



Esri International Developer Summit
Palm Springs, CA

Creating Geoprocessing Tools in a Python Toolbox

David Wynne

Abstract - Creating Geoprocessing Tools in a Python Toolbox

- Python toolboxes offer a way to efficiently build custom geoprocessing tools entirely in Python.
- A Python toolbox and the tools within look and act just like toolboxes and tools created in any other way.
- Join me as we step through examples of creating Python toolbox tools from scratch.



<http://www.esri.com/events/devsummit/session-rater>

Why we create tools

- *Tools solve problems*
- **Tools are:**
 - Easy to share
 - Generic
 - **Becomes part of the geoprocessing framework**
 - Python, Dialogs, ModelBuilder, Services
 - **Basic validation saves coding checks**



Python toolbox tool structure

A tool does 3 types of work:

- Parameters

```
class Tool(object):  
    def __init__(self):  
        """Define the tool (tool name is the name of the class)."""  
        self.label = "Tool"  
        self.description = ""  
        self.canRunInBackground = False
```

```
    def getParameterInfo(self):  
        """Define parameter definitions"""  
        params = None  
        return params
```

- Validation

```
    def isLicensed(self):  
        """Set whether tool is licensed to execute."""  
        return True  
  
    def updateParameters(self, parameters):  
        """Modify the values and properties of parameters before internal  
        validation is performed. This method is called whenever a parameter  
        has been changed."""  
        return  
  
    def updateMessages(self, parameters):  
        """Modify the messages created by internal validation for each tool  
        parameter. This method is called after internal validation."""  
        return
```

- Source

```
    def execute(self, parameters, messages):  
        """The source code of the tool."""  
        return
```

Creating a Python toolbox

```
class Toolbox(object):
    def __init__(self):
        """Define the toolbox (the name of the toolbox is the name of the
        .pyt file)."""
        self.label = "LinesToPoints"
        self.alias = "topoints"

        # List of tool classes associated with this toolbox
        self.tools = [CreatePointsAlongLine]

class CreatePointsAlongLine(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Create Points Along Line"
        self.description = ""
        self.canRunInBackground = False

    def getParameterInfo(self):
        """Define parameter definitions"""
        in_lines = arcpy.Parameter(
            displayName="Input Line Features",
            name="in_line_features",
            datatype="Feature Layer",
            parameterType="Required",
            direction="Input")
        in_lines.filter.list = ['Polyline']

        out_points = arcpy.Parameter(
            displayName="Output Point Feature Class",
            name="out_point_features",
            datatype="Feature Class",
            parameterType="Required",
            direction="Output")

        interval = arcpy.Parameter(
            displayName="Interval",
            name="interval",
            datatype="Double",
            parameterType="Required",
            direction="Input")

        return [in_lines, out_points, interval]
```

Getting more information

- esriurl.com/pythontoolbox

ArcGIS Help (10.2 and 10.2.1)
Resource Center

Welcome to the ArcGIS Help Library

- What's New
- Desktop
 - Mapping
 - Editing
 - Geoprocessing
 - Introduction
 - Commonly used tools
 - Finding tools
 - Executing tools
 - Managing tools and toolboxes
 - Creating tools
 - A quick tour of creating custom tools
 - Running custom (model or script) tools in the background
 - Creating tools with ModelBuilder
 - Creating tools with Python**
 - A quick tour of creating tools with Python
 - What is a script tool?
 - What is a Python toolbox?
 - Comparing custom and Python toolboxes
 - Basic principles of creating tools with Python**
 - Understanding script tool parameters
 - Understanding messages in script tools
 - Understanding the progress dialog box in script tools
 - Understanding validation in script tools
 - Creating script tools in a custom toolbox
 - Creating tools in a Python toolbox
 - Creating a new Python toolbox
 - The Python toolbox template
 - Editing a Python toolbox
 - Defining a tool in a Python toolbox
 - Defining parameters
 - Customizing tool behavior
 - Accessing parameters within a Python toolbox
 - Writing messages in a Python toolbox
 - Documenting a tool in a Python toolbox



Understanding our world.