ARNOLD Workflow

1. Conduct rdsmall QA/QC, particularly:
    a. TWN_LR
    b. AOTMILES
    c. Topology
2. Set up a new ARNOLD_YYYY folder in V:\Projects\Shared\Mapping\ARNOLD_AllRoadNetwork\
    a. Copy the following script folders from the previous year's folder: ARNOLD_AutomationScripts and Scripts
    b. Create a new File Geodatabase (recommended to make a new .gdb for each draft)
    c. Some of the .lyr files from the previous year's folder will also be helpful (especially calibration point symbolization and labeling)
3. Update some scripts to reflect the new year and gdb output path (the "version"), these lines can be found near the beginning of the following scripts.
    a. ARNOLD.py
    b. Flip_RDS_arcs_InMemory.py (this script is now called from ARNOLD.py)
    c. ARNOLD_Calpts.py
        i. In addition to year and version, also update the path to the closed ARNOLD calpts file from which correctly calibrated calpts can be copied to replace the "problem calpts" in the new draft…as long as those route features have not changed.
    d. ARNOLD_Calibrate.py
    e. ARNOLD_PlanarizeAll.py
    f. ARNOLD_PlanarizePrep.py (this script is called by ARNOLD_PlanarizeAll.py)
4. Run **ARNOLD.py** (includes the call to Flip_RDS_arcs_InMemory.py)
    a. Creates uncalibrated versions of ARNOLD without forks and with open or closed loops
    b. Creates a copy of input rdsmall features with all arcs flipped to match the direction of digitization in the uncalibrated draft of ARNOLD
    c. Now includes Flip_RDS_arcs_InMemory.py, which creates a copy of input rdsmall arcs with all of the arcs flipped in the direction of increasing LRS measures
5. Refresh Python Shell (necessary when using Python 2.7 because of memory issues)
6. Run **ARNOLD_Calpts.py**
    a. Determines the measures for each calpt according to its location along the route and the sum of AOTMILES up to that point
    b. Calpts from "problem routes" are automatically deleted from the output and replaced by calpts from a previous version…as long as those route features have not been changed
7. Run **ARNOLD_Calibrate.py** (includes calls to procedures in ARNOLD_PlanarizePrep.py)
    a. Calibrates the local ARNOLD features (with open loops)
    b. Add ARNOLD attributes (same as final schema)
    c. Add and populate fields that keep track of the open and closed locations of the moving vertices
    d. With one copy: Close loops, creating final "dissolved" ARNOLD for VAMIS

            i. Check to make sure that all route features are increasing with digitized direction. Have to check “Ignore cases where consecutive vertices have the same measure value” so that vertices at the beginning and end of ghost sections (which have the same measures) are not flagged. (This isn’t a problem with planarized features)

    e. Keep another copy to be appended into ETE_LRS prepared by Johnathan
    f. Prepare ETE_LRS for planarization
        i. Add and Populate fields that keep track of the open and closed locations of the moving vertices
    g. Append local ARNOLD features into a renamed copy of ETE_LRS

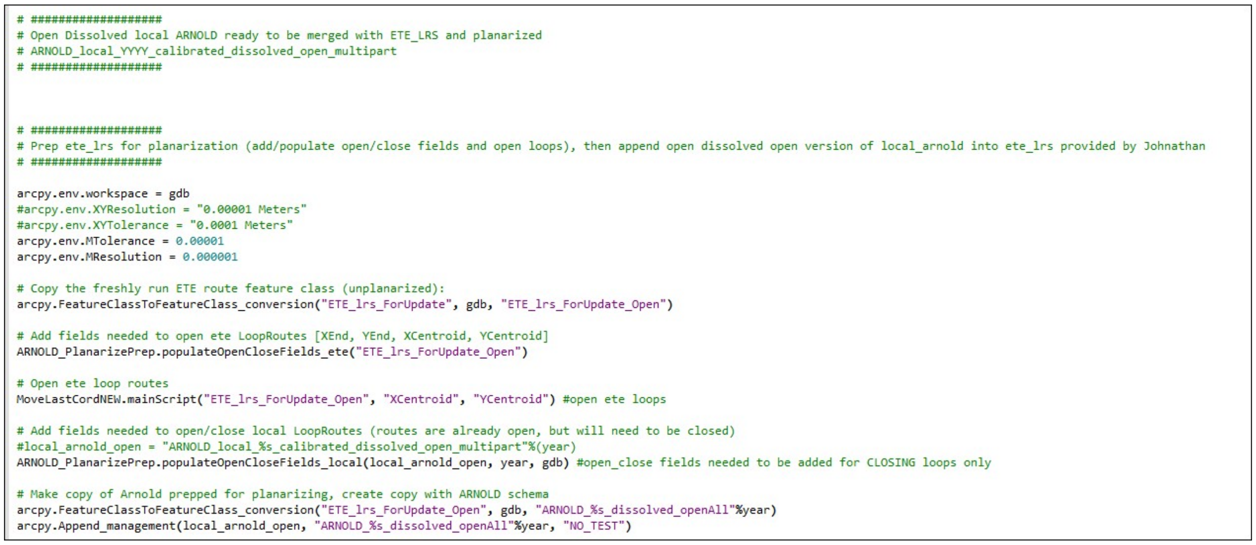
8. Run **ARNOLD_Planarize_All.py** (Details still in script comments)
    a. Use open-loop copies of both lrs_route_ete and local ARNOLD
    b. Planarize and split loops at loop nodes, where they will be closed (eventually)
    c. Close all loops
    d. Re-calculate ARNOLD attributes that can change with planarization (i.e. measure-related fields)
    e. Check to make sure that all planarized route features are increasing with digitized direction. Do NOT check “Ignore cases where consecutive vertices have the same measure value”
9. Create a copy of ARNOLD that meets VAMIS requirements
    a. As dissolved as possible, keeping only routes that have discontinuous mileage between segments (e.g. westbound US-2) as multiple features/records per route.
10. Add and populate Road_Direction attribute for Maureen (Traffic Safety)
    a. Use MasterRouteDefinition Table to obtain the direction for each ETE_LR route
    b. Use Road_Direction.py script to calculate direction of local Arnold routes (use fully dissolved/multipart routes for this step, and transfer directions to planarized ARNOLD using a join on the route codes.

NOTE: Most of the steps above have additional details within the comments written in each of the scripts. These comments generally explain what each section of each script does. See image of code at the bottom of this script.

Possible disruptions to workflow:

1. Incomplete rdsmall QA/QC (depending on the issue, it is often easiest to start from scratch)
2. Forked (non-loop) route(s) found at the end of “Step 3” in ARNOLD.py (script is self checking, and prints out number of forked routes found)
3. If any of the routes with “problem calpts” have been updated since ARNOLD was last created, those calpts will have to be fixed manually instead of simply being replaced from a recent version of ARNOLD.

4. Modification (or addition) of loop geometry so that the offset location of the opening/closing loop vertex still intersects itself. This is more likely to occur with asymmetrical or narrow-necked loops.
    a. Currently there is one ETE_LR route (G-something) that needs to be calibrated manually and left open for planarization, though this specific instance could be added to MoveLastCord.py (this now has a hard-coded fix)



```
# ###################
# Open Dissolved local ARNOLD ready to be merged with ETE_LRS and planarized
# ARNOLD_local_YYYY_calibrated_dissolved_open_multipart
# ###################



# ###################
# Prep ete_lrs for planarization (add/populate open/close fields and open loops), then append open dissolved open version of local_arnold into ete_lrs provided by Johnathan
# ###################

arcpy.env.workspace = gdb
#arcpy.env.XYResolution = "0.00001 Meters"
#arcpy.env.XYTolerance = "0.0001 Meters"
arcpy.env.MTolerance = 0.00001
arcpy.env.MResolution = 0.000001

# Copy the freshly run ETE route feature class (unplanarized):
arcpy.FeatureClassToFeatureClass_conversion("ETE_lrs_ForUpdate", gdb, "ETE_lrs_ForUpdate_Open")

# Add fields needed to open ete LoopRoutes [XEnd, YEnd, XCentroid, YCentroid]
ARNOLD_PlanarizePrep.populateOpenCloseFields_ete("ETE_lrs_ForUpdate_Open")

# Open ete loop routes
MoveLastCordNEW.mainScript("ETE_lrs_ForUpdate_Open", "XCentroid", "YCentroid") #open ete loops

# Add fields needed to open/close local LoopRoutes (routes are already open, but will need to be closed)
#local_arnold_open = "ARNOLD_local_%s_calibrated_dissolved_open_multipart"%(year)
ARNOLD_PlanarizePrep.populateOpenCloseFields_local(local_arnold_open, year, gdb) #open_close fields needed to be added for CLOSING loops only

# Make copy of Arnold prepped for planarizing, create copy with ARNOLD schema
arcpy.FeatureClassToFeatureClass_conversion("ETE_lrs_ForUpdate_Open", gdb, "ARNOLD_%s_dissolved_openAll"%year)
arcpy.Append_management(local_arnold_open, "ARNOLD_%s_dissolved_openAll"%year, "NO_TEST")
```