**Updated Version (V6) / Instructions / Code to create descriptions from polygons 3/20/2023**

**This process is intended to save money while still following MCA and ARM. Specifically:** [**https://rules.mt.gov/gateway/RuleNo.asp?RN=24%2E183%2E1110**](https://rules.mt.gov/gateway/RuleNo.asp?RN=24%2E183%2E1110) **and** [**https://rules.mt.gov/gateway/ruleno.asp?RN=24%2E183%2E1111**](https://rules.mt.gov/gateway/ruleno.asp?RN=24%2E183%2E1111)

**Notes:**

You will have to adjust the variables in the code (step 2) below for your feature class names and for the appropriate output paths. See **Bolded** text in code.

The code snippets need to be copied into the Python Pane (for this version).

I would recommend creating singular districts with this process and then merge the completed datasets together to have one district description with multiple districts within (Commissioner Districts). Alternatively, use your working layers “Definition Query” (as I do in the videos).

The results of this process are intended to be certified by a RLS.

There is a poor quality video of the workflow at:  
 <https://www.youtube.com/watch?v=g9ra6T56Zi4>

The video missed some key steps but does display the workflow (early stages of dev.)

**Process:**

1. Create Polygons for the desired areas. Alternatively, you can create lines directly.
2. Create variables: \*\*\*\****UPDATE VARIABLES FOR NEW DISTRICTS***\*\*\*\*

polygon\_fc = "FinalDraftPrecincts" *# Only needed if converting polygons to lines.*

fc = "All\_2023\_Precincts\_Description"

output\_path = r"C:\Users\tluksha\Desktop\Creating\_Precincts\Creating\_Precincts.gdb"

output\_table = "Complete\_Descriptions"

ExcelFileName = "All\_Precincts.xls"

DistrictName = '"2023 Precinct x-NAME"'

import os *# Do not change this line. It is not a variable.*

1. Convert polygons to lines: (Skip if working with a line FC already)
   1. NOTE: This outputs a feature class titled “Scratch\_PolygonToLine”

arcpy.management.PolygonToLine( in\_features= polygon\_fc, out\_feature\_class= output\_path + “Scratch\_PolygonToLine", neighbor\_option="IDENTIFY\_NEIGHBORS")

1. Separate (split) the lines where any deviation to the description could occur.
2. Join (merge) the lines where they should be continuous but are not.
3. Clear Selection from fc:
   1. Best practice would be to clear within the “Selection” group, In the “Edit” or “Map” tabs versus via code.

arcpy.management.SelectLayerByAttribute(fc, 'CLEAR\_SELECTION')

1. Enter default values:

arcpy.management.CalculateField( in\_table="All\_2023\_Precincts\_Description", field="Area", expression='"Thence,"', expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

arcpy.management.CalculateField( in\_table="All\_2023\_Precincts\_Description", field="Approx", expression='"approximately"', expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

arcpy.management.CalculateField( in\_table="All\_2023\_Precincts\_Description", field="Qualifier", expression='"miles"', expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

arcpy.management.CalculateField( in\_table="All\_2023\_Precincts\_Description", field="Along", expression='"along the"', expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

arcpy.management.CalculateField( in\_table="All\_2023\_Precincts\_Description", field="Factors", expression='"section lines"', expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

arcpy.management.CalculateField( in\_table="All\_2023\_Precincts\_Description", field="ToThe", expression='"to the"', expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

arcpy.management.CalculateField( in\_table="All\_2023\_Precincts\_Description", field="Terminus", expression='"Northeast corner"', expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

arcpy.management.CalculateField( in\_table="All\_2023\_Precincts\_Description", field="OfSection", expression='"of section"', expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

arcpy.management.CalculateField( in\_table=fc, field="T", expression='"T"', expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

arcpy.management.CalculateField( in\_table="All\_2023\_Precincts\_Description", field="Township\_Dir", expression='"S"', expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

arcpy.management.CalculateField( in\_table=fc, field="R", expression='"R"', expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

arcpy.management.CalculateField( in\_table="All\_2023\_Precincts\_Description", field="Range\_Dir", expression='"W"', expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

arcpy.management.CalculateField( in\_table=fc, field="PMM", expression='"PMM"', expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

1. Update “District Name” field: (ONLY if working on one district at a time)

arcpy.management.CalculateField(fc, "District\_Name", DistrictName, "PYTHON3", "", "TEXT", "NO\_ENFORCE\_DOMAINS")

1. If the lines are in appropriate order and NOT NUMBERED, run the following code to insert numbers:

arcpy.management.CalculateField( in\_table="fc", field="ListOrder", expression="autoIncrement()", expression\_type="PYTHON3", code\_block="""rec=0def autoIncrement(): global rec pStart = 3 pInterval = 1 if (rec == 0): rec = pStart else: rec = rec + pInterval return rec""", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

* 1. Manually set “List Order” (if not in order) if needed.
  2. Change the numbers accordingly after the autoincrement() code has been ran.

1. Calculate “Course” field:

arcpy.management.CalculateField( in\_table=fc, field="Course\_Num", expression='"(Course " + str(!ListOrder!) + ")"', expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

* 1. Alternate (to include master list number):

arcpy.management.CalculateField( in\_table=fc, field="Course\_Num", expression='"(Course " + str(!MasterListOrder!) + "-" + str(!ListOrder!) + ")"', expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

1. Calculate “Bearing”

arcpy.AddGeometryAttributes\_management(**fc**, Geometry\_Properties="LINE\_BEARING", Length\_Unit="", Area\_Unit="", Coordinate\_System="PROJCS['NAD\_1983\_StatePlane\_Montana\_FIPS\_2500',GEOGCS['GCS\_North\_American\_1983',DATUM['D\_North\_American\_1983',SPHEROID['GRS\_1980',6378137.0,298.257222101]],PRIMEM['Greenwich',0.0],UNIT['Degree',0.0174532925199433]],PROJECTION['Lambert\_Conformal\_Conic'],PARAMETER['False\_Easting',600000.0],PARAMETER['False\_Northing',0.0],PARAMETER['Central\_Meridian',-109.5],PARAMETER['Standard\_Parallel\_1',45.0],PARAMETER['Standard\_Parallel\_2',49.0],PARAMETER['Latitude\_Of\_Origin',44.25],UNIT['Meter',1.0]]")

1. [Calculate Direction](file:///C:\Users\tluksha\Desktop\Redistricting\Calculate%20Direction%20from%20Bearing%20.txt) (Indentation issues have occurred from MS Word. Specifically with spacing in front of the “if” statement. {See asterisk: \*if (Dir <….}
   1. If Error occurs, Run code from the text file as MS Word continues to crash this (“Calculate Direction from Bearing .txt”).

arcpy.management.CalculateField(in\_table=fc, field="Direction", expression="Reclass(!BEARING!)", expression\_type="PYTHON3", code\_block="""

def Reclass(Dir):

if (Dir < 360 and Dir > 348.75 ):

return "North"

elif (Dir > 0 and Dir < 11.25 ):

return "North"

elif (Dir > 11.25 and Dir < 33.75 ):

return "NNE"

elif (Dir > 33.75 and Dir < 56.25 ):

return "Northeast"

elif (Dir > 56.25 and Dir < 78.75 ):

return "ENE"

elif (Dir > 78.75 and Dir < 101.25 ):

return "East"

elif (Dir > 101.25 and Dir < 123.75 ):

return "ESE"

elif (Dir > 123.75 and Dir < 146.25 ):

return "Southeast"

elif (Dir > 146.25 and Dir < 168.75 ):

return "SSE"

elif (Dir > 168.75 and Dir < 191.25 ):

return "South"

elif (Dir > 191.25 and Dir < 213.75 ):

return "SSW"

elif (Dir > 213.75 and Dir < 236.25 ):

return "Southwest"

elif (Dir > 236.25 and Dir < 258.75 ):

return "WSW"

elif (Dir > 258.75 and Dir < 281.25 ):

return "West"

elif (Dir > 281.25 and Dir < 303.75 ):

return "WNW"

elif (Dir > 303.75 and Dir < 326.25 ):

return "Northwest"

elif (Dir > 326.25 and Dir < 348.75 ):

return "NNW"

else:

return "Error" """, field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

1. Calculate Distance:

arcpy.management.CalculateGeometryAttributes( in\_features=fc, geometry\_property="DistRound LENGTH", length\_unit="MILES\_INT", area\_unit="", coordinate\_system='PROJCS["NAD\_1983\_StatePlane\_Montana\_FIPS\_2500",GEOGCS["GCS\_North\_American\_1983",DATUM["D\_North\_American\_1983",SPHEROID["GRS\_1980",6378137.0,298.257222101]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],PROJECTION["Lambert\_Conformal\_Conic"],PARAMETER["False\_Easting",600000.0],PARAMETER["False\_Northing",0.0],PARAMETER["Central\_Meridian",-109.5],PARAMETER["Standard\_Parallel\_1",45.0],PARAMETER["Standard\_Parallel\_2",49.0],PARAMETER["Latitude\_Of\_Origin",44.25],UNIT["Meter",1.0]]', coordinate\_format="SAME\_AS\_INPUT")

1. Round Distance: NOTE: If error occurs, rerun previous step.

arcpy.management.CalculateField( in\_table=fc, field="DistRound", expression="round(!DistRound!,2)", expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

1. Consider code to remove zeros when rounding presents X.0 or X.00
2. Mile vs Miles

*#Update the Qualifier field with mile or miles*

*arcpy.management.CalculateField(in\_table=fc, field="Qualifier", expression="qualifierMiles(!DistRound!)", expression\_type="PYTHON3", code\_block="""*

*def qualifierMiles(miles):*

*if miles == 1:*

*return 'mile'*

*else:*

*return 'miles'*

*""", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")*

1. Manually complete or modify the wording for the following fields:
   1. Area -Generally beginning or ending components.
   2. Factors -Modify if needed.
   3. ToThe
   4. Terminus -DOUBLE CHECK EACH ONE
   5. Section Number -Numerical only
   6. Township -Numerical only
   7. Range -Numerical only
   8. District Name -Field Calculator is your friend here.
2. Join Concatenation (Step 1a):

arcpy.management.CalculateField( in\_table=fc, field="TR\_concat", expression="\'\'.join(str(i) for i in [!T!, !Township!, !Township\_Dir!, \", \", !R!, !Range!, !Range\_Dir!, \", \" !PMM!, \"; \"] if i)", expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

arcpy.management.CalculateField( in\_table=fc, field="Concatonated", expression="' '.join(str(i) for i in [!Course\_Num!, !Area!, !Direction!, !Approx!, !DistRound!, !Qualifier!, !Along!, !Factors!, !LotNum!, !Subdivision!, !ToThe!, !Terminus!, !OfSection!, !SectionNum!, \",\", !Clarification!, !TR\_concat!] if i)", expression\_type="PYTHON3", code\_block="", field\_type="TEXT", enforce\_domains="NO\_ENFORCE\_DOMAINS")

1. Table to table (Stripping fields), Then to Excel:
   1. The first line of the following code DELETES the previous table (if one exists with the table name set in the variable section above.
   2. Alternatively, you could rename “output\_table”. (I add a numerical if I want to do this)

arcpy.management.Delete(output\_table)

arcpy.conversion.TableToTable( in\_rows=fc, out\_path= output\_path, out\_name=output\_table, where\_clause="", field\_mapping='District\_Name "District Name" true true false 300 Text 0 0,First,#,LegalLineCret,District\_Name,0,300;Concatonated "Concatonated" true true false 2000 Text 0 0,First,#,LegalLineCret,Concatonated,0,2000', config\_keyword="")

#arcpy.management.Delete(ExcelFileName\_Path)

import os

output\_path\_parent = os.path.dirname(output\_path)

ExcelFileName\_Path = output\_path\_parent + """\\""" + ExcelFileName

try:

os.remove(ExcelFileName\_Path)

except:

pass

arcpy.conversion.TableToExcel(

Input\_Table= output\_table,

Output\_Excel\_File=ExcelFileName\_Path,

Use\_field\_alias\_as\_column\_header="ALIAS",

Use\_domain\_and\_subtype\_description="CODE"

)

print(ExcelFileName\_Path)

os.startfile(ExcelFileName\_Path) #This line will open the Excel file

1. Hip Hip Hooray???
2. Consider using/creating topology rules.
3. Turn your lines into Polygons:

arcpy.management.FeatureToPolygon( in\_features=fc, out\_feature\_class=r"C:\Users\tluksha\Desktop\Creating\_Precincts\Creating\_Precincts.gdb\DRAFT\_Precinct\_to\_Polygon", cluster\_tolerance=None, attributes="ATTRIBUTES", label\_features=None)

* 1. Create code to add fields to output as the “Preserve Attributes” option is no longer supported.

1. Set Feature Class variable (fc = fc) before importing code blocks.
2. Set path variable (output\_path=r"C:\Users\tluksha\Desktop\Creating\_Commissioner\_Legal\_Descriptions\Legal\_Lines\_Creation.gdb") before importing code blocks.
3. Future revisions should include a script tool or a major code block to copy/paste into the Python pane.
4. This could be improved by automating the exports to a final feature class, tables, excel, and word document. All within a completed folder and FGDB for the project.
5. I hope to find a way to populate a field based upon the end of a line segment. This would allow automation of the Township and Range values.

UPDATE “TR Concatonated” to allow 25 Characters as compared to the 15 that is currently allowed.