Python:
Building Geoprocessing Tools

David Wynne, Andrew Ortego

SEE
WHAT
OTHERS
CAN'T

Python: Building Geoprocessing Tools

SDCC - Room 03

Being able to build a geoprocessing tool from Python is a fundamental building block for adding your own custom functionality into ArcGIS. Join us as we step through the process of taking your Python code and turning it into fully functional geoprocessing tools. Both script tools and Python toolboxes will be explored.

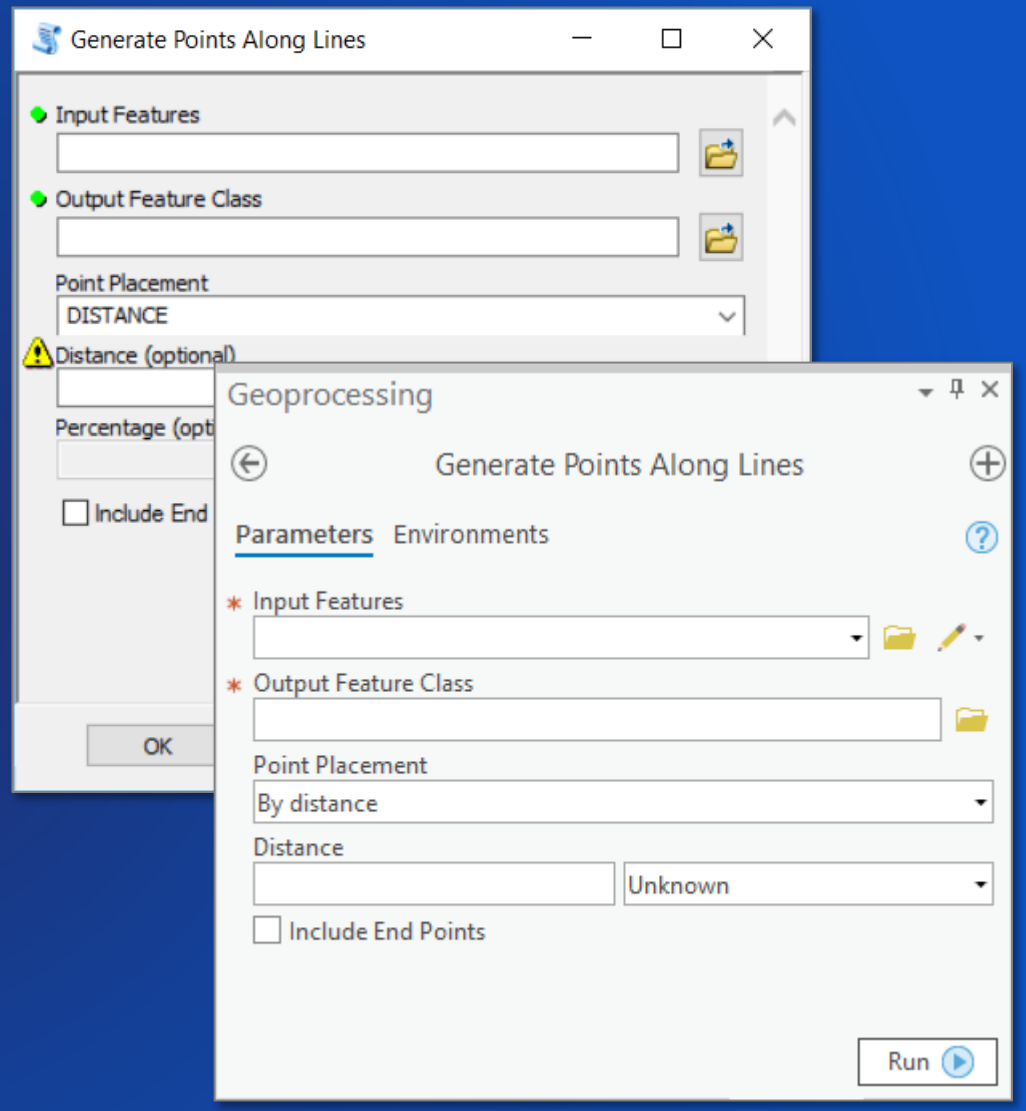
- ☐ Tool basics
- ☐ Tool mechanics
- ☐ Design
- ☐ Script tools
- ☐ Python toolboxes
- ☐ Parameters
- ☐ Validation
- ☐ Migration

http://esriurl.com/uc19buildtools

Why we build geoprocessing tools

- Your work becomes part of the geoprocessing framework

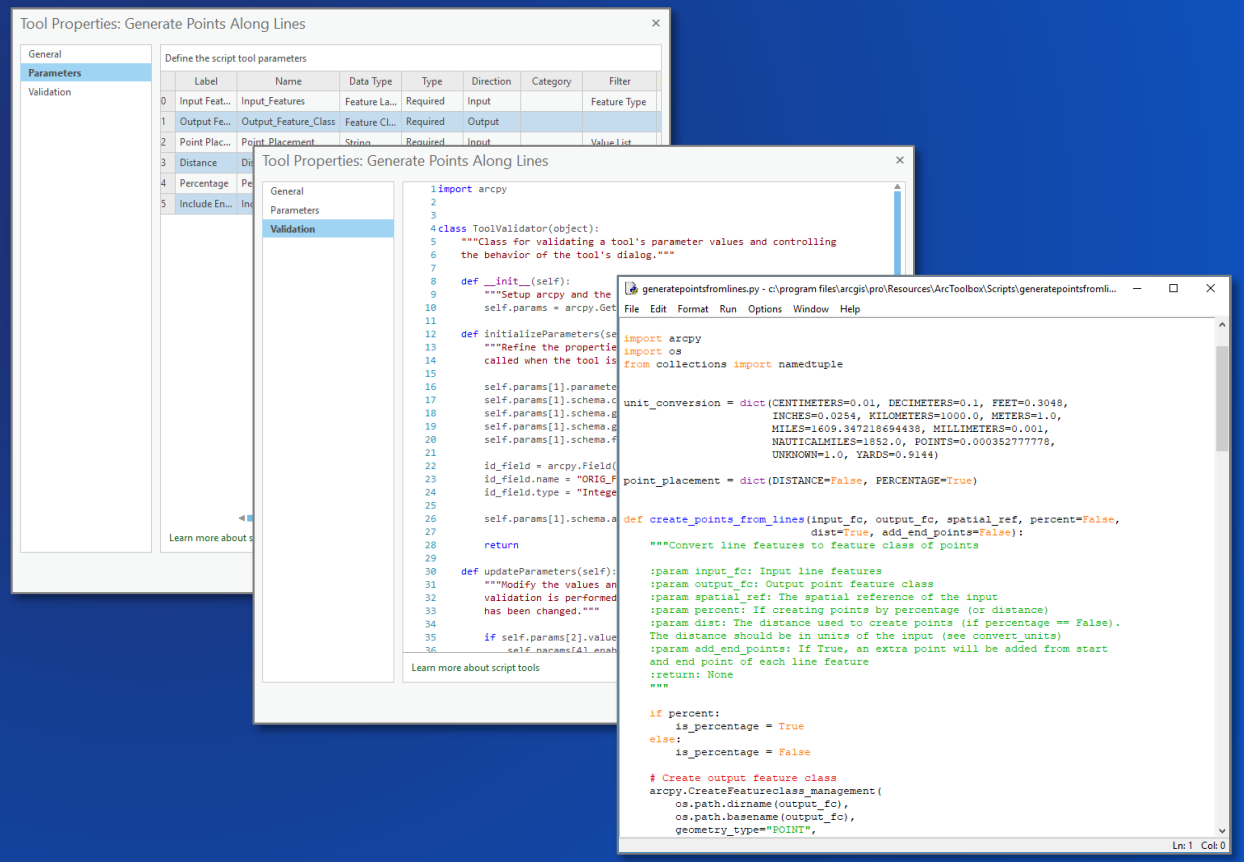
- Easy to access and run from within ArcGIS
- Familiar look and feel
- Make a mistake?
  - Re-run from the previous result
- Run from anywhere you can run a tool
  - Run from Python, ModelBuilder, a service
- Supported in multiple products



Tool recipe

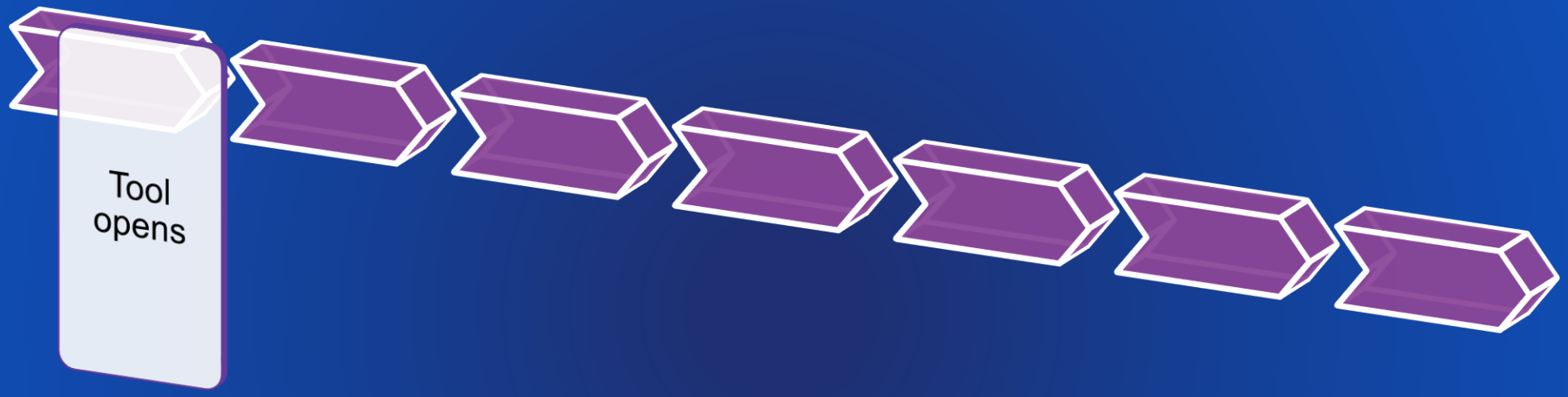
- A geoprocessing tool is made from 3 primary ingredients

1. Parameters
2. Validation
3. Source code



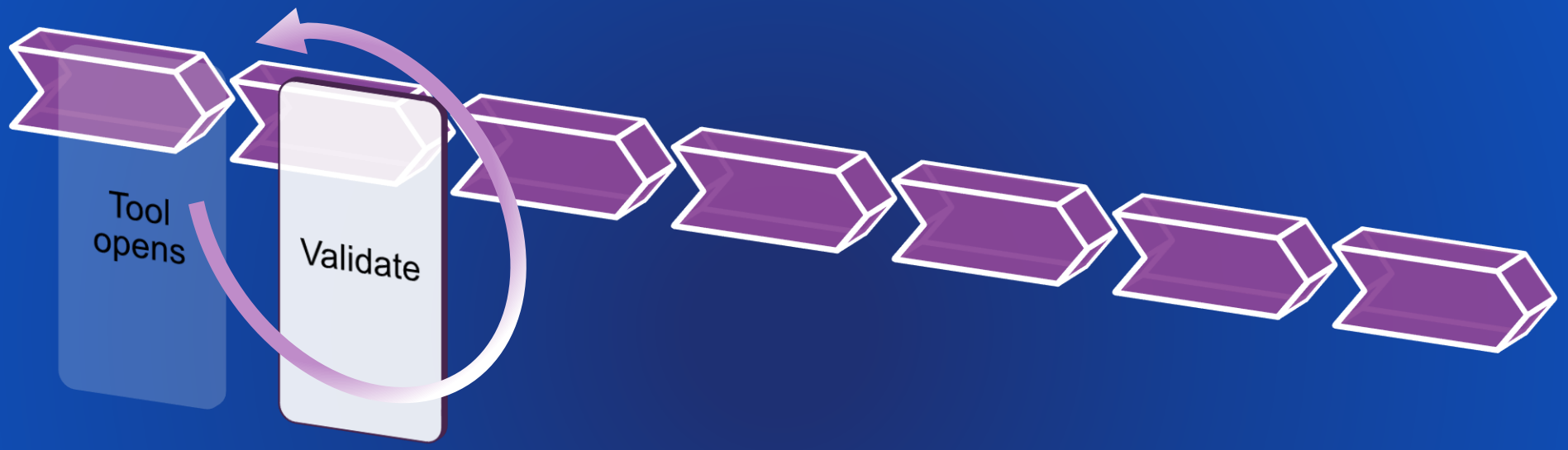
Demo:
From Python to geoprocessing tool

How a tool works



- **Tool parameters are initialized based on their definitions**

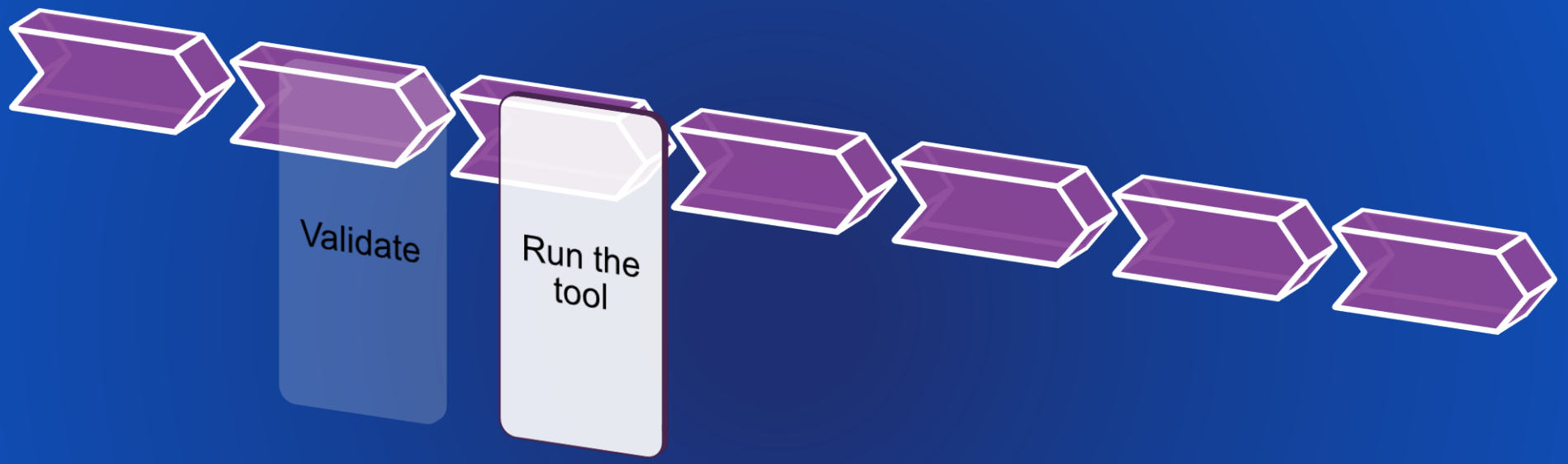
How a tool works

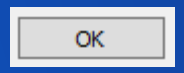


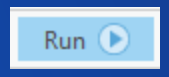
- Any interaction with the tool
  - updateParameters
  - Internal validation
  - updateMessages

- Have all the required parameters been supplied?
- Are the values of the appropriate data types?
- Does the input or output exist?
- Do values match their filter?

How a tool works

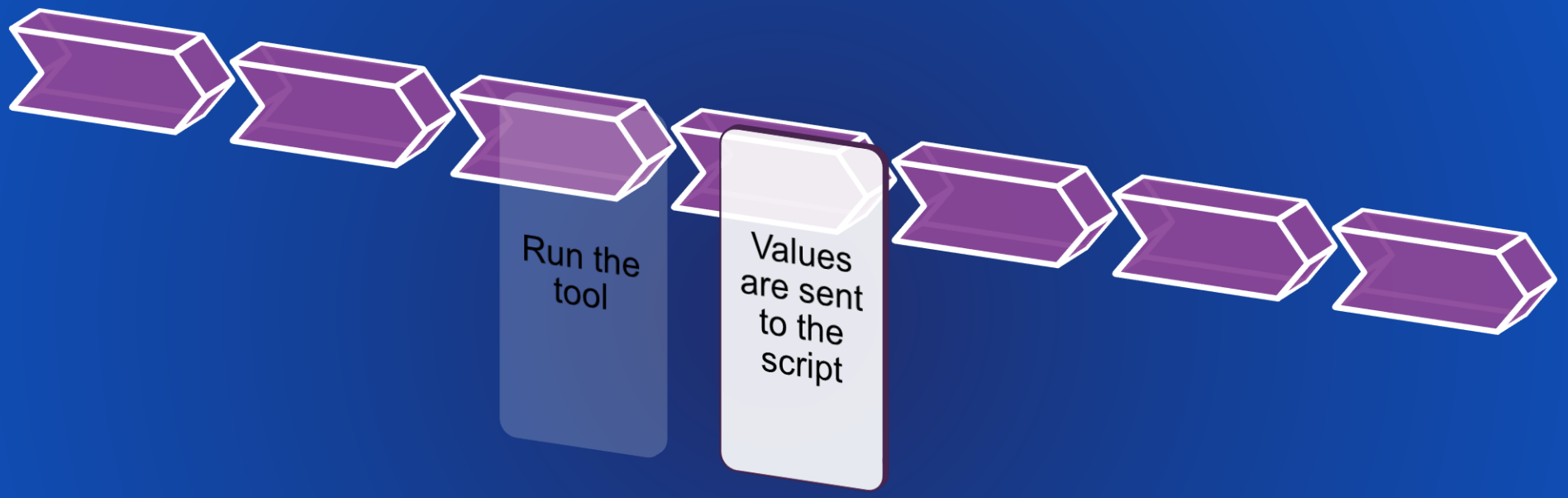


- The 

or 

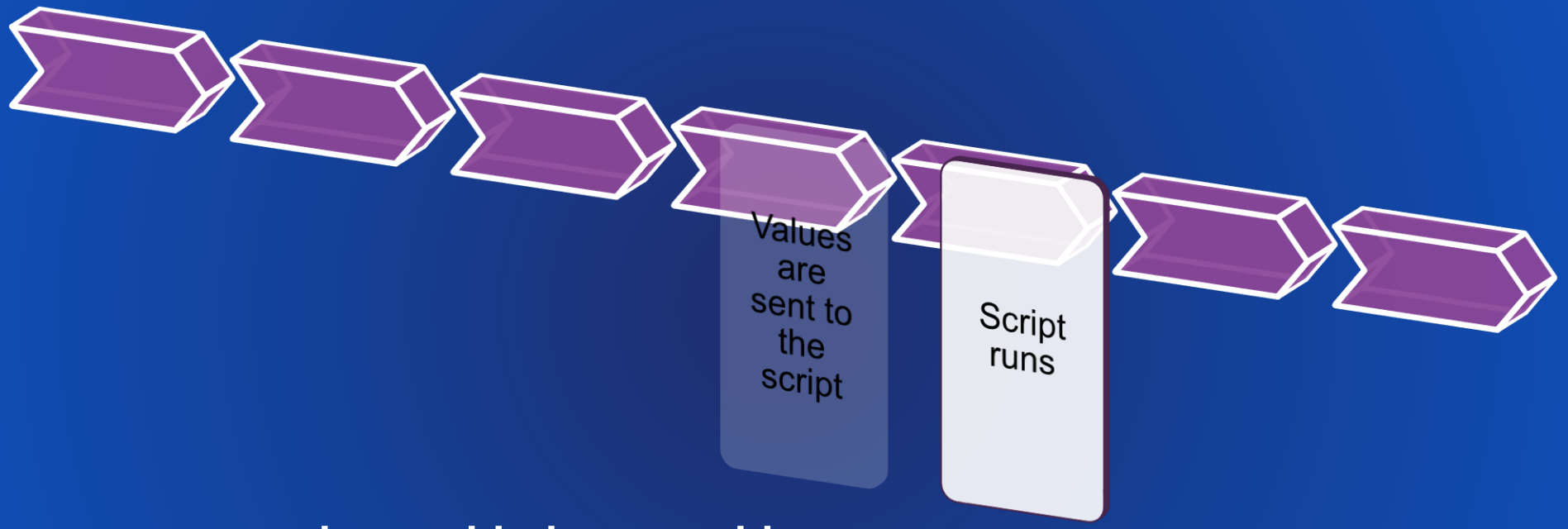
buttons are pushed
  - Parameter values are sent
  - .py is called

How a tool works



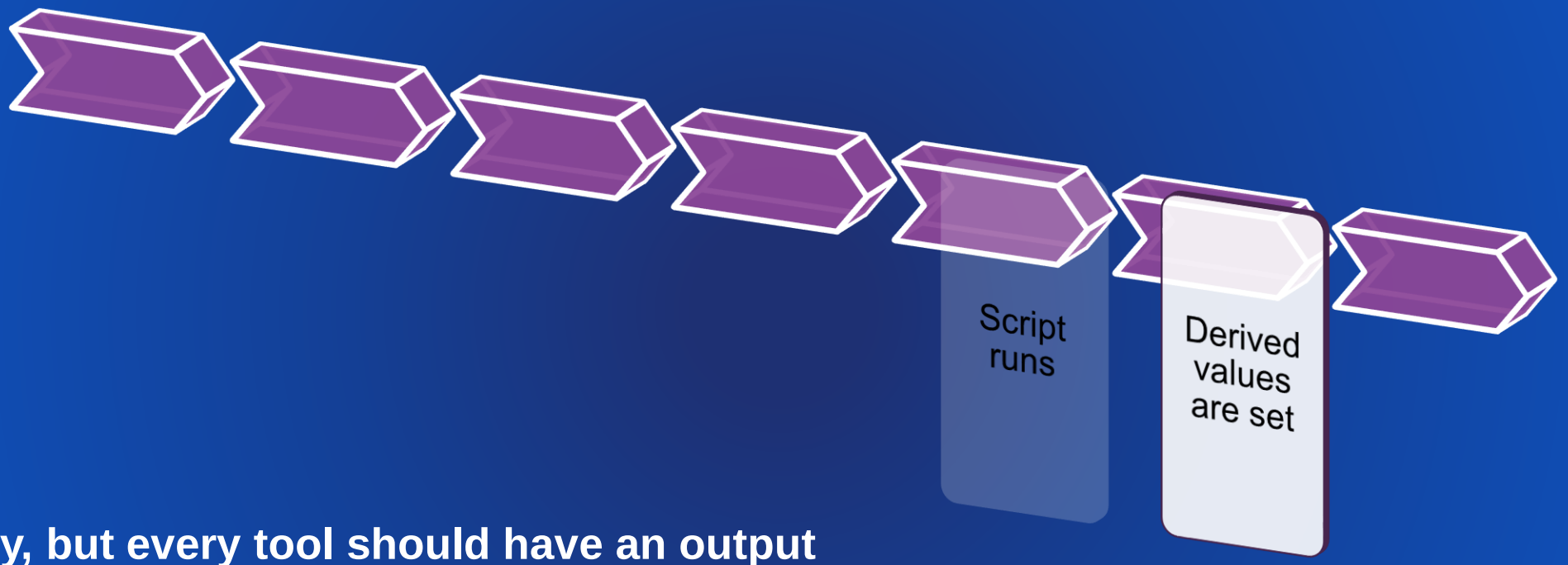
- **Script receives arguments with GetParameterAsText or GetParameter functions**

How a tool works



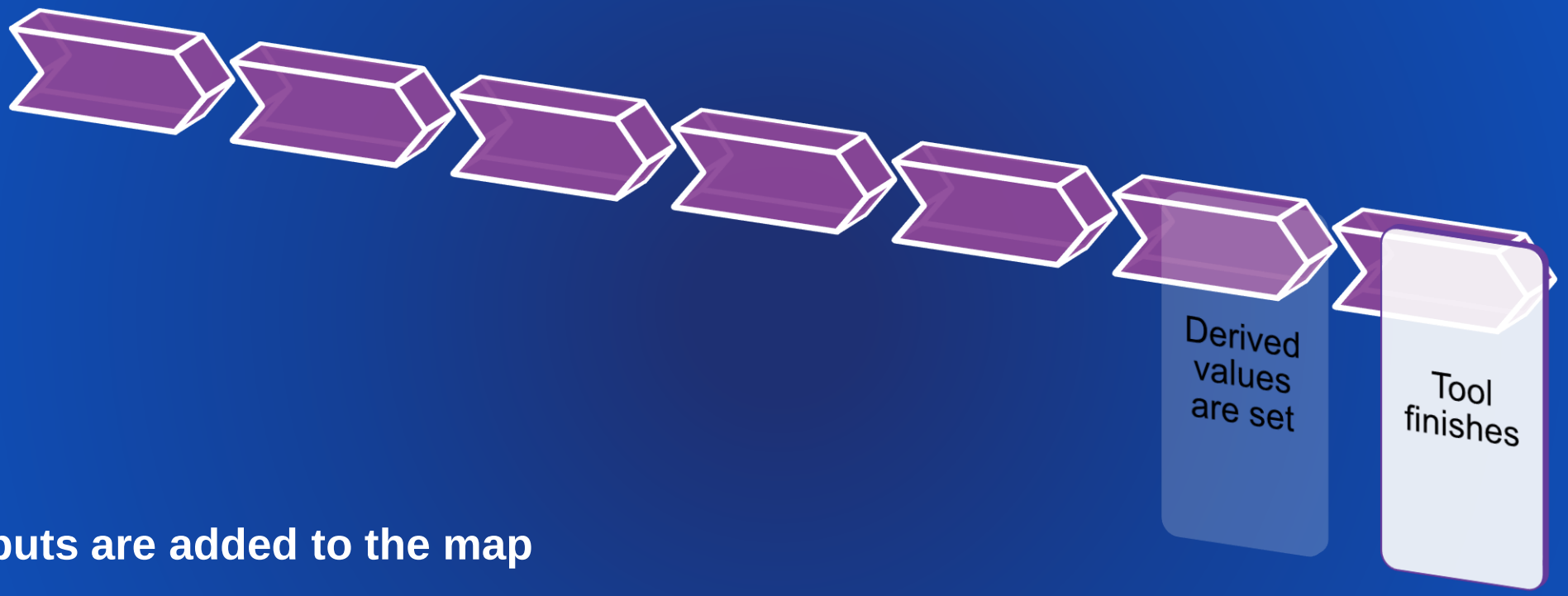
- Script can communicate with the app with:
  - Messages
  - A progressor
- Script can also respond to a cancellation

How a tool works



- If any, but every tool should have an output

How a tool works



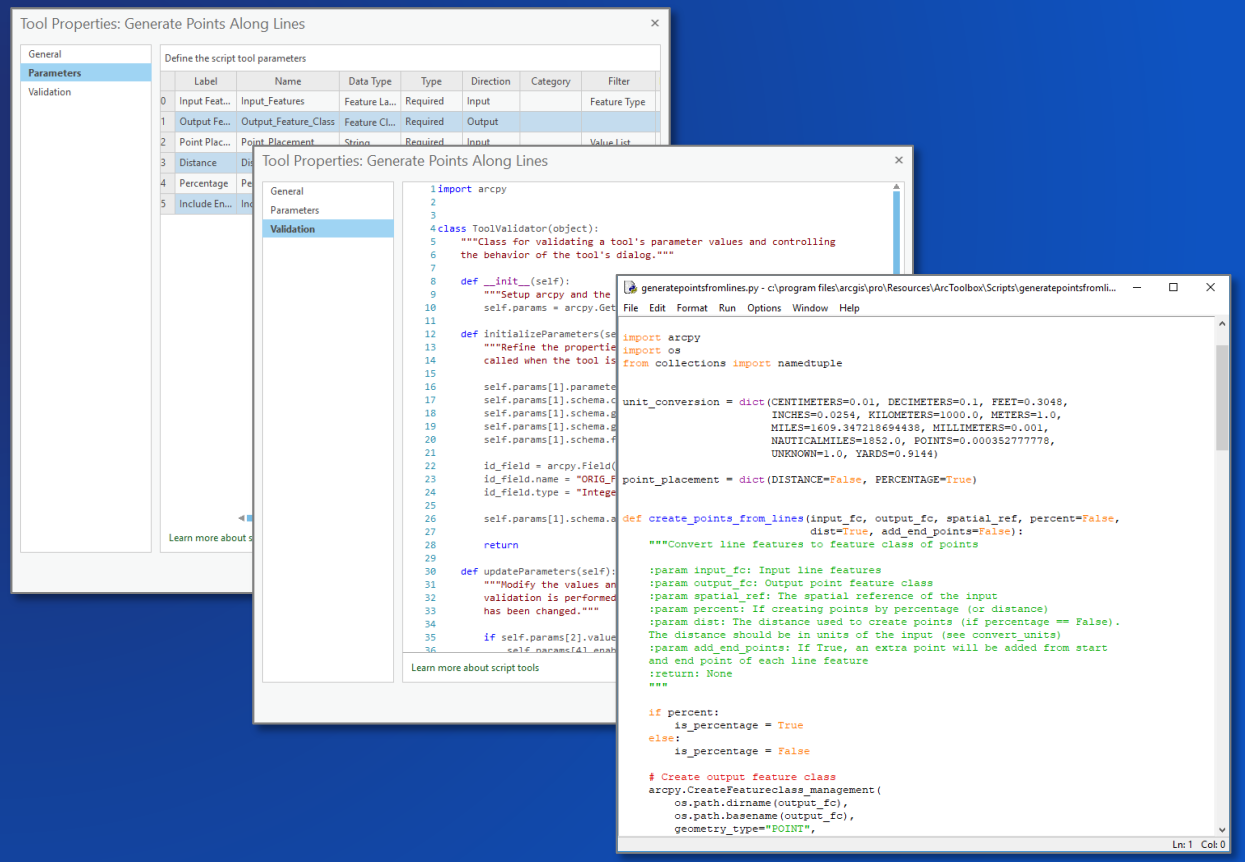
- Outputs are added to the map
- Symbology is applied
- Result is added to the history

Script tools vs Python toolboxes

- **Using Python, we can build tools in two ways:**

Script tools

- Source is Python
- Parameters through wizard
- Validation is Python (stored in toolbox)



Script tools vs Python toolboxes

- **Using Python, we can build tools in two ways:**

Python toolboxes

- Source is Python
- Parameters are Python
- Validation is Python

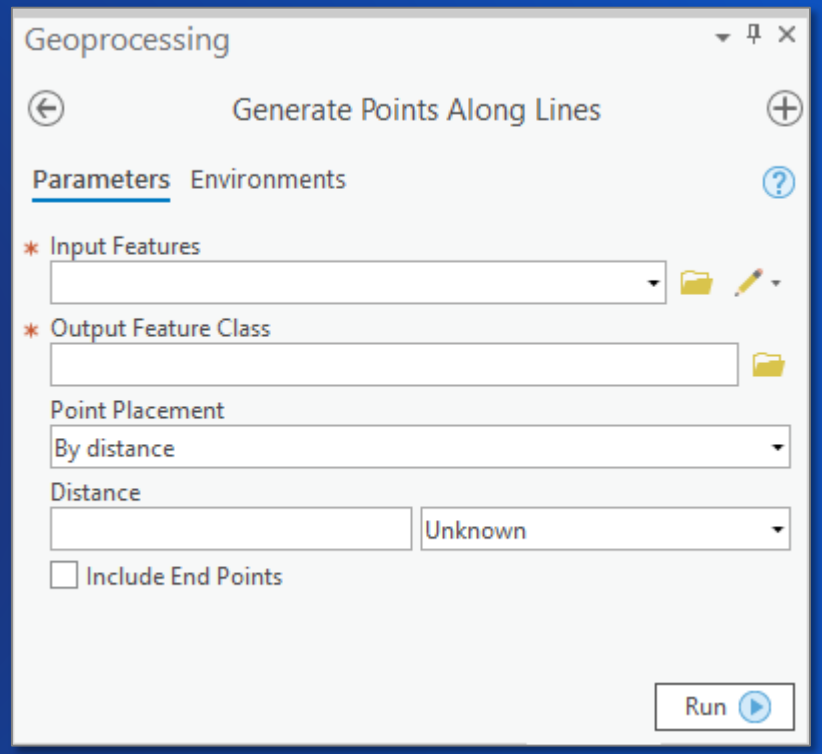
- **Which do I use?**
  - **“A tool is a tool”**

```python
import arcpy


class Toolbox(object):
    def __init__(self):
        """Define the toolbox (the name of the toolbox is the name of the
        .pyt file)."""

        self.label =  "Sinuosity toolbox"
        self.alias  = "sinuosity"

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

class CalculateSinuosity(object):
    def __init__(self):
        self.label       = "Calculate Sinuosity"
        self.description = "Sinuosity measures the amount that a river " + \
                           "meanders within its valley, calculated by " + \
                           "dividing total stream length by valley length."

    def getParameterInfo(self):
        """Define the tool (tool name is the name of the class)."""

        in_features = arcpy.Parameter(
            displayName="Input Features",
            name="in_features",
            datatype="GPFeatureLayer",
            parameterType="Required",
            direction="Input")

        in_features.filter.list = ["Polyline"]

        sinuosity_field = arcpy.Parameter(
            displayName="Sinuosity Field",
            name="sinuosity_field",
            datatype="Field",
            parameterType="Optional",
            direction="Input")

        sinuosity_field.value = "sinuosity"

        out_features = arcpy.Parameter(
            displayName="Output Features",
            name="out_features",
            datatype="GPFeatureLayer",
            parameterType="Derived",
            direction="Output")

        out_features.parameterDependencies = [in_features.name]
        out_features.schema.clone = True

        parameters = [in_features, sinuosity_field, out_features]

        return parameters
```

Parameters

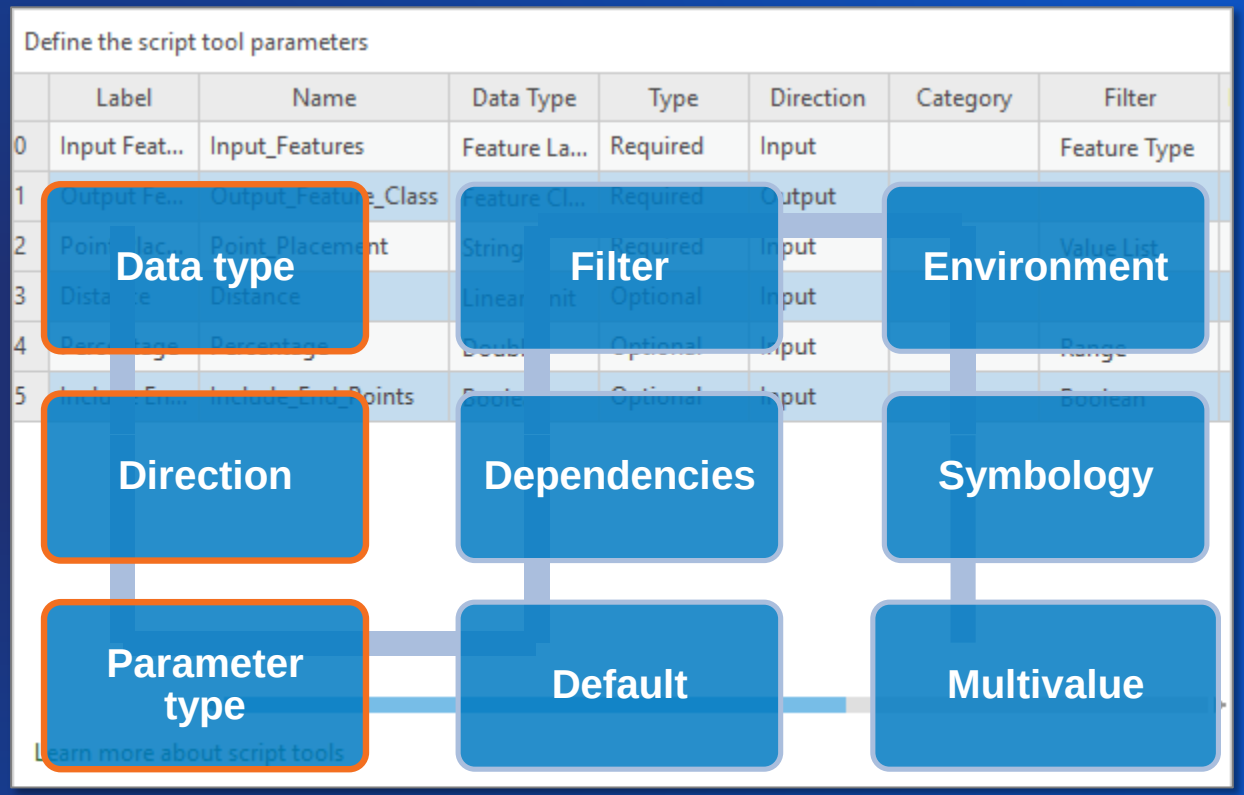
- Parameters are how you interact with a tool

- Simple rules to guide behaviors
  - Does an input exist?
  - Is the input the right type?
  - What are valid fields for this data?
  - Is this value an expected keyword?



Parameter properties

- Data type
  - Feature Layer, Raster Layer, Table View, …
  - String, Boolean, Long, Float, …
- Direction
  - Input, Output
- Parameter type
  - Required, Optional, Derived



Demo:
Working with tool parameters

Validation

- Provides more control
  - Parameter interaction
  - Calculate defaults
  - Enable or disable parameters
- Setting parameter errors and messages
- Defining output characteristics
  - *Chain tools in ModelBuilder*

Validation

- Mechanically, validation is about responding to changes in:

- value / valueAsText
  - Does a parameter have a value?
  - What is the value?
  - Properties of the data (arcpy.Describe)
- altered
  - Has the parameter been altered?
- hasBeenValidated
  - Has internal validation checked the parameter?

```python
def updateParameters(self):
    """Modify the values and properties of parameters before internal
    validation is performed.  This method is called whenever a parameter
    has been changed."""

    if self.params[0].value:
        if not self.params[2].altered:
            extent = arcpy.Describe(self.params[0].value).extent
            if extent.width > extent.height:
                self.params[2].value = extend.width / 100.0
            else:
                self.params[2].value = extent.height / 100.0

    return
```

```python
def updateMessages(self):
    """Modify the messages created by internal validation for each tool
    parameter.  This method is called after internal validation."""

    # Distance should never be negative
    if self.params[2].value <= 0.0:
        self.params[2].setErrorMessage(
            'Distance value cannot be a negative number')

    # If using percentages, distance must be less than 1.0
    elif self.params[3].value:
       if self.params[2].value > 1.0:
            self.params[2].setErrorMessage(
                'Percentages must be between 0.0 and 1.0')
```

Validation: ModelBuilder

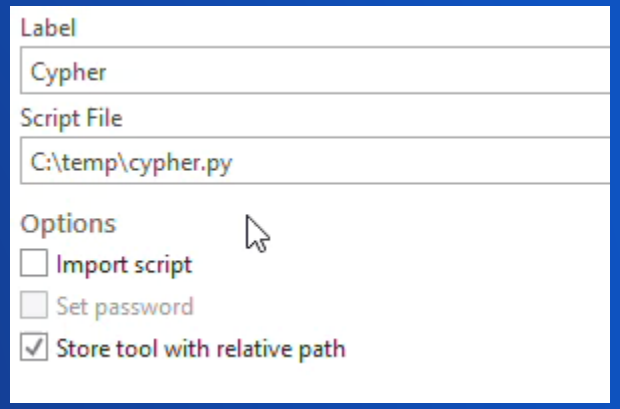
- **Describe outputs for chaining in ModelBuilder**
- **By updating `schema` of outputs, subsequent tools can see pending changes prior to execution**

```
self.params[1].parameterDependencies = [0]
self.params[1].schema.clone = True
self.params[1].schema.geometryTypeRule = 'AsSpecified'
self.params[1].schema.geometryType = 'Point'
self.params[1].schema.fieldsRule = 'FirstDependencyFIDs'

id_field = arcpy.Field()
id_field.name = 'ORIG_FID'
id_field.type = 'Integer'

self.params[1].schema.additionalFields = [id_field]
```

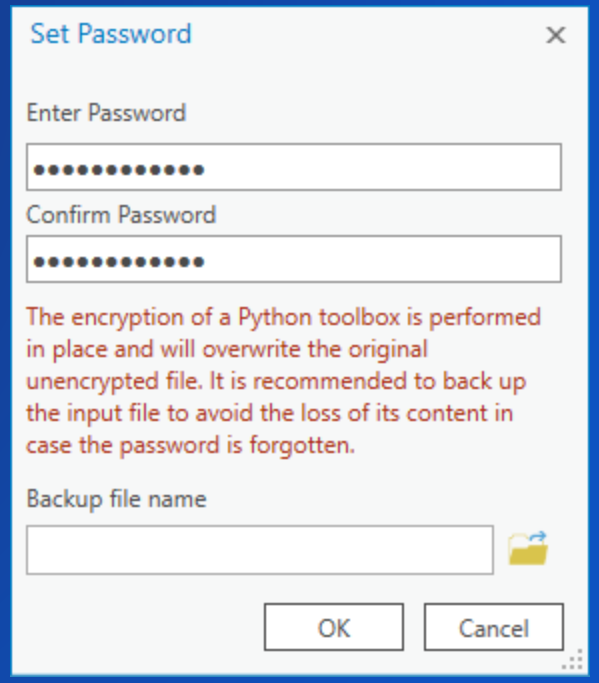
Demo:
Tool validation

Keeping your work private

- Script tools have supported encryption for many releases
  - Embed, then set a password



- Python toolboxes support encryption at 10.5, Pro 1.3
  - Encrypt the toolbox in one step
  - Python toolboxes are encrypted in place

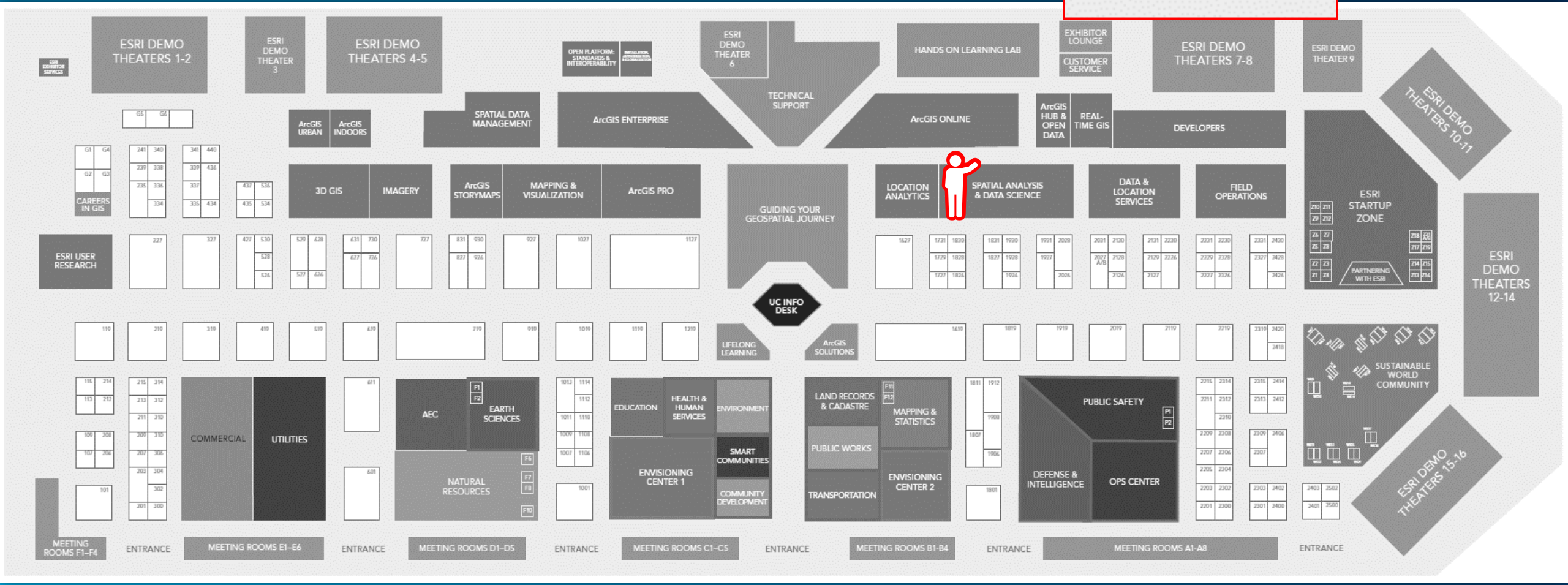


10.x to ArcGIS Pro migration

- Use the Analyze Tools For Pro tool, to identify the following:
  - ArcPy differences
  - Python 2 to 3 compatibility issues

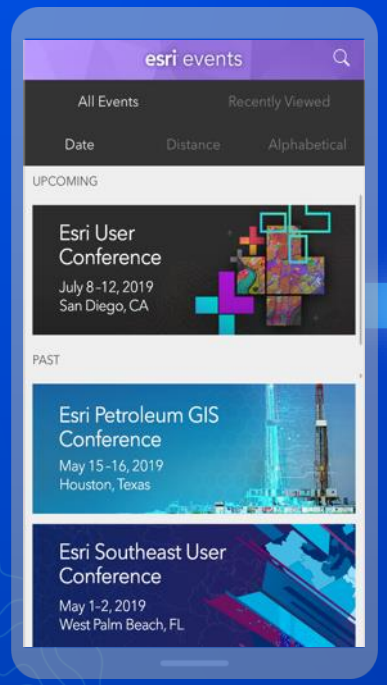
- For Python differences
  - See http://python3porting.com/strategies.html
  - Useful for writing code that will work in both Python 2 and Python 3

Questions?

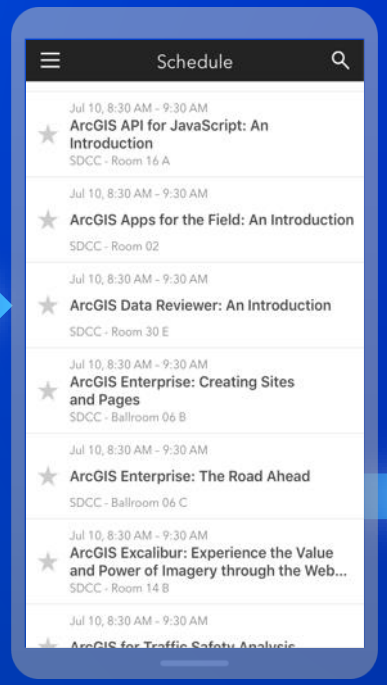

SPATIAL ANALYSIS & DATA SCIENCE
ESRI DEMO THEATERS 1-2
ESRI DEMO THEATER 3
ESRI DEMO THEATERS 4-5
OPEN PLATFORM: STANDARDS & INTEROPERABILITY
ESRI DEMO THEATER 6
TECHNICAL SUPPORT
HANDS ON LEARNING LAB
EXHIBITOR LOUNGE
CUSTOMER SERVICE
ESRI DEMO THEATERS 7-8
ESRI DEMO THEATER 9
ESRI DEMO THEATERS 10-11
ESRI DEMO THEATERS 12-14
ESRI DEMO THEATERS 15-16
ESRI USER RESEARCH
CAREERS IN GIS
ArcGIS URBAN
ArcGIS INDOORS
SPATIAL DATA MANAGEMENT
ArcGIS ENTERPRISE
ArcGIS ONLINE
ArcGIS HUB & OPEN DATA
REAL-TIME GIS
DEVELOPERS
3D GIS
IMAGERY
ArcGIS STORYMAPS
MAPPING & VISUALIZATION
ArcGIS PRO
GUIDING YOUR GEOSPATIAL JOURNEY
LOCATION ANALYTICS
SPATIAL ANALYSIS & DATA SCIENCE
DATA & LOCATION SERVICES
FIELD OPERATIONS
ESRI STARTUP ZONE
PARTNERING WITH ESRI
UC INFO DESK
LIFELONG LEARNING
ArcGIS SOLUTIONS
SUSTAINABLE WORLD COMMUNITY
COMMERCIAL
UTILITIES
AEC
EARTH SCIENCES
NATURAL RESOURCES
EDUCATION
HEALTH & HUMAN SERVICES
ENVIRONMENT
ENVISIONING CENTER 1
SMART COMMUNITIES
COMMUNITY DEVELOPMENT
LAND RECORDS & CADASTRE
MAPPING & STATISTICS
PUBLIC WORKS
TRANSPORTATION
ENVISIONING CENTER 2
PUBLIC SAFETY
DEFENSE & INTELLIGENCE
OPS CENTER
MEETING ROOMS F1–F4
ENTRANCE
MEETING ROOMS E1–E6
ENTRANCE
MEETING ROOMS D1–D5
ENTRANCE
MEETING ROOMS C1–C5
ENTRANCE
MEETING ROOMS B1–B4
ENTRANCE
MEETING ROOMS A1–A8
ENTRANCE

Please Share Your Feedback in the App

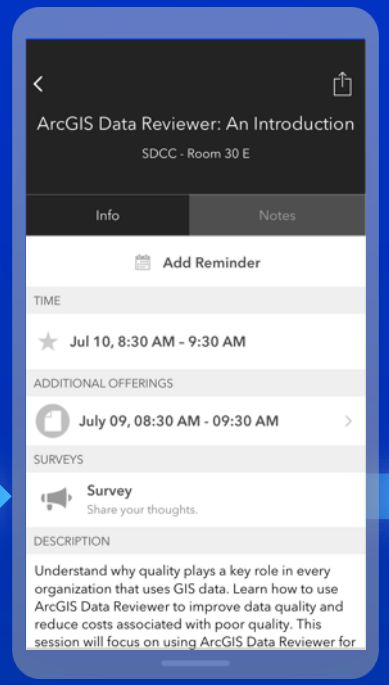
Download the Esri Events app and find your event



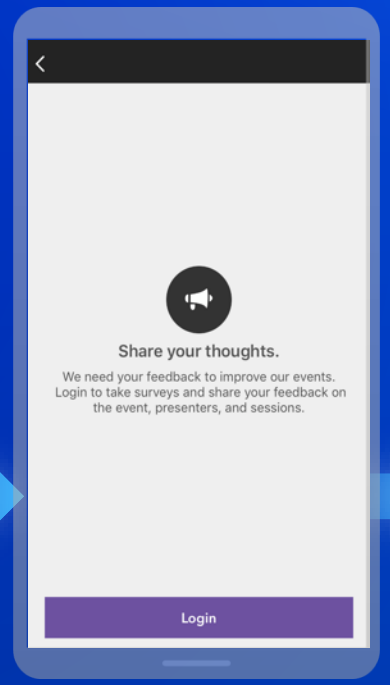
Select the session you attended



Scroll down to "Survey"



Log in to access the survey



Complete the survey and select "Submit"