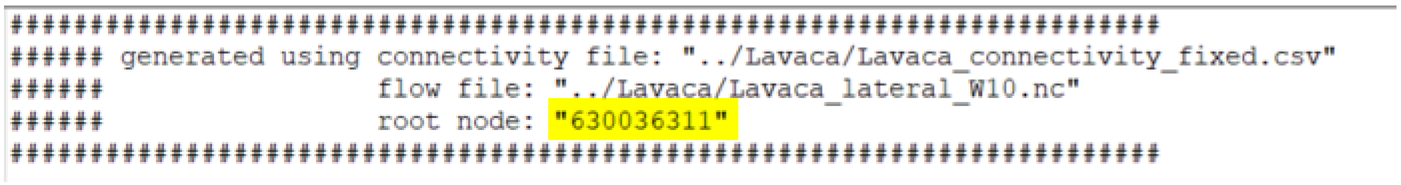
Workflow for changing SPRNT .spt to SWMM .inp file

Steps to convert a SPRNT .spt to a usable SWMM .inp file

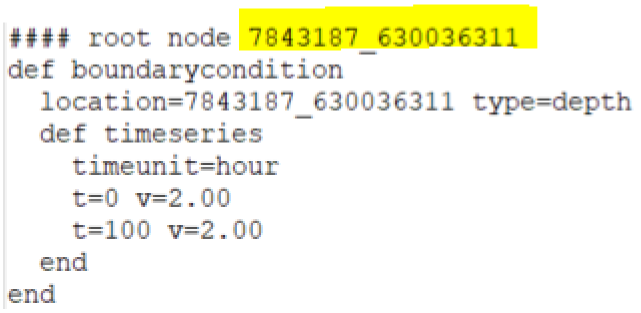
1) Starting Point Requirements –
    a. SPRNT *.spt file
    b. elevation_slope.csv
    c. STEP1_Read.py, STEP2_Process.py, STEP3_Write.py
    d. TRB_Sample.inp in an empty directory
    e. latest SWMM5.1 UI
    f. Python 3
        i. pip install shelve2, numpy

2) Open *.spt and find the root node



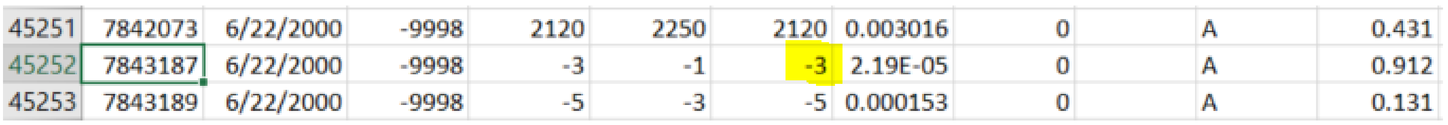
```
##########################################################################
###### generated using connectivity file: "../Lavaca/Lavaca_connectivity_fixed.csv"
######                      flow file: "../Lavaca/Lavaca_lateral_W10.nc"
######                      root node: "630036311"
##########################################################################
```

3) Search the *.spt for the root node's boundary condition, note the root node's full name



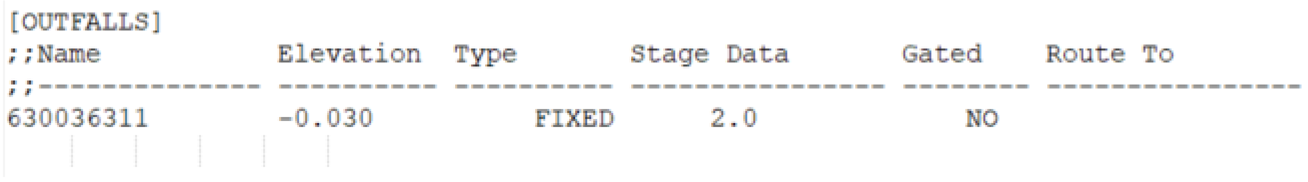
```
#### root node 7843187_630036311
def boundarycondition
  location=7843187_630036311 type=depth
  def timeseries
    timeunit=hour
    t=0 v=2.00
    t=100 v=2.00
  end
end
```

4) Search the elevation_slope.csv for the COMID containing the root node (in this case, '7843187'). Note the minimum elevation of that COMID; this is the elevation of the system outfall (it can be negative). *Must convert from centimeters (csv file) to meters (SWMM).



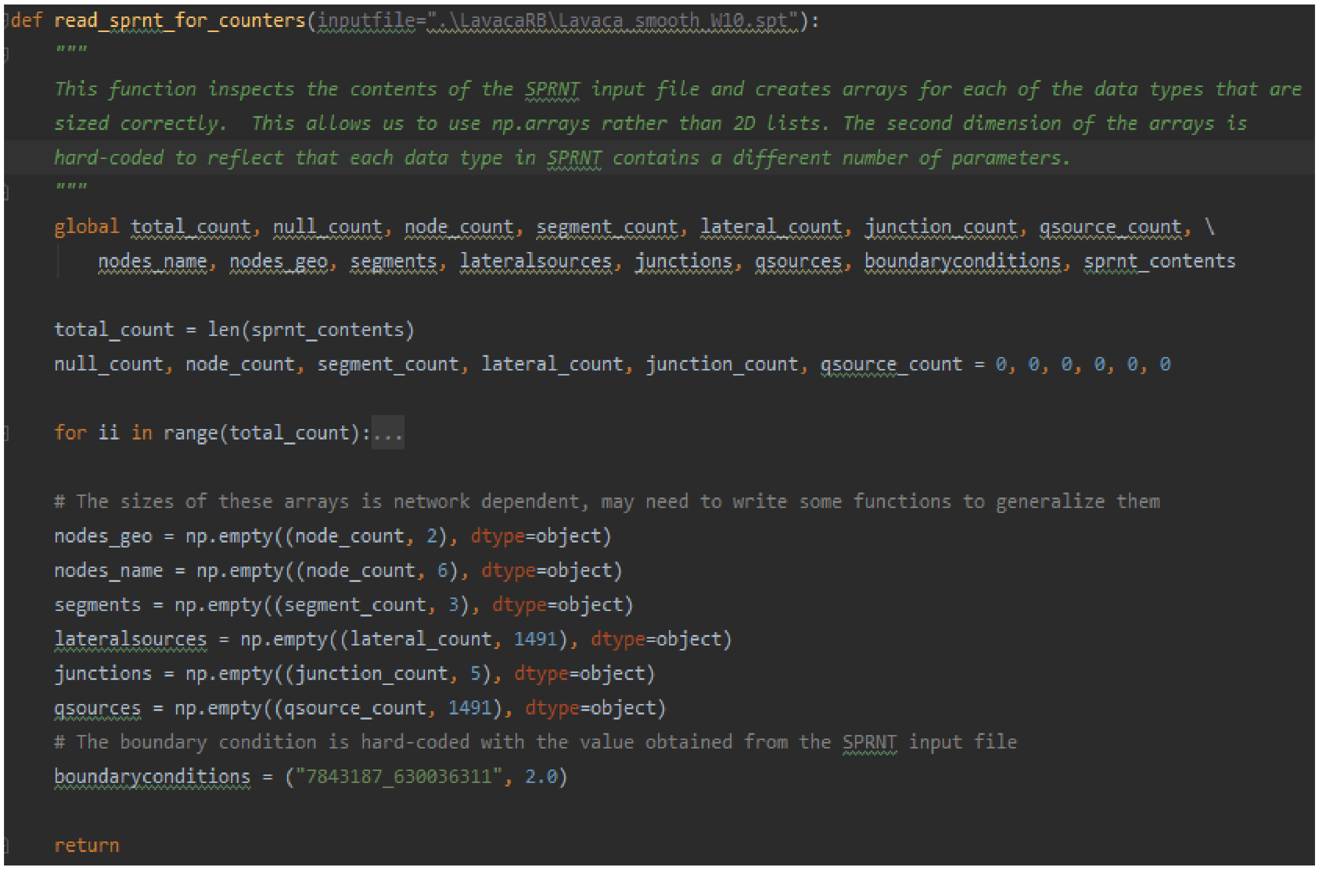
45251	7842073	6/22/2000	-9998	2120	2250	2120	0.003016	0		A		0.431
45252	7843187	6/22/2000	-9998	-3	-1	-3	2.19E-05	0		A		0.912
45253	7843189	6/22/2000	-9998	-5	-3	-5	0.000153	0		A		0.131

5) Open the TRB_Sample.inp and populate the [OUTFALLS] section with the root node name and elevation. If the boundary condition from 2) is constant, include a 'FIXED' B.C. and stage info. If the boundary condition from 2) is not constant, include the name of the timeseries containing the B.C. data.



```
[OUTFALLS]
;;Name            Elevation  Type       Stage Data       Gated    Route To
;;--------------- ---------- ---------- ---------------- -------- ----------------
630036311         -0.030          FIXED      2.0             NO
```

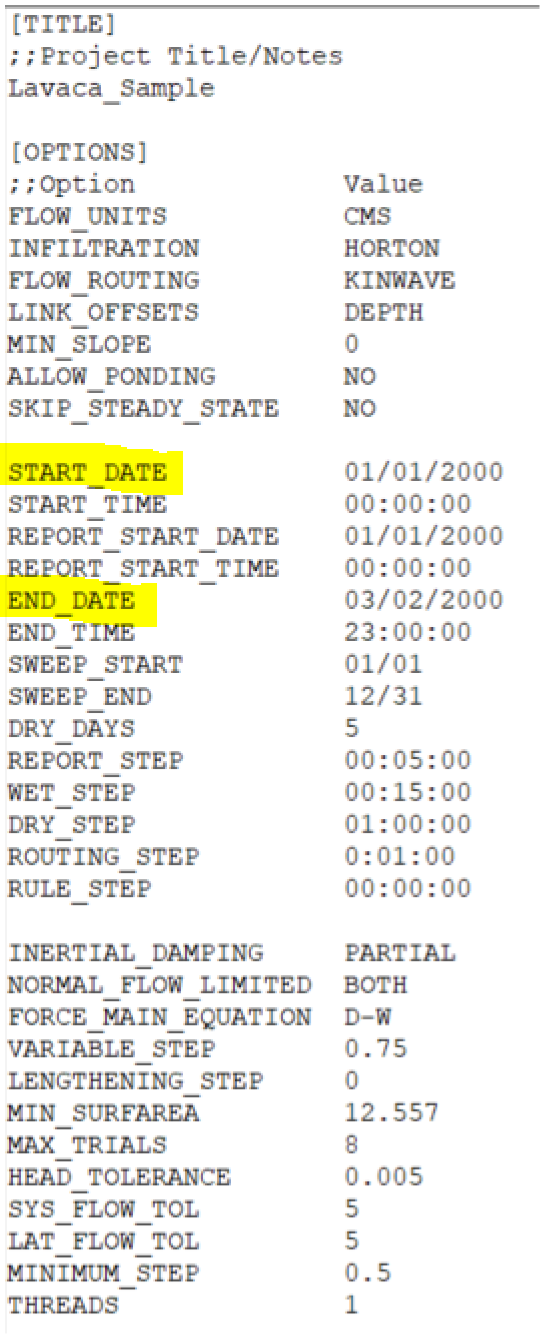
6) The next step is to determine some of the hardcoded parameters in the conversion scripts from the *.spt file. Open STEP2_ Process.py and locate read_sprnt_for_counters().



```
def read_sprnt_for_counters(inputfile=".\LavacaRB\Lavaca_smooth_W10.spt"):
    """
    This function inspects the contents of the SPRNT input file and creates arrays for each of the data types that are
    sized correctly.  This allows us to use np.arrays rather than 2D lists. The second dimension of the arrays is
    hard-coded to reflect that each data type in SPRNT contains a different number of parameters.
    """
    global total_count, null_count, node_count, segment_count, lateral_count, junction_count, qsource_count, \
        nodes_name, nodes_geo, segments, lateralsources, junctions, qsources, boundaryconditions, sprnt_contents

    total_count = len(sprnt_contents)
    null_count, node_count, segment_count, lateral_count, junction_count, qsource_count = 0, 0, 0, 0, 0, 0

    for ii in range(total_count):...

    # The sizes of these arrays is network dependent, may need to write some functions to generalize them
    nodes_geo = np.empty((node_count, 2), dtype=object)
    nodes_name = np.empty((node_count, 6), dtype=object)
    segments = np.empty((segment_count, 3), dtype=object)
    lateralsources = np.empty((lateral_count, 1491), dtype=object)
    junctions = np.empty((junction_count, 5), dtype=object)
    qsources = np.empty((qsource_count, 1491), dtype=object)
    # The boundary condition is hard-coded with the value obtained from the SPRNT input file
    boundaryconditions = ("7843187_630036311", 2.0)

    return
```

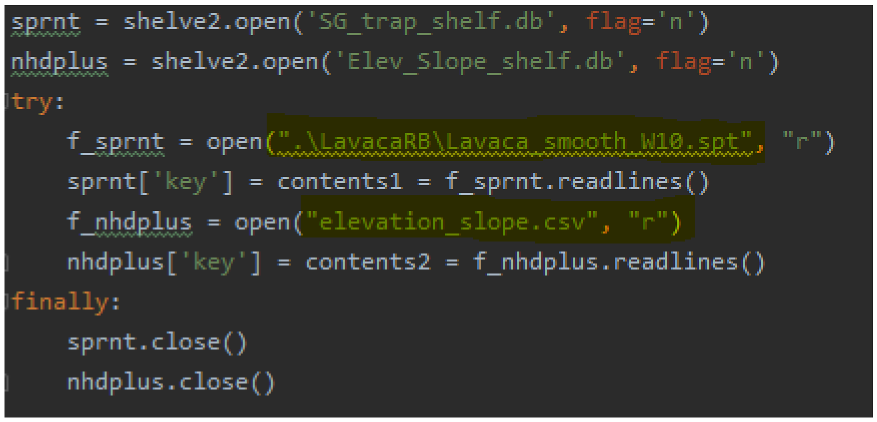
a. Inputfile argument = (relative path of the *.spt)
b. Nodes - If the geometry of the nodes in the *.spt are ‘intrinsic’, nodes_geo should have 2 columns, whereas nodes_name should have 6.
c. Segments - will always have 3 columns
d. Lateralsources - Scan the Lateralsources in the *.spt for the longest timeseries, the number of lateralsources columns = max_timeseries + 3 (to leave room for the source and timeunit). Also note the largest time value in Lateralsources (may not belong to the longest timeseries), this represents the total length of the simulation.
e. Junctions - For TRB, the junctions are limited to having three connections (two upstream and one downstream), so 5 columns suffices (two for the upstream node names, two for the upstream node coefficients, one for the downstream node name).
f. Qsources - Scan the Qsources in the *.spt for the longest timeseries, the number of Qsources columns = max_timeseries + 3 (to leave room for the source and timeunit). If the largest time value in Qsources is larger than the max time value in Lateralsources, it becomes the new total length of the simulation.

7) Set the SWMM.inp START_DATE/TIME and END_DATE/TIME. The START_DATE/TIME is arbitrary, but the END_DATE/TIME is the START_DATE/TIME + the total length of the simulation



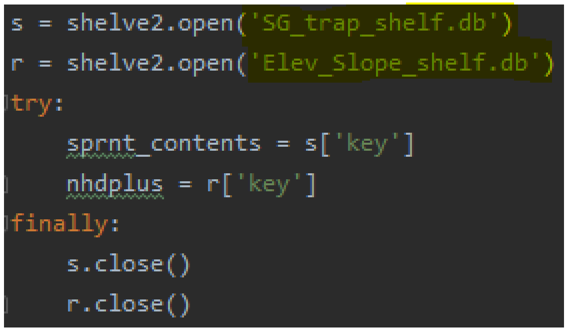
```
[TITLE]
;;Project Title/Notes
Lavaca_Sample

[OPTIONS]
;;Option               Value
FLOW_UNITS             CMS
INFILTRATION           HORTON
FLOW_ROUTING           KINWAVE
LINK_OFFSETS           DEPTH
MIN_SLOPE              0
ALLOW_PONDING          NO
SKIP_STEADY_STATE      NO

START_DATE             01/01/2000
START_TIME             00:00:00
REPORT_START_DATE      01/01/2000
REPORT_START_TIME      00:00:00
END_DATE               03/02/2000
END_TIME               23:00:00
SWEEP_START            01/01
SWEEP_END              12/31
DRY_DAYS               5
REPORT_STEP            00:05:00
WET_STEP               00:15:00
DRY_STEP               01:00:00
ROUTING_STEP           0:01:00
RULE_STEP              00:00:00

INERTIAL_DAMPING       PARTIAL
NORMAL_FLOW_LIMITED    BOTH
FORCE_MAIN_EQUATION    D-W
VARIABLE_STEP          0.75
LENGTHENING_STEP       0
MIN_SURFAREA           12.557
MAX_TRIALS             8
HEAD_TOLERANCE         0.005
SYS_FLOW_TOL           5
LAT_FLOW_TOL           5
MINIMUM_STEP           0.5
THREADS                1
```

8) Open STEP1_ Read.py, change the sprnt and nhdplus keys to the relative paths for the *.spt and elevation_slope.csv, respectively. Then run STEP1_ Read.py.



```
sprnt = shelve2.open('SG_trap_shelf.db', flag='n')
nhdplus = shelve2.open('Elev_Slope_shelf.db', flag='n')
try:
    f_sprnt = open(".\LavacaRB\Lavaca_smooth_W10.spt", "r")
    sprnt['key'] = contents1 = f_sprnt.readlines()
    f_nhdplus = open("elevation_slope.csv", "r")
    nhdplus['key'] = contents2 = f_nhdplus.readlines()
finally:
    sprnt.close()
    nhdplus.close()
```

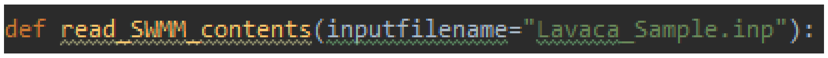
9) Open STEP2_ Process.py. Ensure that the shelve2.open() command is pointing at the databases created by STEP1_ Read.py. Then run STEP2_ Process.py



```
s = shelve2.open('SG_trap_shelf.db')
r = shelve2.open('Elev_Slope_shelf.db')
try:
    sprnt_contents = s['key']
    nhdplus = r['key']
finally:
    s.close()
    r.close()
```

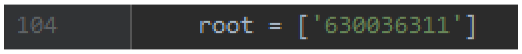
10) Open Step3_ Write.py. Ensure that the shelve.open() command points to the sprnt_info.db created by STEP2_ Process.py.

   a. Locate the read_SWMM_contents() function, include as an argument the relative path to the TRB_Sample.inp.



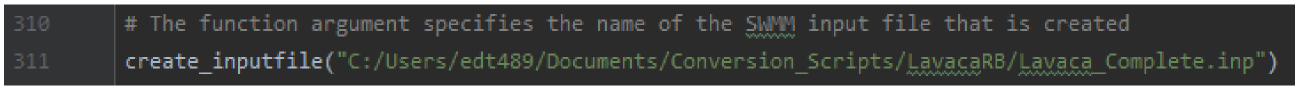
```
def read_SWMM_contents(inputfilename="Lavaca_Sample.inp"):
```

   b. Locate the root variable in the function create_inputfile(). Change it to reflect the root node of *.spt



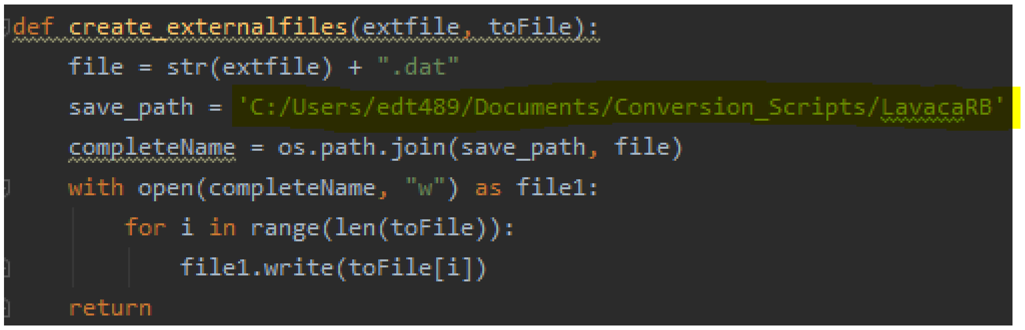
```
        root = ['630036311']
```

   c. Locate the function call for create_inputfile. Include as an argument the absolute path to the TRB_Complete.inp you wish to create



```
# The function argument specifies the name of the SWMM input file that is created
create_inputfile("C:/Users/edt489/Documents/Conversion_Scripts/LavacaRB/Lavaca_Complete.inp")
```

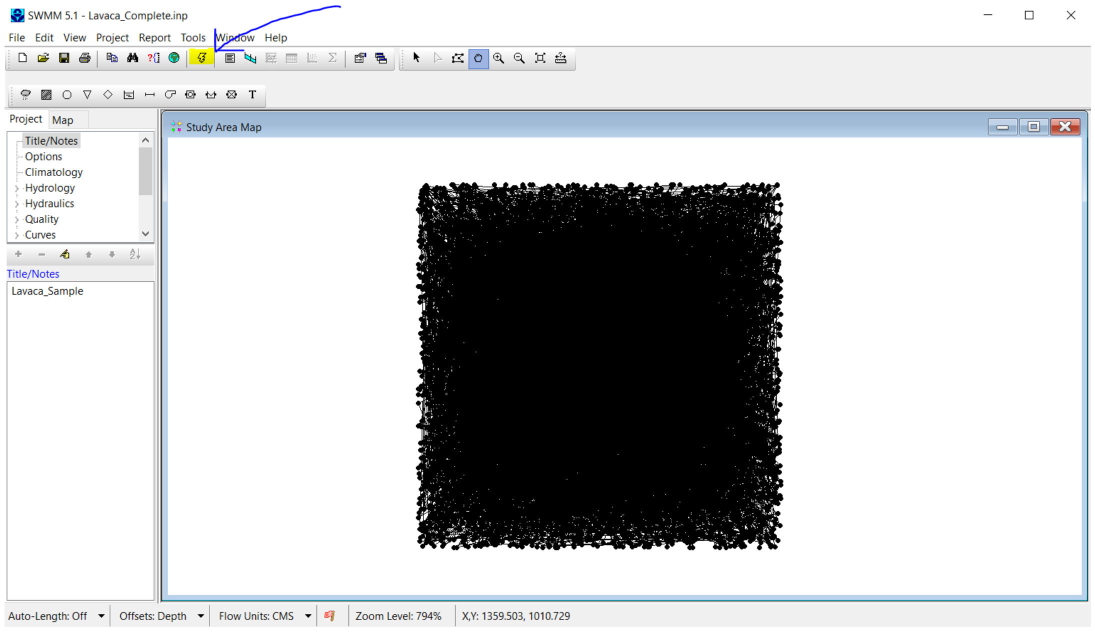
   d. Locate the function create_externalfiles() and variable save_path. Change save_path to reflect the absolute path to the external files.



```
def create_externalfiles(extfile, toFile):
    file = str(extfile) + ".dat"
    save_path = 'C:/Users/edt489/Documents/Conversion_Scripts/LavacaRB'
    completeName = os.path.join(save_path, file)
    with open(completeName, "w") as file1:
        for i in range(len(toFile)):
            file1.write(toFile[i])
    return
```

   e. Run Step3_ Write.py

11) Now we have a TRB_Complete.inp and a bunch of external files. The next thing to do is try to open it in the SWMM5 UI and run it. Ensure that the TRB_Complete.inp and the external files are in the same directory.



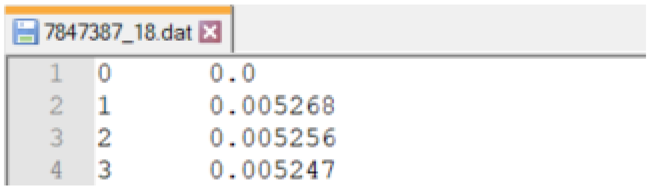
a. The TRB will show up as basically a black square, due to the random coordinate distribution of nodes. Don't worry, this does not matter to the SWMM5 engine. Click the 'Run Simulation' button on the top ribbon.
b. Address any errors.
***As it stands, SWMM5 will not open more than 512 external files. Fixes to this issue are forthcoming.

Notes for Future Work:

Elevation of the nodes - currently there is contradictory information from the *.spt and the elevation_slope.csv

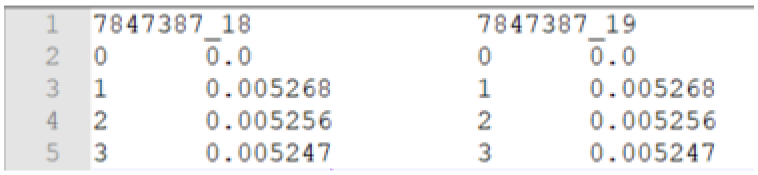
External Files – need to change the SWMM5.1 source code, inflows.c, to accommodate timeseries data in multiple columns in a single external file. This is to side step a Windows 10 OS issue of not being able to open more than 512 files at one time. Example A) is how external timeseries are currently formatted, Example B) is how they should be formatted.

A)


7847387_18.dat

```
0         0.0
1         0.005268
2         0.005256
3         0.005247
```

B)



```
7847387_18                    7847387_19
0         0.0                 0         0.0
1         0.005268            1         0.005268
2         0.005256            2         0.005256
3         0.005247            3         0.005247
```

Additional Information about the input files and scripts used.

SPRNT Input File Data Types and Fields:

- <u>Nodes</u> – connecting points between river sections, locations for lateral inflow
    - Id = COMID_PLACEMENT

        - The COMID on which the node is found is the first part of the node id. After the underscore is the node's placement number within that COMID
        - Some nodes have unique placement numbers indicating that they are a connecting node that exists on a junction between two or more COMIDs.
    - sR = Channel Slope
    - n = Manning's roughness coefficient
    - zR = Elevation of node above sea level
    - hR = reference slope adjustment
    - Coordinates = x and y location data for the node
    - Geometry data - Trapezoidal or Intrinsic
        - Trapezoidal uses Bottomwidth and Slope to define Node Geometry
        - Intrinsic uses Area, Wetted Perimeter, Depth, and Top Width to define Node Geometry
- Boundary Condition – defines the downstream boundary condition
    - Location = the node constituting the outfall of the system
        - *For Texas River Basins there is necessarily only one root, so one B.C.
    - Type = the attribute type of the Dirichlet B.C. (depth, flow, etc)
    - Timeunit = the timeunit used by the timeseries
    - T = time since the simulation began
    - V = Dirichlet B.C. value at the given time
- Segments – river sections defined as being from one upstream node to one downstream node
    - Up = defines the upstream node
    - Down = defines the downstream node
    - Length = defines the length of the segment
- Lateralsources – lateral inflows to interior nodes containing the hydrologic contributions
    - Location = the node id into which this lateral inflow contributes
    - Timeunit = the timeunit used in the timeseries
    - T = time since the simulation began
    - V = flow rate in m^3/s
- Junction – node complexes that define where multiple river segments come together. All nodes within junctions are geographically collocated
    - Down = the id of the node receiving the flows from the upstream branch(es)
    - Up1(2) = the id(s) of the upstream nodes contributing flow
    - Coeff1(2) = the coefficient dictating how much of the downstream flow is contributed by Up1, 2 respectively. The sum of the coefficients is 1.
- Qsources – defines the upstream boundary conditions
    - Location = the node representing the far upstream branch in the network
    - Timeunit = the timeunit used in the timeseries
    - T = time since the simulation began
    - V = flow rate in m^3/s

Details of Read-Process-Write workflow

STEP1_Read.py

This script is the read portion of the read-process-write workflow. This script was used to speed up the workflow used in the conversion from SPRNT to SWMM input files. The SPRNT data is contained in the Texas River Basin (TRB.spt) and "elevation_slope.csv", two quite large files. Separating the reading of the input files from the processing stage helps mitigate the time taken for each run of the processing script, especially considering that the database only needs to be saved once (i.e. the input file isn't changing between runs). Container databases are opened, with the contents of each file being written to it in a line-wise fashion. These databases are more speedily accessed by the processing script.

<u>Components</u>:

Import shelve2

Create empty databases for the TRB.spt and NHDplus Elevation/Slope data

Open and read TRB.spt and NHDPlus files

Copy linewise the opened files into the databases [‘key’]

Close the databases

STEP2_Process.py

This is the processing script of the read-process-write workflow. It handles the conversions of the data structure contained within SPRNT (organized into nodes, segments, junctions, lateral sources, and qsources) into the data structure required by SWMM (organized into nodes, conduits, cross-sections, and time-series).

<u>Components</u>:

Import numpy, random, re, shelve2

Open the TRB and NHDPlus databases and read their contents into variables

<u>Read_sprnt_for_counters()</u> - This function inspects the contents of the SPRNT input file and creates arrays for each of the data types that are sized correctly. This allows us to use np.arrays rather than 2D lists. The second dimension of the arrays is hard-coded to reflect that each data type in SPRNT contains a different number of parameters.

*The hardcoded second dimension of these arrays is TRB dependent. For example, the intrinsic geometry of the Lavaca RB is contained within a 2 column array. Verses the trapezoidal geometry of the Guadalupe – San Antonio RB whose geometry is contained within the nodes_name array. Additionally, the columns of the lateralsources and qsources arrays are sized depending on the length of the timeseries for the given TRB’s simulation.

<u>AY_HEC2_transform()</u> – This function transforms the node geometry data from the Area – Depth – Wetted Perimeter – Top Width tabular relationship used by SPRNT into the Station – Elevation HEC2 structure required by SWMM. The transformation works by using the Depth increments as the Elevation coordinate, and determining the Station from the change in the Top Width.

*Assumptions made in this transformation are that the channel is a concave, symmetrical function. I.e. Station cannot decrease between elevation points.

<u>Transform_check()</u> – This function ensures that the AY_HEC2_transform() conserves the Area and Wetted Perimeter quantities between the two channel geometry structures. Area and Wetted Perimeter are not used in the transformation itself, so they make for adequate validation data.

Area check → $$\left(Area_{max} - \int_{station\_min}^{station\_max} elevation\_max + \int_{station\_min}^{station\_max} elevation_i\right) \Big/ Area_{max}$$

*This normalized difference should be within a 0.1% tolerance representing a conserved channel area during the transformation

Perimeter check →

$$\left(WP_{max} - \sum\left((station_{i+1} - station_i)^2 + (elevation_{i+1} - elevation_i)^2 \right)^{\frac{1}{2}} \right) \Big/ WP_{max}$$

*This normalized difference should be within a 0.1% tolerance representing a conserved wetted perimeter during the transformation

AY_main() – This function controls the AY_HEC2_transform() and transform_check() functions. Each time an “intrinsic” node geometry definition is found within the TRB.spt, the Area – Depth – Wetted Perimeter – Top Width data is stored. Then the AY_HEC2_transform() function is called to convert the geometry data into the correct format and populate the node geometry numpy array. Transform_check() ensures the transformation conserved the Area and Wetted Perimeter.

Read_sprnt_content() - This function populates the various data type arrays with the values parsed from the SPRNT input file database. The function loops through each line in the database; when it finds the keyword that indicates a certain data type, the corresponding values are added to the np.array.

*Each data type has hardcoded token indices related to where each parameter occurs in the data type definition, i.e. the node id token = 0. These token indices are a function of the structure of each TRB.spt.

Read_NHDPlus() – This function parses the NHDplus elevation database into a set of lists that contain the COMID and max/min elevation.

*The NHDPlus COMID name and elevation information is needed in order to give the nodes elevation information. I’m unclear whether this is actually needed.

Merge_comIDs() - The NHDPlus elevation database contains the elevations for every comID in Texas. This function makes a list of each comID that is found in the SPRNT input file and compares that to the list of comIDs in the NDHPlus database. If the comID exists in the domain, its attributes are added to a more exclusive set of lists.

Interpolate_node_elevation() - Elevations for each node are required for the SWMM input file, but absent from the SPRNT data. They must be gathered from the NHDPlus database. Additional complexity is added by the fact that the NHDPlus data only contains elevations for the endpoints of the comID, meaning there are numerous intermediate nodes that do not have directly associated elevation data. The elevation for an intermediate node can be ascertained by determining the comID it exist on, the (thankfully consistent) segment length for that comID, and how many nodes are downstream of the node under inspection. From this, the linear interpolation equation can be used to calculate the elevation for each node that exists on a comID.

*The elevation of the root node is printed by this function. In SWMM the root node is put into a separate OUTFALLS category.

Open a database for the numpy arrays containing the SPRNT data. For each array, assign a [‘key’] from which this array can be retrieved in the write portion of the workflow. Close the database.

STEP3_Write.py

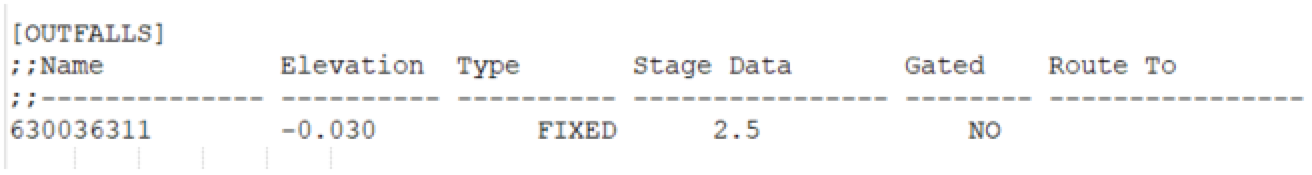
This script constitutes the write step of the read-process-write workflow for converting SPRNT data into SWMM inputfile format. The database result from the processing script is read into variables that will be written into a text file configured to mimic the layout of a SWMM .inp file

Components:

Import shelve2, random, numpy, os.path

<u>Read_SWMM_contents(TRB_Sample.inp)</u> - The general format of a SWMM input file is adopted from a sample file. The sample file is opened and read linewise into a variable

* Outfall node info is pre-coded and derived from the TRB.spt and Process script print statements



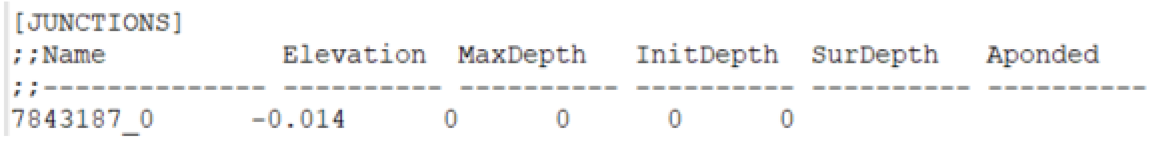
```
[OUTFALLS]
;;Name           Elevation  Type       Stage Data       Gated    Route To
;;-------------- ---------- ---------- ---------------- -------- ----------------
630036311        -0.030          FIXED      2.5              NO
```

<u>Create_inputfile(TRB_Complete.inp)</u> - This function combines the processed data from the np.arrays (organized into the data types from SPRNT) and inserts the data line by line into the SWMM inputfile template. When a header is found in the swmm_contents, the lines beneath that header are populated with the appropriate information from the np.arrays. All of the tokens for which no SPRNT corollary exists are be populated with a default value.

*Where ambiguous, values for conduit/xsection dimensions are defined by the values at the upstream SPRNT node.

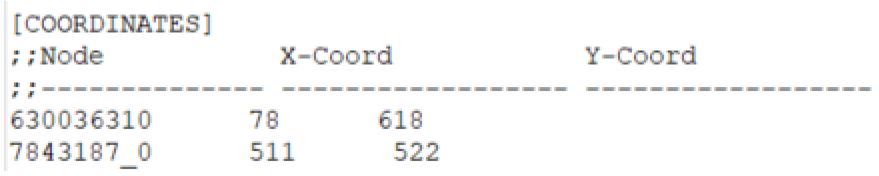
A preprocessing set of for-loops fixes the issue that the same "node" in SPRNT is assigned two different names depending on which comID it appears. Removing the comID prefix makes it so that the same location in space only has one name. This also has the effect of eliminating the need for the Junctions definitions from SPRNT, as the Junctions simply acted to connect the same location in space with each identifier. A corresponding ID trimming must be done wherever nodeID's exist, namely, Segments, Lateralsources, and Qsources.

[JUNCTIONS] – corresponding to the nodes in the SPRNT file. The junction name and elevation are populated. Max depth is a default of 10m (might want to change this to pulling from the A-Y depth). InitDepth, SurDepth, and Aponded are all 0.



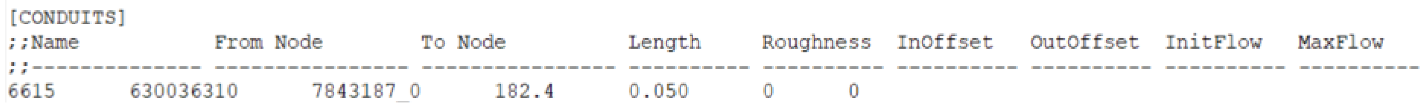
```
[JUNCTIONS]
;;Name           Elevation  MaxDepth   InitDepth  SurDepth   Aponded
;;-------------- ---------- ---------- ---------- ---------- ----------
7843187_0      -0.014      0       0       0       0
```

[COORDINATES] – SPRNT, as a GUI-less software, does not have any express need for coordinates. However, SWMM does need a coordinate value in order to open the .inp in its GUI, so random coordinates are given to each node.



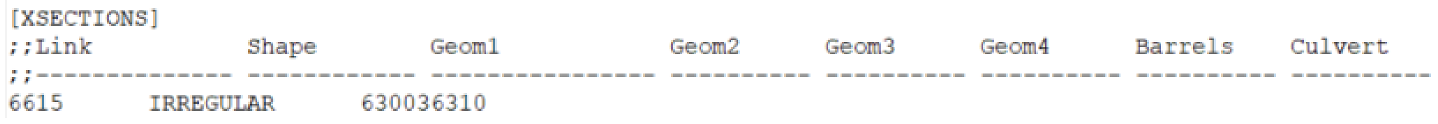
```
[COORDINATES]
;;Node           X-Coord            Y-Coord
;;-------------- ------------------ ------------------
630036310      78      618
7843187_0      511      522
```

[CONDUITS] – corresponding to the segments in the SPRNT file. These conduits will all be open-channels and given arbitrary names. The Manning’s n coefficient info is contained within the SPRNT nodes, but needed in the SWMM conduits; in this instance the upstream node’s roughness value is used. InOffset, OutOffset are given a 0 value; InitFlow, MaxFlow are left blank.



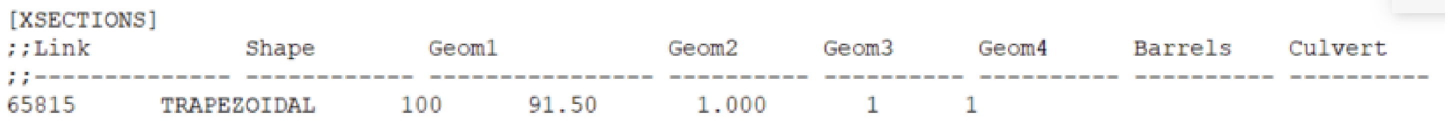
```
[CONDUITS]
;;Name           From Node        To Node          Length     Roughness  InOffset   OutOffset  InitFlow   MaxFlow
;;-------------- ---------------- ---------------- ---------- ---------- ---------- ---------- ---------- ----------
6615      630036310      7843187_0      182.4      0.050      0      0
```

[XSECTIONS] – cross-section data for each conduit is based on the geometry of the upstream node. For irregular channels such as in the Lavaca RB, the TRANSECT name (discussed below) is the name of the upstream node. Link refers to the conduit name the cross-section describes.



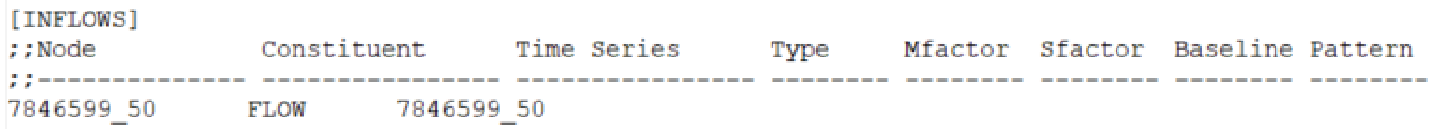
```
[XSECTIONS]
;;Link           Shape        Geom1            Geom2      Geom3      Geom4      Barrels    Culvert
;;-------------- ------------ ---------------- ---------- ---------- ---------- ---------- ----------
6615      IRREGULAR      630036310
```

For trapezoidal channels such as in the Guadalupe – San Antonio RB, the geometries are Max Height (arbitrarily defined as 100m), the Bottom Width, and the Side Slope.



```
[XSECTIONS]
;;Link           Shape        Geom1            Geom2      Geom3      Geom4      Barrels    Culvert
;;-------------- ------------ ---------------- ---------- ---------- ---------- ---------- ----------
65815      TRAPEZOIDAL      100      91.50      1.000      1      1
```

[INFLOWS] – this sections contains the information about which nodes are receiving Lateralsources or Qsources inflow. Each timeseries is unique, so thusly named after the node it feeds into. The Constituent and Type are both 'FLOW', whereas the flow factors are given a default value of '1.0'. However, these extra parameters are not needed for those INFLOWS that will ultimately be stored as external files (i.e. lateralsources).

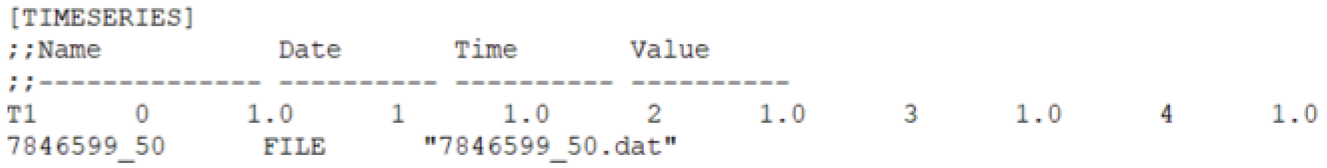


```
[INFLOWS]
;;Node           Constituent      Time Series      Type     Mfactor  Sfactor  Baseline Pattern
;;-------------- ---------------- ---------------- -------- -------- -------- -------- --------
7846599_50      FLOW      7846599_50
```

[TIMESERIES] – this section contains the timeseries values for the INFLOWS. Each timeseries is named after the node associated with the INFLOW, and appears in one of two formats.

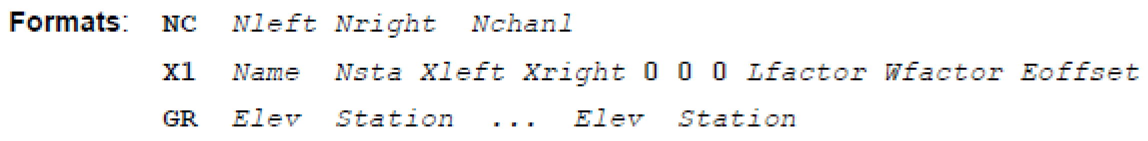
For Qsources, each line is populated as name, time_0, value_0, …. , time_N, value_N.

For Lateralsources, the timeseries data is contained in an external file named after the INFLOW node as well (i.e. "7846599_50.dat")



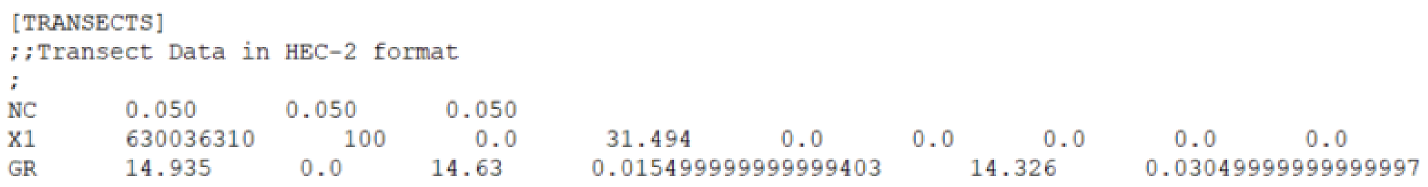
```
[TIMESERIES]
;;Name           Date       Time       Value
;;-------------- ---------- ---------- ----------
T1       0       1.0       1       1.0       2       1.0       3       1.0       4       1.0
7846599_50      FILE      "7846599_50.dat"
```

[TRANSECTS] – Needed in those TRB's with irregular (Area – Depth) defined channel cross-sectional geometries.



```
Formats:  NC  Nleft Nright  Nchanl
          X1  Name  Nsta Xleft Xright 0 0 0 Lfactor Wfactor Eoffset
          GR  Elev  Station  ...  Elev  Station
```

Each conduit has a unique transect defined in the above three line format. The Manning's coefficients are defined in the upstream node array and are constant across the channel. The name of the transect is also the upstream node, the number of stations is equal to the length of the station array for each node, the Xleft and Xright endpoints are the min and max station, respectively. All other factors are set to ‘0.0’. The actual elevation – station pairings are included in the GR line.



```
[TRANSECTS]
;;Transect Data in HEC-2 format
;
NC      0.050      0.050      0.050
X1      630036310      100      0.0      31.494      0.0      0.0      0.0      0.0      0.0
GR      14.935      0.0      14.63      0.015499999999999403      14.326      0.030499999999999997
```

Create externalfiles() – this function is responsible for writing all of the timeseries data from the Lateralsources into external files readable by SWMM. It requires the absolute path for where the files will be written.