使用Arcpy进行数据批处理

关键词：Arcpy，Python，遥感

时空大数据使我们面临前所未有的机遇和挑战，尤其在地学、遥感或空间技术等专业领域，无疑是一个全新的时代。

伴随着时空大数据的到来，海量数据的处理是一个所有科研工作者都无法忽视的重要问题。传统的数据（主要指空间数据）处理工具已无法满足大数据处理的要求，而且笨拙的传统工具数据处理方式无疑是科研道路上的绊脚石，使我们面对大数据处理需求时手忙脚乱。因此，数据批处理的方式很大程度上解决了这一问题，解放大量劳动力。

对于空间数据来说，常用的传统数据处理工具或软件包括ArcGIS、ENVI等，而且现已拥有对应的批处理平台或工具，如Arcpy、ENVI\_IDL。此外许多第三方工具包如GDAL，也很好的支持多种开发语言和环境，以便于编程使用。在此，以Arcpy为例，在Python语言开发环境下，通过实际的编程应用，简单介绍空间数据批处理的实现方法。

1. Python基础

" Life is short, you need Python——Bruce Eckel"，人生苦短，我用Python。

Python是一种解释型、面向对象、动态数据类型的高级程序设计语言，由Guido van Rossum于1989年底发明，第一个公开发行版发行于1991年。

Python的设计哲学：

* 优雅
* 明确
* 简单

对于大多数程序语言，第一个入门编程代码便是"Hello World！"，以下代码为使用Python输出"Hello World！"：

实例(Python 2.0+)

print "Hello World!"

实例(Python 3.0+)

print("Hello World!")

OK，YOU GOT IT!

2. Arcpy介绍

2.1 什么是Arcpy？

在这里简单介绍一下Arcpy，详细说明参见ArcGIS相关帮助文档。ArcGIS采用ArcPy为用户提供了使用Python语言操作所有地理处理工具（包括ArcGIS扩展模块）的方法，并提供了多种有用的函数和类。目的是为以实用高效的方式通过Python执行数据处理分析、数据转换、数据管理和地图自动化创建基础。因此，使用Python和ArcPy，可以实现地理或遥感大数据的批量处理。

2.2 Arcpy的安装和使用

Arcpy无法单独安装使用，其底层实现是完全依托在Arcgiscripting上的，并且由于历史的原因，所有的Arcpy模块都会依赖Geoprocessing模块中的部分函数来实现对Arcgisscripting的访问。

ArcGIS Desktop安装后，在安装目录下会出现Arcpy文件夹，其中包含有Python函数、类和模块。用户可以使用Python语言调用ArcObject的相关类。

Arcpy提供的功能：

* 访问所有地理处理工具
* 数据转换和数据处理
* 数据分析
* 自动化制图等

使用Python和Arcpy，可以开发出大量用于空间数据批处理的实用程序。

3. 数据批处理实现案例

当我们下载了大量遥感影像数据（如风云卫星数据产品，为国产数据打call），一般不可能直接就可以使用，需要进行一定的预数据处理，才能达到我们的使用标准。在这个过程中，一般情况下可能涉及到的数据处理有“定义投影”、“投影变换”、“地图配置”、“数据裁剪”、“重采样”等等。下面具体介绍程序编写实现方法。

以栅格数据裁剪为例。

首先，引入Arcpy包，

**import** arcpy

调用Arcpy的栅格裁剪函数，

# function: RasterClip

**def** RasterClip**(**datadir**,** in\_raster**,** extent\_feature**,** out\_raster**):**

'''

:param datadir: data direction

:param in\_raster: raster data to be clip

:param extent\_feature: cutting boundary

:param file\_name: output

'''

**try:**

arcpy**.**Clip\_management**(**in\_raster**,** "#"**,** out\_raster**,** extent\_feature**,** "#"**,** "ClippingGeometry"**)**

**print(**arcpy**.**GetMessages**(**0**))**

**print** "Clip completed!\nSave as '%s'\n" **%** **(**clipForld **+** os**.**sep **+** tifNameList**[**j**]** **+** "\_clip.tif"**)**

**except:**

**print** arcpy**.**GetMessages**()**

**print** "Process failed."

以上函数可以进行一幅影像的裁剪，接下来便可以实现批处理功能，获取待裁剪栅格数据文件路径及文件名函数如下，

# function: getFileList

**def** getFileList**(**datadir**,** ftype**):**

'''

:param dataDir: forld path

:param ftype: file suffix(.\*)

:return: full path, file name(without suffix)

'''

fileFullPathList **=** **[]**

fileNameList **=** **[]**

filenameList **=** os**.**listdir**(**datadir**)**

**for** fn **in** filenameList**:**

file\_name**,** file\_ext **=** os**.**path**.**splitext**(**fn**)**

**if** file\_ext **==** ftype**:**

fileFullPathList**.**append**(**datadir **+** os**.**sep **+** fn**)**

fileNameList**.**append**(**file\_name**)**

**return** fileFullPathList**,** fileNameList

最后，遍历所有要裁剪的数据，实现数据批量裁剪，

# Define data direction and vector data of boundary

datadir **=** r'D:\data...'

extentFeature **=** r'D:\...\\*.shp'

# Get data

fType **=** '.tif'

**print** "Get %s file......" **%** fType

fileList **=** getFileList**(**datadir**,** fType**)**

# Create forld

clipForld **=** datadir **+** os**.**sep **+** "clip"

**if** **not** os**.**path**.**isdir**(**clipForld**):**

os**.**makedirs**(**clipForld**)**

**for** i **in** range**(**len**(**fileList**[**0**])):**

out\_raster **=** datadir **+** os**.**sep **+** file\_name **+**"\_clip.tif"

RasterClip**(**clipForld**,** fileList**[**0**][**i**],** extentFeature**,** out\_raster**)**

完成数据批处理！