Exercise 1: Scripting geoprocessing tools

Your task in this exercise is to locate high-quality habitat patches—areas that have the best potential to support a coastal bird, the California Gnatcatcher. High quality habitat for this species is defined by a number of criteria:

1. Habitat must be in a **Coastal Sage-Chaparral Scrub**, **Diegan Coastal Sage Scrub**, or **Maritime Succulent Scrub** vegetation area
2. Habitat should not be within **1000 feet** of a highway or major road
3. Habitat must have an **elevation below 250m**
4. Habitat must be mostly flat (**slope less than 40%**)
5. Habitat must be a **contiguous area larger than 100 acres**
6. Habitat must be in a **Coastal** climate

You will write a Python script that performs geoprocessing to find all habitat patches within a study area that meet these criteria. Writing this workflow as a Python script will allow the analysis to be easily shared and reproduced – this is especially important if the desired habitat characteristics may change or if applying the workflow to a different study area.

Examine data

- Navigate to folder …Python_ArcGIS\exercises\exercise1_gp
- Double-click Habitat_Analysis.mxd to open it in ArcMap, and examine the data layers that are available in the table of contents
- The feature classes needed to complete this exercise are stored at …Python_ArcGIS\exercises\exercise1_gp\data\Habitat_Analysis.gdb
- Close ArcMap

Setting up the script

- Open PythonWin
- Click the New button and create a new Python script

- The first thing to do is to write some comments; summary, author, date, etc

```
# Script to find high-quality habitat patches
# Created by San Diego Gnatcatcher Conservation Fund
# On March 15, 2010
```

- The opening lines of a script will usually be to import modules like arcpy and os

```
import arcpy
import os
```

- It is also a good idea to start using geoprocessing environments like workspace to make your scripting easier
  - overwriteOutput will allow you to automatically overwrite a dataset if it already exists (without setting this environment you would get an error)
  - workspace will allow you to set the location where your data is stored, then you can reference feature classes by name instead of full path

```
arcpy.env.overwriteOutput = True
arcpy.env.workspace = os.path.join(os.path.dirname(os.getcwd()), "data", "Habitat_Analysis.gdb")
```

- Set up some variables so you can use these variable names in your geoprocessing commands instead of longer paths
  - You only need to point to the name of these feature classes because they are all contained in the geoprocessing workspace set above

```
roads = "MajorRoads"
veg = "Vegetation"
climate = "ClimateZones"
elev = "Elev_LT_250"
slope = "Slope_LT_40"
```

- Click the Save button and save this script to the directory …Python_ArcGIS\exercises\exercise1_gp\scripts

Scripting geoprocessing tools

With most geoprocessing workflows, there may be more than one way to get the same results. Below is one workflow which will produce areas with the desired habitat characteristics. We will go down the list of criteria and perform a geoprocessing operation to find the areas that meet each specification.

Each geoprocessing tool will have a different set of input and output arguments to be specified when running the tool in Python. ArcGIS desktop help can be a very valuable resource for learning how to use tools in Python. You can open ArcGIS desktop help from **Programs>ArcGIS>ArcGIS Desktop Help>ArcGIS Desktop 10 Help**. Navigate to the geoprocessing tool reference book at **Professional Library>Geoprocessing>Geoprocessing Tool Reference.** Here you can find usage tips as well as Python syntax for all of ArcGIS's geoprocessing tools, broken down by toolbox and toolset. Alternatively, you can get help for a tool using the build-in

Python help() function. In the PythonWin interactive window, import arcpy, then enter help(arcpy.toolname_toolbox) and press enter to get help for that tool. For example, *help(arcpy.Buffer_analysis)*.

Note: all intermediate outputs created by the tools below are written to the in_memory workspace. This location is automatically cleared after processing, so this is a good place to write any outputs that you do not wish to keep after execution.

- The first criterion relates to the type of vegetation. Use the Select_analysis tool to select all vegetation patches of the correct type
  - The required tool parameters are as follows: input, output, SQL expression
  - The key to this operation is specifying the correct SQL expression (third parameter): *""" "VEG_TYPE" In ('Coastal Sage-Chaparral Scrub', 'Diegan Coastal Sage Scrub', 'Maritime Succulent Scrub')"""*

```
# Select vegetation areas with the correct type
select = arcpy.Select_analysis(veg, "in_memory/select", """ "VEG_TYPE" In ('Coastal Sage-Chaparral
```

- The second criterion relates to distance from major roads. Use the Buffer_analysis tool to generate areas that are 1000 feet around major roads.
  - The required tool parameters are as follows: input, output, buffer distance
  - Also, set the optional dissolve parameter (sixth parameter) to ALL, so that overlapping buffers are dissolved

```
# Buffer roads to exclude these areas
buffer = arcpy.Buffer_analysis(roads, "in_memory/buffer", "1000 Feet", "", "", "ALL")
```

- Since suitable habitat cannot be within 1, 000 feet of a major road, use the Erase_analysis tool to remove those buffer areas from the potential vegetation patches (output of the Select tool).
  - The required tool parameters are as follows: input, erasing layer, output

```
# Remove the buffered roads from the the suitable vegetation areas
erase = arcpy.Erase_analysis(select, buffer, "in_memory/erase")
```

- The third and fourth criteria relate to land slope and elevation. Use the Intersect_analysis tool to overlay the climate, slope, elevation, and the output of the Erase tool to find areas where all of these first four criteria are met (correct vegetation type, not within 1000 feet of a major road, less than 40% slope, and less than 250 feet in elevation).
  - The first parameter of the Intersect_analysis tool is a list of the feature classes to overlay. This parameter can be specified as a Python list. The second parameter is the output.

```
# Find all the areas that match the climate, slope, elevation, and vegetation criteria by performi
intersect = arcpy.Intersect_analysis([climate, slope, elev, erase], "in_memory/intersect")
```

- The fifth criterion is that the habitat must be a contiguous area larger than 100 acres. Use the Dissolve_management tool to remove the boundaries between contiguous patches where the vegetation, distance from roads, slope, and elevation criteria are met.
    - The required tool parameters are as follows: input, output, statistic field (only features with the same attribute in this field can be combined)
    - Also, set the optional multi-part/singlepart parameter (fifth parameter) to SINGLE_PART so that each contiguous area is maintained as a single record in the output
    - The key to this operation is setting the “Climate” field as the Dissolve Field

```
# Dissolve the areas
dissolve = arcpy.Dissolve_management(intersect, "in_memory/dissolve", "Climate", "", "SINGLE_PART")
```

- To calculate the area of these patches, we first need a new field to calculate the values into. Use the AddField_management tool to add a new double field “Area” to the Dissolve output
    - The required tool parameters are as follows: input, new field name, type of field
- Use the CalculateField_management tool to calculate the area of each habitat patch in acres
    - The required tool parameters are as follows: input, field to calculate, expression
    - The key to this operation is using a geometric calculation with conversion unit*: !Shape.Area@acres!*
    - Also, since a Python expression is being used, the fourth parameter must be set to PYTHON, as opposed to VB (the ! symbols around field names are required for Python expressions; if this was a VB expression, the field name would be wrapped in square brackets [])

```
# Add an attribute field and calculate the area of the contiguous habitat patches in acres
arcpy.AddField_management(dissolve, "Area", "Double")
arcpy.CalculateField_management(dissolve, "Area", "!Shape.Area@acres!", "PYTHON")
```

- Now we just need to extract the habitat patches that meet the remaining criteria (larger than 100 acres and in the Coastal climate zone). Use the Select_analysis tool again, this time to select all areas from the Dissolve output that are larger than 100 acres and are in the Coastal climate zone
    - The key to this operation is specifying the correct SQL query: *""" "Area" >100 AND "Climate" = 'Coastal' """*

```
# Find patches that are larger than 100 acres in the Coastal zone
habitat = arcpy.Select_analysis(dissolve, "Habitat", """ "Area" >100 AND "Climate" = 'Coastal' """)
```

- Finally, press the Check button to check the script for syntax errors like indentation mistakes or forgotten colons

- And press the Run button and run the full script

Results

Having created a Python script for performing this geoprocessing workflow, you can easily tweak settings used in the analysis, or apply the workflow to another study area to quickly get results without having to re-do any of the individual steps of the analysis.

The successful habitat analysis should find the following eight areas that meet all of our defined criteria.