Exercise 5. Building ArcGIS Tools using Python

GIS in Water Resources, Fall 2014

Prepared by Anthony Castronova

Purpose

The purpose of this exercise is to illustrate how to build ArcGIS tools using the Python programming language. This exercise will guide you through the processes of setting up Python on your computer, installing required libraries, collecting data via ArcGIS services, and building an ArcGIS tool. The purpose of the ArcGIS tool is to provide you with an example of how to manipulate shapefiles, iterate over raster datasets, execute native ArcGIS tools, as well as define ArcGIS tool parameters. Overall, it will provide guidance on how to build your own ArcGIS tool. The tool outlined in this exercise will trace a user-defined point downstream until it hits a watershed outlet.

Learning Objectives

- Understand how to setup and use Python on your own computer
- Students should be capable of basic shapefile and raster manipulation using the Python programming language.
- The ability to extend ArcGIS tools to include custom algorithms
- Understand how to develop a Python script that operates within the ArcGIS toolbox and utilizes input parameters from a user interface.

Computer and Data Requirements

To carry out this exercise, you need to have a computer that runs ArcGIS 10.2 or higher and includes the Spatial Analyst extension. No data is required to start this exercise. All the necessary data will be extracted from ArcGIS.com services. To use these services you need an ArcGIS.com account that has been linked to an ArcGIS license.

This exercise is divided into the following activities:

- 1. Setting up Python and installing 3rd party libraries
- 2. Data Collection
- 3. Model Building and Scripting

Part 1: Setting up Python and Installing 3rd party libraries

If you have ArcGIS 10.2 installed on your computer, then you already have Python 2.7 installed as well. However, you **PATH** variables may also need to be adjusted so that Python is recognized at the command-line. This is purely for running python outside of ArcGIS, but it is useful for other applications such as integrated development environments. Note: you will need administrative privileges to modify the **PATH** on you computer.

Check to see if Python is already in your **PATH** by opening the command prompt and typing **python**. If you get something like the following then you can skip down to installing 3rd party libraries.

000	🏠 tonycastronova — Python — 80×24	M ₂₁
Last login: Mon Tonys-MacBook-P Python 2.7.6 (d [GCC 4.2.1 Comp Type "help", "co >>>	Oct 27 16:13:53 on console Yro-2:~ tonycastronova\$ python efault, Nov 18 2013, 15:12:51) Matible Apple LLVM 5.0 (clang-500.2.79)] on darwin copyright", "credits" or "license" for more information.	

Otherwise, click on the start menu, right click on Computer, and select Properties.



Next, select **Advance System Settings**. This will open the system properties window (below).

			1		_
Computer Name	Hardware	Advanced	System Protect	tion Remote	
You must be lo	gged on as	an Administra	tor to make mos	t of these cha	nges.
- Performance -					
Visual effects	processor s	scheduling m	emory usage ar	nd virtual mem	orv
				Settings	
User Profiles -					
Desktop settir	ngs related t	o your logon			
	-				
				Settings.	
Startup and R	ecovery				
System startu	o, system fai	lure, and deb	ugging informati	on	
					_
				Settings.	
			Envir	onment Variab	les

Click **Environmental Variables.** Find the variable labeled **Path** in the bottom window and select **edit**. Proceed with caution: Deleting items from the variable value field can effect your application and OS settings negatively.

Add the following paths at the end of the variable value textbox (replace with paths on your computer!!!). Make sure you have semicolons between each path (including the one in the front)

;C:\Python27\ArcGIS10.2\Scripts;C:\Python27\ArcGIS10.2;C:\Python27\ArcGIS10.2;C:\Python27\ArcGIS10.2,C:\Python27\A

En	wironment Variable	s	Edit System Variable	e	×
	User variables for To	ony Castroi	Variable name:	Path	-
	Variable	Value	W 111 1		5
	TEMP	%USERF	Variable value:	/\arcGIS10.2;C:\Python2/\arcGIS10.2\lb	s
	TMP	%USERF		OK Cancel	
	System variables	Value			
	OS	Windows	NT		
	Path	C:\Windo	ws\system32;C:\Windo	ows;C:\	
	PATHEXT PROCESSOR_A	.COM;.EX	E;.BAT;.CMD;.VBS;.VB	₹;.JS;	
		New	Edit	Delete	
			ОК	Cancel	

Open an **NEW** command prompt, type **python** and enter. You should now see that python has launched successfully.

Next, we can install a python package manager that will make installing 3rd party applications very easy. Navigate to <u>https://pip.pypa.io/en/latest/installing.html</u> and download get-pip.py. Navigate to this directory using the command line. Type **python get-pip.py**. This will install the pip application.



Install numpy using PIP: **pip install numpy**. Alternatively, you can download and install it using an msi (<u>http://sourceforge.net/projects/numpy/files/NumPy/</u>) or bundled with SciPY (<u>http://www.scipy.org/Installing_SciPy</u>).



Part 2: Data Collection

Add ArcGIS Server		4	ArcGIS Server User Cor	nnection Properties	-X
	This wizard guides you through the process of making a connection to an ArGIS Server. You can create a connection to use, publish, or administer GIS services.		General	[https://hydro.arcgis.com:443/arcgis/services] ArcGIS Server: http://gisserver.domain.com:6080/arcgis	
🔶 🔃	What would you like to do?		Authentication (Op	otional)	
	Our of the services		User Name:	tcastronova	
	Publish GIS services Administer GIS service		Password:	•••••	
				☑ Save Username/Password	
			About ArcGIS Serve	r connections	
	< Back Next > Cancel			OK Cancel Ap	ply

Connect to the ArcGIS hydrology server. We will use this to delineate a watershed.

If added correctly, you should see the following tools listed in ArcCatalog.



Next, add a connection to the ArcGIS landscape1 server. We will use this web service to download and visualize National Hydrography Dataset (version 2) rivers. Use https://landscape1.arcgis.com/arcgis/services as the URL. If added correctly, you will see long list of datasets under the landscape1 service in ArcCatalog.

		😑 🚮 arcgis on landscape1.arcgis.com_443 (user)
		🗉 🚞 Tools
		🗉 🚞 Utilities
		USA_Active_Quaternary_Faults
		USA_AII_Fed_Lands
		🖳 USA_Aquifers
ArcGIS Server User Co	nnection Properties	USA_BLM_Lands
		🖳 USA_Coal_Bed_Methane_Basin
General		USA_Coal_Fields
		USA_Critical_Habitat
Server URL:	https://landscape1.arcgis.com:443/arcgis/services/	USA_Earthquake_Risk
		USA_Flood_Risk
	ArcGIS Server: http://gisserver.domain.com:6080/arcgis	USA_Geology_Units
		USA_Hazardous_Waste_Sites
Authentication (O)	iptional)	USA_Historic_Sites
		🖳 USA_NCED
User Name:	tcastronova	USA_NHDPlusV2
		🖳 USA_NPDES
Password:	•••••	USA_NPS_Lands
		🖳 USA_Ntv_Lands
	Save Username/Password	USA_Oil_Shale_Basins
		USA_Railroads
About ArcGIS Serve	er connections	🖳 USA_Roads
		USA_Soils
		USA_USFS_Lands
		USA_USFWS_Lands
	OK Cancel Apply	🖳 USA_Wetlands
		USA_Wilderness_Areas

Finally, connect to the ArcGIS elevation web service. This will be used to downloading elevation data for the exercise. Use <u>http://elevation.arcgis.com/arcgis/services</u> as the URL. If added correctly, you will see a short list of tools and data available under the elevation service in ArcCatalog.

ArcGIS Server User Co	nnection Properties	
General		
Server URL:	https://elevation.arcgis.com:443/arcgis/services/	
	ArCGIS Server: http://gisserver.domain.com:6080/arcgis	
Authentication (C	ptional)	arcgis on elevation.arcgis.com (use
User Name:	tcastronova	
		🗉 🕎 Elevation
Password:	•••••	🗄 👰 ElevationSync
	Save Username/Password	WorldElevation
About ArcGIS Serve	er connections	🖾 DataExtents
		I Terrain
		TopoBathy
	UK Cancel Apply	III NED30m

Add some template data so that we can zoom into the location that we would like to download data. Select the **Add Data** button:



Navigate to the ArcGIS template data directory (C:\Program Files (x86)\ArcGIS\Desktop10.2\TemplateData\TemplateData.gdb\USA) and add US cities, interstates, and states.

Add Data	
Look in: 🗗	USA 🔹 🔁 🔂 🐨 🗮 🖛 🖾 🖾 🐨 🚳
cities counties intrstat landbnds neighcount states us_lakes us_rivers us_bln	τy
Name:	cities; intrstat; states Add
Show of type:	Datasets, Layers and Results Cancel

The map should now look like this:



Zoom into Logan, UT. Use the **Identify** tool to determine which of these dots is Logan. This will give us an idea of where we are, before we start loading ArcGIS web service datasets.



Add the NHDPlus (version 2) data set from the landscape1.arcgis.com web service.



We are only interested in the stream data, so turn off all NHD layers except Streams. This will help speed up the data load time. The layers in your table of contents should look like this:



Right Hand Fork $\square \times$ Identify Identify from: • <Top-most layer> - NHD Streams Mean Annual Flow Right Fork Logan River ☆ | 万 -111.633275 41.780751 Decimal Degrees Location: . Field Value = OBJECTID Null ComID 664348 GNIS Name **Right Fork Logan River** Length (km) 1.958 Reach Code 16010203000538 Flow Direction With Digitized

Now that we have the NHD rivers loaded, we can zoom into Right Hand Fork.

To delineate a watershed at Right Hand Fork, we will use the ArcGIS online watershed delineation tool. Double click on the ArcGIS server watershed tool.



Select an input point near the outlet of Right Hand Fork (see green dot on map). Don't get too close to the Logan river (downstream), or the delineation tool will snap the outlet to the wrong reach. To ensure that this does not happen, you may have to adjust the snap distance (try 100 meters)

	L Z	ma		N	$\sim \varkappa$
🔨 Watershed					> \
Input Points				*	
Watershed::InputPoints			-	2	n End
InputPoints					112-2
					1 mar
Point Identification Field (optional)				_	Les L
Snap Distance (optional)					Right Hand Fork
Snap Distance Units (optional)					$\sqrt{1}$
Meters Data Source Resolution (optional)				-	SK2 /
				-	
Generalize Watershed Polygons (optional)				
Return Snapped Points (optional)					しょく
					2 8 13
				-	{ { } } ~
	ОК	Cancel Environ	ments Show	Help >>	- Ch-
		λ			The si
\sim		ų į	\sim	\sim	$\neg \uparrow \neg \uparrow, \land$



This operation will result in the Right Hand Fork watershed. Go ahead and turn off all unnecessary layers and change the watershed color to something more meaningful.

Add NED30m elevation from the elevation.arcgis.com server.





Next we want to extract the elevation data within the boundary of our watershed. This will make future data processing faster since we will be using a small subset of the national elevation dataset. In addition, this file will be stored locally so we won't need an Internet connection to perform our processing tasks. To do this, open the search menu and enter "Extract". Make sure to choose the search by "Tools" option above the search textbox. This will limit the search results ArcGIS tools. Since we are dealing with elevation data from an ArcGIS server, we want to select the "Extract Data (server)" tool.

Search		ņ
<	Search	
ALL Maps Data Tools Extract Any Extent ▼	<u>Images</u>	9
Search returned 33 items 🔻		Sort By 🔻
Server (Toolbox) The Server toolbox contain toolboxes\system toolboxe	ns tools to mana s\server tools.t	age ArcGI tbx
Data Interoperability (To The Data Interoperability t toolboxes\system toolboxes	oolbox) oolbox contains es\data interope	a set of t erability to
Extract (Toolset) Summary: not available. toolboxes\system toolboxe	es\analysis tools	s.tbx\ extr
Extract (Toolset) Summary: not available. toolboxes\system toolboxe	es\coverage too	ls.tbx\ana
Extraction (Toolset) Summary: not available. toolboxes\system toolboxe	es\spatial analys	st tools.tb
Data Extraction (Toolset Summary: not available, toolboxes\system toolboxe) es\server tools.1	tbx∖data e
Extract Data (Server) (T Extracts selected layers i toolboxes\system toolboxed	ool) n the specified as sever tools.t	area of int tbx∖data e
Extract Data Task (Serv Extracts the selected layer toolboxes\system toolboxes	er) (Tool) ers in the specif es\server tools.t	ied area o tbx\data e
Extract Data and Email Extracts the data in the s	Task (Server) pecified layers	(Tool) and area

Extract Data
Layers to Clip
🗹 🖻
◆NED30m
Area of Interest
Area_of_interest
Feature Format
File Geodatabase - GDBgdb
Raster Format
ESRI GRID - GRID
Spatial Reference
Same As Input
Custom Spatial Reference Folder (optional)
Output Zip File
2:\windows_shared\exercise 10_28_14\example_data\elevation.zp
OK Cancel Environments Show Help >>

Select the NED 30m elevation raster as the layer to clip. The Area of Interest that will be used to extract the data (i.e. cookie cutter) should be the watershed that you delineated in previous steps. Leave the default options for Feature Format, Raster Format, Spatial Reference, and Custom Spatial Reference Folder. Specify an output ZIP file where the extracted data will be saved.

Open Windows Explorer and navigate to the directory of your output ZIP. Extract the contents, and you should now have an elevation dataset that covers only the watershed area.



Part 3: Model Building and Scripting

The goal of our scripting tool is to trace any point within the watershed downstream to the watershed outlet. This can later be modified to provide statistics regarding the flow path. This example will demonstrate (1) how ArcGIS tools can be used to create a custom model, (2) how to include custom data processing and functionality, and (3) how to build the ArcGIS tool interface for a custom tool.

Activate the ArcToolbox by clicking . Create a new toolbox by right clicking inside the window and selecting Add Toolbox from the context menu. This will open a dialog for you to search for an existing toolbox. Instead, navigate to any directory that you like

and select the create New Toolbox button in the top right corner

		🔊 Arc I oolbox
		🗄 🚳 3D Analyst Tools
		🕀 😂 Analysis Tools
		🗄 🚳 Cartography Tools
		🗄 🚳 Conversion Tools
		표 😂 Data Interoperability Tools
(.		🗄 🗃 Data Management Tools
Add Toolbox		🕀 😂 Editing Tools
Look in: 🔯 H	Home - Exercise 10_28_14\Exar 💌 📤 🏠 🎲 🐺 🕇 🔛 🖆 🗊 🚳	Sercise5
elevation		🕀 😂 Geocoding Tools
Exercise5		🕀 🚳 Geostatistical Analyst Tool
		🗄 🚳 Linear Referencing Tools
		🗄 🚳 Multidimension Tools
		🗄 🚳 Network Analyst Tools
		🕀 🚳 Parcel Fabric Tools
		🕀 🚳 Schematics Tools
		🕀 😂 Server Tools
1		🕀 😂 Spatial Analyst Tools
Name:	Toolbox.tbx Open	🕀 😂 Spatial Statistics Tools
Show of type:	Toolboxes Cancel	🗉 📦 Tracking Analyst Tools

After creating your toolbox (i.e. Exercise 5), right click on it and select New -> Model. You will end up with an empty model. Drag and drop the Fill tool onto the canvas, along with the clipped elevation raster.



From the menu, select Model -> Export -> To Python Script. Open the export Python file to view the code that was written for us by ArcGIS. This is an easy way to extend a model that you have already created.

```
# -*- coding: utf-8 -*-
# -----
# trace_point_downstream.py
# Created on: 2014-10-27 14:23:28.00000
# (generated by ArcGIS/ModelBuilder)
# Description:
# _____
# Import arcpy module
import arcpy
# Check out any necessary licenses
arcpy.CheckOutExtension("spatial")
# Local variables:
ned30m = "ned30m"
Fill_ned30m1 = "C:\\Users\\Tony Castronova\\Documents\\ArcGIS\\Default.gdb\\Fill_ned30m1"
# Process: Fill
arcpy.gp.Fill_sa(ned30m, Fill_ned30m1, "")
```

Notice that there are some strange variable names. Lets modify this code so that the variable names make a little more sense and fix the file paths. Also import the **numpy**, and **math** libraries which we will need later.

```
# Import arcpy module
print 'importing arcpy (this takes a while)...'
import arcpy
from arcpy import env
from arcpy.sa import *
import numpy
import math
# Check out any necessary licenses
arcpy.CheckOutExtension("spatial")
env.overwriteOutput = True
# Local variables:
ned30m = "Z:/windows_shared/exercise 10_28_14/example_data/elevation/zipfolder/ned30m"
fill_outpath = "Z:/windows_shared/exercise 10_28_14/example_data/fill"
# Process: Fill
print 'Run Fill!'
outFill = Fill(ned30m, "")
outFill.save(fill_outpath)
#arcpy.gp.Fill_sa(ned30m, fill_outpath, "")
print 'done'
```

Lets run this code and see what kind of output we get. If the script ran successfully, we should have a new raster called **fill** that can be opened in ArcMap. Note: the original script used the **gp.Fill_sa** tool whereas the documentation states that we should use the **arcpy.sa.Fill** tool. If you encounter this, I suggest that you use the tools outlined in the ArcGIS documentation.

Next, lets calculate flow direction. To determine the syntax for this operation we can google "ArcGIS Flow Direction":

http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//009z00000052000000 .htm

```
# Import arcpy module
print 'importing arcpy (this takes a while)...'
import arcpy
from arcpy import env
from arcpy.sa import *
import numpy
import math
# Check out any necessary licenses
arcpy.CheckOutExtension("spatial")
env.overwriteOutput = True
# Local variables:
ned30m = "Z:/windows_shared/exercise 10_28_14/example_data/elevation/zipfolder/ned30m"
fill_outpath = "Z:/windows_shared/exercise 10_28_14/example_data/fill"
fdr_outpath = "Z:/windows_shared/exercise 10_28_14/example_data/fdr"
                                                                         # Added
# Process: Fill
print 'Run Fill!'
outFill = Fill(ned30m, "")
#outFill.save(fill_outpath)
                                                                        # Modified
# Prcess: FDR
print 'Run FDR!'
                                                                         # Added
outFlowDirection = FlowDirection(outFill, "NORMAL")
                                                                        # Added
                                                                        # Added
outFlowDirection.save(fdr_outpath)
```

```
print 'done'
```

Notice that the output from the fill operation was not **Saved**, but it can still be used in the following step! This is because it is saved temporarily in memory. We can utilize this feature to "hide" intermediary processing outputs. Lets look at the output from the flow direction process.



Now that we have some of the basic raster processing done, lets create a point that can be traced to the outlet. This will be hardcoded for now, but we can change it to a user input later.

```
# Import arcpy module
print 'importing arcpy (this takes a while)...'
import arcpy
from arcpy import env
from arcpy.sa import *
import numpy
import math
# Check out any necessary licenses
arcpy.CheckOutExtension("spatial")
env.overwriteOutput = True
# Local variables:
ned30m = "Z:/windows_shared/exercise 10_28_14/example_data/elevation/zipfolder/ned30m"
fill_outpath = "Z:/windows_shared/exercise 10_28_14/example_data/fill"
fdr_outpath = "Z:/windows_shared/exercise 10_28_14/example_data/fdr"
# create a point object
                                                                        # Added
my_x = -1216071.141
                                                                         # Added
my_y = 307660.098
                                                                         # Added
pnt = arcpy.Point(my_x, my_y)
                                                                         # Added
# Process: Fill
print 'Run Fill!'
outFill = Fill(ned30m, "")
#outFill.save(fill_outpath)
# Prcess: FDR
print 'Run FDR!'
outFlowDirection = FlowDirection(outFill, "NORMAL")
outFlowDirection.save(fdr_outpath)
```

In order to relate this point coordinate with the raster data, we need to do two things: (1) represent the raster grids as arrays of data, and (2) convert the x,y point coordinate into array indices. To convert the raster grids (i.e. **fill** and **fdr**) into arrays, we use the numpy library, specifically RastertoNumPyArray.

```
# Import arcpy module
print 'importing arcpy (this takes a while)...'
import arcpy
from arcpy import env
from arcpy.sa import *
import numpy
import math
# Check out any necessary licenses
arcpy.CheckOutExtension("spatial")
env.overwriteOutput = True
# Local variables:
ned30m = "Z:/windows_shared/exercise 10_28_14/example_data/elevation/zipfolder/ned30m"
fill_outpath = "Z:/windows_shared/exercise 10_28_14/example_data/fill"
fdr_outpath = "Z:/windows_shared/exercise 10_28_14/example_data/fdr"
# create a point object
my_x = -1216071.141
my_y = 307660.098
pnt = arcpy.Point(my_x, my_y)
# Process: Fill
print 'Run Fill!'
outFill = Fill(ned30m, "")
#outFill.save(fill_outpath)
# Prcess: FDR
print 'Run FDR!'
outFlowDirection = FlowDirection(outFill, "NORMAL")
outFlowDirection.save(fdr_outpath)
                                                                            # Added
# convert rasters to arrays
fdr = arcpy.RasterToNumPyArray(outFlowDirection, nodata_to_value=0)
                                                                             # Added
fill = arcpy.RasterToNumPyArray(outFill, nodata_to_value=0)
                                                                             # Added
print 'done'
```

If you print the value of the **fdr** value you will see this:

```
>>> fdr
array([[0, 0, 0, ..., 0, 0, 0],
      [0, 0, 0, ..., 0, 0, 0],
      [0, 0, 0, ..., 0, 0, 0],
      ...,
      [0, 0, 0, ..., 0, 0, 0],
      [0, 0, 0, ..., 0, 0, 0],
      [0, 0, 0, ..., 0, 0, 0],
      [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

It looks like there are lots of 0's, however this is just because we are seeing a small subset of the data. In fact most of the cells near the edge of the raster will be zero. Lets look at some values elsewhere:

```
>>> fdr[100:110, 100:110]
array([[ 2, 4, 8, 4, 4, 4, 4, 4, 4, 8, 16],
      [ 1, 4, 16, 4, 4, 4, 4, 4, 4, 8, 4],
      [ 1, 2, 4, 4, 4, 4, 4, 8, 8, 4],
      [ 2, 1, 2, 2, 4, 4, 4, 8, 8, 4],
      [ 2, 1, 2, 2, 2, 4, 4, 8, 8, 8],
      [ 1, 2, 2, 1, 2, 4, 8, 16, 8, 16],
      [ 1, 2, 1, 2, 2, 4, 8, 16, 16, 32],
      [ 2, 1, 2, 2, 4, 8, 16, 16, 32, 16],
      [ 1, 1, 2, 1, 4, 16, 16, 32, 32, 16]], dtype=uint8)
```

Before we do anymore processing of the raster data, we need to extract some metadata that will enable us to loop over the raster cells. The numpy arrays only contain raster values, so we will need to use the ArcGIS Raster type to retrieve this information.

We can transform our point coordinates into array indices, now that we have the upper left (x,y), cell width, and cell height. This will enable us to access the raster value of the cell associated with our point.

```
. . .
. . .
# convert rasters to arrays
fdr = arcpy.RasterToNumPyArray(outFlowDirection, nodata_to_value=0)
fill = arcpy.RasterToNumPyArray(outFill, nodata_to_value=0)
# create raster object to get metadata
upperLeft = outFill.extent.upperLeft
ux = upperLeft.X
uy = upperLeft.Y
cell_width = outFill.meanCellWidth
cell_height = outFill.meanCellHeight
# convert point coordinates into raster indices
c = abs(int((ux - pnt.X) / cell_width))
r = abs(int((uy - pnt.Y) / cell_height))
. . .
. . .
```

Lets see where our point lives in the raster array:

>>> (pnt.X,pnt.Y), '--->',(c,r) ((-1216071.141, 307660.098), '--->', (62, 210))

Now we are ready to start moving our point around within the raster. Specifically, we want to move our point from its current location (62,210) to the next downstream cell. In order to accomplish this, we need to add a function at the **top** of our script to check the value of our flow direction grid and move the point accordingly. Place this function right below the import statements.

```
def move_to_next_pixel(fdr, row, col):
    # get the fdr pixel value (x,y)
   value = fdr[row, col]
   #
   #| 32 | 64 | 128 |
   #/ 16 / X / 1 /
   #1814121
    #
   # move the pixel
   if value == 1:
       col += 1
   elif value == 2:
       col += 1
       row += 1
   elif value == 4:
       row += 1
   elif value == 8:
       row += 1
       col -= 1
   elif value == 16:
       col -= 1
   elif value == 32:
       row -= 1
       col -= 1
   elif value == 64:
       row -= 1
   else: #value == 128:
       row -= 1
       col += 1
   return (row, col)
```

This function takes in three arguments: fdr (flow direction array), row (current row index), col (current col index). The first thing that it does is extract the value of the flow direction grid at the current (row, col) location. It then checks this value against all the possible flow direction combinations to determine the next downstream neighbor. It increments the current (row, col) pair and returns the result.

Lets pass in the coordinates of our point and see which direction our cell will flow.

>>> r,c (210, 62) >>> move_to_next_pixel(fdr, r, c) (210, 63)

We can verify this by loading the flow direction raster into ArcMap.



Go To XY (Meters)	Identify	Ξ×
x: -1.216.071.141 Y: 307.660.098	Identify from: <pre><top-most layer=""></top-most></pre>	•
	⊡ · fdr 1	
	Location: -1,215,678.974 307,404.166 Meters	
	Pixel value 1	
	Rowid 0 COUNT 10187	
	Identified 1 feature	
	Activities 1 restore	

Lets can modify our code to repeat this process until the point moves beyond the extent of our raster grid (e.g. through the outlet). In order do so, we need to create a loop that will run until the value at location (r,c) is equal to NoDATA (in this case 0).

```
. . .
. . .
# convert rasters to arrays
fdr = arcpy.RasterToNumPyArray(outFlowDirection, nodata_to_value=0)
fill = arcpy.RasterToNumPyArray(outFill, nodata_to_value=0)
# create raster object to get metadata
upperLeft = outFill.extent.upperLeft
ux = upperLeft.X
uy = upperLeft.Y
cell_width = outFill.meanCellWidth
cell_height = outFill.meanCellHeight
# convert point coordinates into raster indices
c = abs(int((ux - pnt.X) / cell_width))
r = abs(int((uy - pnt.Y) / cell_height))
z = fill[r,c]
while (z != 0):
    pass
. . .
. . .
```

This loop will continue to run while the value of z does not equal 0 (i.e. no data value). Currently, this loop will run indefinitely because z is not changing inside the loop. Lets add some code to fix this by moving (r,c) to its downstream neighbor.

```
. . .
. . .
# convert rasters to arrays
fdr = arcpy.RasterToNumPyArray(outFlowDirection, nodata_to_value=0)
fill = arcpy.RasterToNumPyArray(outFill, nodata_to_value=0)
# create raster object to get metadata
upperLeft = outFill.extent.upperLeft
ux = upperLeft.X
uy = upperLeft.Y
cell_width = outFill.meanCellWidth
cell_height = outFill.meanCellHeight
# convert point coordinates into raster indices
c = abs(int((ux - pnt.X) / cell_width))
r = abs(int((uy - pnt.Y) / cell_height))
z = fill[r,c]
while (z != 0):
   r,c = move_to_next_pixel(fdr, r, c)
. . .
. . .
```

This code will move the point (r,c) to its downstream neighbor, and continue to do so until we reach the watershed outlet. Unfortunately, we have no output to visualize. Lets save these points in a list and then create a shapefile that we can visualize in ArcMap.

```
. . .
. . .
# convert rasters to arrays
fdr = arcpy.RasterToNumPyArray(outFlowDirection, nodata_to_value=0)
fill = arcpy.RasterToNumPyArray(outFill, nodata_to_value=0)
# create raster object to get metadata
upperLeft = outFill.extent.upperLeft
ux = upperLeft.X
uy = upperLeft.Y
cell_width = outFill.meanCellWidth
cell_height = outFill.meanCellHeight
# convert point coordinates into raster indices
c = abs(int((ux - pnt.X) / cell_width))
r = abs(int((uy - pnt.Y) / cell_height))
z = fill[r,c]
while (z != 0):
    # move downstream
   last_r = r
    last_c = c
   r,c = move_to_next_pixel(fdr, r, c)
    # adjust x and y
   pntX += (last_c-c)*cell_width
   pntY += (last_r-r)*cell_height
   z = fill[r,c]
    # save this coordinate
    coords.append((pntX,pntY,z))
# write the output to text file
with open('coords.txt','w') as f:
   for c in coords:
        f.write('%5.5f, %5.5f, %5.5f\n' % (c[0],c[1],c[2]))
```

To visualize our output in ArcMap, add the **coords.txt** file to an ArcMap document. Right click on it and select **Display X,Y data**. Choose **Field1** as the X field and **Field 2** as the Y field. You can also symbolize these points by their elevation, **Field 3**

	Display XY Data	×
	A table containing X and Y coordinate data can be added map as a layer	d to the
	Choose a table from the map or browse for another tab	le:
	coords.txt	- 6
	Specify the fields for the X, Y and Z coordinates:	
	X Field: Field1	•
	Y Field: Field2	-
	Z Field: <none></none>	•
	Coordinate System of Input Coordinates	
Add Data	Projected Coordinate System:	*
Look in: 🔁 scripting 🔹 🏠 🛣 🥻 🔛 🖆 🗊 🚳	Name: NAD_1983_Albers	
coords.txt	Geographic Coordinate System: Name: GCS_North_American_1983	
	4	
	Show Details	dit
		uit
Name: coords.txt Add	Warn me if the resulting layer will have restricted fun	ictionality
Show of type: Datasets, Layers and Results Cancel	About adding XY data OK	Cancel
	•••••••	
5.	······································	
· · · · · · · · · · · · · · · · · · ·		
······································		
5		
5		
$\int d$		
5	7	

Since point text file is not an ideal output, lets format it as a PolyLine Shapefile, http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//00170000002p00000 O. In the code snippet below, we first create the polyline feature class that will hold our results. Next we loop over our coordinates are create line segments between each pair. These line segments are then added to the feature class as a polyline.

```
# create the output feature class
arcpy.CreateFeatureclass_management('.','path.shp', "POLYLINE")
# define the point and line segment objects
point = arcpy.Point()
line_seg = arcpy.Array()
featureList = []
cursor = arcpy.InsertCursor('path.shp')
feat = cursor.newRow()
for i in range(1, len(coords)-1):
    # Set X and Y for start and end points
    point.X = coords[i-1][0]
    point.Y = coords[i-1][1]
    line_seg.add(point)
    point.X = coords[i][0]
    point.Y = coords[i][1]
    line_seg.add(point)
    # Create a Polyline object based on the array of points
    polyline = arcpy.Polyline(line_seg)
    # Clear the array for future use
    line_seg.removeAll()
    # Append to the list of Polyline objects
    featureList.append(polyline)
    # Insert the feature
    feat.shape = polyline
    cursor.insertRow(feat)
del feat
del cursor
```



Now that we have some python code that traces a path downstream of any location, we can add some ArcGIS inputs so that it can be used easily.

First create a symbolic layer, which will be used in the next step, to assign a theme to one of our inputs. This will also allow us to incorporate an interactive point input selection feature. To do this, right click inside ArcCatalog and select **New -> Shapefile.** Set a name for this file (e.g. my_point.shp) and set the feature type to **Point.** Change the symbology of this point however you would like. Lastly, right click on the my_point.shp in the Table of Contents and select **Save as Layer File**.

Create New Shapefi	le	×
Name:	my_point	
Feature Type:	Point	•
Spatial Reference		
Description:		
Unknown Coordir	ate System	*
		T
4		P
Show Details		Edit
Coordinates wi	II contain M values. Used to II contain Z values. Used to	store route data. store 3D data.
	ОК	Cancel

Save Layer	
Look in: 📋	data 🔹 🔹 🚖 🔂 🗔 🛛 🖽 🖆 🖆 🗊 🚳
extract_by_	mask_elevation
extract_elev	vation
symbology	
Namer	
Name:	symbology.lyr Save
Save as type:	Layer files (*.lyr) Cancel

Now lets add our new script to the ArcGIS toolbox, so that we can run it like any other tool. Right click on your toolbox (e.g. Exercise5) and select **Add -> Script**. Give it a name and a label, then select next. Specify the location of the python file.

Script 1 Properties	Add Script
General Source Parameters Validation Help Name: TraceDownStream Label:	Script File: Z:\windows_shared\exercise 10_28_14\example_data\scrip
Description:	I Run Python script in process
Stylesheet:	
OK Cancel Apply	<back next=""> Cancel</back>

Now lets add some input parameters. The first input parameter will be the start point of the trace operation. Specify a **Display Name** (such as StartPoint) and set the datatype to **FeatureSet**. Next select the **Schema** property and set its value to the symbology layer that we created in the previous step. (e.g. symbology.lyr)

		Look in:	🔁 data	•	仓 🟠	a 📖	- 1	a e	1	1
										_
		extract_	by_mask_elevation	watershed.sh	р					
l Script		B buffer n	at.shn							
		Export (Output.shp							
Display Name	Data Type	fdr								
(StartPoint	Feature Set	fill								
		my_poir	nt.shp							
		😳 pt.shp								
		symbol	ogy.lyr							
							_		دده	-
		Name:	symbology lyr						40.00	
Click any paramete	er above to see its propertie	Name: est Show of type	symbology.lyr e:				•		ancel	
Click any paramete Parameter Prope	er above to see its propertie erties	Name: Show of type	symbology.lyr				•		ancel	
Click any paramete Parameter Prope Property	er above to see its propertie erties Value	Name: st Show of type	symbology.lyr e:				•		ancel	
Click any parameter Parameter Prope Property Type	er above to see its propertie erties Value Required	Name: st Show of type	symbology.lyr				•		ancel	
Click any parameter Parameter Prope Property Type Direction	er above to see its propertie erties Value Required Input	Name: st Show of type	symbology.lyr				•	C	ancel	
Click any parameter Parameter Prope Property Type Direction MultiValue	er above to see its propertie arties Value Required Input No	Name: st Show of type	symbology.lyr				•	C	ancel	
Click any parameter Parameter Proper Property Type Direction MultiValue Schema	r above to see its propertie rties Value Required Input No	Name: st Show of typ	symbology.lyr e:				•		ancel	
Click any parameter Parameter Proper Property Type Direction MultiValue Schema Environment	er above to see its propertie rties Value Required Input No	Name: Show of typ	symbology.lyr e:				•	C	ancel	
Click any parameter Parameter Proper Property Type Direction MultiValue Schema Environment Filter	r above to see its propertie rties Value Required Input No None	Name:	symbology.lyr e:				•	C	ancel	

Lets also add parameters for Elevation (input), Fill (output), Flow Direction (output), and Path (output). Make sure that the direction parameter for the last three are set to Output.

	raianeten			
Display Name		Data Type		
Start Point		Feature Set		
Elevation		Raster Dataset		4
Filled Elevatio	on	Raster Dataset		_
Flow Direction	n	Raster Dataset		
Path		Shapefile		•
Property	Value		-	
Filiperty	Value			
Direction	Toput			
MultiValue	No		=	
Muluvalue	NU			
Default				
Default	None			
Default Environment Filter			-	
Default Environment Filter	NOTE			
Default Environment Filter Obtained from	None			
Default Environment Filter Obtained from	ameter, typ	e the name into an empty row in t	the	
Default Environment Filter Obtained from	ameter, typ	be the name into an empty row in t a Type column to choose a data ty	the ype,	
Default Environment Filter Obtained from To add a new par name column, clic hen edit the Para	rameter, typ k in the Dat ameter Prop	be the name into an empty row in t a Type column to choose a data ty erties.	the ype,	
Default Environment Filter Obtained from To add a new par name column, clic hen edit the Para	ameter, typ k in the Dat ameter Prop	be the name into an empty row in a Type column to choose a data ty perties.	the ype,	

Now we need to add some code to our python script to utilize these parameters. We use the **arcpy.GetParameter(index)** function to grab user inputs from the ArcGIS UI. The following snippet gets the first parameter (i.e. Start Point) as a feature set, and extracts the (x,y) coordinates. This code should be placed directly under the **move_to_next_pixel(fdr, row, col)** function.

```
fs = arcpy.GetParameter(0)
if fs == '#' or not fs:
    fs = "in_memory\\{87AF799A-1608-483B-9022-3AA58EFEF329}" # provide a default value if unspecified
# create feature set
f = arcpy.FeatureSet(fs)
# parse out the geometry
geom = json.loads(f.JSON)['features'][0]['geometry']
pnt = arcpy.Point(geom['x'], geom['y'])
arcpy.AddMessage('Selected Point = (%5.3f,%5.3f)'% (geom['x'], geom['y']))
```

Next, lets add some code to get the rest of our inputs and outputs:

```
# get elevation input
elevation = arcpy.GetParameterAsText(1)
# get output fill path
fill_outpath = arcpy.GetParameterAsText(2)
# get output fdr path
fdr_outpath = arcpy.GetParameterAsText(3)
# get output trace path
trace_outpath = arcpy.GetParameterAsText(4)
```

Since we are getting these parameters from ArcGIS, we need to remove our old hardcoded paths. We should also add some messages, since our print statements will not appear anywhere. Add or remove the following lines in your script.

```
import os
. . .
# Local variables:
# ned30m = "Z:/windows_shared/exercise 10_28_14/example_data/elevation/zipfolder/ned30m"
# fill_outpath = "Z:/windows_shared/exercise 10_28_14/example_data/fill"
# fdr_outpath = "Z:/windows_shared/exercise 10_28_14/example_data/fdr"
# create a point object
\# my_x = -1216071.141
\# my_y = 307660.098
# pnt = arcpy.Point(my_x, my_y)
# Process: Fill
arcpy.AddMessage('Running DEM Fill...')
# outFill = Fill(ned30m, "")
outFill = Fill(elevation, "")
outFill.save(fill_outpath)
# Process: FDR
arcpy.AddMessage('Running FDR...')
outFlowDirection = FlowDirection(outFill, "NORMAL")
outFlowDirection.save(fdr_outpath)
. . .
#arcpy.CreateFeatureclass_management('.', 'path.shp', "POLYLINE")
directory_path = os.path.dirname(trace_outpath)
file_path = os.path.basename(trace_outpath)
arcpy.CreateFeatureclass_management(directory_path,file_path, "POLYLINE")
#cursor = arcpy.InsertCursor('path.shp')
cursor = arcpy.InsertCursor(trace_outpath)
```

With these changes to our python script, we should be able to successfully run our tool from the ArcGIS toolbox.



Start Point		
Turan Davie Charter Daviet		
TraceDownStream::Start_Point		- E
Start_Point		
Elevation		
Z:\windows_shared\exercise 10_28_14\exam	ole_data\data\ned30m	
Filled Elevation		
Z:\windows_shared\exercise 10_28_14\exam	ole_data\data\fill	2
Flow Direction		
Z:\windows_shared\exercise 10_28_14\exam	ole_data\data\fdr	6
Path		

Note that our output messages appear in the standard ArcGIS output dialog.





Homework Questions

- 1. What is the value of the flow direction array at location (134, 289)? Turn in your result along with the line of code that gave you this answer.
- 2. What is the bounding box of the Flow direction array, (i.e. MinX, MinY, MaxX, MaxY)? Turn in your result along with the line of code that gave you this answer.
- 3. Write a statement to select the maximum and minimum values of the fill raster. Turn in your result along with the line of code that you used.
- 4. Explain how the **move_to_next_pixel** function works. Use examples.
- 5. Modify the **while** loop in this code so that it will terminate at a user defined distance from the input point. For instance, a 1000 meter radius from the input point.
- 6. How could you modify this code to determine the longest flow path in the entire watershed?
- 7. Modify the code to provide the min, max, and average slope along the trace path.
- 8. Explain how you would modify the code to operate on a list of points instead of just one.