Crime Report Automation Guide

# Introduction

To **speed up the report process** and to **integrate ArcGIS Pro and** [**ArcGIS Notebooks**](https://pro.arcgis.com/en/pro-app/latest/arcpy/get-started/pro-notebooks.htm) into the GIS Unit’s workflows, we tested a more automated approach to building the annual PPR crime report.

This automated approach includes the same analysis steps as previous years (see [here](file:///X:\Projects\Special_Projects\Crime\Crime_Statistics\Crime_Annual_PPRSystem_HowTo_GISModel.docx) for documentation on the crime analysis model), but this analysis does not use ModelBuilder; instead, the steps are broken up into individual cells within a Notebook.

The main difference between the current automated approach and previous years’ crime reports is that the automated approach **quickly produces tables and charts for each site using Python** instead of manually creating these elements for each site using Excel. This process is quicker, more organized, and easily reproducible for the creation of future reports.

This automated approach currently uses the same report layout (a PowerPoint presentation with text, tables, and charts placed in slides) with the goal being to eventually automate the report formatting and creation as well.

This crime report consists of **two Notebooks** – one for the crime analysis and the other for the crime report. Both Notebooks are broken into different sections explaining what the code in each section does.

The rest of this guide discusses the 1) requirements needed to run the code in the Notebooks, 2) opening Notebook files, 3) setting up and running Notebook cells, 4) more in-depth descriptions and instructions for both Notebooks including the locations of data, 5) goals for improving and expanding the existing code in the future, and 6) resources for further research/learning.

**Please carefully read this guide before running the Notebooks! Please pay special attention to the “Before Running” sections!**

# Requirements

Below are the requirements for this process.

* ArcGIS Pro 2.5 or more recent
* Access to the data through the X drive
* Project folder in the X drive set up exactly like the example – guide includes instructions
* Little to no understanding of Python needed to run annually, but *more intermediate level needed if expanding existing code*
* Python libraries [Operating System](https://coderslegacy.com/python/libraries-in-python/python-os/#:~:text=Importing%20OS%20library%20OS%20library%20is%20part%20of,is%20the%20file%20path%20of%20the%20python%20program.), [ArcPy](https://pro.arcgis.com/en/pro-app/2.8/arcpy/get-started/what-is-arcpy-.htm), [ArcGIS for Python API](https://pro.arcgis.com/en/pro-app/2.8/arcpy/get-started/arcgis-api-for-python.htm), [pandas](https://pandas.pydata.org/), [matplotlib](https://matplotlib.org/), [numpy](https://numpy.org/) imported – code includes importing at beginning or throughout

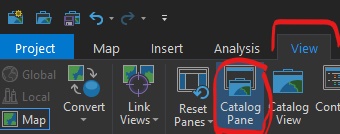
# Making Copies of the Notebook Files

Since any changes made to a Notebook will permanently alter the Notebook, you want to make your own copies of both Notebooks before running your code. To do so, follow the instructions below.

1. Navigate to the folder with the original Notebooks, which use the 2021 crime data: **X:\Projects\Special\_Projects\Crime\Crime\_Statistics\2022\_01\_GIS\_Automation\_Testing**
2. Click on the two Notebooks:
   1. CrimeAnalysis.ipynb
   2. CrimeReport.ipynb
3. Copy both of them (Ctrl + C).
4. Navigate to your project folder, which should be located within the Crime Statistics folder (X:\Projects\Special\_Projects\Crime\Crime\_Statistics) and should include the current year in the title. **Make sure your project folder does not have any spaces in the name.**
5. Paste (Ctrl + V) both Notebook files into your new project folder.
   1. If you’d like, you can rename the files to represent the most recent year.

# Opening Notebook Files

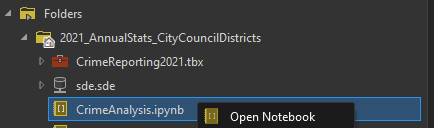
1. Start ArcGIS Pro.
2. Create a new ArcGIS Pro project by clicking on Map under the New section.
3. Name the project something like CrimeReportingXXXX (X = Year) and save the location to your project folder. Hit OK.
4. Once a new project is open, navigate to the “View” tab at the top of the screen. Select Catalog Pane in the Windows subsection under the View tab. This will open up a new pane to the right of the map.



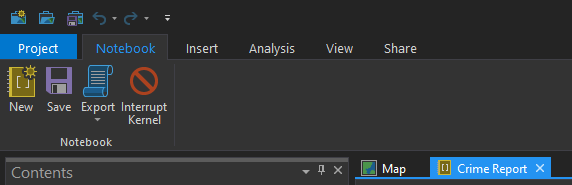
1. In the Catalog Pane’s Project tab at the right of the screen, click on the Folders dropdown caret to expand the connected folders. The project folder you set in the third step should be under Folders.



1. Expand your project folder by clicking on its dropdown.
2. Find your two Notebook copies:
   1. CrimeAnalysis.ipynb (or your new name if you renamed it)
   2. CrimeReport.ipynb (or your new name if you renamed it)
3. Right-click on each Notebook and select Open Notebook. This will open two separate Notebook tabs in the project. The tabs for each Notebook will be next to the Map tab. They may take a few moments to load.



1. Click on one of the Notebook tabs (above the main map area of the project but below the top ribbon with all the tabs and tools). You will see that at the top of the screen (next to Project) when you select a Notebook, there is a Notebook tab activated in the top ribbon. This is where you can add blank new Notebooks, save the Notebook if you’ve made any changes, and interrupt the Notebook processing if necessary.



# Setting Up & Running Notebook Cells

## Notebook Elements

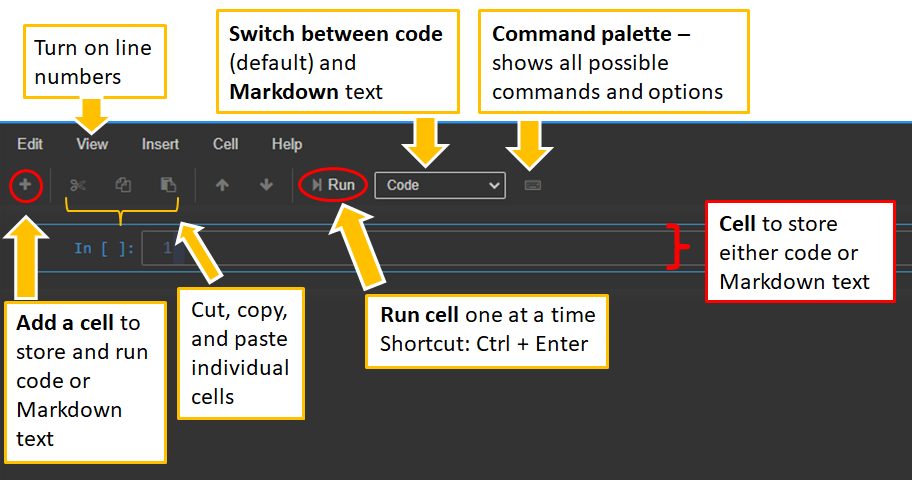
Notebooks consist of two main elements: Markdown text and Python code.

**Markdown** is how you can format regular text to be used for instruction or notes. This text is not attached to any data or functions and is meant to guide the user through the Notebook. This text can be formatted as a title, a heading (with a dropdown arrow to hide the section beneath), a subheading (also with a dropdown arrow), or regular body text.

**Python code** in Notebooks is organized into *cell blocks*. Python code can be used to call existing data, format and edit data, apply existing or customized functions to data, and more.

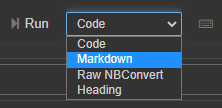
The advantage of using **Markdown and Python code together** is that your Notebook can serve as organized, shared, interactive, easy to follow, and easily reproducible documentation for your work.

## Notebook Tour



## Writing & Editing Markdown Text

In the **Code/Markdown dropdown** at the top of the Notebook (above the cell), choose **Markdown**.



The size and importance of the Markdown text is defined by # symbols. Body text has no # symbol before it, but all titles, headings, and subheadings do. See below for the number of # symbols that correspond to each Markdown text type. Note that for up to 6 # symbols, the cell gets a dropdown to hide the section within it. **Hiding sections is a good way to organize your Notebook better**.



You can style font colors, size, and type (bold, italics, underline, etc.) using HTML, such as in the example below, where the word “Hello” is now bolded, italicized, and teal colored.



To insert images into a Markdown cell, navigate to the Edit tab (above the Add Cell button) and select Insert Image.

To **edit existing Markdown text**, double-click on the Markdown text and then type over the existing text. When editing, you will see the cell block that the text lies within. To make this cell block invisible so that you can see the output of your text styling, click the Run button or use the shortcut Ctrl + Enter.

## Writing & Editing Python Code

To write Python code, ensure that the **Code/Markdown dropdown** at the top of the Notebook is set to **Code**. Type your code.

To indicate lines of code or portions of a line of code that are not meant to be incorporated into the actual code (such as instructions or notes), add a # symbol before the line(s) or text that you want to be muted/not run as actual code. To mute an entire cell/not run it as actual code, select all lines within the cell (Ctrl + A) and then press Ctrl + /

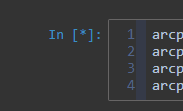
To **edit existing code**, click inside the cell and type over the text.

## Running Cells

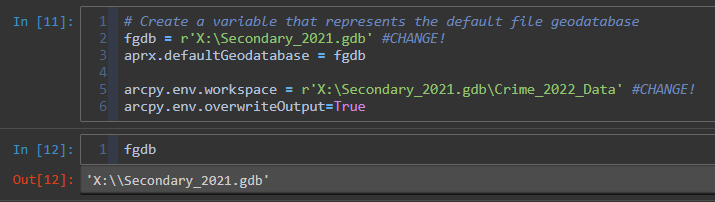
Regardless of whether a cell block is Markdown or Python code, it can **be run by pressing** **Ctrl + Enter** or by clicking the Run button next to the Markdown / Code dropdown at the top of the Notebook.

Running all cells at once is possible and may be faster, but since both Notebooks require the occasional changing of paths and names, **do not run all cells at once**.

When a cell is running, there will be an asterisk (\*) next to the cell block. Multiple cells in the same project cannot be simultaneously running at the same time, but you can tell the [kernel](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#kernel) what cells to run next by pressing Ctrl + Enter for other cell blocks while another cell block is still running. When a cell is finished running, the asterisk will be replaced by the number of the order in which the cells were run.



If you ever want to check that a cell block has successfully run or see the properties of an object you created in a previously run cell block, 1) add another cell block, 2) type the name of the object into the cell block by itself (make sure to keep the letter case and spelling consistent with how you defined the object in the cell you ran), and 3) run the cell block. The Notebook will print the properties of the object that you created to verify that the above cell block was successfully run and that the properties of your object are what you intended them to be. For an example, see below.



In the example above, we set the fgdb path in the first cell block. In the second cell block, we ran just the name of the fgdb object to verify that it was successfully created and that the path is correct.

Now that you know how to edit Markdown text/code and run cell blocks, you can move onto the first Notebook, the Crime Analysis Notebook. **Please read the instructions below carefully before running the Crime Analysis Notebook.**

# Crime Analysis Notebook

## Summary

The goal of this analysis is to create a feature that includes PPR properties information for each crime that occurred within 5ft of a property. This feature includes all information regarding each crime’s season, the time it occurred, and crime type; this information comes with the original PPD crime data, but this information is reshaped in our analysis to meet the standards of our crime report. The outputs of this code will be 1) **a point feature with all crimes**, their original and reshaped information for our report, and all property information that is associated with each crime as well as 2) **a CSV table** with the same information.

## Data Needed and File Locations

* PPD Crime Points
  + Location: **DataBridge -> GIS\_POLICE.INCIDENTS\_PART1\_PART2**
* PPR Properties **(currently BETA)**
  + Location: **X:\PPR\_GIS\_Data.gdb\Asset\_Data\ PPR\_Properties\_BETA**

## Before Running

**IMPORTANT:** There will be numerous times throughout the code where you will have to **manually change the path names or names of features** to represent the most recent year. These instances are all marked in the Notebook as an “IMPORTANT” message with bold red text.

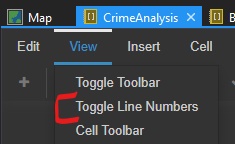
**IMPORTANT:** Before running the code, make sure to **manually place the PPD crime incidents point feature into your ArcGIS Pro project**. This feature can be found in DataBridge and is titled “GIS\_POLICE.INCIDENTS\_PART1\_PART2”.

The name of this crime point feature should remain the same, but **make sure to filter the data to include only crimes that occurred in the year you are reporting on**. The SQL expression looks like this:

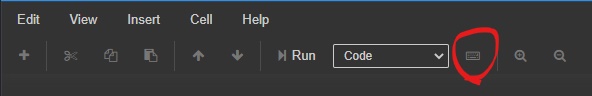
DISPATCH\_DATE >= '20XX-01-01' AND DISPATCH\_DATE <= '20XX-12-31' , with the X values in the year being the last two digits of the year you are reporting on.

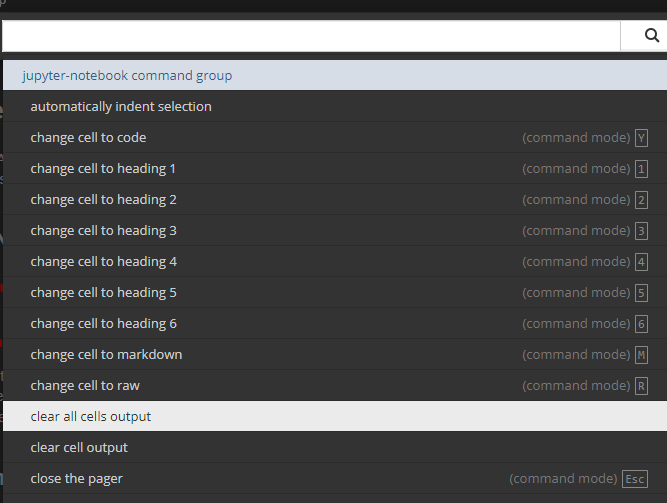
**IMPORTANT:** For organization purposes, **turn on line numbers** and **clear the output of all cells** so that you can track your order in which you run the cells instead of seeing the order of the last person who ran the Notebook.

To **turn on line numbers**, navigate to the View tab of the Notebook (at the top of the Notebook but below the ArcGIS Pro ribbon). Click on the View tab and select “Toggle Line Numbers.”



To **clear the output of all cells**, navigate to the Notebook Command Palette (to the right of the code/Markdown dropdown) and select “clear all cells output” (you can search for it in the Command Palette search bar if necessary).





## Run Notebook

Please **turn your attention to the Crime Analysis Notebook**, which will now walk you through the code and anything you need to change.

Click on each code cell block and press **Ctrl + Enter** to run **one cell at a time**. Make sure to follow the order of the Notebook. Follow the steps to change the path names or file names where indicated in **bolded red text with an “IMPORTANT:” prefix**.

When you are finished running all cells and you have followed all instructions in the Crime Analysis Notebook carefully, please **return to this document** to read the information below about the Crime Report Notebook before you move on to the Crime Report Notebook.

# Crime Report Notebook

## Summary

The Crime Report Notebook takes the spatially joined crime points layer that was created in the Crime Analysis Notebook and creates new [dataframes](https://www.tutorialspoint.com/python_pandas/python_pandas_dataframe.htm), or tables, from this layer in order to summarize and display our data into the pie charts, overview tables, and individual site tables that our annual reports consist of.

This Notebook consists of 4 main sections, each with many subsections. The structure **breaks up repeating report elements** (such as pivot tables, site pie charts, etc.) **by district** so that we can tailor the code to specific districts and more quickly locate potential errors. The estimated time to run this Notebook (excluding reading instructions) is 15-25 minutes.

Below are the 4 main sections and their subsections.

* Setting Up
  + Import libraries and set environment
  + Add new Council District field for more accurate sorting
  + Convert PPR\_Crime\_5ft to spatial dataframe for quicker manipulation
* Overview/Page 1 Tables and Charts
  + Center Pivot Table
  + Pie Chart
  + Bar Graph
* Summary Tables and Charts for Each District
  + District 1
    - Main Summary Analysis Table (on District 1 overview page)
    - Main Pivot Table (on District 1 overview page)
    - District Pie Chart (on District 1 overview page – creates dataframe)
    - Pie Charts for Each Site
    - Tables for Each Site
      * Table 1
      * Table 2
      * Table 3
      * Table 4
  + (Other 9 districts with the same structure as District 1 above)
* Pie Charts for the 10 District Overview Pages (creates all 10 pie charts at once)

Currently, the output includes 1) hundreds of tables saved as HTML files and styled with CSS stylesheets, 2) pie charts for each district and each site within each district saved as PNGs, and 3) an overview bar graph also saved as a PNG.

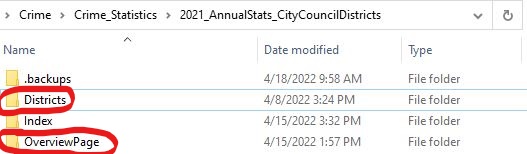
The goal is to eventually be able to programmatically convert each styled HTML table to its own PNG within the same Notebook, but due to limitations regarding ArcGIS Pro’s Python library manager and the difficulties associated with importing additional third-part libraries, **each HTML table will need to be screenshotted manually after running the code** to be added as an image in the report.

## Data Needed and File Locations

The only layer that you will need in order to run this Notebook is the **“PPR\_Crime\_5ft” layer**, which is the spatially joined crime point layer output that you created during the Crime Analysis Notebook. Make sure it is in your Contents pane before running the Crime Report Notebook. Both Notebooks should be running out of the same ArcGIS Pro project.

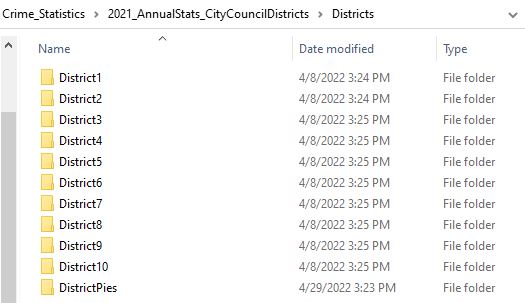
Before running the Crime Report Notebook, **make sure your project folder’s structure is the same as the example 2021 folder** (X:\Projects\Special\_Projects\Crime\Crime\_Statistics\2021\_AnnualStats\_CityCouncilDistricts).

You want two subfolders within the annual project folder: 1) one titled **OverviewPage** for the charts and graphs that go into the first page of the crime report and 2) another titled **Districts** for the overview tables, pie charts, individual site tables, and pie charts for each district.



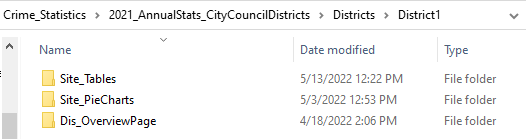
The OverviewPage folder will have PNGs and HTML files directly saved to it and does not have any subfolders. The Districts folder has **one subfolder for each City Council district within it (10 total)** and a **subfolder for the 10 overview district pie charts**.

Please see the screenshot below for the Districts folder structure and names of the subfolders. **Please keep the folder and subfolder names consistent to those in the screenshot. The code will not work if the names are not the same as in the screenshot.**



Each of the ten District subfolders has three subfolders: 1) the first for **individual site tables**, 2) the second for **individual site pie charts**, and 3) the third **for each district’s overview page tables**.

Please see the screenshot below for the structure and names of these three subfolders within each District subfolder. Again, **please keep the folder and subfolder names consistent to those in the screenshot. The code will not work if the names are not the same as in the screenshot.**



Numerous lines of code throughout the Notebook depend on this folder structure. Please **keep this folder structure and look out for notices** (bolded and in red) **to change lines of code** in certain cell blocks.

## Before Running

**IMPORTANT:** The 12th step of the Crime Analysis Notebook instructs you to manually 1) **delete any duplicate crime features** resulting from the spatial join and 2) **change any duplicate districts** so that each crime took place in only one district. Make sure you followed these instructions carefully before moving on to the Crime Report Notebook!

**IMPORTANT:** As the previous section instructs, **make sure your project folder structure is exactly like the example 2021 folder** before running the code. Refer to the previous section for guidance on structuring these folders. **The Crime Report Notebook code** **will not work if you do not do this.**

**IMPORTANT:** There will be numerous times throughout the code where you will have to **manually change the path names or names of features** to represent the most recent year. These instances are all marked in the Notebook as an “IMPORTANT” message with bold red text.

**IMPORTANT:** Make sure you **run the Notebook in the order that the sections are organized**.

**IMPORTANT:** Try to **run all cell blocks in one session**, meaning that you want to avoid closing Pro before you’re finished successfully running the entire Notebook.

If you don’t run all cell blocks in one session, you will have to rerun the Notebook from the beginning since the memory (where most of your objects are stored) is cleared when you exit Pro.

## Run Notebook

Like the Crime Analysis Notebook, click on each code cell block and press **Ctrl + Enter** to run **one cell at a time**. As instructed in the previous section, make sure to follow the order of the Notebook.

For the most part, you should be able to get through the Notebook smoothly without any errors or having to change many lines of code. However, there are **some sections that require more attention** than others, so be sure to watch out for the **bolded red text with an “IMPORTANT:” prefix**. For more complex situations in which you need to change the code, detailed instructions will be provided above the cell block that needs to be changed. Below are additional instructions for these two situations.

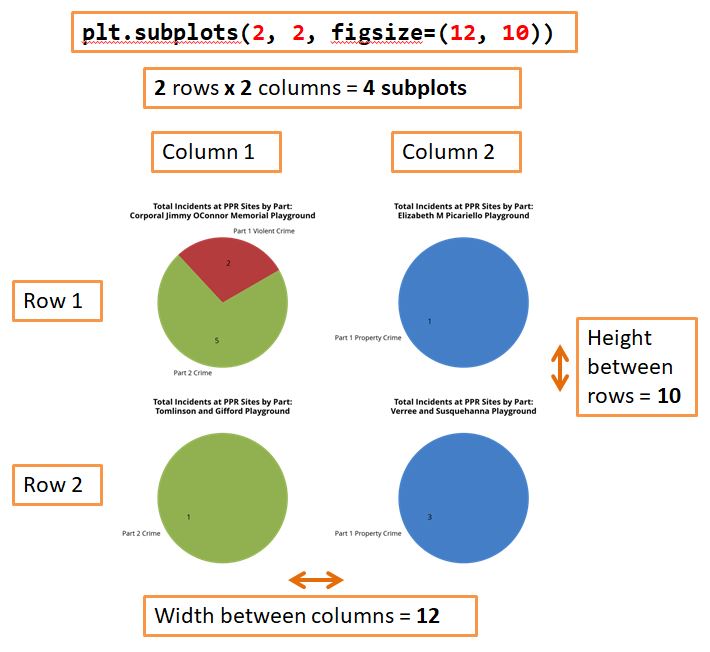
### Pie Charts for Each Site

One of the main functions of this Notebook is to produce pie charts for each individual site. Under each district section’s Pie Charts for Each Site subsection, you are running two cell blocks to accomplish this; these cell blocks format and export all pie charts for each site within a certain district.

Because the number of sites and therefore the number of pie charts changes from district to district, **one specific line of code will have to be changed** for each district’s Pie Charts for Each Site section. If you do not change this line of code for each district, the code will give an error and a pie chart figure will not be produced**.** This line of code in the second cell block (marked in the Notebook with bolded red text) is responsible for creating the **figure**, or single “page”, **that has all the pie charts for the specified district displayed in it.**

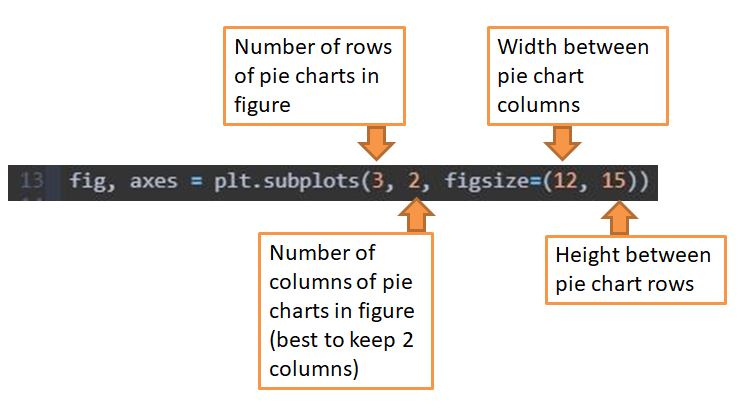
In order to format the pie chart figure for a single district, the aforementioned line of code must specify the 1) **number of individual pie charts**, hereafter referred to as **subplots**, that will need to be created for each district. This number is broken up into rows and columns that, when multiplied, make up the total number of desired subplots. This line of code also specifies 2) the width and height between the subplots in the figure, which is important for screenshotting the individual subplots within the figure.

The image below represents this line of code (at the top) and how the line of code directly impacts the number of subplots produced and the figure size (or the space between the subplots). The example shows 4 subplots, or 4 pie charts to represent 4 sites in that district, being made. These 4 sites are broken up into 2 rows and 2 columns (2 x 2 = 4), and the figure size allows just enough room between the subplots to take screenshots of each individual subplot.



Now that we understand what the pie chart figure output is supposed to look like, we can look more closely at the line of code that produces this output.

Another example of this line that you will have to change is shown below. The **subplots() parameter will need to be changed** to represent the number of pie charts. The first number (in the example below, 3) represents the number of rows in the overall figure, and the second number (2 in the example) represents the number of columns in the figure. 3 multiplied by 2 is 6, which is the number of sites or individual pie charts that need to fit into the figure. We’re breaking these 6 pie charts into 3 rows and 2 columns.



It’s best to keep 2 columns so that there are fewer numbers to change and so that the figure size parameters don’t have to be altered as much. Note that if the **column number is 2 but you have a total site/pie chart number that is not divisible by 2, you should round up** so that you have one empty subplot. For example, a total of 27 sites/pie charts could turn into 28 (or 14 rows and 2 columns) with one empty subplot that wouldn’t be screenshotted.

The **figsize=() parameter** represents the space between the rows and columns in the figure. This part of the code’s improper configuring will not result in error messages, but it should be changed when the number of subplots drastically changes so that the pie charts and their corresponding text stay mostly consistent from district to district. The first number (in the example above, 12) represents the width between pie chart columns and the second number (15) represents height between pie chart rows. For taking screenshots of individual pie charts, you want to make sure that the pie chart labels do not overlap other pie charts. You may need to export the pie charts and manually check the outputs before finding the appropriate figure size for each district’s pie chart figure.

### Tables for Each Site: *Tables 3 & 4*

Under each district’s Tables for Each Site section (particularly tables 3 and 4), you are running multiple cell blocks to format and export tables summarizing the crime count per crime category, season, and active hours for each site.

Some districts **did not have any crimes occur at any site during either the summer or off-season**, which would make any code that is run for either Table 3 or Table 4 **invalid** (meaning you will get error messages such as TypeError: other must be a MultiIndex or a list of tuples) and no tables would be produced). Make sure to pay attention to the dataframe previews that you create so that you can anticipate when a district will not produce these tables for a certain season.

In many cases, a district will have crimes occur during both the summer and the off-season, but **only one type of hour (active hour vs. non-active) has incidents recorded**. Tables 3 and 4 will be generated in these cases, but the code will need modifying so that a column that does not exist will no longer be called. If you do not change this in the code, the tables will not be produced for the district.

If you get an error message like **“Columns not found: ‘Non-Active’”** or **“[‘Non-Active’] not in index”** after running the cell blocks for Tables 3 or 4, **REMOVE the hour type** (either ‘Non-Active’ or ‘Active’ – in the above example, it would be ‘Non-Active’ removed since that is the type that the error message called out) from the below example code.

**dis1table3col = dis1table3[['PUBLIC\_NAME', 'GIS\_CRIME', 'Active', 'Non-Active']]**

would turn into:

**dis1table3col = dis1table3[['PUBLIC\_NAME', 'GIS\_CRIME', 'Active']]**

Make sure to remove this type for all cell blocks in that table (either Table 3 or Table 4). Note that just because a certain hour type is not present for Table 3 does not mean that it will also not be present in Table 4, so make sure to only change the cell blocks for the table that is getting error messages. It’s best to get an error message to note when a column is not present before changing the code.

Please **turn your attention to the Crime Report Notebook**, which will now walk you through the code and anything you need to change.

### After Running

After running the entire Notebook one cell block at a time, **make sure your folders are all populated correctly.**

The overview page folder should have one PNG bar chart, one PNG pie chart, and one HTML pivot table.

The site tables folder for each district should have many HTML tables that are named after the table number and the site name. The site pie charts folder for each district should (for now) have only one PNG file for all site pie charts. The overview page for each district should have one PNG pie chart, one HTML pivot table, and one HTML summary table.

The district pie charts folder (for now) should have at least one PNG file for all district pie charts.

# Goals for Future Reports

* I designed [2 site tables that could replace our 4 per site](TablesRedesign.xlsx). These 2 new site tables combine the information presented in our current 4 tables and include grand total (total per site) and multiple subtotal (by category) rows/columns for each site. I’d like to incorporate this design into future reports.
  1. I anticipate this to be done by creating new dataframes for each subtotal and grand total and then concatenating these new dataframes to the original dataframe. The next steps (exporting one table per site by using a dictionary) would be the same as the code I’ve already written.
* Currently, each pie chart is saved into one large figure that requires the user to take screenshots of each pie chart/subplot. I’d like to find a way to save each pie chart as a separate PNG (one PNG file per pie chart) so that screenshots are not necessary. My current code for the overview district pie charts almost accomplishes this (files are set up to represent each site or district number), but the actual subplots are not properly split and saved into their corresponding PNG file.
  1. I anticipate the solution being to change something in the last few lines of the current code for the overview district pie charts. The extent and export lines of code in particular would be the lines that would need to be changed.
* Currently, the number of site pie charts being produced varies per district, which requires the user to manually change the subplot and figure size parameters for each district. This may result in inconsistent pie chart proportions. I’d like to find a better way to design these figures so that the user won’t have to change it as much.
* Currently, the site tables are exported as HTML tables so that the CSS style sheets can be applied to them. This requires the user to take screenshots of each HTML table, which is time-consuming and tedious. I would love to programmatically save each styled HTML table as a PNG in the same cell block that produces the HTML files instead; I believe that [this is possible](https://stackoverflow.com/questions/60598837/html-to-image-using-python), but it requires the installation of open-source packages that are not natively available to ArcGIS Pro. To install new packages, you have to go to the Python Package Manager (Project/Pro Settings -> Python tab), clone your current Python environment or create a new one with the same packages installed, and then look for and add new packages. I ran into the issue of not being able to clone or create a new Python environment (possibly due to administrator or firewall issues?).
  1. A workaround for this is finding an alternative to PowerPoint (like further automating the report using HTML and CSS) that allows embedded HTML tables so that you do not have to screenshot each HTML table to insert it into the report.
* The Crime Report Notebook is lengthy and could potentially be further automated to produce all charts and tables for all 10 districts at once instead of the current method of breaking up the production into 10+ sections. This would likely be accomplished with custom if-then functions that are able to automatically reconcile inconsistencies between districts.