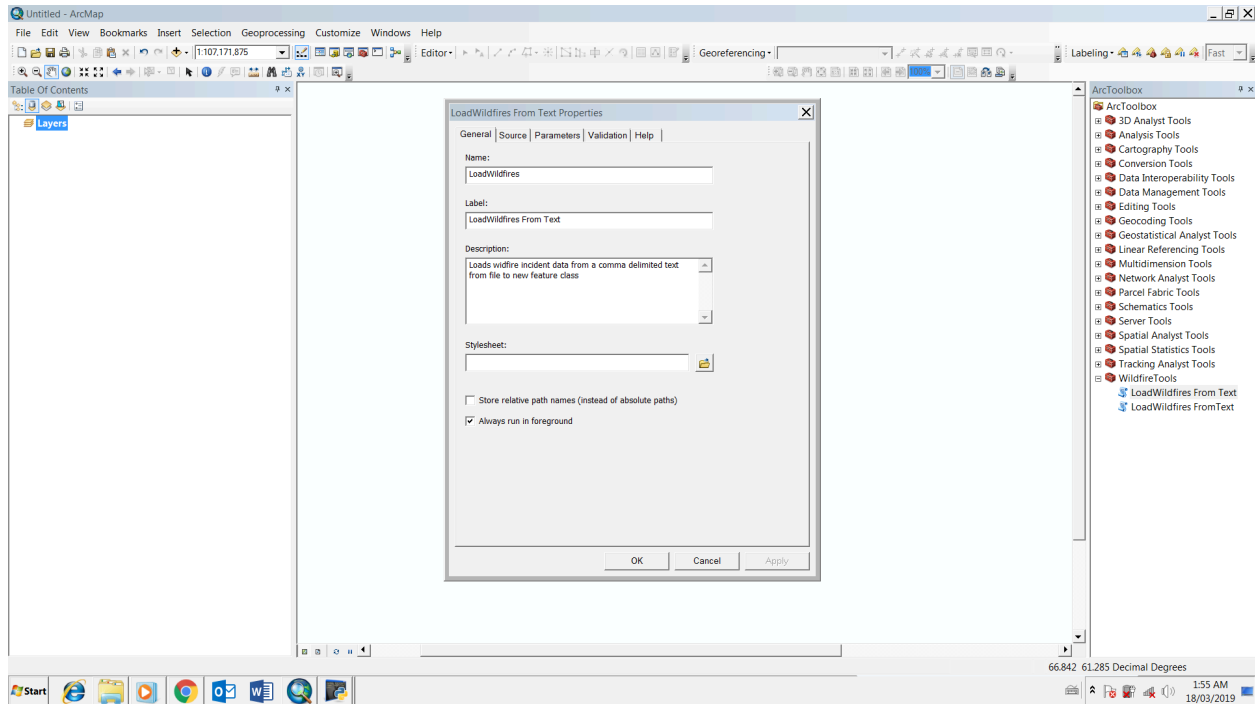
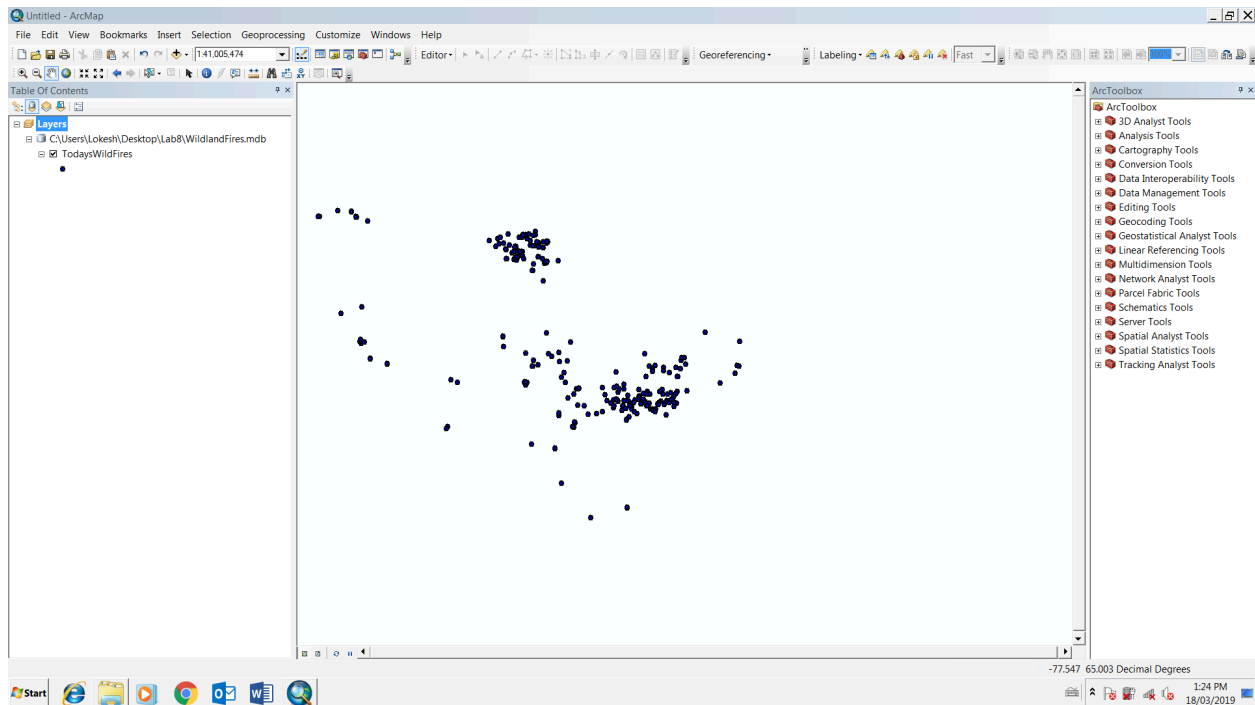


# LAB 8

## Deliverable 1 :

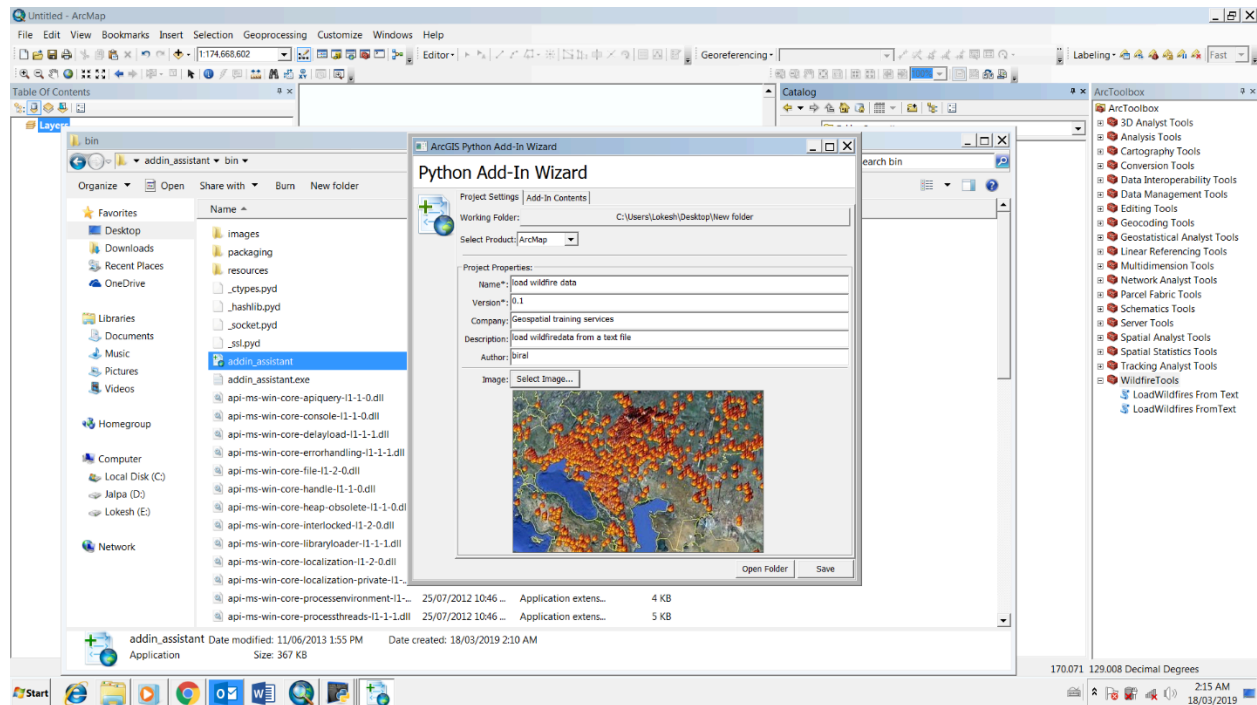


## Deliverable 2:

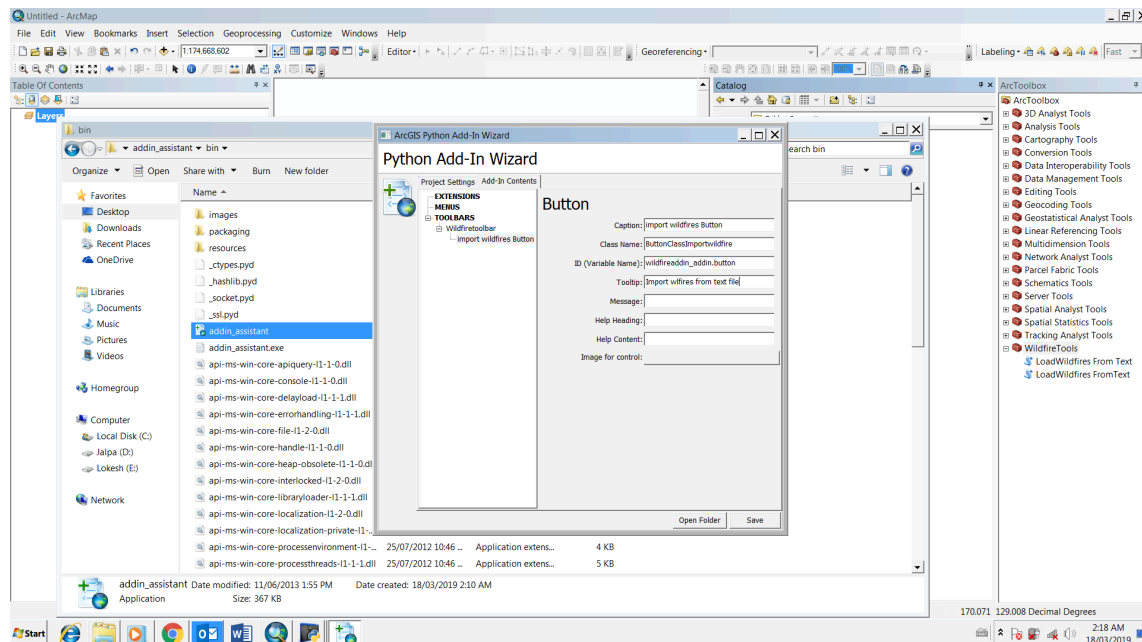


### Deliverable 3:

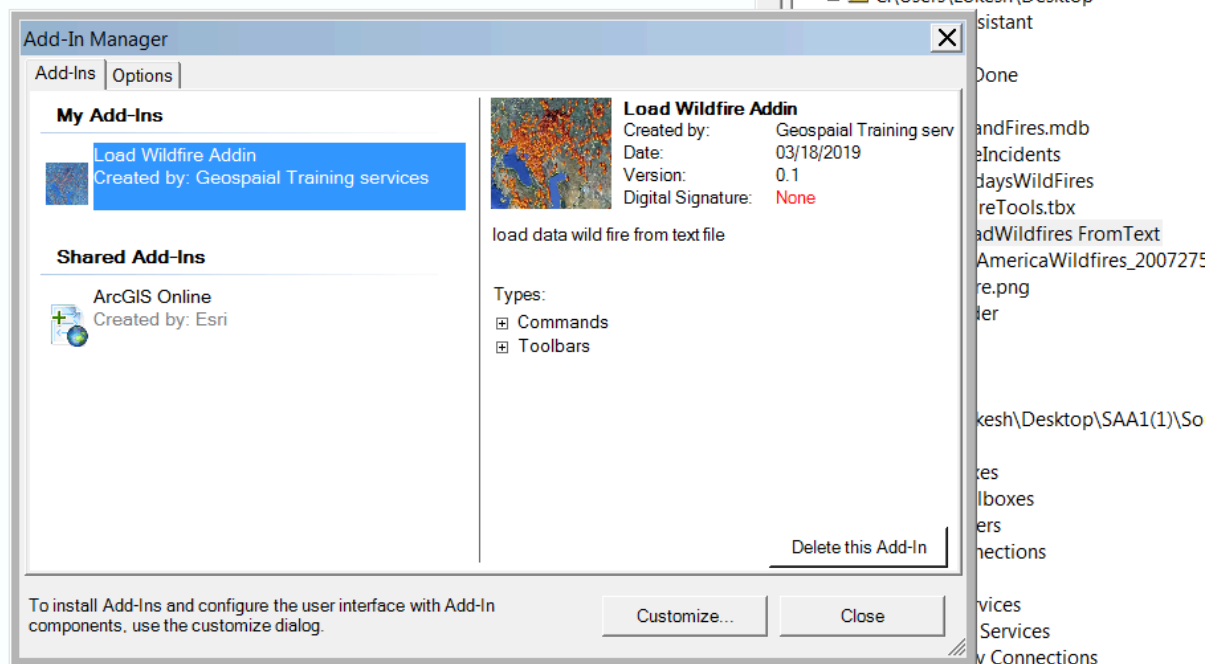
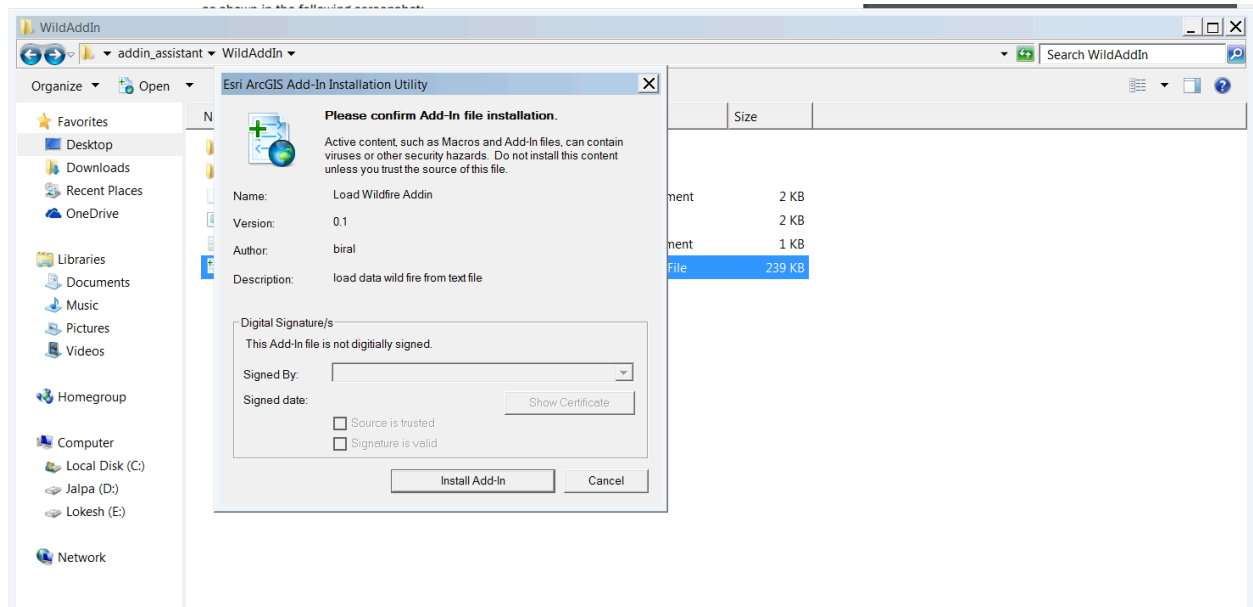
#### Downloading and installing the Python Add-In wizard

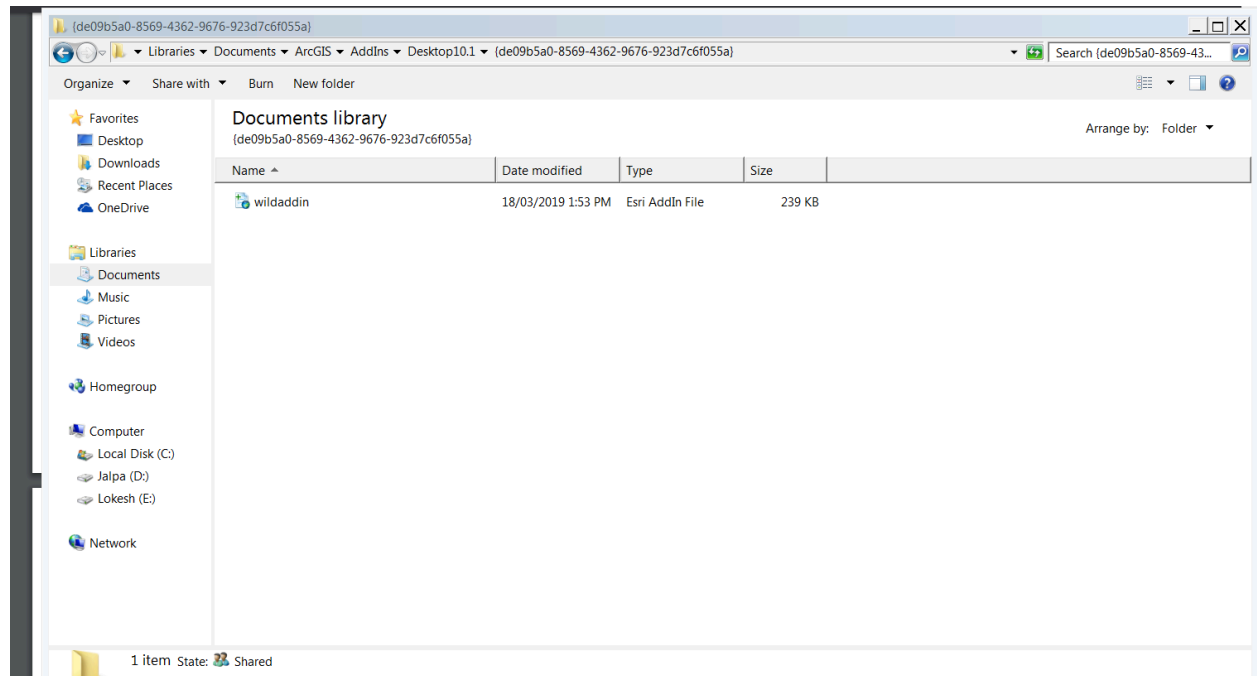


#### Creating a button add-in



## INSTALLING AND TESTING





#### Deliverable 4: [Creating a tool add-in](#)

## Python Add-In Wizard



Project Settings | Add-In Contents

Working Folder: C:\Users\Lokesh\Desktop\addin\_assistant\Generate\_Random\_Points

Select Product: ArcMap ▼

### Project Properties:

Name\*: Generate\_Random\_Points

Version\*: 0.1

Company: GEOSPATIAL TRAINING SERVICES

Description: GENERATE RANDOM POINTS FROM USER DATA

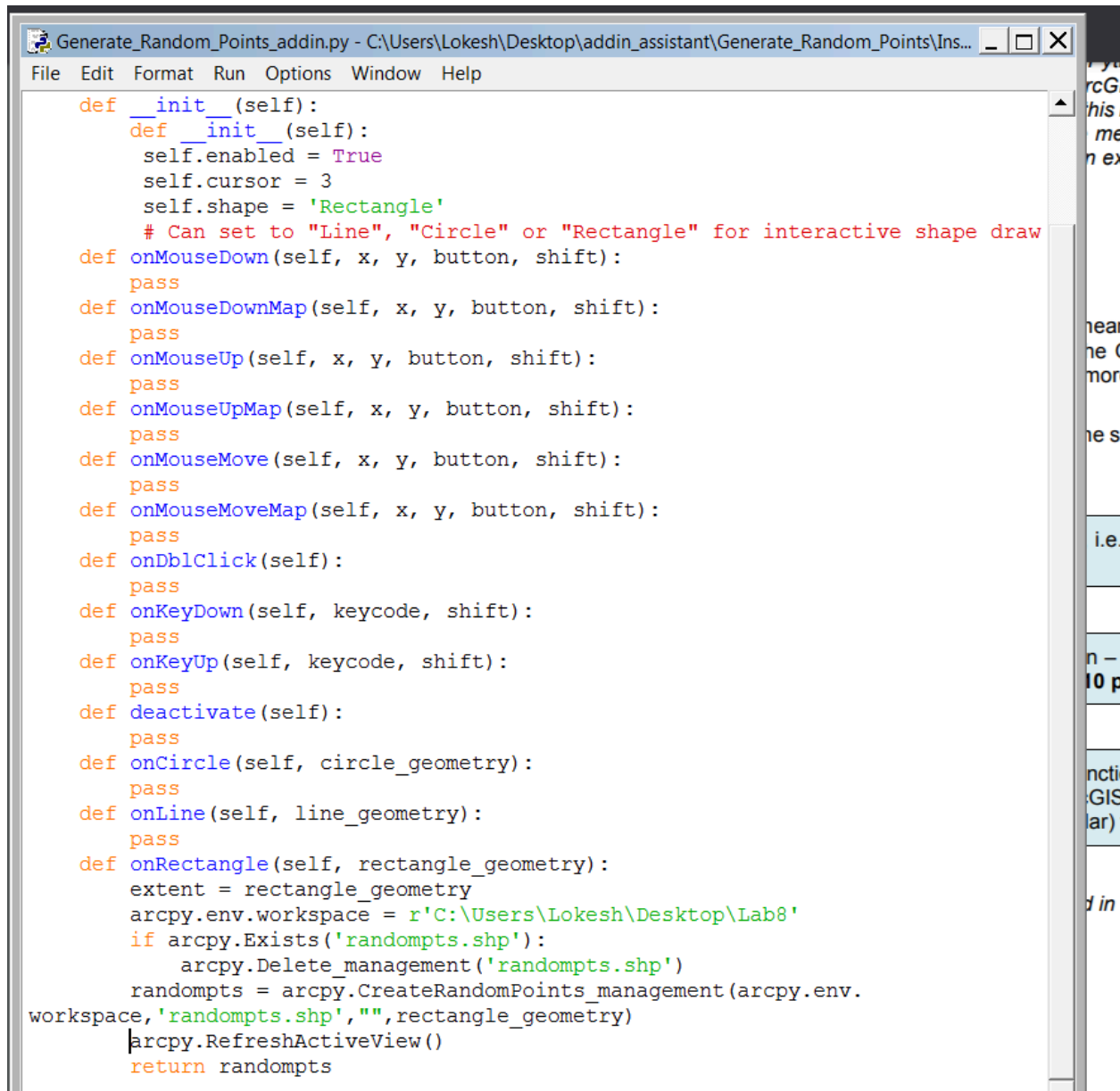
Author: BIRAL

Image: Select Image...



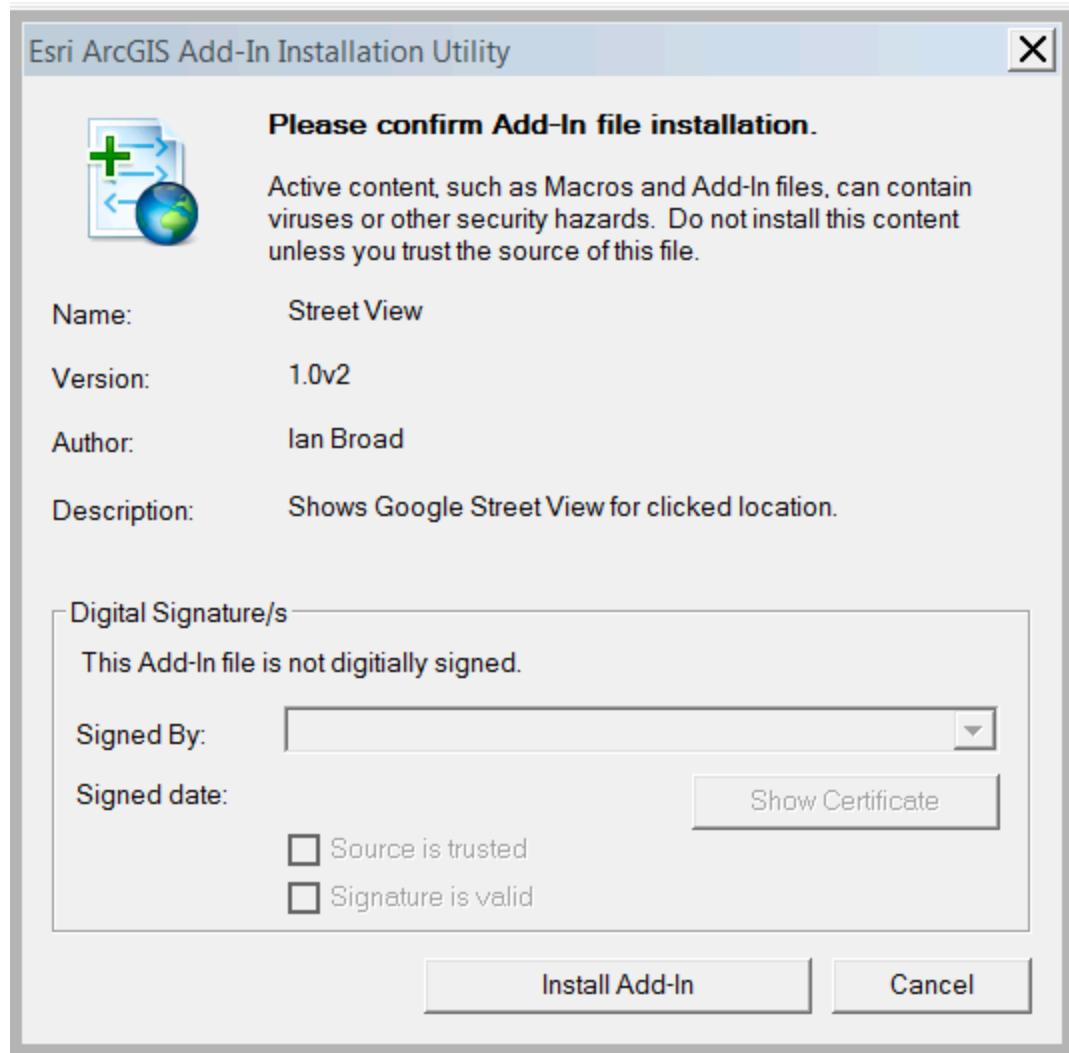
Open Folder

Save

A screenshot of a Python script editor window titled "Generate\_Random\_Points\_addin.py - C:\Users\Lokesh\Desktop\addin\_assistant\Generate\_Random\_Points\Ins...". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code is written in Python and defines a class with various methods for interacting with a GIS environment. The methods include initialization, mouse and keyboard event handling, and geometry creation. The code is as follows:

```
def __init__(self):
    def __init__(self):
        self.enabled = True
        self.cursor = 3
        self.shape = 'Rectangle'
        # Can set to "Line", "Circle" or "Rectangle" for interactive shape draw
    def onMouseDown(self, x, y, button, shift):
        pass
    def onMouseDownMap(self, x, y, button, shift):
        pass
    def onMouseUp(self, x, y, button, shift):
        pass
    def onMouseUpMap(self, x, y, button, shift):
        pass
    def onMouseMove(self, x, y, button, shift):
        pass
    def onMouseMoveMap(self, x, y, button, shift):
        pass
    def onDoubleClick(self):
        pass
    def onKeyDown(self, keycode, shift):
        pass
    def onKeyUp(self, keycode, shift):
        pass
    def deactivate(self):
        pass
    def onCircle(self, circle_geometry):
        pass
    def onLine(self, line_geometry):
        pass
    def onRectangle(self, rectangle_geometry):
        extent = rectangle_geometry
        arcpy.env.workspace = r'C:\Users\Lokesh\Desktop\Lab8'
        if arcpy.Exists('randompts.shp'):
            arcpy.Delete_management('randompts.shp')
        randompts = arcpy.CreateRandomPoints_management(arcpy.env.
workspace, 'randompts.shp', "", rectangle_geometry)
        arcpy.RefreshActiveView()
        return randompts
```

Deliverable 6: Screen capture of the installed Python add-in.



Deliverable 7 :

Owner: ian\_broad ian\_broad

Description: This version is for 10.1, please visit the following link for the 10.2 and 10.3 version.

Allows a user to click a road within ArcMap, and the browser will open and load the Google Street View data for that location. Created: 11 Feb 2015 Updated: 5 Jun 2015

He works for an electric utility, and Google Street View is a very handy tool to do a quick “field check” to verify, for example, equipment or pole specifications. It can save a lot of time, and gas! Of course, the Street View data may not always accurately depict what’s in the real world since things change, but it’s still a very good source to have available.

User review : garixmartin64 commented 4 months ago

The screenshot shows the ArcMap application window. The main map area displays a map of North America with numerous black dots representing wildfire locations. The map includes labels for major cities and states. The interface includes a top menu bar with options like File, Edit, View, Bookmarks, Insert, Selection, Geoprocessing, Customize, Windows, and Help. Below the menu bar is a toolbar with various icons. On the left side, there is a 'Table of Contents' pane showing the 'Layers' list, which includes 'C:\Users\Lokesh\Desktop\Lab8\WildlandFires.mdb' and 'World Street Map'. On the right side, there is a 'Catalog' pane showing the project structure, including folders like 'bin', 'images', 'resources', 'Generate\_Random\_Points', 'WildAddin', 'Images', 'Almost\_Done', 'New folder', 'working', 'New folder', 'Lab8', 'WildlandFires.mdb', 'FireIncidents', 'Today's Wildfires', 'WildfireTools.txt', and 'NorthAmericaWildfires\_2007275.txt'. Below the Catalog is an 'ArcToolbox' pane listing various tools such as '3D Analyst Tools', 'Analysis Tools', 'Cartography Tools', 'Conversion Tools', 'Data Interoperability Tools', 'Data Management Tools', 'Editing Tools', 'Geocoding Tools', 'Geostatistical Analyst Tools', 'Linear Referencing Tools', 'Multidimension Tools', 'Network Analyst Tools', 'Parcel Fabric Tools', 'Schematics Tools', 'Server Tools', 'Spatial Analyst Tools', 'Spatial Statistics Tools', and 'Tracking Analyst Tools'.

**Deliverable 8:**

If it is required to make a customization that performs an action in response to an event, or requires the use of the mouse to interact with the display, you should consider making an add-in. An example is a tool that requires the user to click and drag a rectangle over a map to define an area of interest. Another example is an application extension that saves the map document automatically anytime a layer is added or removed from the table of contents. (Python add-in, n.d.)

- You define parameters through the wizard
- You create validation code that lives in the toolbox
- Plus, you create and maintain the source script separately

All of these parts are segregated and more difficult to manage collectively. In a Python toolbox, parameter definitions, validation code, and the source code are all handled with Python code, making it easier to create and maintain Python tools. Additionally, Python toolboxes support capabilities that script tools do not, such as value tables, composite data types, and custom license checking. (comparition, n.d.) Python toolboxes support an `isLicensed` method that allows you to control whether a tool can be opened based on your criteria