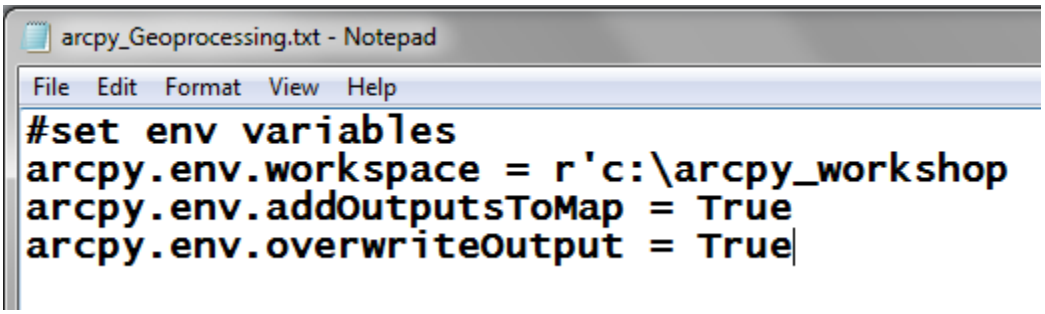
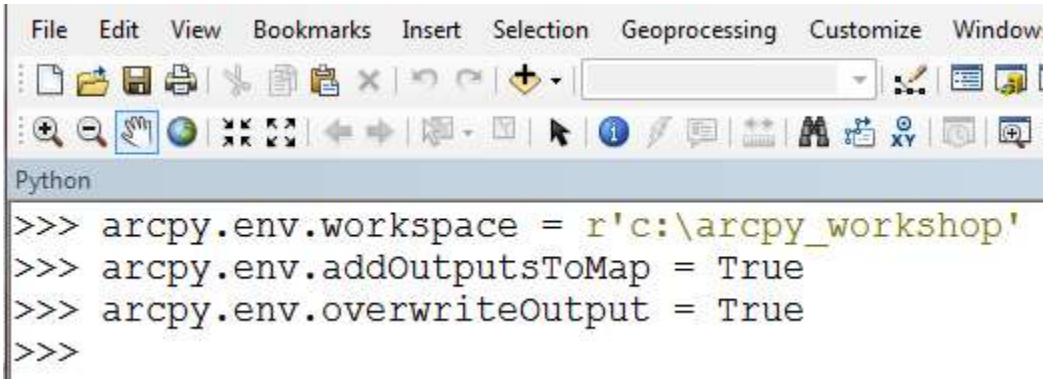
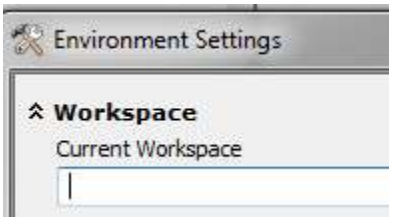
Session 3: Python Geoprocessing

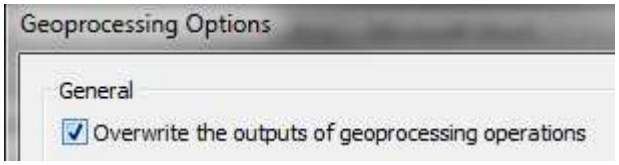
In this session we use ArcGIS geoprocessing tools in the Python window. Typically you first set your environment and extensions. For example, copy (Ctrl-C) following from the ***arcpy_Geoprocessing.txt*** file and paste (Ctrl-V) each line into your arcmap Python window

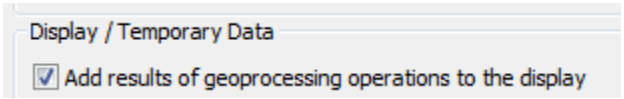




This is analogous to when you are working interactively and manually set these environment variables:







You must also load an extension to get access to the geoprocessing tools associated with an extension. For example,

arcpy.CheckInExtension("spatial")

randomRaster = arcpy.sa.CreateRandomRaster()

results in..

```
ERROR 000824: The tool is not licensed.
```

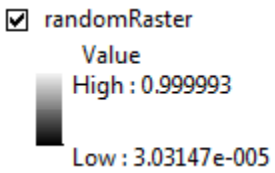
So you need to check out the spatial analyst extension first, then you have access to all the tools associated with that extension.

arcpy.CheckOutExtension("spatial")

press the up-arrow key to recall your .CreateRandomRaster() line and then execute that line once again by pressing the Enter key

```
>>> arcpy.CheckOutExtension("spatial")
u'CheckedOut'
>>> randomRaster = arcpy.sa.CreateRandomRaster()
>>>
```

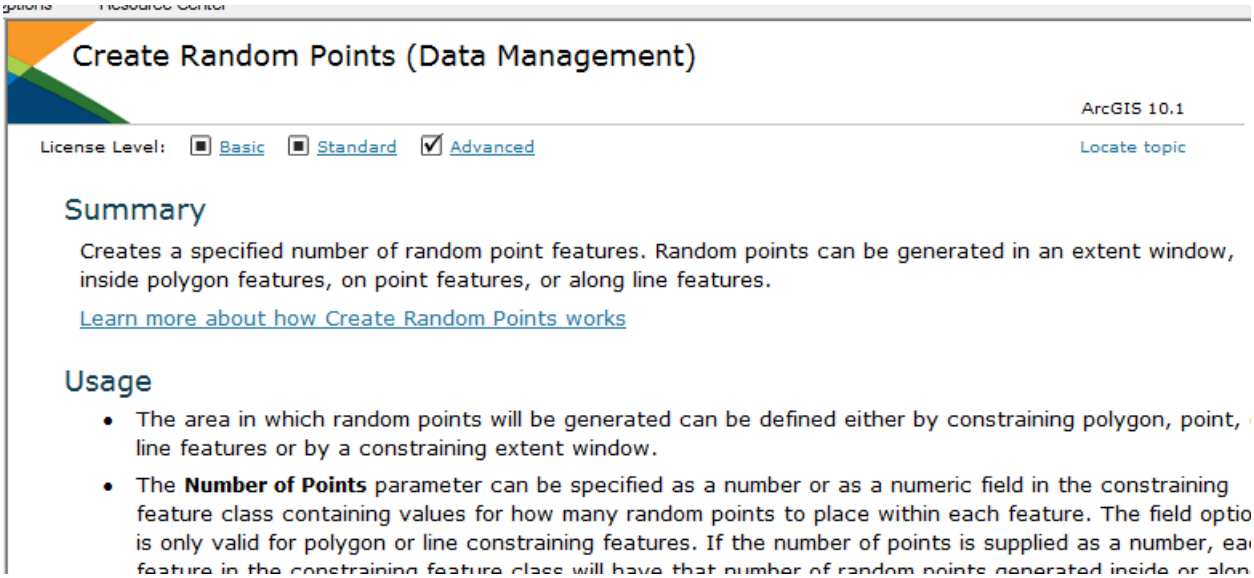
```
Start Time: Tue Mar 18 03:41:47 2014
Succeeded at Tue Mar 18 03:41:48 2014 (Elapsed Time: 1.00 seconds)
```



Your random raster will have pixel values ranging from 0.0 up to 1.0

Executing Geoprocessing Tools

First use ArcGIS Help to read about the geoprocessing tool and look at an example Python script..



Create Random Points (Data Management)

ArcGIS 10.1

License Level: Basic Standard Advanced Locate topic

Summary

Creates a specified number of random point features. Random points can be generated in an extent window, inside polygon features, on point features, or along line features.

Learn more about how Create Random Points works

Usage

- The area in which random points will be generated can be defined either by constraining polygon, point, line features or by a constraining extent window.
- The **Number of Points** parameter can be specified as a number or as a numeric field in the constraining feature class containing values for how many random points to place within each feature. The field optio is only valid for polygon or line constraining features. If the number of points is supplied as a number, ea feature in the constraining feature class will have that number of random points generated inside or alon

.

.

Code Sample

CreateRandomPoints example 1 (Python window)

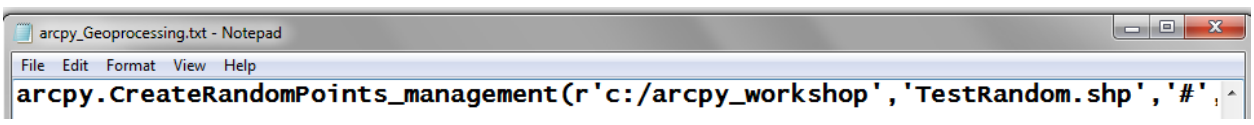
The following Python window script demonstrates how to use the CreateRandomPoints tool in immediate mode:

```
import arcpy
arcpy.CreateRandomPoints_management("c:/data/project", "samplepoints", "c:/data/studyar
```

CreateRandomPoints with Random Values example 2 (stand-alone Python script)

The following stand-alone Python script demonstrates how to create random points with random values:

Test drive the tool to create one shapefile containing 20 random points…copy and paste the Python geoprocessing line from your ***arcpy_Geoprocessing.txt*** file to your arcmap Python window.

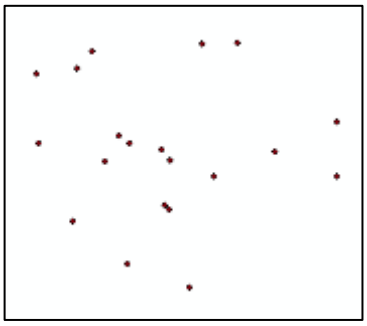


arcpy_Geoprocessing.txt - Notepad

File Edit Format View Help

```
arcpy.CreateRandomPoints_management(r'c:/arcpy_workshop','TestRandom.shp','#',
```

```
Python
>>> arcpy.CreateRandomPoints_management
 (r'c:/arcpy_workshop','TestRandom.shp','#','#',20)
<Result 'c:\\arcpy_workshop\\TestRandom.shp'>
>>>
Start Time: Sat Jan 11 07:46:59 2014
Succeeded at Sat Jan 11 07:46:59 2014 (Elapsed Time: 0.00 seconds)
```



If you want to skip a geoprocessing tool parameter, use the "#"..what 2 parameters are skipped in the above example?

Syntax

CreateRandomPoints_management (out_path, out_name, {constraining_feature_class}, {constraining_extent}, {number_of_points_or_field}, {minimum_allowed_distance}, {create_multipoint_output}, {multipoint_size})

Parameter	Explanation	Data Type
out_path	The location or workspace in which the random points feature class will be created. This location or workspace must already exist.	Feature Dataset;Workspace
out_name	The name of the random points feature class to be created.	String
constraining_feature_class (Optional)	Random points will be generated inside or along the features in this feature class. The constraining feature class can be point, multipoint, line, or polygon. Points will be randomly placed inside polygon features, along line features, or at point feature locations. Each feature in this feature class will have the specified number of points generated inside it (for example, if you specify 100 points, and the constraining feature class has 5 features, 100 random points will be generated in each feature, totaling 500 points).	Feature Layer
constraining_extent (Optional)	Random points will be generated inside the extent. The constraining extent will only be used if no constraining feature class is specified.	Extent;Feature Layer;Raster Layer
number_of_points_or_field (Optional)	The number of points to be randomly generated.	Field;Long

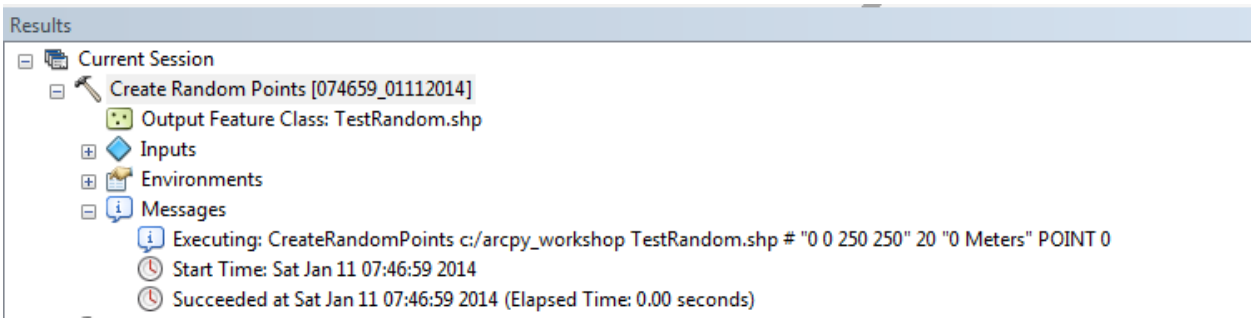
See if you can figure out how to run the ***Add XY*** geoprocessing tool with your random points layer **in your Python window**….this geoprocessing tool will create the fields Point_X, Point_Y:

TestRandom

FID	Shape	Id	POINT_X	POINT_Y
0	Point	0	210.307082	152.966785
1	Point	0	152.999548	66.136257
2	Point	0	178.855861	191.418011
3	Point	0	185.102145	178.861404

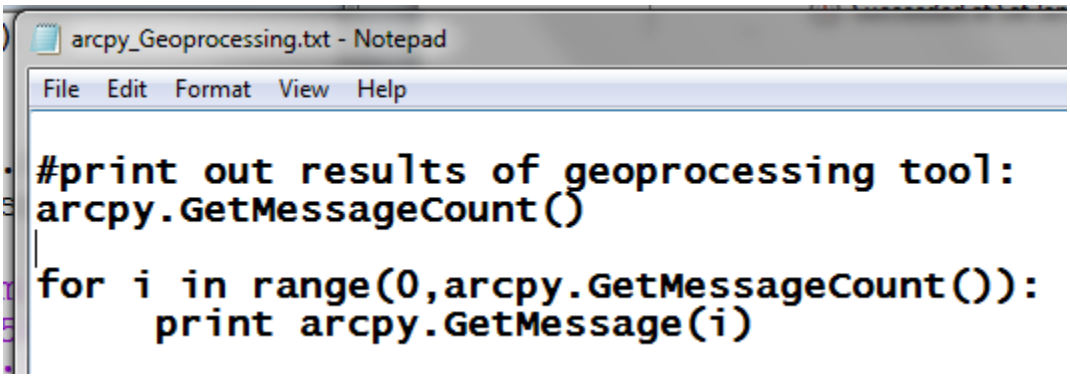
Geoprocessing Results

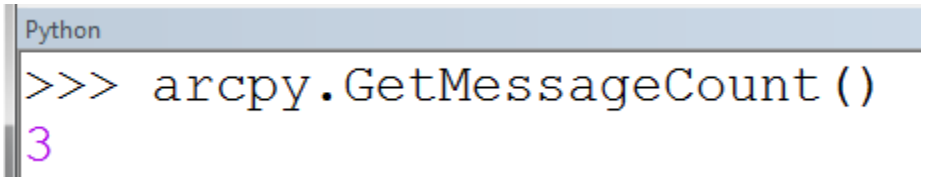
Every time you execute a Geoprocessing tool, the results are output to the Results window…



You can access the same messages as text strings in your Python window.

Copy and paste the following from your ***arcpy_Geoprocessing.txt*** file:





```
>>>
... for i in range(0,arcpy.GetMessageCount()):
...       print arcpy.GetMessage(i)
...
Executing: AddXY TestRandom
Start Time: Tue Mar 18 03:54:49 2014
Succeeded at Tue Mar 18 03:54:49 2014 (Elapsed
```

So Message 0 is : `Executing: AddXY TestRandom`

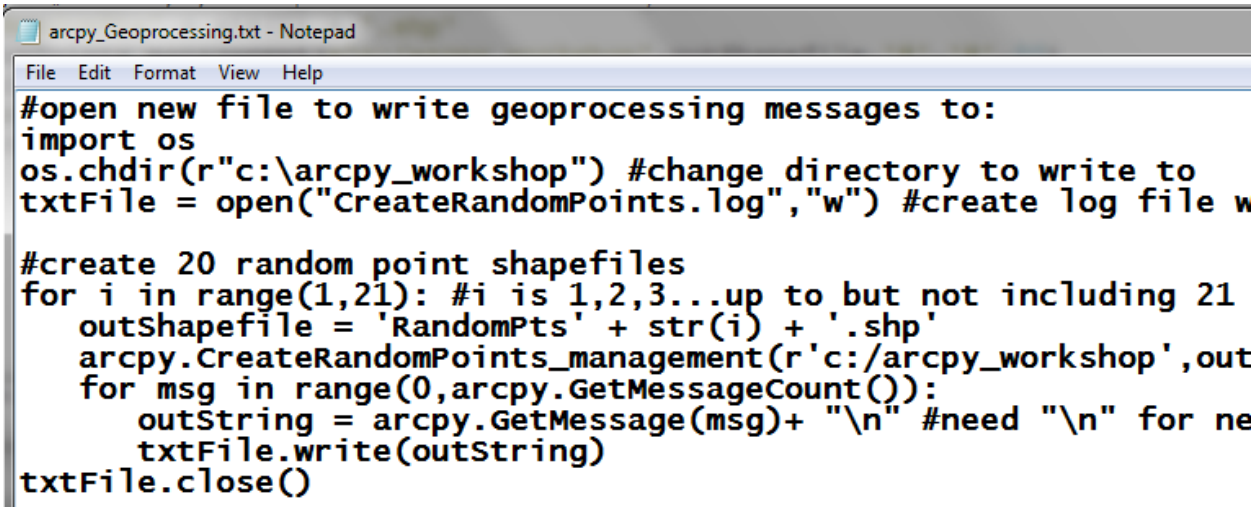
Message 1 is : `Start Time: Tue Mar 18 03:54:49 2014`

Message 2 is :

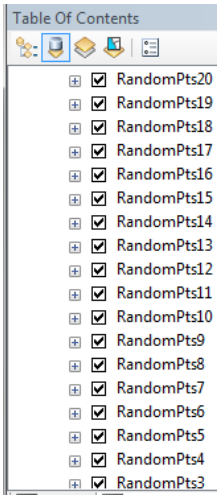
```
Succeeded at Tue Mar 18 03:54:49 2014 (Elapsed Time: 0.00 seconds)
```

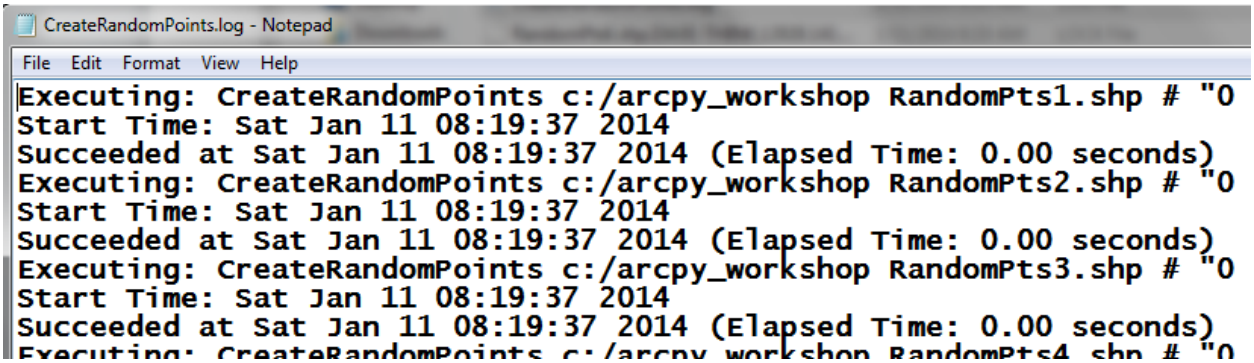
Geoprocessing Loops

Now lets create 20 shapefiles, each containing 20 random points…and output to a log file the results of the geoprocessing. Copy and paste the following from your text file…



Here you opened a new file names “CreateRandomPoints.log”, then created a loop going from 1 to 20, naming our output shapefiles RandomPts1.shp, RandomPts2.shp . . . RandomPts20.shp and outputting the geoprocessing results to your file.

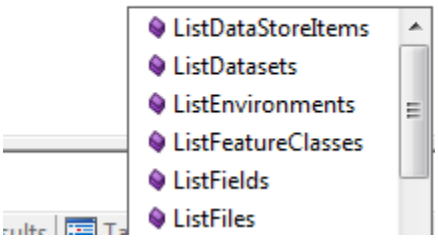




Arcpy Listing Commands

There is a suite of arcpy ***.List*** commands..

```
arcpy.List
```



Try the following to obtain a list of all feature classes that are POINT shapetype, and end with ".shp".

```
Python
>>> arcpy.env.workspace = r'c:\arcpy_workshop'
>>> lstShapefiles = arcpy.ListFeatureClasses("*.shp","POINT")
>>> lstShapefiles
[u'RandomPts1.shp', u'RandomPts10.shp', u'RandomPts11.shp', u'RandomPts12.shp', u'Ra
 u'RandomPts14.shp', u'RandomPts15.shp', u'RandomPts16.shp', u'RandomPts17.shp', u'R
```

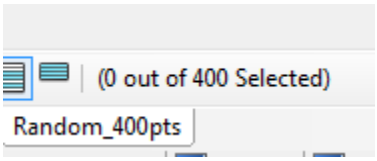
Let's create a list of point feature classes that start with "Random" and merge these into one point feature class:

```
>>> lstShapefiles = arcpy.ListFeatureClasses("Random*.shp","POINT")
>>> lstShapefiles
[u'RandomPts1.shp', u'RandomPts10.shp', u'RandomPts11.shp', u'RandomPts12.shp',
u'RandomPts13.shp', u'RandomPts14.shp', u'RandomPts15.shp', u'RandomPts16.shp',
u'RandomPts17.shp', u'RandomPts18.shp', u'RandomPts19.shp', u'RandomPts2.shp',
u'RandomPts20.shp', u'RandomPts3.shp', u'RandomPts4.shp', u'RandomPts5.shp',
u'RandomPts6.shp', u'RandomPts7.shp', u'RandomPts8.shp', u'RandomPts9.shp']
```

```
>>> arcpy.Merge_management(lstShapefiles,'Random_400pts.shp')
<Result 'c:\\arcpy_workshop\\Random_400pts.shp'>
>>>
```

```
Start Time: Tue Mar 18 04:14:00 2014
Succeeded at Tue Mar 18 04:14:00 2014 (Elapsed Time: 0.00 seconds)
```

Open your point attribute table…you should have 400 records…



Or you can use the arcpy .GetCount() geoprocessing command:

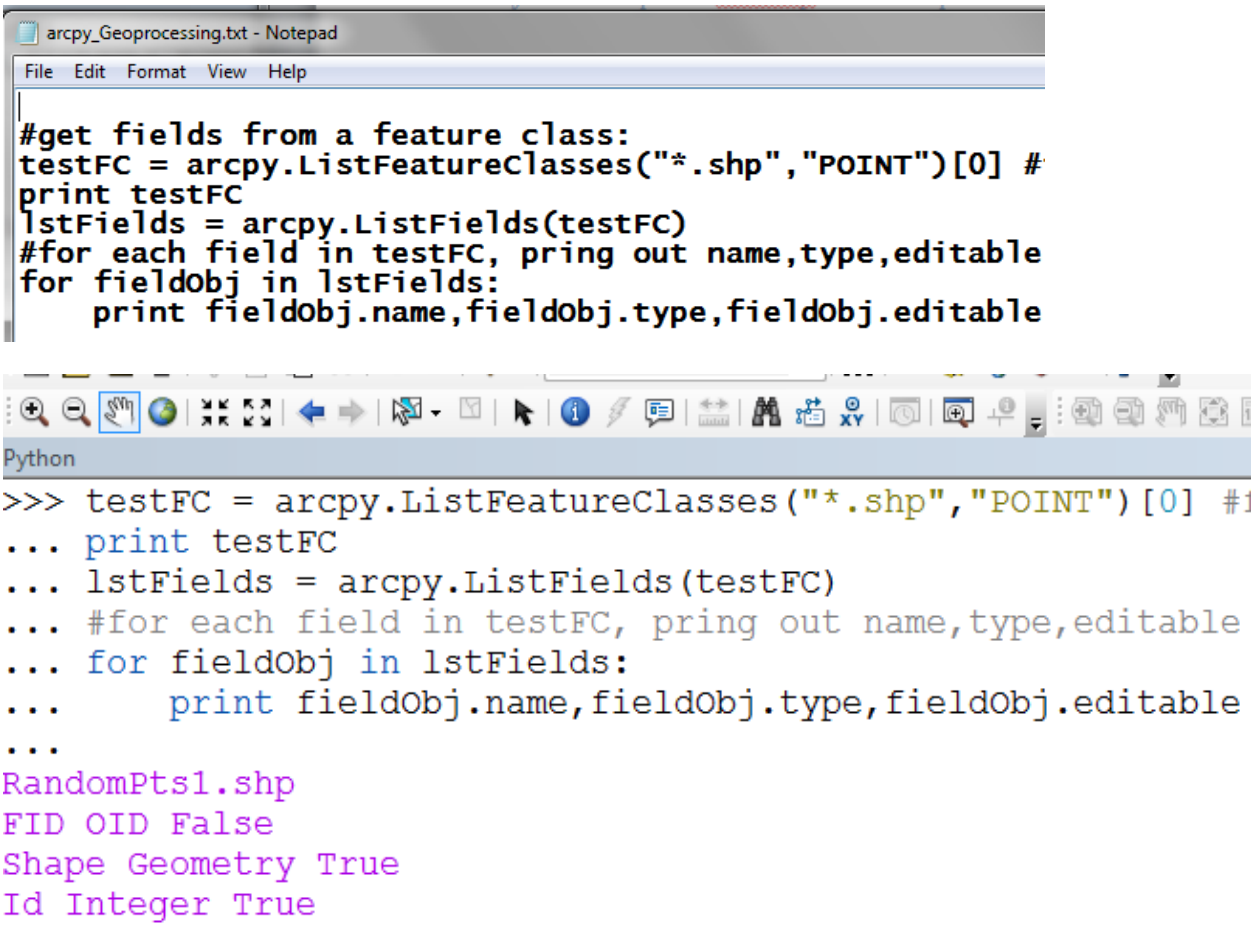
```
>>> txtCount = str(arcpy.GetCount_management('Random_400pts.shp'))
>>> txtCount
'400'
```

The ***.ListFields*** command returns a list of field objects. Each field object has properties

Properties

Property	Explanation	Data Type
aliasName (Read and Write)	The alias name of the field.	String
baseName (Read and Write)	The unqualified field name.	String
domain (Read and Write)	The name of the associated domain.	String
editable (Read and Write)	The editable state: True if the field is editable.	Boolean
isNullable (Read and Write)	The nullable state: True if the field allows null values.	Boolean
length (Read and Write)	The field's length.	Integer
name (Read and Write)	The name of the field.	String
precision (Read and Write)	The precision of the field.	Integer
required (Read and Write)	The required state: True if the field must contain a value.	Boolean
scale (Read and Write)	The field's scale.	Integer
type (Read and Write)	The field type. • Blob —Blob • Date —Date • Double —Double • Geometry —Geometry • Guid —Guid • Integer —Integer (Long Integer) • OID —Object ID • Raster —Raster • Single —Single (Float) • SmallInteger —Small Integer (Short Integer) • String —String (Text) Learn more about ArcGIS field data types	String

Try the following:



So the testFC has three fields: FID, Shape, and ID

To access field values, you will use the data access cursors…we will cover that in the last session.

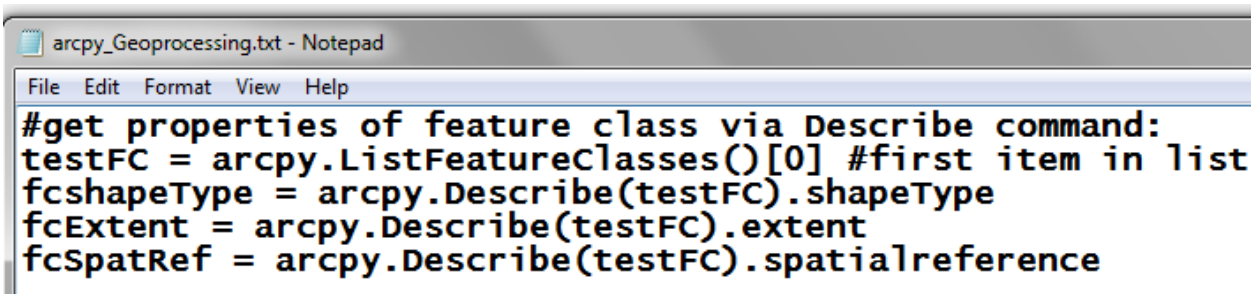
Getting Feature Counts

You can use the arcpy ***.GetCount*** command to get the count of features in a feature class. The result of the .GetCount command can then be converted to a string. For example:

```
#get the feature count of every feature class:
arcpy.env.workspace = r'c:\arcpy_workshop'
lstFC = arcpy.ListFeatureClasses()
for fc in lstFC:
   featureCount = arcpy.GetCount_management(fc)
   print fc,str(featureCount) + " features"
```

Describing Feature Classes

The arcpy ***.Describe*** command can be used to get properties about objects such as workspace, table, raster, and feature class properties. Try the following:



shapeType will be a text string..

shapeType (Read Only)	The geometry shape type. • Polygon • Polyline • Point • MultiPoint • MultiPatch	String

```
>>> print fcshapeType
Point
```

Extent will be an object with properties….

Properties

Property	Explanation	Data Type
MMax (Read Only)	The extent MMax value. None if no M value.	Double
MMin (Read Only)	The extent MMin value. None if no M value.	Double
XMax (Read Only)	The extent XMax value.	Double
XMin (Read Only)	The extent XMin value.	Double
YMax (Read Only)	The extent YMax value.	Double
YMin (Read Only)	The extent YMin value.	Double
ZMax (Read Only)	The extent ZMax value. None if no Z value.	Double
ZMin (Read Only)	The extent ZMin value. None if no Z value.	Double
depth (Read Only)	The extent depth value. None if no depth.	Double
height (Read Only)	The extent height value.	Double
lowerLeft (Read Only)	The lower left property: A point object is returned.	Point
lowerRight	The lower right property: A point object is returned.	Point

Try the following:

```
#some extent properties:
xMin = fcExtent.XMin
xMax = fcExtent.XMax
print fcExtent.width
print xMax - xMin
```

Let's work with a feature class that has the projection defined, and then look at its spatial reference properties.

arcpy_Geoprocessing.txt - Notepad

File Edit Format View Help

```
#spatial reference propetries of Polygons.shp
polyFC = r'C:\arcpy_workshop\Session2_Field_Calculations\GIS_Data\Polygons.shp'
fcSpatRef = arcpy.Describe(polyFC).spatialreference
fcSpatRef.exporttostring() #same information you would find in prj file
SpatReftype = fcSpatRef.type #either Unknown, Geographic, or Projected
print SpatReftype
```

```
>>> print SpatReftype
Projected
```

If a polygon feature class is projected, you can use the Shape.area property to compute polygon area in Hectares, Acres, etc.

Geometry unit conversions

Shape and length properties of the geometry field can be modified with unit types expressed with an @ sign.

- Areal unit of measure keywords:
  - ACRES | ARES | HECTARES | SQUARECENTIMETERS | SQUAREDECIMETERS | SQUAREINCHES | SQUAREFEET | SQUAREKILOMETERS | SQUAREMETERS | SQUAREMILES | SQUAREMILLIMETERS | SQUAREYARDS | SQUAREMAPUNITS | UNKNOWN

Try the following Python script…if the feature class is a polygon shape type, check that the coordinate system is projected, if it is, add a field named "Ha" and calculate the polygon areas in hectares…

arcpy_Geoprocessing.txt - Notepad

File Edit Format View Help

```
#compute polygon areas in Hectares if feature class is projected:
if(arcpy.Describe(polyFC).ShapeType == "Polygon"):
    if( SpatReftype == "Projected"):
        arcpy.AddField_management(polyFC,"Ha","DOUBLE")
        arcpy.CalculateField_management(polyFC,"Ha","!Shape.area@Hectares!","PYTHON")
```

Polygons

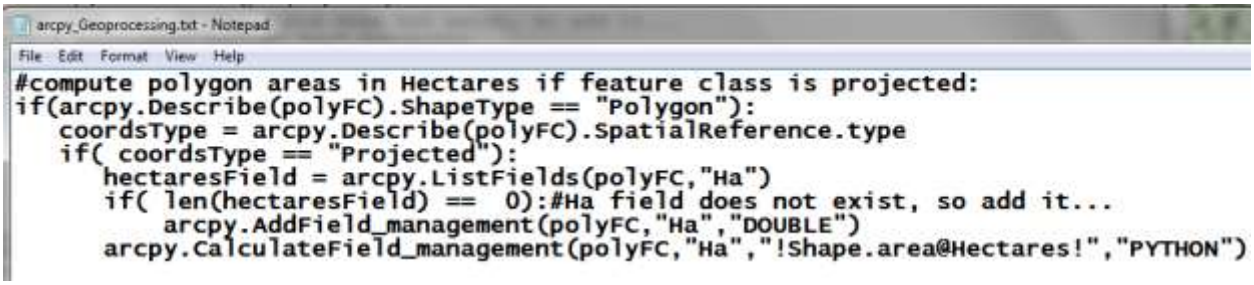
	Shape	Ha
▶	Polygon	189021.670616
	Polygon	112.091976
	Polygon	1264.286267
	Polygon	13.723777

Typically your script will add the field if it does not already exist. Try the following…use the .ListFields to search for a field named “Hectares” and then search for a field named “Ha”

```
#check if field already exists:
hectaresField = arcpy.ListFields(polyFC,"Hectares")
print len(hectaresField) #0---field does not exist

hectaresField = arcpy.ListFields(polyFC,"Ha")
print len(hectaresField) #1---field exists
```

And finally the modified script that checks if the field “Ha” already exists…



Take a 10-minute break.

The next session will be creating Python script tools in your personal toolbox…