Integrating Google Sheets to an Esri SDE Feature Class

Collaborative Data-Entry for non-GIS Users

Jordan Carmona, GISP
City of McKinney, Texas
jcarmona@mckinneytexas.org
linkedin.com/in/jordancarmona

Wrangling data from non-database sources manually?

- time-consuming
- frustrating
- untenable

Google Sheets

+ Collaborative
+ Real-Time
+ Web-Based

– Cumbersome to Export

Esri SDE

+ Collaborative
+ Real-Time
+ Native GIS

– "Hard" to Script Against?

This presentation demonstrates:

Google Sheet → Esri SDE

House-Keeping

Necessary Libraries

In [ ]:
```
import pandas as pd          #used for ingesting our Google Sheet
import numpy as np           #used for converting to an Esri Table
import arcpy                 #used in the ArcGIS environment
import datetime              #used for timestamps
import time                  #used for timers and sleeping
import os                    #used for making file paths
import tempfile              #used for creating a temporary file location
import shutil                #used for closing our temporary file location
import traceback             #used for error messaging
print("We can run fun code!")
```

Methods

In [ ]:

```
def auto_truncate(val):
    return val[:255]

def write_to_log(content):
    log_time = datetime.datetime.now()
    with open(r"C:\Users\jcarmona\Desktop\SCAUG 2019 PRESENTATION\Presentation_log.txt",
"a") as log_file:
        log_file.write(f"\n{log_time.strftime('%Y-%m-%d %H:%M:%S')} --- {content}")

print("Definition")
```

Timers are Great

In [ ]:
```
starttime = time.time() #goes at the beginning of your code
```

In [ ]:
```
endtime = time.time() - starttime #goes at the end of your code
```

In [ ]:
```
print(f"This process took {endtime} seconds") #these are F-strings
print(f"This process took {round(endtime,2)} seconds") #the "F" is for FUN-ction.
```

In [ ]:
```
print(f"This process took {endtime // 60} minutes and {round((endtime % 60),1)} seconds"
)
```

Code Part I

Wrangling the Google Sheet

Accessing the Google Sheet Programatically

Cleaning the Google Sheet

Accessing the Google Sheet Programatically

CIP Project Tracking

File Edit View Insert Format Data Tools Add-ons Help Last edit was made 7 minutes ago by anonymous

Project Name:	Substandard Utility & Street Rehab (South of Virginia)
Project No.:	CO1705
Assoc. Project No.	

General Project Information

Location:	Various locations (See project description)
Project Type:	Street, Water, Wastewater
Description:	Various replacement of previously identified substandard water/wastewater mains and failing pavement generally located south of Virginia, north of Eldorado, west of SH 5.
City Contact:	Jason Horn

Project Funding Summary

FUNDING START DATE: 4/4/2017

FUNDING SOURCE	PRIOR	FY19 - DEBT	FY20	FY21-FY23	TOTAL
FUND 1 (GENERAL FUND)	$ 470,000.00				$ 470,000.00
FUND 20 (UTILITY)	$ 2,830,000.00				$ 2,830,000.00
					$ -
					$ -
					$ -
TOTAL	$ 3,300,000.00	$ -	$ -	$ -	$ 3,300,000.00

Project Funding Allocation

INITIAL CONTRACT DATE: 4/18/2017

FUNDING COMPONENT	ENC	PRIOR	FY19 - REVENUE	FY20	FY21-FY23	TOTAL
Design	x	$ 327,000.00				$ 327,000.00
Construction Inspection	x	$ 66,000.00				$ 66,000.00
Materials Testing	x	$ 49,950.00				$ 49,950.00
Construction	x	$ 2,857,050.00				$ 2,857,050.00
						$ -
						$ -
						$ -
						$ -
						$ -
TOTAL		$ 3,300,000.00	$ -	$ -	$ -	$ 3,300,000.00
PROJECT FUNDING OVER/UNDER		$ -	$ -	$ -	$ -	$ -
TOTAL ENCUMBERED		$ 3,300,000.00		% ENCUMBERED		100%

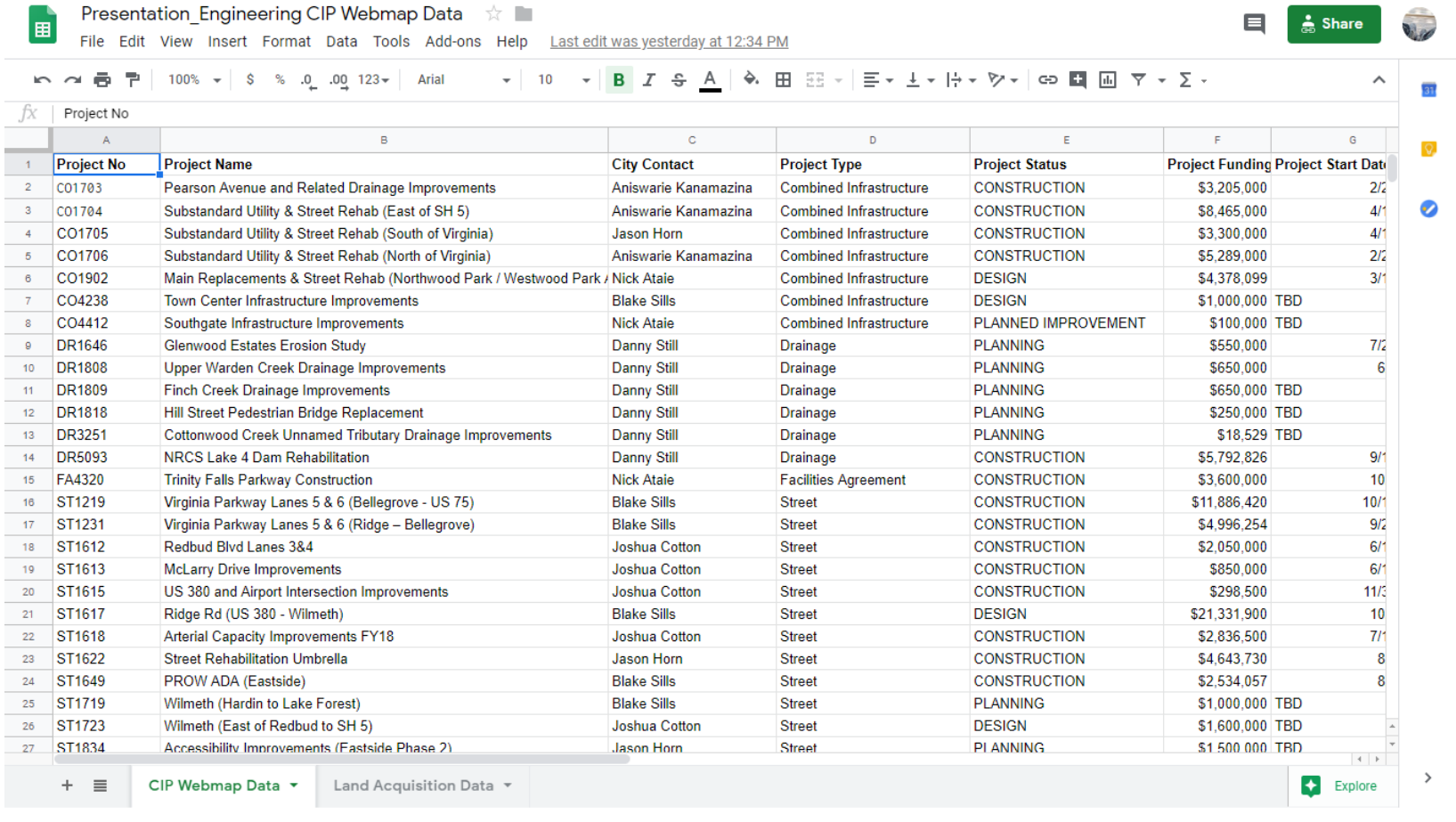
Project Status Summary

NO — Track Schedule (YES or NO)

PROJECT COMPONENT	% COMP	PLANNED START	PLANNED END	CURRENT START	CURRENT END	DAYS	CRITICAL STATUS UPDATES & NEEDS
PLANNING	100%	4/4/17	4/18/17	4/4/17	4/18/17		
DESIGN	100%	4/18/18	11/13/17	4/18/18	2/9/18		Complete
FRANCHISE RELOCATION	100%						AT&T potholed duct bank. No conflict is anticipated
						0	PENDING, CONFLICT, OR CONSTRUCTION
						12	CLEAR
LAND ACQUISITION	100%						
		PARCELS REQUIRED:					
		PARCELS OBTAINED:					
BIDDING & AWARD	100%	7/1/18	7/16/18	7/1/18	7/16/18		Camino construction low bidder and approved by council
CONSTRUCTION	70%	10/1/18	7/5/19	10/1/18	7/18/19		Preconstruction meeting 9/26 with construction scheduled to start 10/1

PROJECT SCHEDULE DELAY SUMMARY (CALENDAR DAYS)

UNALLOC.	GENERAL CONSULTANT DELAY	DESIGN CHANGES	SURVEY GEOTECH	3RD PARTY COORD.	CITY (OTHER)	LAND ACQ.	UTIL. RELOC.	CONST.	CITY (PM)	TOTAL
0										0



Presentation_Engineering CIP Webmap Data

File Edit View Insert Format Data Tools Add-ons Help Last edit was yesterday at 12:34 PM

Share

Project No

	A	B	C	D	E	F	G
1	Project No	Project Name	City Contact	Project Type	Project Status	Project Funding	Project Start Dat
2	CO1703	Pearson Avenue and Related Drainage Improvements	Aniswarie Kanamazina	Combined Infrastructure	CONSTRUCTION	$3,205,000	2/2
3	CO1704	Substandard Utility & Street Rehab (East of SH 5)	Aniswarie Kanamazina	Combined Infrastructure	CONSTRUCTION	$8,465,000	4/1
4	CO1705	Substandard Utility & Street Rehab (South of Virginia)	Jason Horn	Combined Infrastructure	CONSTRUCTION	$3,300,000	4/1
5	CO1706	Substandard Utility & Street Rehab (North of Virginia)	Aniswarie Kanamazina	Combined Infrastructure	CONSTRUCTION	$5,289,000	2/2
6	CO1902	Main Replacements & Street Rehab (Northwood Park / Westwood Park /	Nick Ataie	Combined Infrastructure	DESIGN	$4,378,099	3/1
7	CO4238	Town Center Infrastructure Improvements	Blake Sills	Combined Infrastructure	DESIGN	$1,000,000	TBD
8	CO4412	Southgate Infrastructure Improvements	Nick Ataie	Combined Infrastructure	PLANNED IMPROVEMENT	$100,000	TBD
9	DR1646	Glenwood Estates Erosion Study	Danny Still	Drainage	PLANNING	$550,000	7/2
10	DR1808	Upper Warden Creek Drainage Improvements	Danny Still	Drainage	PLANNING	$650,000	6
11	DR1809	Finch Creek Drainage Improvements	Danny Still	Drainage	PLANNING	$650,000	TBD
12	DR1818	Hill Street Pedestrian Bridge Replacement	Danny Still	Drainage	PLANNING	$250,000	TBD
13	DR3251	Cottonwood Creek Unnamed Tributary Drainage Improvements	Danny Still	Drainage	PLANNING	$18,529	TBD
14	DR5093	NRCS Lake 4 Dam Rehabilitation	Danny Still	Drainage	CONSTRUCTION	$5,792,826	9/1
15	FA4320	Trinity Falls Parkway Construction	Nick Ataie	Facilities Agreement	CONSTRUCTION	$3,600,000	10
16	ST1219	Virginia Parkway Lanes 5 & 6 (Bellegrove - US 75)	Blake Sills	Street	CONSTRUCTION	$11,886,420	10/1
17	ST1231	Virginia Parkway Lanes 5 & 6 (Ridge – Bellegrove)	Blake Sills	Street	CONSTRUCTION	$4,996,254	9/2
18	ST1612	Redbud Blvd Lanes 3&4	Joshua Cotton	Street	CONSTRUCTION	$2,050,000	6/1
19	ST1613	McLarry Drive Improvements	Joshua Cotton	Street	CONSTRUCTION	$850,000	6/1
20	ST1615	US 380 and Airport Intersection Improvements	Joshua Cotton	Street	CONSTRUCTION	$298,500	11/3
21	ST1617	Ridge Rd (US 380 - Wilmeth)	Blake Sills	Street	DESIGN	$21,331,900	10
22	ST1618	Arterial Capacity Improvements FY18	Joshua Cotton	Street	CONSTRUCTION	$2,836,500	7/1
23	ST1622	Street Rehabilitation Umbrella	Jason Horn	Street	CONSTRUCTION	$4,643,730	8
24	ST1649	PROW ADA (Eastside)	Blake Sills	Street	CONSTRUCTION	$2,534,057	8
25	ST1719	Wilmeth (Hardin to Lake Forest)	Blake Sills	Street	PLANNING	$1,000,000	TBD
26	ST1723	Wilmeth (East of Redbud to SH 5)	Joshua Cotton	Street	DESIGN	$1,600,000	TBD
27	ST1834	Accessibility Improvements (Eastside Phase 2)	Jason Horn	Street	PLANNING	$1,500,000	TBD

CIP Webmap Data | Land Acquisition Data | Explore

Normal Google URL:

https://docs.google.com/spreadsheets/d/1QB1pZC
ObWNQW5FjY/edit#gid=0

The magic is right at the end:

/export?format=csv

It is now readable using Pandas!

We'll use the following command: `pd.read_csv()`

This can ingest a file location or URL

In [ ]:
```
CIP_sheet = "https://docs.google.com/spreadsheets/d/1QB1pZQ3CN45ZYS2sc4LnuoOaImL5lQsP-ObWNQW5FjY/export?format=csv"
dataframe = pd.read_csv(CIP_sheet)
print(dataframe)
```

Cleaning the Google Sheet

A Google Sheet is not a database.
While some input standards can be enforced,
more often, they aren't.

- `Date` fields contain TBD entries
- Funding column has $'s
- Project Type needs to be all-caps

Pandas allows you to chain commands using the `.` notation

We'll use Pandas `astype()`, `replace()`, and `upper()` functions

In [ ]:

```
df_dirty = dataframe.copy()
df_dirty = df_dirty[['Project Funding','Project Start Date', 'Estimated Project Completi
on Date','Project Type']]
print(df_dirty)
```

In [ ]:

```
df_clean = df_dirty.copy()

df_clean['Project Funding'] = df_clean['Project Funding'].replace('[\$,]', '', regex=Tru
e).astype(float)
df_clean['Project Start Date'] = df_clean['Project Start Date'].replace('[TBD]', '', reg
ex=True)
df_clean['Estimated Project Completion Date'] = df_clean['Estimated Project Completion D
ate'].replace('[TBD]', '', regex=True)
df_clean['Project Type'] = df_clean['Project Type'].astype(str).str.upper()
print(df_clean.dtypes)
print(df_clean)
```

The data are now clean and ready to be taken inside the ArcGIS Environment

Let's take a look at what we've done so far!

In [ ]:

```
import pandas as pd
print("Pandas at {}".format(pd.__version__))
import numpy as np
print("Numpy at {}".format(np.version.full_version))
import arcpy
print("ArcPy imported successfully.")
import datetime, time
import os, shutil, tempfile
import traceback

def auto_truncate(val):
        return val[:255]

def write_to_log(content):
        log_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        with open(r"C:\Users\jcarmona\Desktop\SCAUG 2019 PRESENTATION\CIP_GoogleSheet_lo
g.txt", "a") as log_file:
                log_file.write(f"\n{log_time} --- {content}")

starttime = time.time()
```

In [ ]:

```
CIP_sheet = "https://docs.google.com/spreadsheets/d/1QB1pZQ3CN45ZYS2sc4LnuoOaImL5lQsP-ObWNQW5FjY/export?format=csv"

print("Reading and cleaning CIP Google Sheet.")
df = pd.read_csv(CIP_sheet, converters ={'Project Description': auto_truncate, 'Project Notes': auto_truncate})

df['Project Funding'] = df['Project Funding'].replace('[\$,]', '', regex=True).astype(float)
df['Project Start Date'] = df['Project Start Date'].replace('[TBD]', '', regex=True).astype(str)
df['Estimated Project Completion Date'] = df['Estimated Project Completion Date'].replace('[TBD]', '', regex=True).astype(str)
df['Project Type'] = df['Project Type'].astype(str).str.upper()

print(df.head(5))
```

Code Part II

Sheet to Table 2-Step

Converting Pandas Dataframe to NumPy Array

Numpy Array to Esri Table

Table Additions and Calculations

Converting Pandas Dataframe to NumPy Array

Currently, there is not native support for Pandas dataframes within Desktop ArcGIS

We'll need to have an intermediate step using NumPy which is supported

Broadly, we will turn our records into a NumPy `array` using `np.array()` and `np.rec.fromrecords()`

This allows us to plug in our dataframe values with this chain command `dataframe.values` , using the following:
`np.array(np.rec.fromrecords(dataframe.values))`

Then we'll gather our column names into a `list` , with this:
`dataframe.dtypes.index.tolist()`

And assign the `list` of column names into the previous `array` as a `tuple`

This process takes 3 Lines of Code:

In [ ]:
```
fromPandas_array = np.array(np.rec.fromrecords(df.values))
fromPandas_columns = df.dtypes.index.tolist()
fromPandas_array.dtype.names = tuple(fromPandas_columns)
print(f"These are our column names: \n{fromPandas_columns}")
print()
print(f"This is an array: \n{fromPandas_array}")
```

NumPy Array to Esri Table

With an `array` in hand, let's push it into a temporary table

We have two workspace options for this:

- Scratch GDB
- In-memory

In [ ]:
```
disk_table = r"%scratchGDB%\CIP"
memory_table = r"in_memory\CIP"
```

Now we'll run an ArcPy tool from the Data Access module

```
NumPyArrayToTable (in_array, out_table)
```

This will accept our `fromPandas_array` as input, and will output to our table location

In [ ]:

```
arcpy.da.NumPyArrayToTable(fromPandas_array, memory_table)
print("Transferred to memory")
arcpy.da.NumPyArrayToTable(fromPandas_array, disk_table)
print("Transferred to disk")
```

In [ ]:

```
if arcpy.Exists(disk_table):
        print("Previous CIP Google Sheet Table found in Scratch. Deleting now.")
        arcpy.Delete_management(disk_table)

if arcpy.Exists(memory_table):
        print("Previous CIP Google Sheet Table found in Memory. Deleting now.")
        arcpy.Delete_management(memory_table)
```

In [ ]:

```
arcpy.da.NumPyArrayToTable(fromPandas_array, memory_table)
print("Transferred to memory")
arcpy.da.NumPyArrayToTable(fromPandas_array, disk_table)
print("Transferred to disk")
```

In [ ]:

```
field_names = [f.name for f in arcpy.ListFields(memory_table)]
field_types = [f.type for f in arcpy.ListFields(disk_table)]
print(f"Here are some field names: \n{field_names} \n\nHere are some field types: \n{fie
ld_types}")
```

Table Additions and Calculations

This next bit is something of a work-around

We'll need to convert our dates and also run a new general status rule

A quick call to:

```
arcpy.AddField_management (in_table, field_name, field_type, ...)
```

```
arcpy.AddField_management(googletable, "StartDate", "DATE")
arcpy.AddField_management(googletable, "CompleteDate", "DATE")
arcpy.AddField_management(googletable, "GeneralStatus", "TEXT", field_length=50)
```

Next we'll call:

```
arcpy.CalculateField_management (in_table, field, expression,
{expression_type}, {code_block})
```

```
arcpy.CalculateField_management(in_table, "Field_Name",
                                "python_method(!Field_Name!)",
                                "PYTHON3", code_block)
```

The optional parameter `codeblock` requires some further explanation

This is our `date_codeblock`

```
def skip_nulls(field):
    if field != '':
        return field
    else:
        return
```

We'll need to wrap it in """ to feed it into the ArcPy parameter

```
date_codeblock = """
def skip_nulls(field):
    if field != '':
        return field
    else:
        return"""
```

This is our `status_codeblock`

```
def general_status(field):
    if field in ["PLANNING", "PLANNED IMPROVEMENT"]:
        return "PLANNED IMPROVEMENT"
    elif field in ["DESIGN", "BIDDING", "LAND ACQUISITION", "FRANCHISE RELOCATION"]:
        return "DESIGN"
    elif field == "CONSTRUCTION":
        return "CONSTRUCTION"
    elif field == "COMPLETE":
        return "COMPLETE"
    else:
        return
```

We've made more progress!

In [ ]:

```
input_array = np.array(np.rec.fromrecords(df.values)) #create numpy array from pandas da
taframe
col_names = df.dtypes.index.tolist() #grab column names from pandas dataframe as a list
input_array.dtype.names = tuple(col_names) #set column names in numpy array

#googletable = r'%scratchGDB%\CIP_Google_Sheet_Table'
googletable = r'in_memory\CIP_Google_Sheet_Table'

#check needed if loop continues past 1st iteration, OR if using ScratchGDB
if arcpy.Exists(googletable):
        print("Previous CIP Google Sheet Table found. Deleting now.")
        arcpy.Delete_management(googletable)
arcpy.da.NumPyArrayToTable(input_array, googletable)
print("Converting CIP Google Sheet from CSV to Table in memory.")
print("Adding new fields to CIP Google Sheet Table.")

arcpy.AddField_management(googletable, "StartDate", "DATE")
arcpy.AddField_management(googletable, "CompleteDate", "DATE")
arcpy.AddField_management(googletable, "GeneralStatus", "TEXT", field_length=50)
```

In [ ]:

```
print("Calculating new date fields in CIP Google Sheet Table.")
date_codeblock = """def skip_nulls(field):
        if field != '':
                return field
        else:
                return"""
arcpy.CalculateField_management(googletable, "StartDate", "skip_nulls(!Project_Start_Dat
e!)",
                                "PYTHON3", date_codeblock)
arcpy.CalculateField_management(googletable, "CompleteDate", "skip_nulls(!Estimated_Proj
ect_Completion_Date!)",
                                "PYTHON3", date_codeblock)

print("Calculating general status field in CIP Google Sheet Table.")
status_codeblock = """def general_status(field):
        if field in ["PLANNING", "PLANNED IMPROVEMENT"]:
                return "PLANNED IMPROVEMENT"
        elif field in ["DESIGN", "BIDDING", "LAND ACQUISITION", "FRANCHISE RELOCATION"]:
                return "DESIGN"
        elif field == "CONSTRUCTION":
                return "CONSTRUCTION"
        elif field == "COMPLETE":
                return "COMPLETE"
        else:
                return"""
arcpy.CalculateField_management(googletable, "GeneralStatus", "general_status(!Project_S
tatus!)",
                                "PYTHON3", status_codeblock)
```

Code Part III

Managing Your Chuck Wagon

Making SDE Connections

Checking Versions

New Item Maintenance

Making SDE Connections

Let's make sure that we connect to our SDE

We'll create a fresh path each time using `tempfile.mkdtemp()` and assign it to a variable

In [ ]:
```
sdeTempPath = tempfile.mkdtemp()
```

Now we'll call

```
arcpy.CreateDatabaseConnection_management (out_folder_path, out_name, database_platform, instance, ...)
```

In [ ]:
```
arcpy.CreateDatabaseConnection_management(sdeTempPath, "Editor.sde",
                                          "SQL_SERVER", "ITT701",
                                          account_authentication = "OPERATING_SYSTEM_AUT
H")
```

In [ ]:
```
EditSDE = os.path.join(sdeTempPath, "Editor.sde")
print(EditSDE)
```

Checking Versions

Let's assume we're being good and not editing the default instance

We'll need to do the following:

- check if our version exists
- make it if it doesn't

We'll iterate through the `arcpy.da.ListVersions()` function
Adding each version name to `sdeVersionLIST`
If we find our version, we'll check to make sure the description is filled out

In [ ]:
```
#psst Jordan don't run this. It breaks. xoxo Jordan
sdeVersionLIST = []
for version in arcpy.da.ListVersions(EditSDE):
    sdeVersionName = version.name.split(".")[0]
    sdeVersionLIST.append(sdeVersionName)
    if sdeVersionName == '"MCKINNEY\\JCARMONA"':
        if version.description == "":
            print("No description found. Fixing.")
            arcpy.AlterVersion_management(EditSDE, '"MCKINNEY\JCARMONA".CARMONA',
                                          description = "jcarmona@mckinneytexas.org | ex
t 7422")
```

Now we have a `list` of all the versions

We'll use `not in` to find our missing version

In [ ]:
```
if '"MCKINNEY\\JCARMONA"' not in sdeVersionLIST:
    print("\tCARMONA version not found, creating now.")
    arcpy.CreateVersion_management(EditSDE,"sde.DEFAULT" ,"CARMONA", "PUBLIC")
    arcpy.AlterVersion_management(EditSDE, '"MCKINNEY\JCARMONA".CARMONA',
                                  description = "jcarmona@mckinneytexas.org | ext 7422")
    print("Version created.")
else:
    print("\tCARMONA version exists, connecting now.")
    arcpy.CreateDatabaseConnection_management(sdeTempPath, "CARMONA.sde", "SQL_SERVER",
                                              instance, "OPERATING_SYSTEM_AUTH",
                                              version = '"MCKINNEY\JCARMONA".CARMONA')
```

New Item Maintenance

Sometimes something new crops up

If our goal is a set it / forget it solution
We need to add in a mechanism to alert us

We'll make a `list` to store our CIP project number codes from the Google Sheet

A `cursor` will help us create a second `list` of CIP projects from our SDE

Then we'll check them against each other to create a final `list` for inclusion in our alert

In [ ]:

```
sde_CIP1923 = os.path.join(EditSDE, "SDE.DBO.DemoSection_CIP1923")
list_googleprojects = df['Project No'].tolist()
list_sde = []
list_email = []
print("Checking CIP Google Sheet Table for new projects.")
with arcpy.da.SearchCursor(sde_CIP1923, "CIPProjectNumber") as scursor:
        for srow in scursor:
                list_sde.append(srow[0])
for item in list_googleprojects:
        if item not in list_sde:
                list_email.append(item)
if list_email:
        print(f"\tThe following projects are new: {list_email}")
        write_to_log(f"The following projects are new: {list_email}")
else:
        print("\tNo new projects were found.")
```

Although there is some overhead to working within the Esri SDE,

the benefits of a versioned environment are immense.

We're also informed if we need to do some manual labor!

In [ ]:

```
print("Connecting to the SDE and appropriate version.")
sdeTempPath = tempfile.mkdtemp()
arcpy.CreateDatabaseConnection_management(sdeTempPath, "Editor.sde", "SQL_SERVER",
                                          "ITT701", "OPERATING_SYSTEM_AUTH")
EditSDE = os.path.join(sdeTempPath, "Editor.sde")
sdeVersionNameFULL = ""
sdeVersionLIST = []
for version in arcpy.da.ListVersions(EditSDE):
        sdeVersionLIST.append(version.name)
        if version.name == "DBO.CarmonaVersion":
                if version.description == "":
                        arcpy.AlterVersion_management(EditSDE, "CarmonaVersion",
                                                      description = "jcarmona@mckinneytexas.org | ex
t 7422")
if "DBO.CarmonaVersion" not in sdeVersionLIST:
        print("\tCarmona version not found, creating now.")
        arcpy.CreateVersion_management(EditSDE, "sde.DEFAULT" ,"CarmonaVersion", "PUBLI
C")
        arcpy.AlterVersion_management(EditSDE, "CarmonaVersion",
                                      description = "jcarmona@mckinneytexas.org | ext 7422")
        print("CarmonaVersion created.")
else:
        print("\tCarmonaVersion version exists, connecting now.")
        arcpy.CreateDatabaseConnection_management(sdeTempPath, "Carmona.sde", "SQL_SERVE
R", "ITT701",
                                                  "OPERATING_SYSTEM_AUTH", version = "DBO.Ca
rmonaVersion")
```

In [ ]:

```
EditSDE_Carmona = os.path.join(sdeTempPath, "Carmona.sde")
sde_CIP1923 = os.path.join(EditSDE_Carmona, "SDE.DBO.DemoSection_CIP1923")
sde_CIPFY1923 = os.path.join(EditSDE_Carmona, "SDE.DBO.DemoSection_CIPFY1923")
sde_CIP1923_Point = os.path.join(EditSDE_Carmona, "SDE.DBO.DemoSection_CIP1923_Point")

list_googleprojects = df['Project No'].tolist()
list_sde = []
list_email = []
print("Checking CIP Google Sheet Table for new projects.")
with arcpy.da.SearchCursor(sde_CIP1923, "CIPProjectNumber") as scursor:
        for srow in scursor:
                list_sde.append(srow[0])
for item in list_googleprojects:
        if item not in list_sde:
                list_email.append(item)
if list_email:
        print(f"\tThe following projects are new: {list_email}")
        write_to_log(f"The following projects are new: {list_email}")
else:
        print("\tNo new projects were found.")
```

Code Part IV

The Code-ttle Drive

Tandem Cursors

Tandem Cursors

Cursors can seem a little daunting.

They do require clear understanding
In order to use them effectively

At their most basic, a `cursor` is comprised of:

- table or layer reference
- fields to be used
- query

There are individual cursors to:

- search records
- update records
- insert records

This script uses a search cursor to iterate through our new data from the Google Sheet and an update cursor to edit the old data in the SDE

In [ ]:
```
updatefields_google = ["Project_No", "Project_Name"]
with arcpy.da.SearchCursor(googletable, updatefields_google) as scursor:
    for srow in scursor:
        print(f"My project number is: {srow[0]} --- {srow[1]})")
```

In [ ]:

```
updatefields_google = ["Project_No", "Project_Name", "Project_Funding"] #fields in table
updatefields_sde = ["CIPProjectNumber", "ProjectName", "ProjectBudget"] #fields in SDE
arcpy.env.workspace = EditSDE_Carmona
edit = arcpy.da.Editor(EditSDE_Carmona)
edit.startEditing(True, True)

with arcpy.da.SearchCursor(googletable, updatefields_google) as scursor:
    for srow in scursor:
        print(f"Google project number is: {srow[0]} ---")
        update_query = f"CIPProjectNumber = '{srow[0]}'"
        ucursor = arcpy.da.UpdateCursor(sde_CIP1923, updatefields_sde, update_query)
        edit.startOperation()
        for urow in ucursor:
            print(f"\tSDE data for budget {urow[2]}")
            print(f"\tSDE data for project {urow[1]}")
        edit.stopOperation()
edit.stopEditing(True)
```

Within the actual script, this nested pattern has three update cursors in sequence

Allowing us to update three layers share similar records but differ in geometry type

In [ ]:

```
counter =1
with arcpy.da.SearchCursor(googletable, updatefields_google) as scursor:
        for srow in scursor:
                update_query = f"CIPProjectNumber = '{srow[0]}'"
                print(f"\tCIP Project {srow[0]}, {counter-1} edits pending")
                ucursor = arcpy.da.UpdateCursor(sde_CIP1923, updatefields_sde, update_qu
ery)
                edit.startOperation()
                for urow in ucursor:
                        urow[1] = srow[1]
                        urow[2] = srow[2]
                        urow[3] = srow[3]
                        urow[4] = srow[4]
                        urow[6] = srow[6]
                        urow[7] = srow[7]
                        if (
                                urow[9] != srow[9] or
                                urow[5] != srow[5]
                        ):
                                urow[10] = datetime.date.today()
                        urow[5] = srow[5]
                        urow[9] = srow[9]
                        ucursor.updateRow(urow)
                        counter += 1
                edit.stopOperation()
```

In [ ]:

```
counter = 1
with arcpy.da.SearchCursor(googletable, updatefields_google) as scursor:
        for srow in scursor:
                update_query = f"CIPProjectNumber = '{srow[0]}'"
                print(f"\tCIP Project {srow[0]}, {counter-1} edits pending")
                ucursor = arcpy.da.UpdateCursor(sde_CIPFY1923, updatefields_sde, update_
query)
                edit.startOperation()
                for urow in ucursor:
                        urow[1] = srow[1]
                        urow[2] = srow[2]
                        urow[3] = srow[3]
                        urow[4] = srow[4]
                        urow[6] = srow[6]
                        urow[7] = srow[7]
                        if (
                                urow[9] != srow[9] or
                                urow[5] != srow[5]
                        ):
                                urow[10] = datetime.date.today()
                        urow[5] = srow[5]
                        urow[9] = srow[9]
                        ucursor.updateRow(urow)
                        counter += 1
                edit.stopOperation()
```

In [ ]:

```
counter = 1
with arcpy.da.SearchCursor(googletable, updatefields_google) as scursor:
        for srow in scursor:
                update_query = f"CIPProjectNumber = '{srow[0]}'"
                print(f"\tCIP Project {srow[0]}, {counter-1} edits pending")
                ucursor = arcpy.da.UpdateCursor(sde_CIP1923_Point, updatefields_sde, upd
ate_query)
                edit.startOperation()
                for urow in ucursor:
                        urow[1] = srow[1]
                        urow[2] = srow[2]
                        urow[3] = srow[3]
                        urow[4] = srow[4]
                        urow[6] = srow[6]
                        urow[7] = srow[7]
                        if (
                                urow[9] != srow[9] or
                                urow[5] != srow[5]
                        ):
                                urow[10] = datetime.date.today()
                        urow[5] = srow[5]
                        urow[9] = srow[9]
                        ucursor.updateRow(urow)
                        counter += 1
                edit.stopOperation()

print("Editing loops complete for CIP layers.")
print(f"Editing environment active: {edit.isEditing}")
edit.stopEditing(True)
print("Edits saved.")
print(f"Editing environment active: {edit.isEditing}")
arcpy.ReconcileVersions_management(EditSDE, "ALL_VERSIONS", "sde.DEFAULT", r'CarmonaVers
ion', "LOCK_ACQUIRED",
                                   "NO_ABORT", "BY_OBJECT", "FAVOR_TARGET_VERSION", "POS
T", "KEEP_VERSION", None, "PROCEED")
print("Version has reconciled and posted to Default.")
```

In [ ]:

```
print("Editing loops complete for CIP layers.")
print(f"Editing environment active: {edit.isEditing}")
edit.stopEditing(True)
print("Edits saved.")
print(f"Editing environment active: {edit.isEditing}")
arcpy.ReconcileVersions_management(EditSDE, "ALL_VERSIONS", "sde.DEFAULT", r'CarmonaVers
ion', "LOCK_ACQUIRED",
                                   "NO_ABORT", "BY_OBJECT", "FAVOR_TARGET_VERSION", "POS
T", "KEEP_VERSION", None, "PROCEED")
print("Version has reconciled and posted to Default.")
```

Fin

Jordan Carmona, GISP

jcarmona@mckinneytexas.org

linkedin/in/jordancarmona