Nunavut Project Structure

# Progress so far:

1. Can extract 16 of 25 Nunavut hamlets’ building numbers and their geographical coordinates.
2. Can cluster those hamlets with building numbers into clusters by three algorithms
3. Can color clustering results in original PDF maps

# Current plans:

1. Get to know the building number standards in the remaining 9 hamlets
2. Devise a more reasonable distance measuring algorithm

# Python Version: 3.9

# Packages used in the project:

[re](https://docs.python.org/3/library/re.html) : used for parsing and analyzing strings

[pdfplumber](https://pypi.org/project/pdfplumber/) : used for extracting chars, rectangles and lines from a PDF file.

[xlwt](https://xlwt.readthedocs.io/en/latest/) : used for writing and editing .xls files.

[xlrd](https://xlrd.readthedocs.io/en/latest/) : used for reading .xls files.

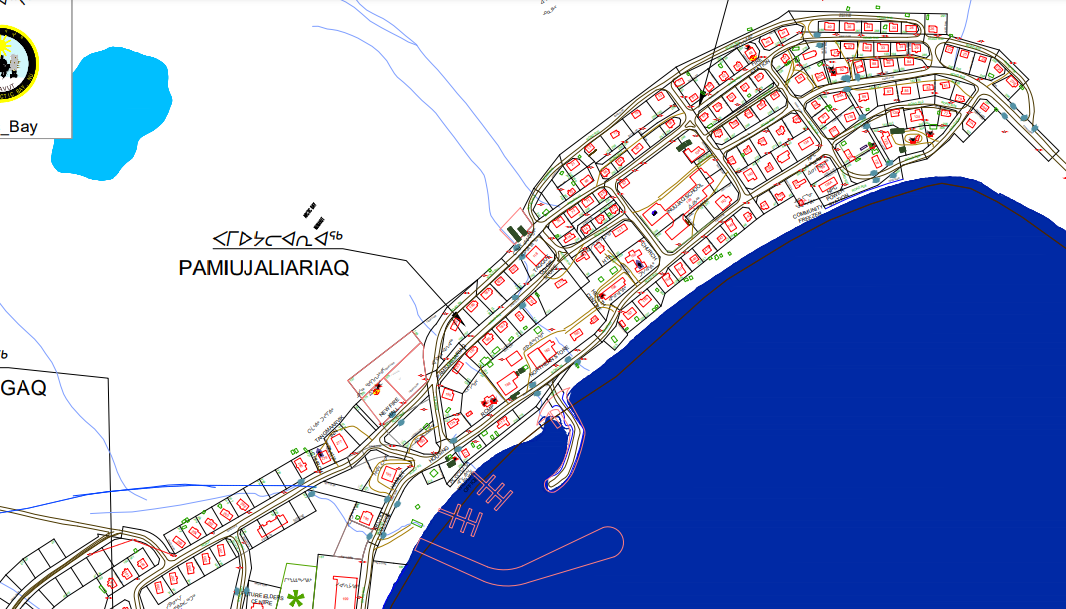
[xml.etree.ElementTree](https://docs.python.org/3/library/xml.etree.elementtree.html) : used for parsing and writing .xml file

[pdf\_annotate](https://github.com/plangrid/pdf-annotate) : used for annotating PDF files.

# Folders:

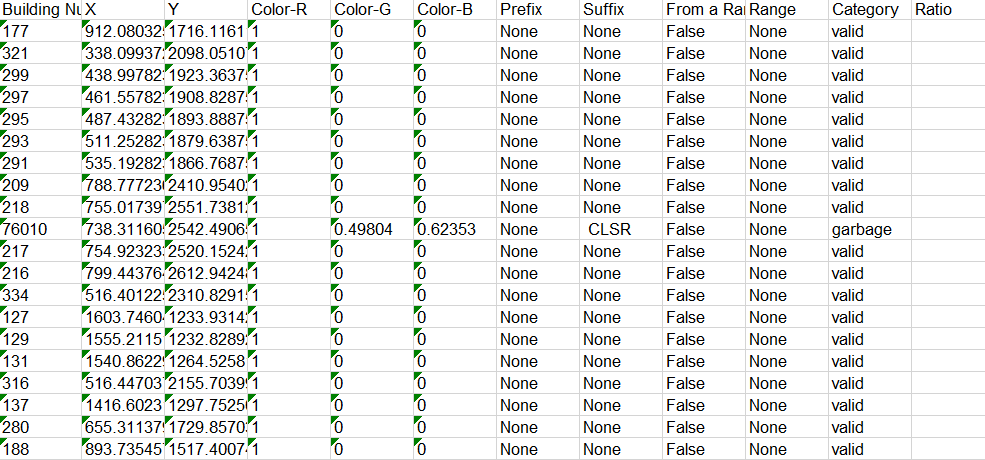
## **/Building\_Number\_Maps\_2021 Gonzalo**

PDF files which are Nunavut building number maps provided by Gonzalo. There are 25 PDF maps in total. Currently 16 of them have valid building numbers.



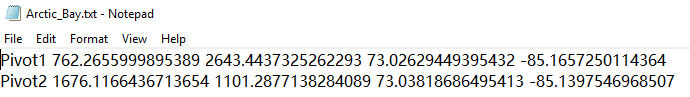
## **/semi\_data\_geo coordinates**

.xls files that are generated from **all\_map\_parser.py** and used to obtain geographic coordinates by **find\_geo\_coordinates.py**. In addition to geographical coordinates, other information, including color, prefix, suffix, whether from a range and category, is also stored under this directory.



## **/pivot points**

.txt files that contain 2~4 pivot points that help find geographic coordinates in each hamlet. Each pivot point contains both its relative coordinates in the PDF map and its geographical coordinates(Longitude and Latitude). They are manually selected from Google Maps. Used in **find\_geo\_coordinates.py**

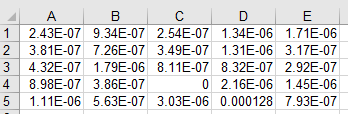
****

## **/geo coordinates**

.xls files generated from **find\_geo\_coordinates.py**.Each .xls file contains every building’s relative coordinates in the PDF map and geographic coordinates. Used for clustering in **clustering\_SOM.py**, **clustering\_k\_means.py** and **clustering\_hierarchical\_clustering.py**

## **/division\_SOM**

.xml and .xls files generated from **clustering\_SOM.py**. .xml files are the ways a hamlet is divided into clusters by SOM(Self Organizing Maps) algorithm. .xls files are matrices that are used to evaluate the clustering results. Also contains PDF files, which are visualized clustering results by **map\_coloring.py**.



## **/division\_k\_means**

.xml and .xls files generated from **clustering\_k\_means.py.** .xml files are the ways a hamlet is divided into clusters by K-means algorithm. .xls files are matrices that are used to evaluate the clustering results. Also contains PDF files, which are visualized clustering results by **map\_coloring.py**.



## **/division\_hierarchical\_clustering**

.xml and .xls files generated from **clustering\_hierarchical\_clustering.py** and **hierarchical\_clustering\_divider.py.** .xml files are the ways a hamlet is divided into clusters by hierarchical clustering algorithm. .xml files that end with “\_detailed.xml” are clustering results in fully hierarchical structure, and those end with “\_division.xml” are clustering results being cut in a certain level. .xls files are matrices that are used to evaluate the clustering results. Also contains PDF files, which are visualized clustering results by **map\_coloring.py**.

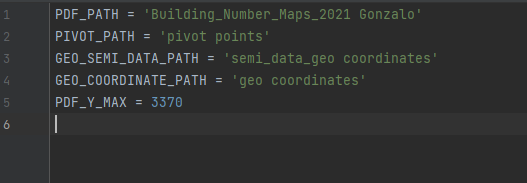
# .py files:

## **common\_modules.py**

Common modules used in other files. It includes the **Building\_Number class**, which mainly stores building numbers, their relative coordinates and geographical coordinates. It also includes **read\_data\_from\_xls function**,

## **constants.py**

constants used in other files. It includes:



**PDF\_PATH** : the path of Gonzalo’s pdf maps

**PIVOT\_PATH** : the path of pivot points

**GEO\_SEMI\_DATA\_PATH** : the path of semi data that are used for finding geographical coordinates

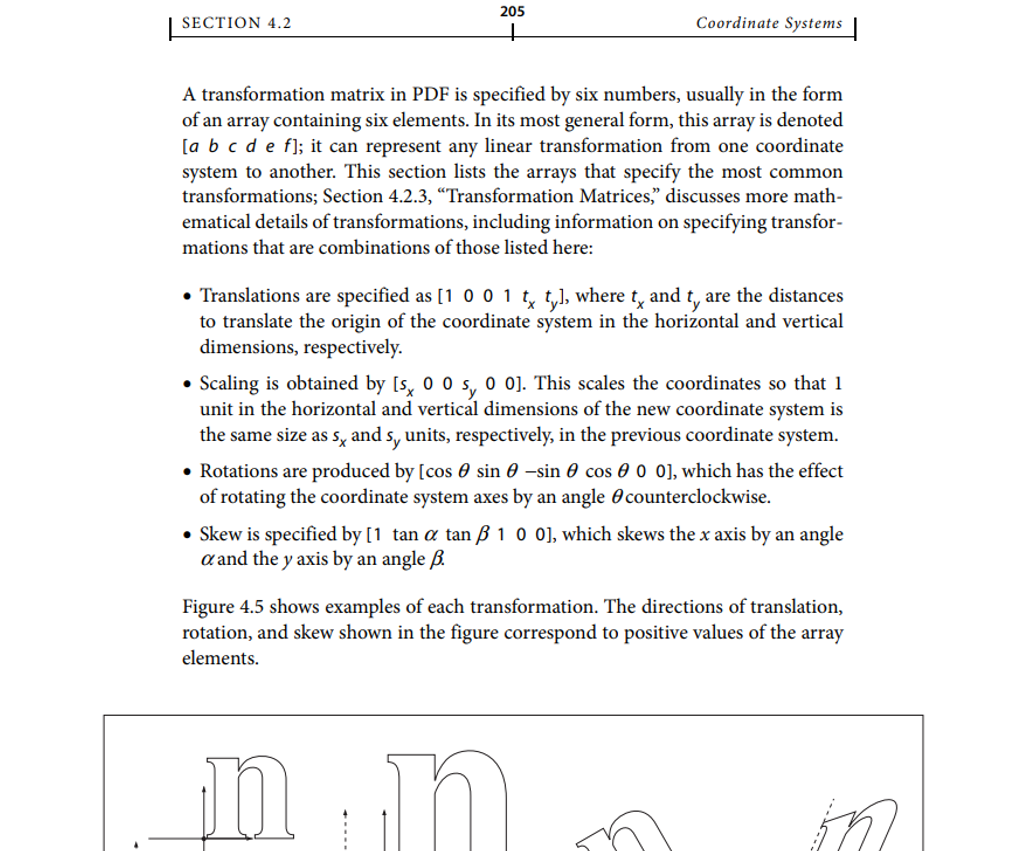
**GEO\_COORDINATE\_PATH** : the path of buildings’ information

**PDF\_Y\_MAX** : the height of PDF files in PDF\_PATH

## **all\_map\_parser.py**

This program parses maps in **/Building\_Number\_Maps\_2021 Gonzalo**, and extracts all building numbers from each PDF map. [pdfplumber](https://pypi.org/project/pdfplumber/) package is used in this program to extract chars and their properties from PDF maps.

This program mainly focuses on assembling individual chars in those PDF maps into continuous strings that may contain building numbers. Although there is a method in [pdfplumber](https://pypi.org/project/pdfplumber/) package that can obtain all strings in a PDF file, it is quite time consuming and inefficient, because this method deals with many irrelevant information in a PDF file.

The basic algorithm of this program has something to do with Current Transformation Matrices, which tell the rotations of PDF chars. Chars in the same string should have the same rotation. 

Furthermore, since every building number in PDF maps of **/Building\_Number\_Maps\_2021 Gonzalo** are colored red, the color of PDF chars is an important indicator of whether a string is a valid building number.

## **find\_geo\_coordinates.py**

This program converts buildings’ relative coordinates obtained from **all\_map\_parser.py** to geographic coordinates. The main algorithm of the program is to use 2 (In Iqaluit, use 4) pivot points that have their relative coordinates. Manually locate their geographic coordinates in Google Maps. Then every other buildings’ geographic coordinates can be calculated.

## **clustering\_SOM.py**

This program uses the SOM(Self Organizing Maps) algorithm to categorize buildings into different clusters. The number of clusters is given for every hamlet.

Its performance is evaluated by three matrices: Number of buildings, Standard deviation and Average distances to three closest weights(geographical centers of clusters). In an ideal clustering solution, every cluster should have a similar number of buildings, small standard deviations and a similar average distance to three closest weights.

Currently, the distance between two buildings is simply the Euclidean distance. Better ways to measure distances may be used in the future.

## **clustering\_k\_means.py**

This program uses the K-means clustering algorithm. It uses the same set of matrices to evaluate its clustering results.

The times of iteration is given, but the number of clusters varies for different hamlets. Generally, its performance is better than SOM; on the other hand, it’s more time-consuming.

## **clustering\_hierarchical\_clustering.py**

This program uses the hierarchical clustering algorithm to cluster. The result is stored in a tree that can reflect its hierarchical structure.

For convenience, the way to measure proximity of two clusters is Single Leakage, i.e. the shortest distance between two clusters.

This algorithm is the most time consuming one. In the worst scenarios, its time complexity is O(n^4) (n is the number of buildings).

## **hierarchical\_clustering\_divider.py**

This program cuts the results of clustering\_hierarchical\_clustering.py at a certain level and turns them into many clusters that can be evaluated through matrices used in SOM and K-means clustering.

Generally, the result is the worst among the three algorithms, because many clusters only have one building, which is not desirable.

## **map\_coloring.py**

This program visualizes clustering results. It colors buildings of the same cluster the same color, and neighboring clusters have quite different colors.

# Flow Chart of the project

