Data Cleaning Using Python

CIG-NL CONSERVATION GIS WORKSHOP

Summary

Introduction	• Python Fundamentals
Part 1	• Setting Up
Part 2	• Reading Coordinates
Part 3	• Locating Water Supplies
Part 4	• Writing to Output
Part 5	• Results
Final notes	

INTRODUCTION

Interactive Exercise	• Python Fundamentals • Geolocation of Data • Reading and Writing to .csv file
Materials	• TutorialWaterSupplies.csv • Coords.csv • Watershed_tutorial_exercise.py

INTRODUCTION (to Python)

Python uses **whitespace** indentation to delimit blocks – rather than curly braces or keywords. This helps improve readability.

Python doesn't require **semicolons**

There are many ways/resources to learn Python, a slideshow before lunch is generally not recommended.

INTRODUCTION (to Python)

There are many **data types** in Python but we will use the following for this tutorial:

- List **[]** or **list()**, an editable collection of things
- Dictionary **{}** or **dict(),** a series of key: value pairs where the value can be almost anything
- String **'Text'**, can use either single or double quotes, **r'** makes the text be read as raw.
- **None**, this is EXPLICITLY nothing

INTRODUCTION (to Python)

Python has some need features that we will be using to make life easier:

- **with keyword**, this is a built in property of the open command that handles opening and closing behavior when the code block exits.
- **dictionary.get(KEY_VALUE)** this allows us to get a **None** when we have no entry in a dictionary instead of a **KeyError**
- **String formatting '{0}'**, lets you make a string with placeholder values, takes care of a lot of the busy work of formatting outputs.
- **For** loop, let you loop through any collection that can has a sequence
- **If/elif/else**, allows you to have logic within a process.

Part 1: Setup

1. Open **Watershed_tutorial_exercise.py** in a code or text editor:
   - IDLE
   - Notepad++
   - PythonWin
   - Pycharm

2. Fill in datasets
   - Shift click the **.csv** file (alt click in Os X) select **'Copy as path'** (Copy X as pathname in Os X
   - Place the **COORDS.csv** path in the **COORD_SOURCE** constant
   - Place the **TutorialWaterSupplies.csv** path in the **WS_SOURCE** constant

Part 2: Reading Our Coordinates

- **COORDS.csv** comma delimited file with 3 attributes (**PLACE, LONG, LAT**)
- Derived from **geonames.org** daily place name records
- ~34k Place names in NL. Can have multiple hits to same location

	A	B	C
1	PLACE	LONG	LAT
2	101 Ponds	-53.97733	47.811
3	Aaron Arm	-57.63176	47.61668
4	Aaron Arm	-57.66516	47.63328
5	Aaron Cove	-55.71468	52.45009
6	Aaron Island	-57.64846	47.59998
7	Aaron Island	-55.71468	52.45009
8	Aarons Hill	-55.3502	47.21884
9	Abbate Point	-63.64815	59.367
10	Abbot Pond	-55.28146	49.53323
11	Abbot Rock	-53.59806	49.5666
12	Abbots Pond	-57.83186	48.50004
13	Abbott Cove	-55.29817	49.53323
14	Abbott Pond	-55.28146	49.53323
15	Abbott Rock	-55.81505	46.83325
16	Abbotts Cove	-53.86478	49.53321
17	Abbotts Steady	-53.28136	48.51659

Part 2: Reading Our Coordinates

Below the **HEADER** line write the following code:

```
22
23      geolocations = dict()
24      with open(COORD_SOURCE) as coord_file:
25          for row in coord_file:
26              row = row.replace('\n', '')
27              split_row = row.split(',')
28              place, lon, lat = split_row
29              coordinates = lon, lat
30              if place in geolocations:
31                  geolocations[place].append(coordinates)
32              else:
33                  geolocations[place] = [coordinates]
34
```

Part 2: Reading Our Coordinates

Line by Line

```
geolocations = dict()  # Create an empty Dictionary to hold our coordinates
with open(COORD_SOURCE) as coord_file:  # open the file
    for row in coord_file:  # we are going to loop over the rows in the csv
        row = row.replace('\n', '')  # clean up the text by removing newline(\n)
        split_row = row.split(COMMA)  # split the csv line by commas)
        place, lon, lat = split_row  # create variables for the 3 columns
        coordinates = lon, lat  # assign the lat long to a single coord
        # NOTE: where place names are not unique we need to account for multiple
        # NOTE: place name results
        if place in geolocations:  # check to see if we got an existing result
            geolocations[place].append(coordinates)  # add the result
        else:  # if the first check fails this executes
            geolocations[place] = [coordinates]  # create a new result
```

Part 3: Locating Our Water Supplies

- Derived from Water Supply Dataset on Government Open data portal:
- http://opendata.gov.nl.ca/public/opendata/page/?page-id=datasetdetails&id=302
- **TutorialWaterSupply.csv** comma delimited file with multiple attributes
- Original Dataset was scraped for best geolocation results and stored in the **BEST_MATCH** column
- Was able to scrape 554 of the 647 entries with no alteration of the downloaded file

BEST_MATCH	COMMUNITY	AREA	SOURCE	SUPPLY_TYPE	STATUS
Blaketown	Blaketown	Blaketown North	#4 Well	GW	Unprotected
Brigus South	Brigus South	Dunphey's Hill	#2 Well Dunphey's Hilll	GW	Unprotected
Forge Hill	Brigus South	Forge Hill	#1 Well Forge Hill	GW	Unprotected
Brigus South	Brigus South	Near highway	#3 Well Main Road	GW	Unprotected
Bryant's Cove	Bryant's Cove	Bryant's Cove South Side	Well Field	GW	Protected
Bunyan's Cove	Bunyan's Cove	Bunyan's Cove	#1 Well	GW	Protected
Cannings Cove	Canning's Cove	Lower Canning's Cove	#1 Well - Pleman Pitts	GW	Protected
Cannings Cove	Canning's Cove	Upper Canning's Cove	#2 Well - Eugene Ellis	GW	Protected
Cannings Cove	Canning's Cove	Centre Canning's Cove	#3 Well - Glenda Penney	GW	Protected
Cavendish	Cavendish	North Side Cavendish	#1 Well - Max Bishop	GW	Unprotected
Chance Cove	Chance Cove	Upper Cove Centre	Angus Brace Well	GW	Unprotected
Chance Cove	Chance Cove	Upper Cove South	Edgar Crann Well	GW	Unprotected
Chance Cove	Chance Cove	Back Cove	Olive Smith Well	GW	Unprotected
Chance Cove	Chance Cove	New Housing	New Housing Area Well	GW	Unprotected
Clarke's Beach	Clarke's Beach	Otterbury	#1 Well - Quinlon Well	GW	Unprotected
Clarke's Beach	Clarke's Beach	Otterbury	#2 Well - Delaney Well	GW	Unprotected
Colliers	Colliers	Main Road	#1 Mahoney's Well	GW	Protected
Admiral's Beach	Admirals Beach	Admiral's Beach	2 Well Fields	GW	Unprotected
Badger	Badger	Badger	Well Field 3 wells on standby	GW	Protected
Badger	Badger	Badger	Well Field 3 wells on standby	GW	Protected

Part 3: Locating Our Water Supplies

Below the previous section write the following code:

```
water_locations = []
with open(WS_SOURCE) as pws_file:
    reader = DictReader(pws_file)
    for record in reader:
        place_name = record.get('BEST_MATCH').replace(COMMA, BLANK)
        status = record.get('STATUS')
        locations = geolocations.get(place_name)
        if not locations:
            continue
        number_of_coords = len(locations)
        for location in locations:
            lon, lat = location
            output_row = OUTPUT_TEMPLATE.format(
                lon, lat, place_name, status, number_of_coords)
            water_locations.append(output_row)
```

Part 3: Locating Our Water Supplies

```
water_locations = []  # Create an empty List to hold our location results
with open(WS_SOURCE) as pws_file:  # Open the Public water supply file
    # NOTE: a DictReader is part of the built in csv library that was
    # NOTE: imported earlier, that allows you to access row data by column name
    reader = DictReader(pws_file)  # transform the file into a DictReader
    for record in reader:  # Loop through the records
        # We will lookup the record by columns here, cleaning up the comma from
        # best match
        place_name = record.get('BEST_MATCH').replace(COMMA, BLANK)
        status = record.get('STATUS')  # Look up status
        locations = geolocations.get(place_name)  # Try to get the
        # Coordinates of the location from our geolocation dictionary
        if not locations:  # the get keyword is a Null if we don't have an entry
            continue  # if we don't have an entry, skip to the next water supply
        number_of_coords = len(locations)  # count the number of results to find
        # out the level of certainty we have to the location
        for location in locations:  # loop through all the dictionary results
            lon, lat = location  # get long and lat from the location
            # NOTE: This could be on one line, but you can break lines with
            # NOTE: Parentheses without changing code exectution
            output_row = OUTPUT_TEMPLATE.format(
                lon, lat, place_name, status, number_of_coords)
            # Populate the output row template
            water_locations.append(output_row)  # add our results to our list
```

Part 4: Writing to Output

Below the previous section write the following code:

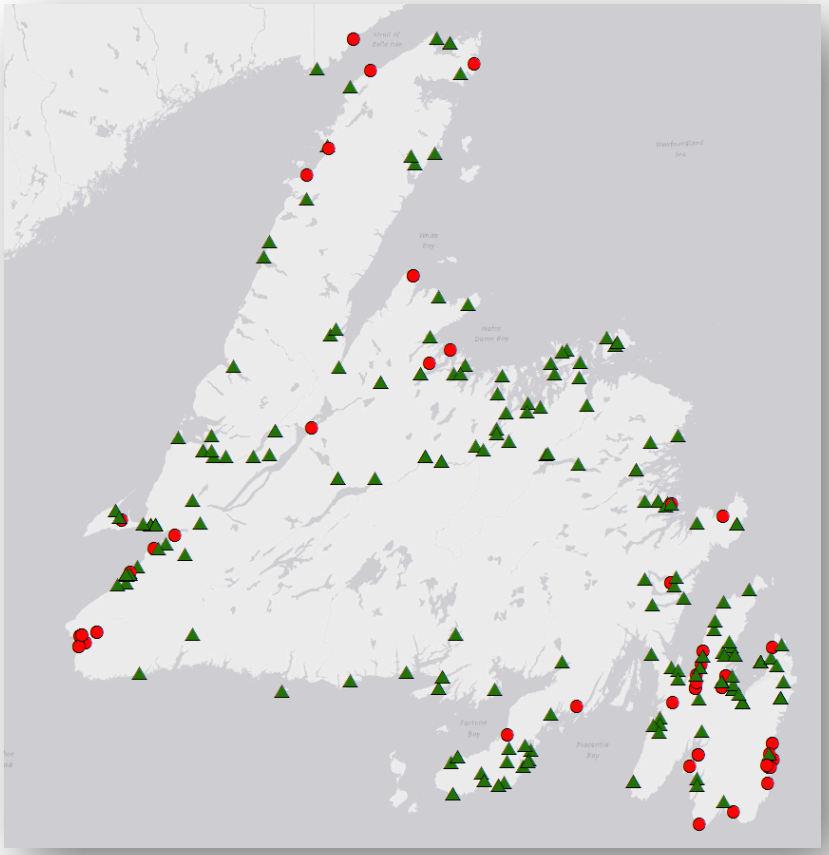
```
with open(OUTPUT_LOCATION, 'w') as out_file:
    out_file.write(HEADER)
    out_file.writelines(water_locations)
```

Part 4: Writing to Output

```
# NOTE: we are using the 'w' arguement on our outfile, this indicates we are in
# NOTE: 'w'rite Mode, wiht no argment it defaults to 'r'eadmode
with open(OUTPUT_LOCATION, 'w') as out_file:# Open our output file
    out_file.write(HEADER)# write our file header
    out_file.writelines(water_locations)# write all our results at once
```

Part 5: Results

We have now created located **potential water supply locations**, using your GIS platform of choice you can now add the Lat/Lon Events and finally get around to doing your analysis.



Results plotted in ArcGIS of exact matches, by status.

▲ Green Triangle = Protected
● Red Circle = Unprotected

Final Note

- There are additional resources available within the distribution of this workshop.
- I included the original scratch python files (I say scratch because they are not production quality) I used to prepare the data along with the original downloaded input files.
- They are included in the Resources Subfolder.