

Professional Python Development in ArcGIS Pro

Clinton Dow

High Quality PDF (2MB)

Introduction

Today's Presentation

Overview of ArcGIS Pro

- Work accomplished in the past year
- Tour of Documentation and Code samples
- Esri's open source participation and integration
- Explanation of Tool creation concepts

Product Engineer - Geoprocessing at Esri California (1 year 8 months)

- Python
- ArcPy
- Conda Integration
- C#/WPF
- Python Backstage
- Charts and Graphs
- Presentations

GIS Developer at Matrix Solutions in Calgary (2 years)

- Civil Engineering/Environmental Consultant Firm
- Created several dozen custom Geoprocessing tools in Python
- Customized ArcGIS with Python and C#/WPF



Esri ArcGIS

Setting up ArcGIS Pro

Windows Only

Free Windows VMs - <http://bit.ly/FreeWindowsVM>

Free ArcGIS Pro - (<http://bit.ly/ArcPyProTrial>

ArcPy Documentation - <http://bit.ly/ArcPyDocs>

Why ArcGIS? - Powerful and Proven technology

ARC/INFO released in 1982

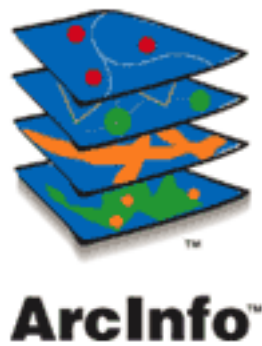


Figure 1:

ArcGIS in 1999

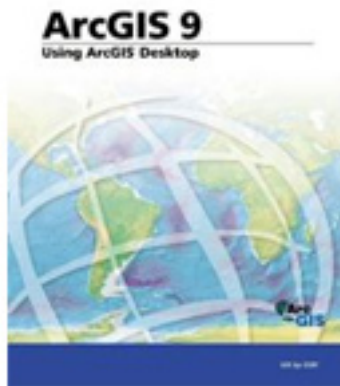


Figure 2:

Why ArcGIS? - Powerful and Proven technology

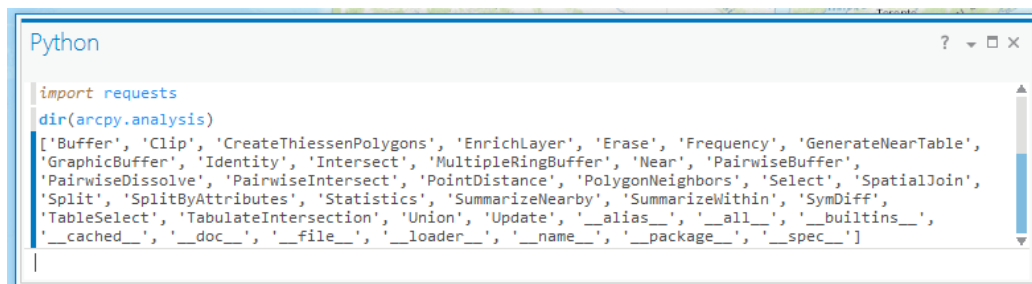
ArcGIS Pro in 2015



Extensive Python Support:

- ArcPy
- ArcGIS Python API

Integrated Python Interpreter



```
Python
import requests
dir(arcpy.analysis)
['Buffer', 'Clip', 'CreateThiessenPolygons', 'EnrichLayer', 'Erase', 'Frequency', 'GenerateNearTable',
'GraphicBuffer', 'Identity', 'Intersect', 'MultipleRingBuffer', 'Near', 'PairwiseBuffer',
'PairwiseDissolve', 'PairwiseIntersect', 'PointDistance', 'PolygonNeighbors', 'Select', 'SpatialJoin',
'Split', 'SplitByAttributes', 'Statistics', 'SummarizeNearby', 'SummarizeWithin', 'SymDiff',
'TableSelect', 'TabulateIntersection', 'Union', 'Update', '__alias__', '__all__', '__builtins__',
'__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__']
```

Figure 3:

Why ArcGIS? - Documentation

Second-to-None Documentation

Esri Blog - <https://blogs.esri.com/esri/arcgis/category/subject-python/>

Esri Press - <http://esripress.esri.com/display/index.cfm>

ArcGIS Help - <https://pro.arcgis.com/en/pro-app/help/main/welcome-to-the-arcgis-pro-app-help.htm>

GeoNet - <https://geonet.esri.com>

- Esri Forums/Social Network
- Python Community
- <https://geonet.esri.com/community/developers/gis-developers/python>

Why ArcGIS? - Versatility

Supports multiple GIS Applications

- ArcGIS Desktop (Includes ArcGIS Pro)
- ArcGIS Enterprise
- ArcGIS Online

Includes over 1000 Geoprocessing Tools

- From simple (Buffer, Spatial Join)
- To Complex (Space-Time Cube, Generate Tessellation)

Advanced Projection Engine

- Supports dozens of projections/transformations

Why ArcGIS? - Geospatial Data

Rich source of GIS Data

- ArcGIS Online provides thousands of ready-to-go data sets.
 - Living Atlas of The World
 - ArcGIS Online

Large selection of Plugins

- <https://marketplace.arcgis.com/>

Why ArcGIS? - Industry Standard

Included suite of cartographic symbology.

Industry Standard solution

- Used by Governments, Fortune 500 companies and Individuals.
 - Swiss Gov't examples (Canton Governments, Swiss Institute of Forest, Snow and Landscape Research etc)

Why ArcGIS Pro?

Modern View for ArcGIS:

- .Net 4.5 concurrency model
- Integrated 2D and 3D views
- In active development, New Features
- Modern Python Experience
- Supports environments and packages via Conda

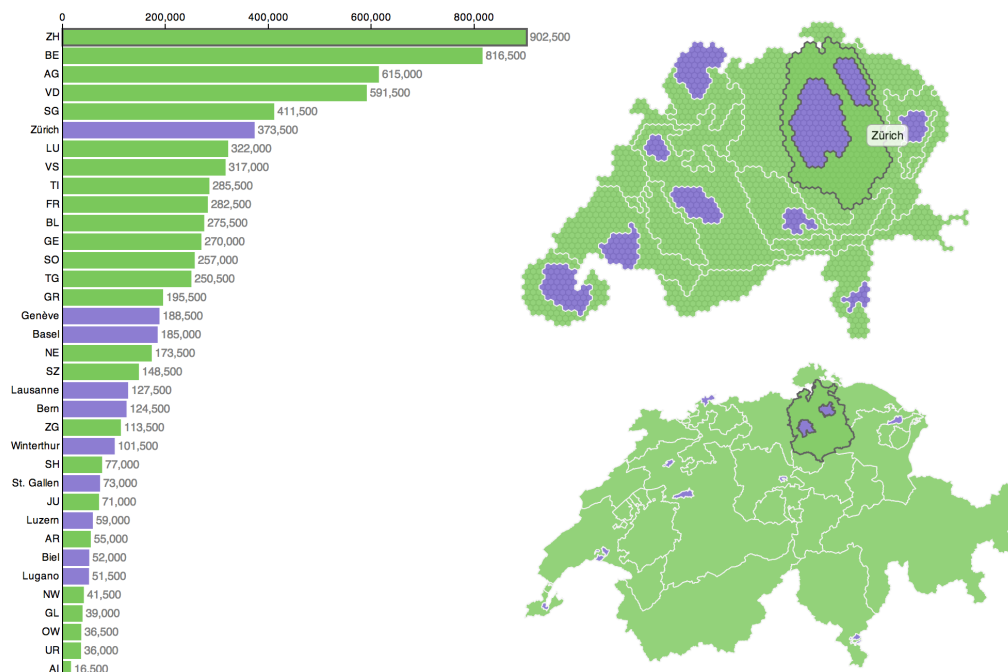


Figure 4:

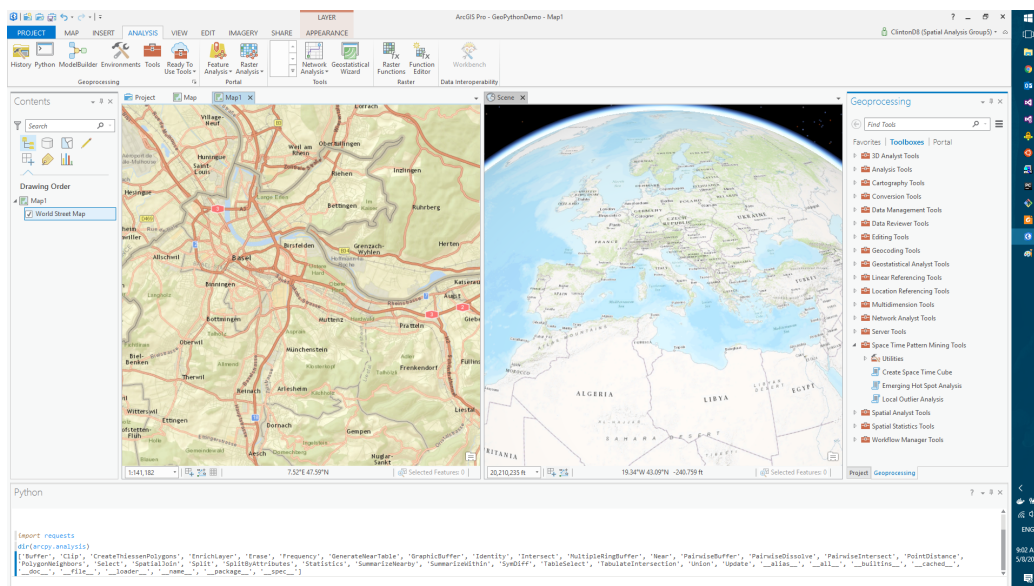


Figure 5:

ArcGIS Pro

Python Packages and Environments

Package Management on Windows

Using pip, wheels, virtualenvs

- Packaged with distributions of Python
- These tools handle the harder problem of system dependencies, considered out of scope by Python packagers — does it end up in site-packages?
- Package devs: On OSX and Linux, ‘easy’ to get the deps! Use a system package manager (e.g. apt, brew, yum)
- Included Compiler (e.g. clang, gcc).

Virtual Environments

What are Virtual Environments

- Self-contained instances of Python
- Separate from main Python installation
- Can contain a unique set of packages
- Useful when working on multiple projects at one time

What about Windows?

Windows lacks broadly used package management

- Only developers have a C compiler on their machine (Typically Visual Studio)
- A hard problem for many organizations to reliably solve
- “Works on my machine but not yours” problem.
- Supporting users takes up valuable dev time
- No guarantee that customers will be supported

Enter Conda



Why Conda?

Scientific Python community identified that there was a gap not being addressed by the core Python infrastructure, limiting their ability to get packages into the hands of users

Industry standard built by people who care about this space — Continuum Analytics

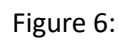
Handles dependencies for many languages (C, C++, R and of course Python)

Built for Python first, but it really solves a much broader infrastructural issue.

Conda in ArcGIS Pro

Significant effort has been made at Esri to integrate the conda package manager and virtual environment experience into the ArcGIS Pro experience.

- ## Conda in ArcGIS Pro



Coming Soon

- Public Channels
- Private Channels

- Currently only available in cmd line
- Share packages with colleagues and customers

9

- Conda packages and environments fully supported on Enterprise
- ArcGIS Python API Notebooks served on Enterprise

Using Packages to our Advantage

Open Source Ecosystem

The Python Ecosystem includes thousands of open-source packages

Esri is using several packages in ArcGIS

- NumPy
- SciPy
- matplotlib
- Pandas (coming soon)

Open Source Ecosystem

Automate or Extend your ArcGIS capabilities

- Thousands of open-souce Python packages
- Conda-Forge community

Easily package and share your work

- Conda-build
- Anaconda.org
- Host packages on network

Setting up a Development Environment

What can we install? Not just scientific packages.

- Documentation
- Datasets
- GUI toolkits (PyQt, TKinter)
- Database Drivers (psycopg2)

- C++ Libraries (Boost)
- IDEs
 - Spyder
 - Jupyter

Setting up a Production Environment

Proven environments that will 'Just work'

- Solves 'works on my machine' problem
- Metapackages
 - Packages which only have requirements
- Environment.yaml
 - More complex requirements.txt

Working with Customers

Requirements Analysis

Determining user expectations for a new or modified product

- Identify Stakeholders
- Eliciting Requirements
- Stakeholder interviews
- Analyzing Requirements
- Clear
- Complete
- Consistent
- Recording Requirements
- User Stories
- Use Cases

Test Driven Development

Turning valid requirements into testable code

- Identifying units of work
- Preexisting tools?
- Defining functional extent of units
- Writing tests to encapsulate functionality



Figure 7:

TDD in Python

Several test suites available

- Pytest
- Nose
- unittest

Efficient Testing with ArcPy

Extra considerations for Geospatial tools

- Data set types
- Feature Class (Geodatabase)
- Shapefiles (Esri Open-Source Format)

- GeoJSON (Generic Open-Source Format)
- Projections
- Data Frame (View)
- Data Sets
- UI interactions
- Data Validation

Version Control as a Communication Tool

What is Git?

Git

A distributed version control system.

- Originally made for linux developers
- Arguably the most popular open-source version control
- Heavily used in Conda packaging workflow

Why Git?

Features of Git that support Conda packaging

- Commit hash as version number
- Excellent for testing development versions
- Packages can 'cherry-pick' parts of repositories
- Production packages may not include tests etc.
- Develop with a team
- Intuitive branching and merging

What is Github

Github

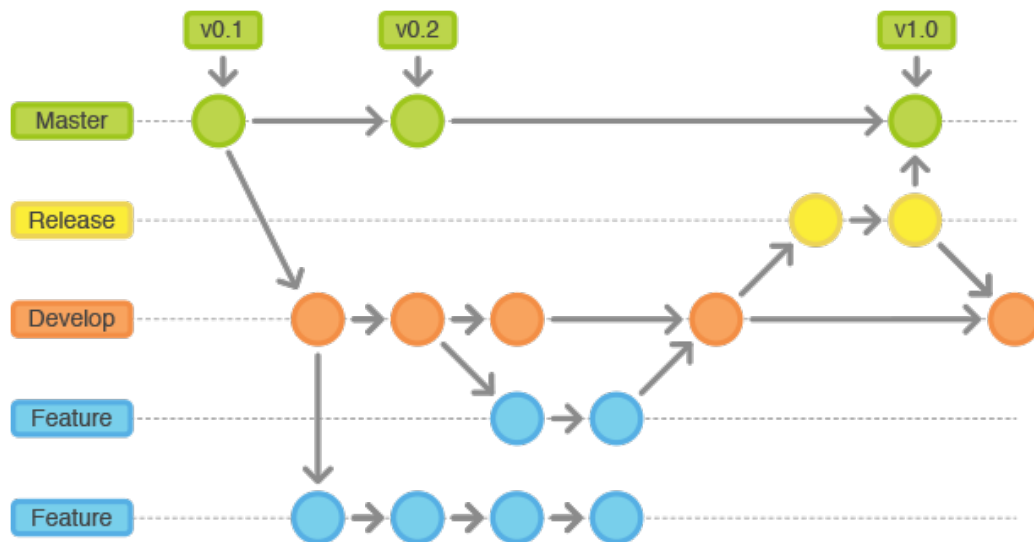


Figure 8:

Github promotes 'Social Coding' a combination of Version Control and Social Media

- Ease of Collaboration
- Ease of Communication
- Ease of Distribution

Esri on Github

- <https://www.github.com/Esri>
- <https://www.github.com/arcpy>

Github as a Communication Tool

Github offers several features which enhance communication - Issue tracking - Tags - User Notifications - Repository Forking - Create your own version of code

Beginner Python - Python Features in ArcGIS Pro UI

Python Window

Quick means of interacting with ArcGIS Pro using Python

- Good way to practice and test concepts
- View help documentation for ArcPy functions
- Import and Export Python Scripts

Interactive Selections

- Operate on data selected on Map
- Automatically show result in Map

Field Calculator

Leverage the power of Python to enhance your data sets

- Utilize Python to calculate new fields
- Available through Pro UI
- Extract code to Scripts
- Data access ArcPy module
- Calculate fields using Cursors without UI interaction

Intermediate Python - Creating Tools in ArcGIS Pro

TBX Toolboxes

The original ArcGIS toolbox. - Created via the ArcGIS GUI - Tool Validation not set in Python script - Set via 'Properties' menu of tool in ArcGIS

Python Toolboxes

ArcGIS toolboxes revisited in Python - All steps defined in a .pyt file - Ease of testing &

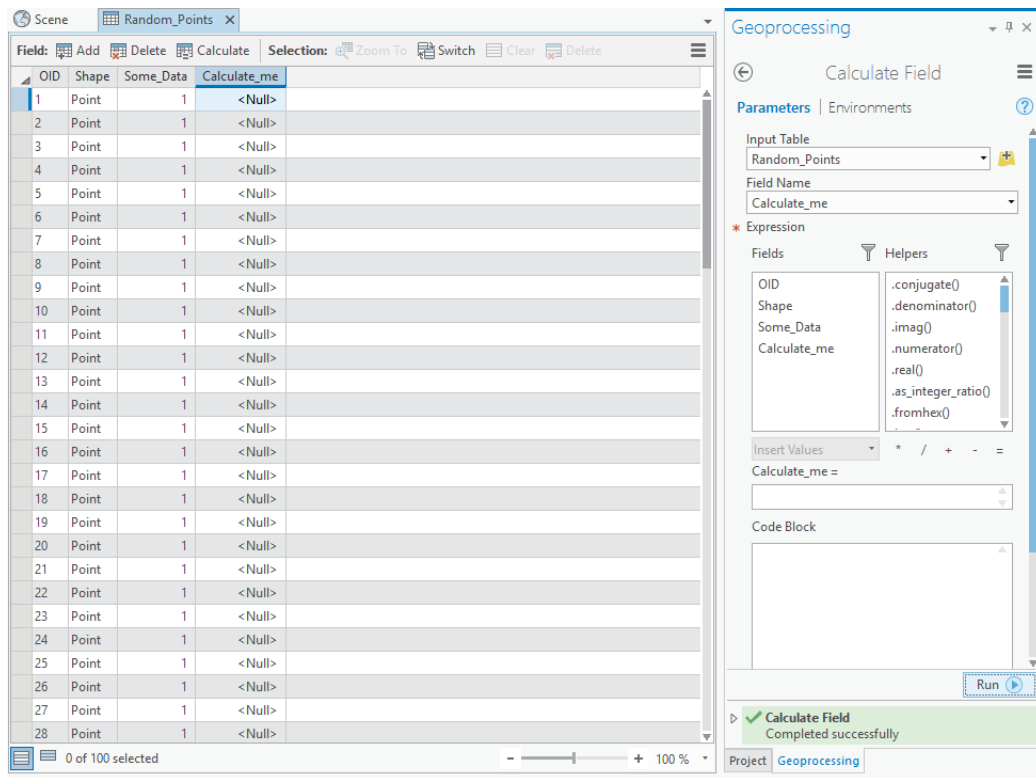


Figure 9:

debugging - Validation defined within script - Works with Python IDEs - Define .pyt as a Python file type in settings - Syntax Highlighting - Autocomplete

Python Toolboxes

Python Toolbox (.pyt) Files - Toolbox Class - **init** - self.tools - Tool Class - **init** - Validation - Logic

Tool Validation

ArcGIS Supports Dozens of Data Types - Ensure the inputs supplied by the user are valid - Dynamically populate fields with values - Inform users when unexpected or unusable data is present

Input/Output Parameters

Defining Parameters Working with Input Parameters - GetInputParameter - GetInputParameterAsText - Parameters from Command Line Working with Output Parameters - Schema - ParameterDependencies

getParameterInfo

Populate the values of a Tool's Parameters - Called when the tool is opened. - Populate input parameters with initial values

updateParameters

Refine and Modify the values of a Tool's Parameters - Called whenever a parameter has been changed in the ArcGIS GUI - Frequent calls - Make method 'inexpensive' if possible - May use 'global' values to store results after first call

updateMessages

Modify the Messages created when a Tool's Parameters have changed - Called after validation has been performed. - Display a Warning or Error to users if Parameters have bad values

isLicensed

Query the license system to ensure the tool can run at the current license level. - Checking for licenses

execute

The tool's source code which is run upon tool execution. - Error Handling - AddMessage - AddWarning - AddError

Where the Geoprocessing of data is accomplished - Set output dataset to allow for chaining in ModelBuilder - SetOutputParameter

Model Builder

Model Builder

Allows for visual creation of Geoprocessing tools

- Consume custom geoprocessing tools
- Ensure you are setting an output!
- Export to Python code
- A good way to introduce scripting concepts

Advanced Python - Modular Design and Packages

Validation modules

For each validation requirement:

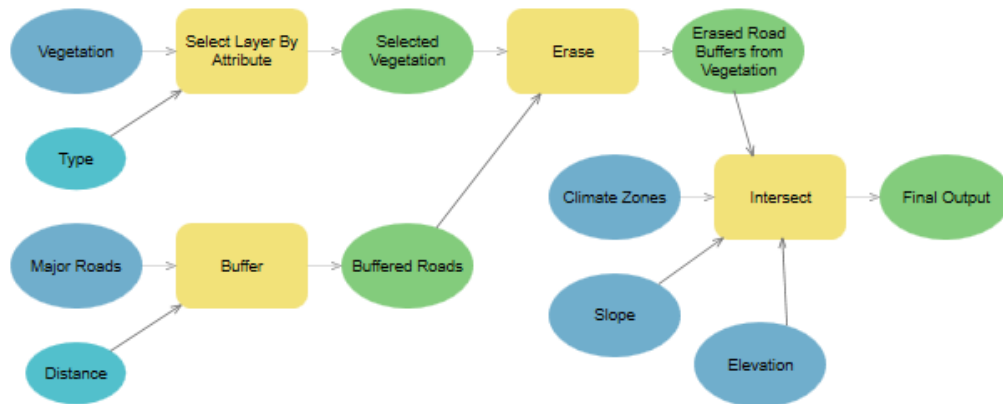


Figure 10:

- Create a function which will validate a dataset
- Create a dataset which satisfies the validation requirements
- Create one or more datasets which do not satisfy requirements
- Write validation functions

Debugging/Testing Tool Validation

Creating tests for validation:

- Validation accomplishes two things
- Ensuring the data set is 'clean'
- Ensuring the data set will not crash the tool
- Two types of tests
- Correct data does not trigger any errors
- Incorrect data is error handled and returns a message

Planning the Logic of a Tool

For each requirement in a tool:

- Create a function which accomplishes the requirement statement

- Call the function from the applicable test method(s)
- Commit when the code passes the test(s)

Granularity in Tool Design

Can we break a tool into multiple tools?

- Does any requirement make sense as a standalone tool?
- Can we chain the tools together based off input/output parameters?
- Can these sub-tools be made more robust?
- Handle more data types
- Improve processing speed

Reusing Modular Code

Tool Metadata

- Tags
- Documentation

Conda Environments

Development Environments

Contains features in development

- May use alpha/beta code
- Contain test modules & data
- Mirrored by version control

Production Environments

Stable environments in which to run tools/services

- Ship reference to env with Project, Tool etc

- Requires stable versions of packages

Packaging Tools the Right Way

Creating a Package

Using setuptools and distutils to create a Python Package.

- setuptools
- setup
- pkg_resources
- distutils
- Legacy, use setuptools if possible

Creating a conda package

- bld.bat/bld.sh
- LICENSE
- Manifest.in
- meta.yaml
- README.md
- setup.py

Deploying a Package Internally

On a network - Point at a network location containing a package Via a http server - All advantages of any http service - Authentication - Metrics - Availability

Deploying a Package Publicly

Python Package Index Anaconda.org Custom Server

Viewing the Tool Output

Tools in ArcGIS Pro

Demonstration

- Geoprocessing History
- Stack-trace when tool fails
- Result File

Deploying a Tool as a Service

Tools and data sets can be uploaded to ArcGIS Enterprise or ArcGIS Online as services.

Demonstration

Consuming services via the ArcGIS Python API

The ArcGIS Python API is a cross-platform solution which allows for users to view, administer and interact with services on ArcGIS Enterprise or ArcGIS Online.

- More info to come tomorrow!

Thank you!! Questions/Comments?? cdow@esri.com