District	NaN	NaN
4	256	ZWE	Zimbabwe	3	Manicaland	5	Chipinge	District	District	NaN	NaN
5	256	ZWE	Zimbabwe	3	Manicaland	6	Makoni	District	District	NaN	NaN
6	256	ZWE	Zimbabwe	3	Manicaland	7	Mutare	District	District	NaN	Umtali
7	256	ZWE	Zimbabwe	3	Manicaland	8	Mutasa	District	District	NaN	NaN
8	256	ZWE	Zimbabwe	3	Manicaland	9	Nyanga	District	District	NaN	NaN
9	256	ZWE	Zimbabwe	4	Mashonaland Central	10	Bindura	District	District	NaN	NaN
10	256	ZWE	Zimbabwe	4	Mashonaland Central	11	Centenary	District	District	NaN	Muzarambani
11	256	ZWE	Zimbabwe	4	Mashonaland Central	12	Guruve	District	District	NaN	Sipolilo
12	256	ZWE	Zimbabwe	4	Mashonaland Central	13	Mazowe	District	District	NaN	NaN
13	256	ZWE	Zimbabwe	4	Mashonaland Central	14	Mount Darwin	District	District	NaN	Mt Darwin
14	256	ZWE	Zimbabwe	4	Mashonaland Central	15	Rushinga	District	District	NaN	NaN
15	256	ZWE	Zimbabwe	4	Mashonaland Central	16	Shamva	District	District	NaN	NaN
16	256	ZWE	Zimbabwe	5	Mashonaland East	17	Chikomba	District	District	NaN	NaN
17	256	ZWE	Zimbabwe	5	Mashonaland East	18	Goromonzi	District	District	NaN	NaN
18	256	ZWE	Zimbabwe	5	Mashonaland East	19	Marondera	District	District	NaN	Marandellas
19	256	ZWE	Zimbabwe	5	Mashonaland East	20	Mudzi	District	District	NaN	NaN
20	256	ZWE	Zimbabwe	5	Mashonaland East	21	Murehwa	District	District	NaN	NaN
21	256	ZWE	Zimbabwe	5	Mashonaland East	22	Mutoko	District	District	NaN	NaN
22	256	ZWE	Zimbabwe	5	Mashonaland East	23	Seke	District	District	NaN	NaN
23	256	ZWE	Zimbabwe	5	Mashonaland East	24	UMP	District	District	NaN	Murehwa U.M.P.
24	256	ZWE	Zimbabwe	5	Mashonaland East	25	Wedza	District	District	NaN	Hwedza
25	256	ZWE	Zimbabwe	6	Mashonaland West	26	Chegutu	District	District	NaN	NaN
26	256	ZWE	Zimbabwe	6	Mashonaland West	27	Hurungwe	District	District	NaN	Karoi|Urungwe
27	256	ZWE	Zimbabwe	6	Mashonaland West	28	Kadoma	District	District	NaN	Gatooma
28	256	ZWE	Zimbabwe	6	Mashonaland West	29	Kariba	District	District	NaN	NaN
29	256	ZWE	Zimbabwe	6	Mashonaland West	30	Makonde	District	District	NaN	Lomagundi
30	256	ZWE	Zimbabwe	6	Mashonaland West	31	Zvimba	District	District	NaN	NaN
31	256	ZWE	Zimbabwe	7	Masvingo	32	Bikita	District	District	NaN	NaN
32	256	ZWE	Zimbabwe	7	Masvingo	33	Chiredzi	District	District	NaN	Hartley

	ID_0	ISO	NAME_0	ID_1	Province	ID_2	District	TYPE_2	ENGTYPE_2	NL_NAME_2	VARNAME_2
33	256	ZWE	Zimbabwe	7	Masvingo	34	Chivi	District	District	NaN	NaN
34	256	ZWE	Zimbabwe	7	Masvingo	35	Gutu	District	District	NaN	NaN
35	256	ZWE	Zimbabwe	7	Masvingo	36	Masvingo	District	District	NaN	Victoria
36	256	ZWE	Zimbabwe	7	Masvingo	37	Mwenezi	District	District	NaN	NaN
37	256	ZWE	Zimbabwe	7	Masvingo	38	Zaka	District	District	NaN	NaN
38	256	ZWE	Zimbabwe	8	Matabeleland North	39	Binga	District	District	NaN	NaN
39	256	ZWE	Zimbabwe	8	Matabeleland North	40	Bubi	District	District	NaN	NaN
40	256	ZWE	Zimbabwe	8	Matabeleland North	41	Hwange	District	District	NaN	Wankie
41	256	ZWE	Zimbabwe	8	Matabeleland North	42	Lupane	District	District	NaN	NaN
42	256	ZWE	Zimbabwe	8	Matabeleland North	43	Nkayi	District	District	NaN	NaN
43	256	ZWE	Zimbabwe	8	Matabeleland North	44	Tsholotsho	District	District	NaN	NaN
44	256	ZWE	Zimbabwe	8	Matabeleland North	45	Umguza	District	District	NaN	NaN
45	256	ZWE	Zimbabwe	9	Matabeleland South	46	Beitbridge	District	District	NaN	NaN
46	256	ZWE	Zimbabwe	9	Matabeleland South	47	Bulilima (North)	District	District	NaN	Bulalimamangwe|Bulilima
47	256	ZWE	Zimbabwe	9	Matabeleland South	48	Gwanda	District	District	NaN	NaN
48	256	ZWE	Zimbabwe	9	Matabeleland South	49	Insiza	District	District	NaN	NaN
49	256	ZWE	Zimbabwe	9	Matabeleland South	50	Mangwe (South)	District	District	NaN	Bulalimamangwe South
50	256	ZWE	Zimbabwe	9	Matabeleland South	51	Matobo	District	District	NaN	NaN
51	256	ZWE	Zimbabwe	9	Matabeleland South	52	Umzingwane	District	District	NaN	NaN
52	256	ZWE	Zimbabwe	10	Midlands	53	Chirumhanzu	District	District	NaN	Chirumanzu
53	256	ZWE	Zimbabwe	10	Midlands	54	Gokwe North	District	District	NaN	Gokwe
54	256	ZWE	Zimbabwe	10	Midlands	55	Gokwe South	District	District	NaN	Gokwe
55	256	ZWE	Zimbabwe	10	Midlands	56	Gweru	District	District	NaN	Gwelo
56	256	ZWE	Zimbabwe	10	Midlands	57	Kwekwe	District	District	NaN	Que Que
57	256	ZWE	Zimbabwe	10	Midlands	58	Mberengwa	District	District	NaN	Mberingwe
58	256	ZWE	Zimbabwe	10	Midlands	59	Shurugwi	District	District	NaN	Selukwe
59	256	ZWE	Zimbabwe	10	Midlands	60	Zvishavane	District	District	NaN	Shabani, Shavani

Verify Data

We just want to check if there is any column in our dataset that contains an empty field among the fields that we have selected as our Geocoding fields.

If there is a column with an empty field, the `ValueError` error will be raised as we have stated in the condition below:

In [8]:

```
if address_column_name not in data.columns:
    raise ValueError("Missing Address column in input data")
```

Format addresses & write as list

We want to make life easier for the ESRI Geocoding Service. So we want to make sure all our addresses are in the format; *District, Province* So as to make sure the Geocoding Service knows that the district that has been selected, is in the particular Province which also improves the speed of the function when it is executed.

for example: Mazowe District is in Mashonaland Central. So the geocoding service will read this as:

Look for Mazowe in Mashonaland Central

The `len()` Function

The `len()` function returns the number of items in an object. When the object is a string, the len() function returns the number of characters in the string. In our case, it returns the number of records contain in the variable we just passed to it as the parameter.

Below, we placed *addresses* as the parameter, hence it is going to return the number of records that are stored within that variable.

In [9]:

```
addresses = (data[address_column_name] + "," + data['Province']).tolist()
len(addresses) # display length
addresses # display the data
```

Out[9]:

```
['Bulawayo,Bulawayo',
 'Harare,Harare',
 'Buhera,Manicaland',
 'Chimanimani,Manicaland',
 'Chipinge,Manicaland',
 'Makoni,Manicaland',
 'Mutare,Manicaland',
 'Mutasa,Manicaland',
 'Nyanga,Manicaland',
 'Bindura,Mashonaland Central',
 'Centenary,Mashonaland Central',
 'Guruve,Mashonaland Central',
 'Mazowe,Mashonaland Central',
 'Mount Darwin,Mashonaland Central',
 'Rushinga,Mashonaland Central',
 'Shamva,Mashonaland Central',
 'Chikomba,Mashonaland East',
 'Goromonzi,Mashonaland East',
 'Marondera,Mashonaland East',
 'Mudzi,Mashonaland East',
 'Murehwa,Mashonaland East',
 'Mutoko,Mashonaland East',
 'Seke,Mashonaland East',
 'UMP,Mashonaland East',
 'Wedza,Mashonaland East',
 'Chegutu,Mashonaland West',
 'Hurungwe,Mashonaland West',
 'Kadoma,Mashonaland West',
 'Kariba,Mashonaland West',
 'Makonde,Mashonaland West',
 'Zvimba,Mashonaland West',
 'Bikita,Masvingo',
 'Chiredzi,Masvingo',
 'Chivi,Masvingo',
 'Gutu,Masvingo',
 'Masvingo,Masvingo',
 'Mwenezi,Masvingo',
 'Zaka,Masvingo',
 'Binga,Matabeleland North',
 'Bubi,Matabeleland North',
 'Hwange,Matabeleland North',
 'Lupane,Matabeleland North',
 'Nkayi,Matabeleland North',
 'Tsholotsho,Matabeleland North',
 'Umguza,Matabeleland North',
 'Beitbridge,Matabeleland South',
 'Bulilima (North),Matabeleland South',
 'Gwanda,Matabeleland South',
 'Insiza,Matabeleland South',
 'Mangwe (South),Matabeleland South',
 'Matobo,Matabeleland South',
 'Umzingwane,Matabeleland South',
 'Chirumhanzu,Midlands',
 'Gokwe North,Midlands',
 'Gokwe South,Midlands',
 'Gweru,Midlands',
 'Kwekwe,Midlands',
 'Mberengwa,Midlands',
 'Shurugwi,Midlands',
 'Zvishavane,Midlands']
```

Geocoding The List of Addresses

Let's geocode the list of addresses as we have specified above.

All we need to do, is simply call the `batch_geocode()` function and pass the variable which contains the data which is going to be geocoded.

In [10]:

```
results = batch_geocode(addresses)
len(results)
```

Out[10]:

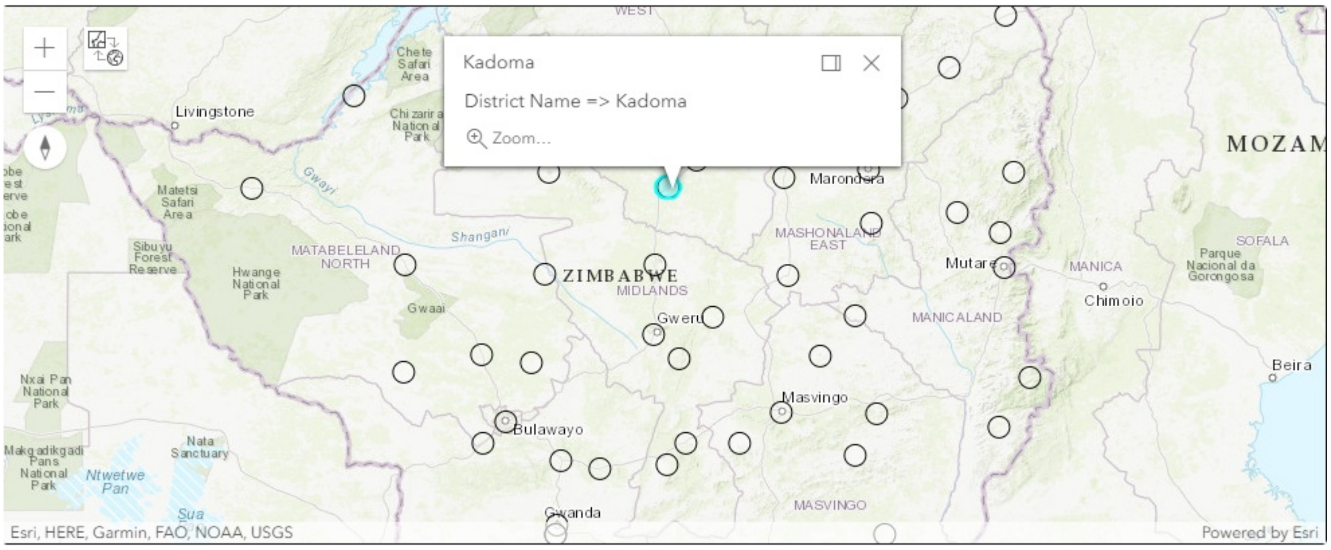
```
60
```

Visualize the Geocoded Addresses on a Simple Map

Let's call a map which we are going to use to display our data.

In [11]:

```
map = gis.map("Zimbabwe", 7)
map
```


Kadoma

District Name => Kadoma

Format Pop-up & Draw Geocoded Point

We are going to be explaining this section is more detail. Its probably the first time to see a different and new syntax like the one below.

So here, we want to display a popup with an `onclick` event. This simply means when the user clicks a certain region on the map within the data range, a popup message with all the information that we have passed will be displayed to that user. So basically here, we are using the Javascript which is embedded within the maps API to display the information that we want our users to view.

The `for` Loop

If you have been coding before this tutorial you probably know about the for loop, but if you are new to programming I am just going to provide a small definition and some referal link below.

For Loop - A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string). *This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.*

For more information about **For Loops** in Python, navigate to W3Schools to get an understanding about how this works.

In this case, we have our sequence which is the records that are stored within the `results` variable which contains the Geocoded information. We are going to loop through these records to make sure every record stored has its own popup.

The Popup Dictionary

Dictionaries are used to store data values in key:value pairs. This is where we are going to specify the data to be displayed within our popup. Read more about Dictionaries here

`title` - gets a column from the dataset which assigns a unique to all our different popups. So here we are going to be using the name of the *District* as our title.

`content` - Is the information that we want to show to the user which is contained within the popup. Here you can display various types of information. For the purposes of this tutorial, we are just going to display the name of the Province in which the District lies in so as to show you how these popups work.

We will discuss about the various types of information in a later session.

Calling the Popup

Now that we have defined all our parameters for the popup, we now need to call the popup and display it on a map.

- using the map function, we simply attach it together with the `draw()` function for displaying our geocoded data to the map
- call the popup function with the popup dictionary

Please Take Note Here: We have not yet altered our data sets (adding X,Y fields) yet. We are simply using the Geocoding Service to show us the locations of the places which are within our CSV file. and then use the data in our file to label these locations with the popups when the user clicks on them.

Take a look 'upstairs' to see what your map looks like now if you have successfully executed the cell block below.

In [12]:

```
for address in results:
    popup = {
        "title": address['attributes']['ShortLabel'],
        "content": "District Name =>  " + address['attributes']['ShortLabel'] # to add more content
for the popup here
    }
    map.draw(address['location'], popup)
```

Format Geocoded Coordinates to Table

This is the step where we alter our data. We want to add the Latitude and Longitude (Lat, Lon) data but we will not be altering the original CSV in case anything goes wrong. We are going to duplicate the records within the original into the new CSV file and then append the new fields (Lat, Long).

First, we need to actually get these coordinates. We do this by creating a **list**.

The variables `latcoords` and `longcoords` below are declaring lists which will store our data and then use the same list to append to the file one by one.

What is a list?

Lists are used to store multiple items in a single variable.

You can read more about *lists* on W3Schools

continuing...

- We are going to loop through each and every record within our dataset to get the `latitude` and the `longitude` of the location.
- After looping through, we are going to be calling the lists that we declared and instruct the function to **append** each and every successful loop into the list of coordinates which are `latcoords` and `longcoords` as indicated at the beginning of the cell block below.

In [13]:

```
latcoords = []
longcoords = []

for coordinates in results:
    latitude = "{:.3f}".format(float(int(coordinates['location']['y']*100))/100)
    longitude = "{:.3f}".format(float(int(coordinates['location']['x']*100))/100)
    latcoords.append(latitude)
    longcoords.append(longitude)
```

- create the new columns and give them names

Since these will be carrying the location information, I just named them accordingly.

- the list will self align itself with the number of records in the CSV file based on how much times the loop function has been executed. This is the main reason why, we had to go through the data *verification* step above in order to make sure that the list contains the same number of records as in the CSV file. If this was not the case, we would end up having misplaced coordinate values.

Execute the block below and see the first 30 records. *Notice at the far right, the two new fields* `Long` *and* `Lat` can have been added to our data.

In [14]:

```
data['Long'] = longcoords
data['Lat'] = latcoords
data.head(30)
```

Out[14]:

	ID_0	ISO	NAME_0	ID_1	Province	ID_2	District	TYPE_2	ENGTYPE_2	NL_NAME_2	VARNAME_2	Long	Lat
0	256	ZWE	Zimbabwe	1	Bulawayo	1	Bulawayo	District	District	NaN	NaN	28.580	-20.140
1	256	ZWE	Zimbabwe	2	Harare	2	Harare	District	District	NaN	Salisbury|Harare Urban	31.040	-17.820
2	256	ZWE	Zimbabwe	3	Manicaland	3	Buhera	District	District	NaN	NaN	31.430	-19.320
3	256	ZWE	Zimbabwe	3	Manicaland	4	Chimanimani	District	District	NaN	NaN	32.870	-19.800
4	256	ZWE	Zimbabwe	3	Manicaland	5	Chipinge	District	District	NaN	NaN	32.620	-20.190
5	256	ZWE	Zimbabwe	3	Manicaland	6	Makoni	District	District	NaN	NaN	32.280	-18.530
6	256	ZWE	Zimbabwe	3	Manicaland	7	Mutare	District	District	NaN	Umtali	32.650	-18.950
7	256	ZWE	Zimbabwe	3	Manicaland	8	Mutasa	District	District	NaN	NaN	32.630	-18.680
8	256	ZWE	Zimbabwe	3	Manicaland	9	Nyanga	District	District	NaN	NaN	32.740	-18.210
9	256	ZWE	Zimbabwe	4	Mashonaland Central	10	Bindura	District	District	NaN	NaN	31.320	-17.300
10	256	ZWE	Zimbabwe	4	Mashonaland Central	11	Centenary	District	District	NaN	Muzarambani	31.110	-16.720
11	256	ZWE	Zimbabwe	4	Mashonaland Central	12	Guruve	District	District	NaN	Sipolilo	30.690	-16.650
12	256	ZWE	Zimbabwe	4	Mashonaland Central	13	Mazowe	District	District	NaN	NaN	30.970	-17.500
13	256	ZWE	Zimbabwe	4	Mashonaland Central	14	Mount Darwin	District	District	NaN	Mt Darwin	31.570	-16.770
14	256	ZWE	Zimbabwe	4	Mashonaland Central	15	Rushinga	District	District	NaN	NaN	32.010	-16.630
15	256	ZWE	Zimbabwe	4	Mashonaland Central	16	Shamva	District	District	NaN	NaN	31.560	-17.290
16	256	ZWE	Zimbabwe	5	Mashonaland East	17	Chikomba	District	District	NaN	NaN	30.890	-19.020
17	256	ZWE	Zimbabwe	5	Mashonaland East	18	Goromonzi	District	District	NaN	NaN	31.370	-17.850
18	256	ZWE	Zimbabwe	5	Mashonaland East	19	Marondera	District	District	NaN	Marandellas	31.540	-18.190
19	256	ZWE	Zimbabwe	5	Mashonaland East	20	Mudzi	District	District	NaN	NaN	32.670	-16.980
					Mashonaland								

	ID_0	ISO	NAME_0	ID_1	Province	ID_2	District	TYPE_2	ENGTYPE_2	NL_NAME_2	VARNAME_2	Long	Lat
20	256	ZWE	Zimbabwe	5	Mashonaland East	21	Murehwa	District	District	NaN	NaN	31.770	-17.640
21	256	ZWE	Zimbabwe	5	Mashonaland East	22	Mutoko	District	District	NaN	NaN	32.210	-17.400
22	256	ZWE	Zimbabwe	5	Mashonaland East	23	Seke	District	District	NaN	NaN	30.850	-18.250
23	256	ZWE	Zimbabwe	5	Mashonaland East	24	UMP	District	District	NaN	Murehwa U.M.P.	31.690	-17.980
24	256	ZWE	Zimbabwe	5	Mashonaland East	25	Wedza	District	District	NaN	Hwedza	31.570	-18.610
25	256	ZWE	Zimbabwe	6	Mashonaland West	26	Chegutu	District	District	NaN	NaN	30.150	-18.120
26	256	ZWE	Zimbabwe	6	Mashonaland West	27	Hurungwe	District	District	NaN	Karoi|Urungwe	29.680	-16.810
27	256	ZWE	Zimbabwe	6	Mashonaland West	28	Kadoma	District	District	NaN	Gatooma	29.910	-18.330
28	256	ZWE	Zimbabwe	6	Mashonaland West	29	Kariba	District	District	NaN	NaN	28.800	-16.520
29	256	ZWE	Zimbabwe	6	Mashonaland West	30	Makonde	District	District	NaN	Lomagundi	30.180	-17.350

Even though we have managed to add this field, it does not mean that this data has been stored. Techically, this data is stored within the Notebook and now we need to actually save it to a file

Remember the file that did not exist that we targeted with our `output_file` variable at the beginning of this tutorial? We are going to be saving this new data to that file and if it does not exist, a new file with that name will be created.

So right below, lets use **pandas** to actually create this file for us using the `DataFrame()` function by passing our `data` variable as a parameter since this is where our new data technically exists and instruct it to save this as a CSV file with `.to_csv` command which houses the location of our output as a parameter.

We have also notified these functions to encode this as **utf8**.

Check the number of records. Still the same and they haven't changed.

```
In [15]:
```

```
pd.DataFrame(data).to_csv(output_file, encoding='utf8')
len(results)
```

```
Out[15]:

60
```

Check in the folder where you stored your output file.

We have our new CSV file with the name we have gave it on the `output_file` variable. I named mine as **final_geocoded_dataset.csv**

If you didn't face any errors along the way and managed to get your places geocoded then;

CONGRATULATIONS

You can easily save this as a template for any Geocoding processes you would like to perform in the future.

For anyone having trouble or fails to understand this tutorial, I am reachable via LinkedIn. Just send me a direct message and I will be sure to respond to any questions relating to the tutorials that you might have.

About Author

- 3rd Year BSc Hons in Surveying & Geomatics
- Interested in GIS for Health and Land Administration, Spatial Data Science and Programming
- currently working on a book titled: **GIS Step by Step: A Practical Guide to GIS**