# Itinerary.xml to .csv Python Script

## Instructions

The runtime varies depending on the size of the itinerary.xml, and particularly the number of routes and vertices. A 120,000-line Infrastructure Data.xml file takes about 6 minutes to run.

1. Run the script.
2. When prompted, select the itinerary.xml
3. Wait for completion.
4. Open the XML\_To\_CSV\_Output.csv found in the same folder as the script.
   1. Open with Excel either with “Open with” windows functionality or by dragging the csv to an excel window.
5. Rename the output.csv as needed.

If an xml output is desired for debugging or other purposes, un-comment line 144 in the script.

## Inputs and Outputs

Input:

* Infrastructure Data.xml (exported from OT (trafIT-Format))

Output:

* XML\_To\_CSV\_Output.csv

Optional output:

* XML\_To\_CSV\_Output.xml

## Maintenance/Code Structure

Almost all functional code is contained in functions because, in Python, nested loops tend to run faster in functions. Due to how the Element Tree module deals with hierarchical data, nested for loops are needed to be able to .append and .remove nodes.

|  |  |
| --- | --- |
| **Variable Name** | **Variable Value/Purpose** |
| branch | Iterable used when referring to objects just below the root (Eg. <itineraries>, <paths>, <edges> |
| child | Iterable used to refer to the direct children of an object (Eg. <path> object is a child of <itinerary> objects |
| cols | List for storing column headers. Used when writing to csv |
| copied[Path/Route/Vertex] | variable used to store outputs of searchNode function calls. These are element tree objects which may have child nodes. Any attributes are also stored |
| df | Dataframe handle of post-xml and post-cleanup data. Any future data processing can be added after this point |
| documentName | Variable used to store “documentname” attribute of a vertex |
| id | Variable used to store “id” attribute of a vertex |
| input | handle used to store opened input file |
| inputFile | variable used to store handle of the user-selected input file |
| itineraryNode | iterable used to refer to <itinerary> elements |
| neighbourid | Variable used to store “id” attribute of a vertex’s neighbour vertex |
| node | Iterable used to refer to a node. Context dependent usage |
| pathNode | iterable used to refer to <path> elements |
| root | Root element of the tree object |
| routeNode | iterable used to refer to <route> elements |
| rows | List for storing row data. Used when loading data to dataframe |
| tree | ElementTree object used as a handle for xml data |
| vertexName | Variable used to store “name” attribute of a vertex |
| vertexNode | iterable used to refer to <vertex> elements |

## Areas for Improvement

* Optimization: This script contains many nested for loops, which dramatically slow it down. Future improvements could optimize the script by reducing the number of nested loops. Better results may be easier to achieve using the lxml python module, as it appears you don’t need nested loops to append or remove nodes.
* Extended Functionality: Adding a way to specify which columns to include may be useful in the future.
* Improving Resilience: Making sure the script can work with all Infrastructure Data.xml files.

# Turnback Check Python Script

## Instructions

The turnback check is conducted after the xml file has been converted to a csv file. This is run within the same python script but both the itinerary csv file and the final turnback check excel file are outputted. The turnback check automatically begins after the xml to csv conversion is done so the setup and input for the python script are the same as above.

## Inputs and Outputs

Inputs:

* XML\_To\_CSV\_Output.csv
* course\_xml.csv (exported from OT as Courses (trafIt-Format) xml file and converted to csv using CourseXML Converter.exe)

Output:

* Turnback\_Check\_Output.xlsx

## Maintenance/Code Structure

The structure of the turnback check is the same as the xml to csv conversion portion of the script, in that the majority of the code is sectioned into functions. Essentially, the code takes the csv file in as a dataframe, loops through each itinerary’s vertexes, and returns “Turnback” if the list of vertices includes a vertex ID and it’s neighbour ID within the same itinerary. This list of turnbacks is then added to the output file along with the name of the itinerary. The final step is to output the potential courseID’s for each turnback by referencing the course.xml file.

|  |  |
| --- | --- |
| **Variable Name** | **Variable Value/Purpose** |
| df\_in | The csv file input as a dataframe providing the itinerary names, and vertex information used to identify turnbacks |
| df\_course\_xml | Course.xml file input as dataframe to connect itinerary to CourseID |
| df\_true\_false | A copy of the csv df used to filter out ‘na’ values from input df |
| turnbacks | Variable used to store a list of whether each itinerary contains a turnback or not |
| vertexID\_array | Variable used to a store 2-value array including the vertex ID and corridor of the current row in the itinerary |
| vertexID\_documentName\_list | Variable used to store a list of each vertexID\_array |
| neighbourID\_array | Variable used to a store 2-value array including the neighbouring vertex ID and corridor of the current row in the itinerary |
| neighbourID\_documentName\_list | Variable used to store a list of each neighbourID\_array |
| turnback\_found | Variable used to store a True or False value identified by the comparison of the vertexID and neighbourID lists contain a match (and therefore the itinerary contains a turnback) |
| itineraries | Variable used to store a list of every itinerary name |
| course\_ID | Variable used to store a list of all possible courseID’s for itineraries containing a turnback |
| df\_final\_output | Dataframe made to put all output information together into one file |

## Areas for Improvement

* Optimization: The turnback check portion of this script takes around 10 seconds to run through approximately 200,000 lines of itinerary information in the XML\_To\_CSV\_Output.csv file so it probably doesn’t require much more optimization.
* Extended Functionality: The turnback check portion of the script relies on the output from the xml to csv conversion portion being formatted a specific way. If this were to be changed, the turnback section would also need to be changed. It may be useful to alter this portion to be able to run based on a more general csv input.
* Improving Resilience: Making sure the script can work with all Infrastructure Data.xml files.