

## Geoprocessing mit Python

Ab ArcGIS 9 kann das Geoprocessing von ArcGIS Desktop mittels Scriptsprachen ausgeführt werden.

Das Geoprocessing umfasst die Funktionen der ArcToolbox. Als groben Vergleich kann man sagen, dass damit die Aufgaben übernommen werden können, die bisher mit AML als Scriptsprache in ArcInfo Workstation erledigt wurden. Allerdings liegt der Focus deutlich auf Geoprocessing im Gegensatz zu Benutzerschnittstellen und GUIs.

Prozesse, die routinemäßig wiederholt werden oder die lange laufen, können als Batch automatisiert werden und in lastarme Zeiten verlegt werden.

Der Geoprocessor implementiert die COM I Dispatch Schnittstelle und das bedeutet, dass Sie als Anwender die Wahl unter mehreren Scriptsprachen haben, die COM unterstützen. Die bekanntesten sind VBScript, JScript und Python. Viele Entwickler bei ESRI bevorzugen Python aus folgenden Gründen: Python ist plattformunabhängig (Windows, UNIX, Linux), leicht zu erlernen, unterstützt objektorientierte Programmierung, der Code ist gut lesbar, Python lässt sich mit Java, C++ und Fortran integrieren. Python ist frei als Download verfügbar ([www.python.org](http://www.python.org) und [PythonWin.starship.python.net/crew/mhammond](http://PythonWin.starship.python.net/crew/mhammond)).

### Erster Zugang zu Python

Technische Voraussetzungen / Installation: Python gibt es kostenlos als Download unter [www.python.org](http://www.python.org) und dort gibt es auch einen Verweis auf PythonWin für Windows. Mit der Installation von ArcGIS 9 wird auch Python 2.1 angeboten. Es ist jedoch zu empfehlen, eine aktuellere Version, z.Zt. 2.3, zu installieren.

Erzeugen Sie sich mit ArcToolbox aus ArcGIS 9 ein einfaches Modell, das ein Shapefile projiziert. Exportieren Sie dieses Modell in ein Python-Script.

Das folgende Script wurde, wie im Kommentarkopf vermerkt ist, auf diese Weise von ArcGIS/ModelBuilder generiert. Ein paar ergänzte Kommentarmarken sollen helfen, Details zu erläutern.

#1 Hier wird nur win32com.client wirklich benötigt. sys, string und os gehören zu den am häufigsten gebrauchten Modulen und werden standardmäßig in die generierten Scripts eingebaut.

#2 Wird immer für Geoprocessing benötigt.

#3 Der Zeilenumbruch der folgenden Zeile entstand durch den Abdruck. In den Scripts entspricht in der Regel eine logische Zeile einer physikalischen Zeile. Auch diese Zeile wurde generiert. Für AML-Programmierer: Ihre Funktion entspricht in etwa `&amp;amp;path`. In diesem Kontext ist sie nicht unbedingt notwendig. Das GP-Tool `project_management` wird auch ohne diesen Eintrag gefunden.

#4 Python unterscheidet Groß- und Kleinschreibung. Strings können wahlweise durch „oder“ begrenzt werden.

#5 Es folgt der Aufruf eines Geoprocessing Tools – im Original in einer sehr langen Zeile.

```
# -----
# gp_prj_shapefile.py
# Created on: Fr Feb 06 2004 11:30:11
# (generated by ArcGIS/ModelBuilder)
# -----

# Import system modules
import sys, string, os, win32com.client #1
# Create the Geoprocessor object
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1") #2

# Load required toolboxes... #3
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/Data Management Tools.tbx")

# Local variables... #4
Coast_geo = "C:/testdatasets/Coast_geo"
CoastLine_project_shp = "C:/testdatasets/Coast_project.shp"

# Process: Project... #5
gp.Project_management(Coast_geo, Coast_project_shp,
"PROJCS['British_National_Grid',GEOGCS['GCS_OSGB_1936',DATUM['D_OSGB_1936',
SPHEROID['Airy_1830',6377563.396,299.3249646]],PRIMEM['Greenwich',0.0],UNIT
['Degree',0.0174532925199433]],PROJECTION['Transverse_Mercator'],PARAMETER[
'False_Easting',400000.0],PARAMETER['False_Northing',-100000.0],
PARAMETER['Central_Meridian',-2.0],PARAMETER['Scale_Factor',0.999601272],
PARAMETER['Latitude_Of_Origin',49.0],UNIT['Meter',1.0]],"OSGB_1936_To_WGS_1984_1")
```

Die aus Modellen exportierten Scripts können in zwei Arten von Umgebungen weiterverwendet werden. Entweder als neues ArcToolbox-Script in der bisherigen Applikation (ArcMap, ArcCatalog, etc.) oder außerhalb der Applikation unter dem Python-Interpreter oder unter PythonWin. Im letzteren Fall ist die exportierte Version selten eigenständig lauffähig, eignet sich aber als Vorlage. Dieses einfache Beispiel soll nun umgebaut werden – z.B. im PythonWin-Editor, und dabei lernen Sie ein paar andere Syntax- und Konstruktionselemente von Python kennen. Die erste Variante tut dasselbe und ist im Code fast maximal verkürzt:

```
# gp_prj_shapefile_0.py
'''Project Coast_geo.shp to Coast_project.shp'''

import win32com.client; gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")
gp.OverwriteOutput = 1
gp.Project_management("C:/testdatasets/Coast_geo.shp",\
"C:/testdatasets/Coast_project.shp",\
"27700","OSGB_1936_To_WGS_1984_1")
```

### Die Änderungen sind im Einzelnen:

- Ein Doc-String wurde eingefügt. Er ist durch dreifache Quote-Zeichen begrenzt. Der Doc-String ist ein Python-Attribut dieses Scripts namens `__doc__`, das dieses Script beschreibt. Es muss sich dabei um das erste Statement handeln.
- Ausgabedaten werden automatisch überschrieben.
- Zwei logische Zeilen in einer physikalischen Zeile – durch ; getrennt.
- Lange logische Zeilen auf drei physikalische Zeilen verteilt. Trennzeichen am Zeilenende vor einer Fortsetzungszeile ist \. Ein Kommentar darf dann in der Zeile nicht mehr folgen.

- Die lange textuelle Beschreibung des Koordinatensystems wurde durch die entsprechende EPSG-Code-Nummer 27700 ersetzt. Achtung: Zwischen ArcGIS 8 und ArcGIS 9 gab es Änderungen bei einigen EPSG-Codenummern. 27700 ist nicht betroffen.
- Auf die inneren Quote-Zeichen kann bei einem einzelnen Transformationsnamen verzichtet werden. Transformationen können nicht durch die EPSG-Codenummer ersetzt werden.

Der kleine Unterschied beim Ablauf besteht darin, dass die zweite Variante jeweils die Ausgabedaten überschreibt. Diese vereinfachte Variante sollte ein paar neue Sprachelemente von Python vorstellen, ist aber noch nicht so flexibel, wie es die Praxis häufig verlangt. Die dafür typischen Änderungen am Script folgen nun:

- Übergabe von Argumenten
- Ausnahmebehandlung
- Umgang mit Objekten im Filesystem
- Speziell hier wird der Gebrauch einer hart kodierten Code-Nummer durch den Namen des Scripts deutlich gemacht.

```
# gp_prj_WGS84_to_27700.py
'''Project from WGS84 to British National Grid.'''

print __doc__ #0
import sys, os #1

try: #2
    inshp=sys.argv[1] #3
    outshp=sys.argv[2]
except: #4
    print "Usage: gp_prj_WGS84_to_27700 <inshp> <outshp>" #5
    sys.exit(0) #6

# find new output name
num=0 #7
file=os.path.basename(outshp) #8
filename = os.path.splitext(file)[0]
pathname=os.path.dirname(outshp)
while os.path.exists(outshp): #9
    if num >= 10: #10
        print "giving up."
        sys.exit(0)
    num=num+1
    outshp = os.path.join(pathname,filename+"_"+str(num)+".shp") #11

import win32com.client #12
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1") #13
gp.SetProduct("ArcView") #13
gp.OverwriteOutput = 1

if not gp.exists(inshp): #14
    print "Input",inshp,"does not exist." #15
    sys.exit(0)

try:
    gp.Project_management(inshp,outshp,"27700", "OSGB_1936_To_WGS_1984_1") #16
    print outshp,"successfully created."
except:
    print "Error in", os.path.splitext(os.path.basename(sys.argv[0]))[0] #17
    print gp.GetMessages(0) #18
```

- #0 Das Script meldet sich mit seinem Doc-String
- #1 Die Module sys und os werden diesmal verwendet.
- #2 try-Block. : schließt immer die Zeile ab, die einen Block einleitet. Alle Anweisungen im Block sind anschließend einheitlich eingerückt. Das Einrücken von Zeilen gehört bei Python zur Syntax! Zur Überprüfung hat PythonWin z.B. einen „TabNanny“-Button.
- #3 Übernahme der Argumente aus dem Aufruf.
- #4 Im Fall eines Fehlers im try-Block springt Python in einen except-Block. Für Ausnahmebehandlungen gibt es viele spezielle Optionen. Dies ist die einfachste Standardvariante für beliebige Ausnahmen.
- #5 Ausgabe auf Standard-Out.
- #6 Beendet das Skript
- #8 Zerlegung eines vollständigen Dateinamens. Die Hauptzerlegung in Dateiname und Verzeichnisname erfolgt durch die Methoden basename und dirname. Splitttext zerlegt den Dateinamen in ein Tupel. Tupel ist ein eingebauter Datentyp. Der erste Teil steht vor dem ersten Punkt, der zweite Teil ab Punkt.
- #9 Beginn eines while Block.
- #10 Geschachtelter if Block. Ein Block ist beendet, wenn die Einrücktiefe wieder zurückgeht. Begrenzung und Absicherung gegen unendliche Schleife. Bei num=num+1 geht es mit dem „Body“ des while Blocks weiter.
- #11 Bildung eines neuen Namens. Die Methode os.path.join ist analog zu [joinfile] von AML und plattformunabhängig.
- #12 Beginn des GP
- #13 Bezug auf eine bestimmte Lizenz. Generell optional, muss aber bei Single Use Lizenz außerhalb der ArcGIS Umgebung genau passen, sonst „verabschiedet“ sich Python.
- #14 Prüft im Gegensatz zu #9 die Existenz einer FeatureClass (hier erfüllt durch Shapefile)
- #15 Die einzelnen Bestandteile der Ausgabe werden automatisch durch Leerzeichen getrennt.
- #16 Das eigentliche Projektionskommando.
- #17 Dateiname des Scripts. Es werden Funktionen des os-Moduls verwendet.
- #18 Ausgabe der Geoprocessing Messages

Das war nun ein GP-Tool, aber schon viel Python. Mit der Standardinstallation von Python haben Sie schon viele Module mit nützlichen Funktionen, und weitere wie z.B. Numeric gibt es im Internet.



Dr. Werner Flacke  
ESRI Geoinformatik GmbH  
Kranzberg  
W.Flacke@ESRI-Germany.de