Community Experience Distilled

Explore the robust features of Python to create real-world ArcGIS applications through exciting, hands-on projects

ArcGIS Blueprints

Eric Pimpler

This book is an immersive guide that will take your ArcGIS Desktop application development skills to the next level. It starts off by providing a detailed description and examples of how to create ArcGIS Desktop Python toolboxes that will serve as containers for many of the applications that you will build. We provide several practical projects that involve building a local area/community map and extracting wildfire data. You will then learn how to build tools that can access data from ArcGIS Server using the ArcGIS REST API. Along the way, you will be introduced to advanced ArcPy Mapping and ArcPy Data Access module techniques and use Data Driven Pages to automate the creation of map books. Finally, you will learn advanced techniques of working with video and social media feeds. By the end of the book, you will have your own desktop application without having spent too much time learning sophisticated theory.

Who this book is written for

If you have prior experience building simple apps with ArcGIS and now fancy developing a more challenging and complex desktop application in ArcGIS, then this book is ideal for you.

What you will learn from this book

- Automate the creation of creative output data visualizations including maps, charts, and graphs
- Explore ways to use the ArcPy mapping module and Data Driven Pages to automate the creation of map books in your own project
- Develop applications that use the Plotly platform and library to create stunning charts and graphs that can be integrated into ArcGIS Desktop
- Build tools that access REST services and download data to a local geodatabase
- Design, build, and integrate advanced GUIs with wxPython and ArcGIS Desktop in ArcGIS
- Get clued up about constructing applications that export data to Google Earth Pro to automate time-consuming and complex processes
- Maximize the access to ArcGIS Server and ArcGIS Online using the ArcGIS REST API with Python

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 3 'Automating the Production of Map Books with Data Driven Pages and ArcPy'
- A synopsis of the book’s content
- More information on ArcGIS Blueprints
About the Author

**Eric Pimperl** is the founder and owner of GeoSpatial Training Services (http://geospatialtraining.com/) and has over 20 years of experience implementing and teaching GIS solutions using Esri, Google Earth, Google Maps, and open source technologies. Currently, he focuses on ArcGIS application development with Python and the development of custom ArcGIS Server web and mobile applications using JavaScript.


He has a bachelor's degree in geography from Texas A&M University and a master of applied geography degree with a concentration in GIS from Texas State University.
ArcGIS Desktop 10.3 is the leading desktop solution for GIS analysis and mapping. The ArcPy site package, which contains the ArcPy mapping and Data Access modules, enables Python programmers to access all the GIS functionalities provided through ArcGIS Desktop. ArcPy can be integrated with other open source Python libraries to enhance GUI development; create stunning reports, charts, and graphs; access REST web services; perform statistics analysis; and more. This book will teach you how to take your ArcGIS Desktop application development skills to the next level by integrating the functionality provided by ArcPy with open source Python libraries to create advanced ArcGIS Desktop applications.

In addition to working with the ArcPy, ArcPy Mapping, and ArcPy Data Access modules, the book also covers the ArcGIS REST API and a wide variety of open source Python modules, including requests, csv, plotly, tweepy, simplekml, wxPython, and others.

**What this book covers**

*Chapter 1, Extracting Real-Time Wildfire Data from ArcGIS Server with the ArcGIS REST API*, describes how to use the ArcGIS REST API with Python to extract real-time wildfire information from an ArcGIS Server map service and write the data to a local geodatabase. A Python Toolbox will be created in ArcGIS Desktop to hold the tools.

*Chapter 2, Tracking Elk Migration Patterns with GPS and ArcPy*, teaches you how to read a CSV file containing the GPS coordinates of elk migration patterns and write the data to a local geodatabase. The ArcPy mapping module will then be used to visualize time-enabled data and create a series of maps that show the migration patterns over time.
Chapter 3, *Automating the Production of Map Books with Data Driven Pages and ArcPy,* shows you how to use the Data Driven Pages functionality in ArcGIS Desktop along with the ArcPy mapping module to automate the production of a map book. The use of Python add-ins for ArcGIS Desktop for user interface development will also be introduced.

Chapter 4, *Analyzing Crime Patterns with ArcGIS Desktop, ArcPy, and Plotly - Part 1,* is the first of two chapters that covers the creation of ArcPy scripts for crime analysis. In this first chapter, you'll learn how to use the Python requests module to extract crime data from the city of Seattle's open database and write to a local geodatabase. You'll then write custom script tools to aggregate the crime data at various geographic levels and create maps that can be printed or exported to a PDF format.

Chapter 5, *Analyzing Crime Patterns with ArcGIS Desktop, ArcPy, and Plotly - Part 2,* is the second of two chapters that covers the creation of ArcPy scripts for crime analysis. In this chapter, you'll learn how to use the Plotly platform and the Python module to create compelling graphs and charts of crime data that can be integrated into the ArcGIS Desktop layout view for printing and export.

Chapter 6, *Viewing and Querying Parcel Data,* teaches you how to use the wxPython module to create advanced graphical user interface (GUI) applications for ArcGIS Desktop using Python within the context of an application that queries and views parcel data. Python add-ins for ArcGIS Desktop will also be used in the creation of the application.

Chapter 7, *Using Python with the ArcGIS REST API and GeoEnrichment Service for Retail Site Selection,* teaches you how to use the ArcGIS Online GeoEnrichment Service with Python to retrieve demographic and lifestyle information to support the site selection process of a new store location. You'll also build tools to interactively select the potential geographic location of stores based on demographic factors.

Chapter 8, *Supporting Search and Rescue Operations with ArcPy, Python Add-Ins, and simplekml,* teaches you how to build a Search and Rescue (SAR) application that identifies the last known location of the subject, creates search sectors in the support of operations, and exports the data to Google Earth for visualization purposes.

Chapter 9, *Real Time Twitter Mapping with Tweepy, ArcPy, and the Twitter API,* covers the mining of a live stream of tweets containing specific terms and hash tags. Tweets that contain geographic coordinates will be written to a local geodatabase for further analysis. In addition, several tools will be created to enable the analysis of this social media data. Finally, the results will be shared with the public through the ArcGIS Online service.
Chapter 10, Integrating Smart Phone Photos with ArcGIS Desktop and ArcGIS Online, covers the creation of a real estate application that reads photo metadata, extracts the coordinate information, retrieves the nearest address to the photo, and writes this information to a local feature class. In addition, the photos will be copied to a Dropbox account using the Python Dropbox module so that the photos can be accessed through a web application. Finally, the property feature class will be uploaded to ArcGIS Online, integrated with the Dropbox photos, and shared as a web-based map.
Many organizations have a need to create map books that contain a series of individual maps that cover a larger geographic area. These map books contain a series of maps and some optional additional pages, including title pages, an overview map, and some other ancillary information, including reports and tables. For example, a utility company might want to generate a map book detailing their assets across a service area. A map book for this utility company could include a series of maps, each of a large scale, along with a title page and an overview map. These resources would then be joined in a single document that could be printed or distributed as a PDF file.

In this chapter, we will cover the following topics:

- The preparation of a map document to handle Data Driven Pages
- Using the data-driven pages toolbar in ArcGIS Desktop
- Using the DataDrivenPages object in the ArcPy mapping module
- Exporting map books with the ArcPy mapping module
- Creating a Python add-in for ArcGIS Desktop
Design
From a design perspective, ArcGIS Desktop isn't terribly complex. We'll spend a fair amount of time up front, preparing the map document. We'll add a floodplain layer to the map document file, create a grid index layer, enable data-driven pages, and prepare the layout view. After the map document has been prepared, we'll create a new Python add-in containing a button that will trigger the creation of the book map. The Python add-in will have a single button, which will contain an onClick() method. The onClick() method will append the individual map pages to a single output PDF file using the PDFDocument class in the arcpy.mapping module as shown in the following screenshot:

Let's get started and build the application.

Setting up the Data Frame
The ArcGIS Desktop provides the ability to efficiently create a map book through a combination of the Data Driven Pages functionality along with an arcpy.mapping script. With a single map document file, you can use the Data Driven Pages toolbar to create a series of maps using the layout view along with your operational data and an index layer.
The index layer contains features that will be used to define the extent of each map in the series. It divides the map into sections, with each section representing a map that will be generated. These sections are sometimes called tiles or areas of interest, and they are often rectangular or square shapes.

If you need to include additional pages in the map book, including a title page, an overview map, and other ancillary pages, you'll need to combine the output from the Data Driven Pages toolbar with the functionality provided by the arcpy.mapping module.

In the following steps, we will learn how to use the Data Driven Pages toolbar, to set up a map document file for the Data Driven Pages functionality. We'll create a map book that will display a series of floodplain maps for a water-management district:

1. Open ArcMap with a Blank Map document and add the World_Topo_Map Basemap layer using the Add Basemap button.
2. Add the floodplain_100yr_capcog geodatabase and reg_wtr_planning_dist.shp layers from the C:\ArcGIS_Blueprint_Python\data\Floodplain_100yr_capcog folder.
3. Style the layers as shown in the following screenshot:
4. Save the map document as C:\ArcGIS_Blueprint_Python\ch3\Floodplains.mxd.

5. We need to define the map and page layout before creating the series. Let's create the reference series on a letter page size (8.5 by 11 inches) with a portrait orientation. The map scale will be 1:100,000. In addition, the map will have a title, labels for adjacent pages in the series, and some items in the map margin.

6. Double-click on the Layers data frame to display **Data Frame Properties** and select the **General** tab, as shown in the following screenshot:

![Data Frame Properties](image)

7. Rename the data frame to **Detail Map**.
8. Click on the Size and Position tab. Enter a value of 1 inch for X position and 2.5 inch for Y position.

9. For size, enter 6.25 inch as Width and 7.5 inch as Height.

10. Click on the Coordinate System tab. The current coordinate system should be defined as NAD 1983 StatePlane Texas Central FIPS 4203 (US Feet). If not, change it now.

11. Click on Apply and then click on OK.

12. Switch to Layout View in ArcMap and make sure that you have left enough space for the title at the top and the footer area has ample space for text, scale bar, north arrow, and other marginalia, as shown in the following screenshot:
Creating the Grid Index Features

A grid index feature class can be used to set the map extent for each map in the series. We can use the Grid Index Features geoprocessing tool to create this layer, the following steps will guide you through, how to create Grid Index Features:

1. Using the selection tools in ArcMap, select the Lower Colorado region from the Planning Districts layer, as shown in the following screenshot. We're going to create an index grid for this region. The process will be as, though, if you decide to select other regions of the map:

2. Open the Grid Index Features geoprocessing tool found in the Data Driven Pages toolset in the Cartography Tools toolbox.
3. Define the parameters seen in the following screenshot:
4. Click on **OK** to generate the grid index layer. It should look similar to what is shown in the following screenshot. Note that I have altered the zymology for the layer to only include an outline with no fill for the index polygons and have zoomed in on the map:

![Grid Index Layer Screenshot](image)

5. We have some additional work that needs to be done on the grid index layer, including the addition of a field for the labeling of adjacent pages and a field to determine the correct **UTM** zone for each page. To do this, we'll use the **Calculate Adjacent Fields** and **Calculate UTM Zone** geoprocessing tools.

6. Open the **Calculate Adjacent Fields** geoprocessing tool from the **Data Driven Pages** toolset in the **Cartography Tools** toolbox. Define the parameters, as shown in the following screenshot, where **Input Features** is the grid index layer that you created in the preceding steps and **PageName** is the default field that has already been added to the grid index layer:
7. Click on OK to execute the tool. Open the attribute table for the GridIndexFeatures feature class to view the attribute data that has been created for the PageName_* fields. Attributes that define the adjacent index pages have been populated.

8. Open the Add Field geoprocessing tool found in the Fields toolset in the Data Management toolbox.

9. Add the field parameters shown in the following screenshot. This will add a field that we’ll then populate with the Calculate UTM Zone geoprocessing tool:
10. Open the **Calculate UTM Zone** tool found in the **Data Driven Pages** toolset in the **Cartography Tools** toolbox.

11. Define the parameters, as shown in the following screenshot:

![Calculate UTM Zone tool](image)

12. Click on **OK** to execute the tool. If you'd like, open the attribute table and view the contents of the **UTM_Zone** field.

### Enabling Data Driven Pages

In this section, you’ll enable the **Data Driven Pages** functionality for the map document. The following steps will guide you through, how to enable **Data Driven Pages**:

1. In **ArcMap**, click on the **Scale Control** on the **Standard** toolbar and type `1:100,000` as the map scale.
2. Open the **Data Driven Pages** toolbar by navigating to **Customize** | **Toolbars** | **Data Driven Pages** from the main **ArcMap** menu.
3. Click on the **Data Driven Pages Setup** button.
4. **Enable Data Driven Pages** by clicking on the checkbox shown in the following screenshot. Also, select `GridIndexFeatures` as index **Layer** and `PageName` as **Name Field**. Make sure all the other values are set, as shown in the following screenshot:
5. Click on the Extent tab, choose Center and Maintain Current Scale, and then click on OK, as shown in the following screenshot:

![Set Up Data Driven Pages](image)

6. Save the map document file.

Creating the Locator Map

The Locator Map provide an overview of the spatial location of the current map within the context of a large geographic area. They provide an overview of the location of the current map in the series. In this section, you'll create a Locator Map for the layout view. We'll use a feature class copied from the grid index layer and edited to create a mask layer and a current page layer. The mask layer is used to gray out the features that are not in the current map while the current page layer highlights the current map:

1. In ArcMap, create a new data frame and name it Locator Map.
2. Copy the Basemap, Floodplains, and GridIndexFeatures layers from the Detail Map and paste them into the Locator Map. Rename the GridIndexFeatures layer to Page Labels.

3. Right-click on the Page Labels feature class and navigate to Data | Export Data. Save it to the same location as the grid index features and name it LocatorMask. Add the layer to the map. Your ArcMap table of contents should now appear, as shown in the following screenshot:

4. Right-click on the Page Labels feature class and select Properties.

5. Click on the Labels tab and choose PageName for the Label field and check the box next to Label features in this layer. Click on OK.
6. In the Locator Map data frame, click on the symbol for the LocatorMask feature class.

7. Using Symbol Selector, change Fill Color to black, Outline Width to 1, and Outline Color to white.

8. In the Properties dialog box for the LocatorMask feature class, click on the Display tab and type 60 for the transparency value. Click on OK.

9. Next, we'll create a layer that will serve as the highlight layer. Right-click on the LocatorMask feature class and select Copy.

10. Right-click on the Locator Map data frame and select Paste Layer.


12. Click on the symbol for Locator_Mask Current Page in the Locator Map data frame. From Symbol Selector, choose Hollow, set Outline Width to 1, and click on OK.

13. Now, we'll set the page definition queries for the LocatorMask and Locator_Mask Current Page layers so that they are displayed correctly.

14. Double-click on the LocatorMask layer and then click on the Definition Query tab.

15. Click on the Page Definition Query button and set the properties shown in the following screenshot. Click on OK when you're done:
16. Click on OK to exit the Layer Properties dialog box.

17. Your view should now appear as shown in the following screenshot. Any features that do not match the current page are drawn so that the area outside the current page is displayed as a gray mask:
18. Open the **Properties** dialog box for the Locator_Mask Current Page layer and select the **Definition Query** tab and then select the **Page Definition Query** button.

19. Set the properties shown in the following screenshot. Click on **OK** when you're done and then click on **OK** again to dismiss the **Layer Properties** dialog:

![Page Definition Query](image)

**What are page definition queries?**

Features can be filtered using the current data driven page. The attribute table must have a field that contains page names.

- **Enable**
- **Page Name Field:** `PageName`
- **Show features that**
  - **Match**
  - **Don't Match**

[OK] [Cancel]
20. Switch to **Layout View** and resize the **Locator Map** data frame on the layout so that it appears just below the main data frame, as shown in the following screenshot:
Adding dynamic text to the layout

The last thing that we need to do before writing our script to automate the process of generating the map book is add dynamic text to the layout. Dynamic text includes a title, a page number, a label for an adjacent page number, and other items added to the margins of the map. Dynamic text items are necessary when we have text items that will change for each map that is created. We'll also add a north arrow and scale bar:

1. First, we'll add the north arrow and scale bar. In **Layout View** inside **ArcMap**, add a north arrow and scale bars, as shown in the following screenshot. You don't have to select the same style as mine:
2. Now, we’ll add the page number. If necessary, open the Data Driven Pages toolbar and navigate to Page Text | Data Driven Page Name from the toolbar.

3. The page number will be placed directly in the center of the main data frame for the map. Drag it just above the first scale bar, as shown in the following screenshot. You may want to make the text larger than the default font size of 10. I’ve changed mine to 16 by right-clicking on the Text, and selecting Properties, and then clicking on the Change Symbol button.

4. Next, we’ll add the current page number. In the Data Driven Pages toolbar, navigate to Page Text | Data Driven Page Count. Like earlier, it will add a new text element to the center of the main data frame. Drag this text item to a new location somewhere in the margin.
5. You can add additional margin items, such as the date the map was saved, the author, the username, the coordinate system, and more by navigating to **Insert | Dynamic Text** from the main **ArcMap** menu. I've added the reference scale and date saved dynamic text items to my layout.

6. You'll also want to add a title to the map by navigating to **Insert | Title in ArcMap**. Call it **Floodplain Map for Lower Colorado River Planning District**.

7. Your map should look similar to what is shown in the following screenshot, though you may choose to add or change some of the items, as you see fit:

   ![Floodplain Map for Lower Colorado River Planning District](image)

8. Save your map document file.
Exporting the map series with ArcPy mapping

In this final section, we'll use the arcpy.mapping module along with a Python add-in to automate the process of exporting the map series to PDF files. Python add-ins are an easy way to add user interface items to ArcGIS Desktop. The Python Add-In Wizard is a utility that greatly simplifies the creation of the user interface items. In this section of the chapter, you'll use the Python Add-In Wizard to create a toolbar containing a button that will trigger the export of your map book to a PDF file. The following steps will guide you through, how to export map series with ArcPy mapping:

1. If necessary, download and install the Python Add-In Wizard from http://www.arcgis.com/home/item.html?id=5f3aefe77f6b4f61ad3e4c62f30bf3b#.f
2. In the folder where you unzipped the Python Add-In Wizard, find and double-click on the addin_assistant.exe file to start the wizard.
3. Choose or create a directory to be used as the add-in project root. Remember the name of the folder because you'll need it later. I'm going to use C:\MapBook.
4. The Python Add-In Wizard has two tabs: Project Settings and Add-In Contents. In the Project Settings tab, define the parameters, as shown in the following screenshot. Your working folder may be different than mine depending on your action in the last step, and you'll obviously want to change the author and company:

![Python Add-In Wizard](image)

5. Select the Add-In Contents tab.
6. Create a new toolbar by right-clicking on **Toolbars** and selecting **New Toolbar**. Define the caption as **Create Map Book** and click on **Save**. The toolbar will serve as a container for the button that triggers the creation of the map book.

7. Right-click on the new toolbar that you just created, and click on **New** button. Fill in the parameters, as shown in the following screenshot:

![Python Add-In Wizard](image)

8. Click on **Save**.
9. Click on the **Open Folder** button to display the folders and files that the **Python Add-In Wizard** has created. You should see something very similar to what is shown in the following screenshot:

![Open Folder screenshot](image)

10. Now, it's time to add the Python code that will be executed when the button is clicked on. Go to the Install folder, as shown in the preceding screenshot, and you should see a single Python script called `MapBook_addin.py`. Open this file in your Python development environment.

11. Find the `onClick(self)` method shown in the following code. This method is executed when the button is clicked:

   ```python
   import arcpy
   import pythonaddins

   class CreateMapBook(object):
       """Implementation for MapBook_addin.button (Button)""
       def __init__(self):
           self.enabled = True
           self.checked = False
       def onClick(self):
           pass
   ```

12. Remove the `pass` statement from the `onClick()` method.

13. Add an `import` statement for the `arcpy.mapping` and `os` modules. In this line of code, we're importing the mapping module and assigning it to a variable called `MAP`. It will make referencing the functions in the mapping module easier as we proceed:

   ```python
   import arcpy
   import pythonaddins

   import arcpy.mapping as MAP
   import os
   ```
14. The rest of our code will go inside the `onClick()` method. Use the `pythonaddins.SaveDialog()` function to display a dialog box that will allow the end user to save the map book as a PDF file. Also, create an output directory variable:

```python
def onClick(self):
    # Create an output directory variable
    finalpdf_filename = pythonaddins.SaveDialog("Save Map Book", "MapBook.pdf", r"C:\ArcGIS_Blueprint_Python\ch3")
    outDir = os.path.split(finalpdf_filename)[0]
```

15. If you are using ArcGIS Desktop 10.3, create a new `pythonaddinsProgressDialog` object using a `WITH` statement. This will display a progress dialog while the map book is being created. Set the title, description, and animation properties. If you are using ArcGIS Desktop 10.1 or 10.2, please skip this step and proceed to step 16:

```python
def onClick(self):
    # Create an output directory variable
    finalpdf_filename = pythonaddins.SaveDialog("Save Map Book", "MapBook.pdf", r"C:\ArcGIS_Blueprint_Python\ch3")
    outDir = os.path.split(finalpdf_filename)[0]

    with pythonaddinsProgressDialog as dialog:
        dialog.title = "Progress Dialog"
        dialog.description = "Creating a map book....this will take awhile!"
        dialog.animation = "File"
```

16. Create a new, empty PDF document in the specified output directory with `pythonaddinsProgressDialog` as dialog:

```python
def onClick(self):
    # Create an output directory variable
    finalpdf_filename = pythonaddins.SaveDialog("Save Map Book", "MapBook.pdf", r"C:\ArcGIS_Blueprint_Python\ch3")
    outDir = os.path.split(finalpdf_filename)[0]

    with pythonaddinsProgressDialog as dialog:
        dialog.title = "Progress Dialog"
        dialog.description = "Creating a map book....this will take awhile!"
        dialog.animation = "File"

    # Create a new pdf document in the output directory
    if os.path.exists(finalpdf_filename):
        os.remove(finalpdf_filename)
    finalPdf = MAP.PDFDocumentCreate(finalpdf_filename)
```
17. To simplify the process, a map title page (TitlePage.pdf), and overview map page (IndexMap.pdf), have also been created for you. These files are located in your C:\ArcGIS_Blueprint_Python\ch3 folder. Add the title and index pages:

```python
# Create a new, empty pdf document in the specified output directory
if os.path.exists(finalpdf_filename):
    os.remove(finalpdf_filename)
finalPdf = MAP.PDFDocumentCreate(finalpdf_filename)

# Add the title page to the pdf
finalPdf.appendPages(r"C:\ArcGIS_Blueprint_Python\ch3\TitlePage.pdf")

# Add the index map to the pdf
finalPdf.appendPages(r"C:\ArcGIS_Blueprint_Python\ch3\IndexMap.pdf")
```

18. Export the **Data Driven Pages** to a temporary PDF file, and then add it to the final PDF:

```python
# Add the title page to the pdf
finalPdf.appendPages(r"C:\ArcGIS_Blueprint_Python\ch3\TitlePage.pdf")

# Add the index map to the pdf
finalPdf.appendPages(r"C:\ArcGIS_Blueprint_Python\ch3\IndexMap.pdf")

# Export the Data Driven Pages to a temporary pdf and then add it to the final pdf.
mxd = MAP.MapDocument("CURRENT")
ddp = mxd.dataDrivenPages

temp_filename = outDir + r"\tempDDP.pdf"

if os.path.exists(temp_filename):
    os.remove(temp_filename)
ddp.exportToPDF(temp_filename, "ALL")
finalPdf.appendPages(temp_filename)
```

19. Update the properties of the final PDF and save it:

```python
if os.path.exists(temp_filename):
    os.remove(temp_filename)
ddp.exportToPDF(temp_filename, "ALL")
```
finalPdf.appendPages(temp_filename)
# Update the properties of the final pdf.
finalPdf.updateDocProperties(pdf_open_view="USE_THUMBS", pdf_layout="SINGLE_PAGE")

# Save your result
finalPdf.saveAndClose()

20. Remove the temporary Data Driven Pages file:

# Update the properties of the final pdf.
finalPdf.updateDocProperties(pdf_open_view="USE_THUMBS", pdf_layout="SINGLE_PAGE")

# Save your result
finalPdf.saveAndClose()

# remove the temporary data driven pages file
if os.path.exists(temp_filename):
    os.remove(temp_filename)

21. Check your script against the solution file found in C:\ArcGIS_Blueprint_Python\ch3\scripts\CreateMapBook.py for accuracy.

22. Save your script and close the file.

23. Now, it's time to install and test your Python add-in. Inside the main folder where you created the add-in (C:\MapBook, in my case), you will find a Python script called makeaddin.py. Double-click on this file to open.

24. A new .esriaddin file called MapBook.esriaddin will be created in the same folder, as shown in the following screenshot:
25. To install your new add-in using ArcGIS Desktop, double-click on the MapBook.esriaddin file to launch the **Esri ArcGIS Add-In Installation Utility** window seen in the following screenshot:

![Esri ArcGIS Add-In Installation Utility](image)

26. Click on **Install Add-In**. If everything is successful, you should see a success message.

27. Open ArcMap with the Floodplains.mxd file that you created earlier in the chapter, to test the add-in. Navigate to **Customize | Create Map Book** from the **ArcMap** menu. This will display the **Create Map Book** add-in, as shown in the following screenshot:
28. Click on the Create Map Book button to execute the code.

**Summary**

ArcGIS Desktop provides the ability to efficiently create a map book through a combination of Data Driven Pages along with an arcpy.mapping script. With a single map document file, you can use the Data Driven Pages toolbar to create a basic series of maps using the layout view along with your operational data and an index layer. The index layer contains features that will be used to define the extent of each map in the series. However, if you need to include additional pages in the map book, including a title page, an overview map, and other ancillary pages, you'll need to combine the output from the Data Driven Pages toolbar with the functionality provided by the arcpy.mapping module. With the arcpy.mapping module, you can automate the export of the map series and append the ancillary files into a single map book document. While it is certainly programmatically to generate the entire map book using only Python and the arcpy.mapping module, it is more efficient to use a combination of programming and the Data Driven Pages toolbar.
Where to buy this book

You can buy ArcGIS Blueprints from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.