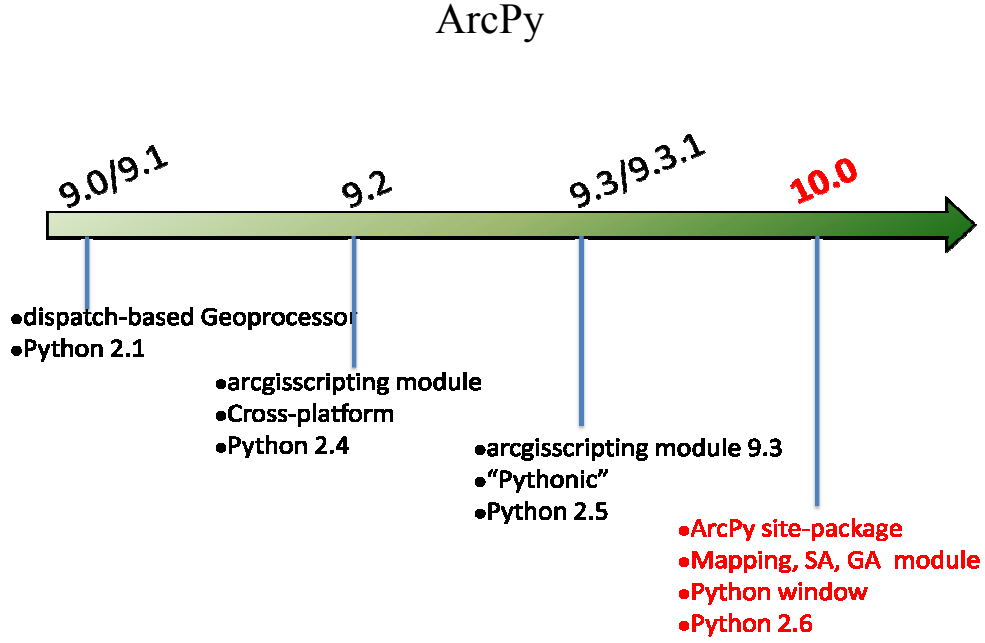
ArcGIS 10 中的 Python

Geoprocessing 作为 GIS 的三个基本视点(以及 GeoVisualization 和 GeoData) 之一，从来就不乏众人的关注；而作为 Geoprocessing 的主要实现途径之一（ 以及 Tools 和 Modeling)，Python 语言自从 9.0 版本出现在 ArcGIS 中以后，就体现出了强大的生命力。Python 作为一种灵活而完善的脚本语言，具有诸多优势：

- 易于上手，且功能强大；
- 高度的可伸缩性。既适用于大型工程，也适用于小型脚本；
- 跨平台；
- 可嵌入；
- 稳定而成熟；
- 用户群体众多。

在 ArcGIS 10 中，众多新特性再次使 Python 的活力有了质的飞跃。本文将带您领略其中的魅力。



Python 在 ArcGIS 的发展历史中与时俱进。9.2 版本中引入了可跨平台的 arcgisscripting module；9.3 版本中亲密接触了 Python 语言的特性，比如返回 List 类型的结果；ArcGIS 10 中用 ArcPy site-package 完全替代了 arcgisscripting module，在完全向下兼容的基础上，ArcPy 旨在为用户提供一种快速而高效的途径来利用 Python 完成地理数据分析，数据转换，数据管理以及地图自动化工作。

通过 ArcPy，你可以：

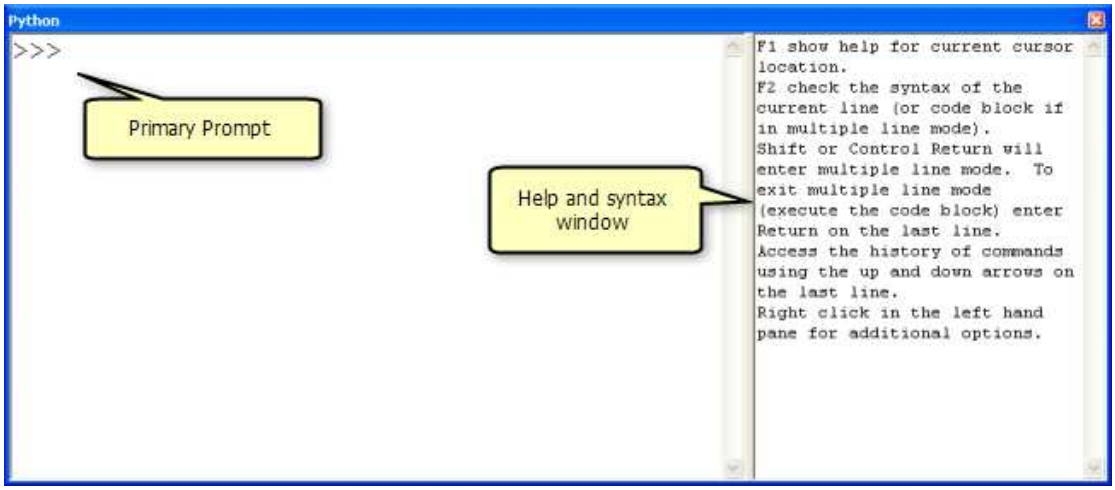
- 访问超过 800+的 Geoprocessing Tools；
- 内嵌的详细使用文档；
- 在你钟爱的 Python IDE 中提供更好的智能感知；
- cursors，geometry，原生 classes 等的改进；
- NumPy 的利用：NumPyArrayToRaster，RasterToNumPyArray；
- ……

例 1：arcgisscripting 与 arcpy

```
# 9.3
import arcgisscripting
gp = arcgisscripting.create(9.3)
array1 = gp.createobject("array")

# 10.0
import arcpy
array2 = arcpy.Array()
```

Python Window



ArcGIS 10 中引入了全新的 Python Window 来增强内嵌的 Python 体验，它作为 Python 的解释器，替代了先前版本中的 command line window（但你依然可以用 ArcPy 提供的 Command 函数来执行 command line 语法）。在 Python Window 中，键入的内容可以是单独的 Python 命令，也可以是完整而复杂的代码内容。通过 Python Window，你可以：

- 访问 ArcPy 提供的功能，包括 tools 和 environments；
- 使用其他的 Python 内置功能；
- 感受良好的编程体验。Python Window 提供比其他同类 IDE（比如 PythonWin 的 Interactive Window）更好的智能感知和代码自动完成功能；
- 快速试验你的想法；
- 感受和学习 Python；
- 简单地执行 Geoprocessing tools；
- 快速构建 Python 中的工作流；
- ……

需要注意的是，Python Window 是 Python 的一个解释器。尽管 Python Window 拥有诸多优点，但它并不是为了取代其他的 Python IDE，当你需要完成大量代码

工作的时候，还是需要用到它们。

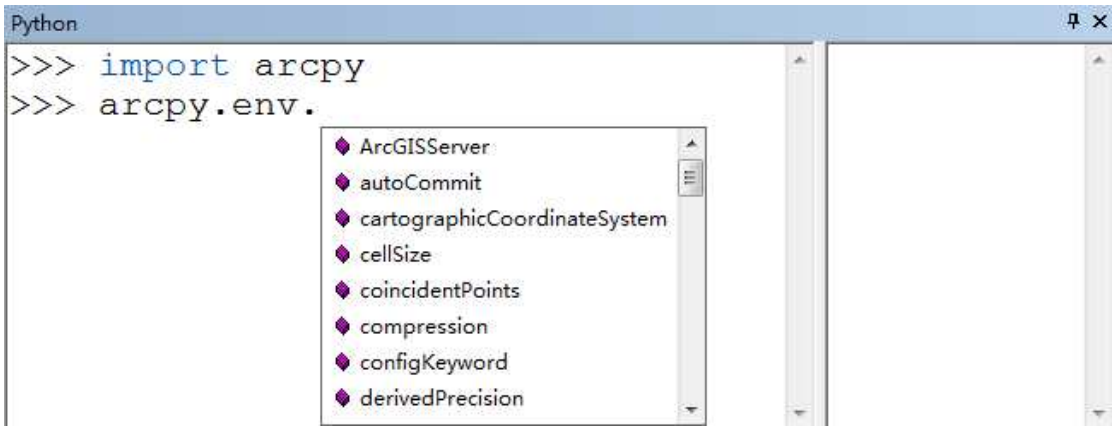
下面就通过几个简单的例子，体验一下 ArcPy 和 Python Window 结合给您工作带来的便利。

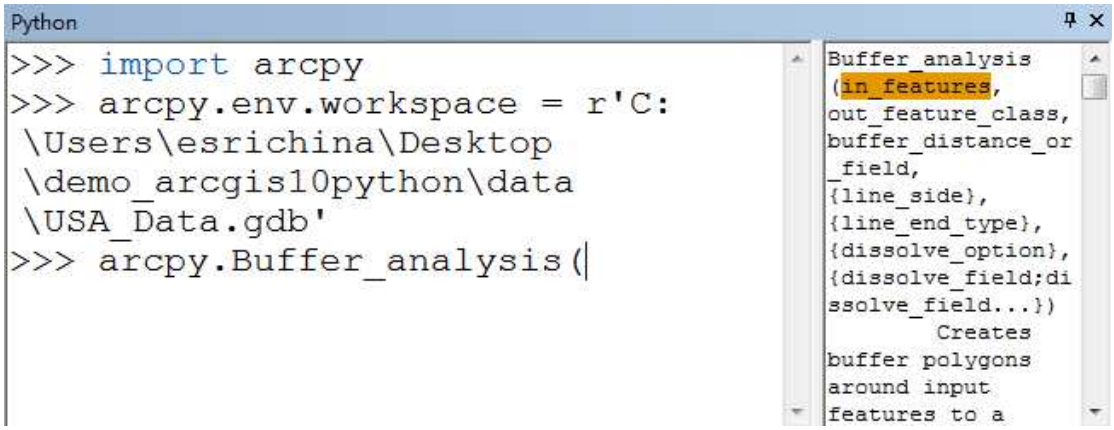
例 2：调用 Geoprocessing tools

```
    >>> import arcpy
    >>> arcpy.env.workspace =
r'C:\Users\esrichina\Desktop\demo_arcgis10python\data\USA_Data.gdb'
    >>> arcpy.Buffer_analysis("Cities","Cities_buffer","5 Kilometer")
    <Result
'C:\\Users\\esrichina\\Desktop\\demo_arcgis10python\\data\\USA_Data.gdb\\Cities_buffer'>
    >>>
```

在 Python Window 中，“>>>”提示符后面敲入上述代码，即可调用 Buffer 工具，为指定点图层生成 5 千米的缓冲区。

Tip1：ArcPy 的智能感知和完善的文档





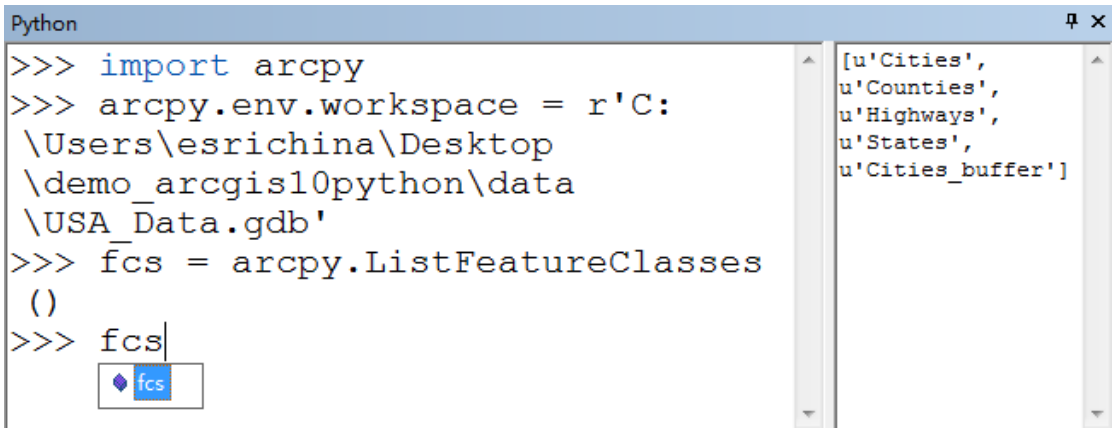
Tip2：快速的路径输入

在 ArcGIS 10 的 Python Window 中，路径输入变得十分容易。比如：

“ r'C:\Users\esrichina\Desktop\demo_arcgis10python\data\USA_Data.gdb'”

这个路径，将 Catalog 窗体中的对应 FeatureClass 拖入 Python Window 中即可生成。

Tip3：字符串和数字内容的变量屏幕会实时打印



如上图，fcs 是 Python 中 List 类型的变量，键入其名称时，右侧帮助窗口会显示出其内容（u 代表 unicode）。

例 3：Automation

```
    >>> import arcpy
    >>> arcpy.env.workspace =
r'C:\Users\esrichina\Desktop\demo_arcgis10python\data\US
A_Data.gdb'
    >>> for feature in
arcpy.SearchCursor("States","STATE_NAME = 'California'"):
    ...
arcpy.Clip_analysis("States",feature.Shape,"California")
    ...
    >>>
```

本例中首先利用 SearchCursor 方法从 States 图层中筛选出名为 California 的 Feature，然后将其图形作为输入，对 States 图层进行 Clip 操作，从而生成新的 California 多边形图层。

Tip4：灵活的操作

注意，如果双击打开 ArcToolbox 中的 Clip 工具，第二个参数输入一般是图层，而利用 Python 脚本，我们能够灵活地取到图层中要素的图形作为 Clip 参数进行操作。

```
    >>> fcs = arcpy.ListFeatureClasses()
    >>> fcs.remove("California")
    >>> for fc in fcs:
    ...
arcpy.Clip_analysis(fc,"California",fc+"_clip")
    ...
    >>>
```

接下来，在一个 for 循环里，用新生成的 California 图层去裁剪工作空间中的其它 FeatureClass，将需要重复的手动操作自动化。

Tip5：可在 Python Window 中调用 Python 的功能

如上，fcs 是 Python 中 List 类型变量，用 remove 函数将 California 图层移除，达到只裁剪其他图层的目的。

Tip6：Background Geoprocessing

ArcGIS 10 中引入了 Background Geoprocessing 的概念，允许你在运行 Geoprocessing 工具的时候继续对 ArcGIS 进行操作，而使用多核 CPU 的“另一个进程”（实际上是由 ArcGIS 管理的 3 个进程）来完成正在进行的 Geoprocessing 工作。本例中档进行图层裁剪时你依然可以操作 ArcMap，Background Geoprocessing 结束后会弹出气泡窗口的通知。

当然，本例中也可使用嵌套的 for 循环将两个步骤合并起来。

ArcPy 中的 Modules

除了本身包含的诸多类与函数，ArcPy 中还新增了 3 个模块用来细化 Python 脚本的的功能：

arcpy.mapping：Mapping module，可直接操作地图文档（.mxd）和图层（.lyr）之中的内容，用来自动化地图文档的输出打印工作，创建 PDF 地图集等；

arcpy.sa：Spatial Analyst module，用来进行空间分析工作。比如利用 ArcGIS 10 中新增的 ArcPy.Raster 类进行栅格图像的代数运算等；

arcpy.ga：Geostatistical Analyst module，用来进行地理统计分析。

这些 modules 将为您的工作带来前所未有的便利。

下面介绍两个 Mapping module 的例子。

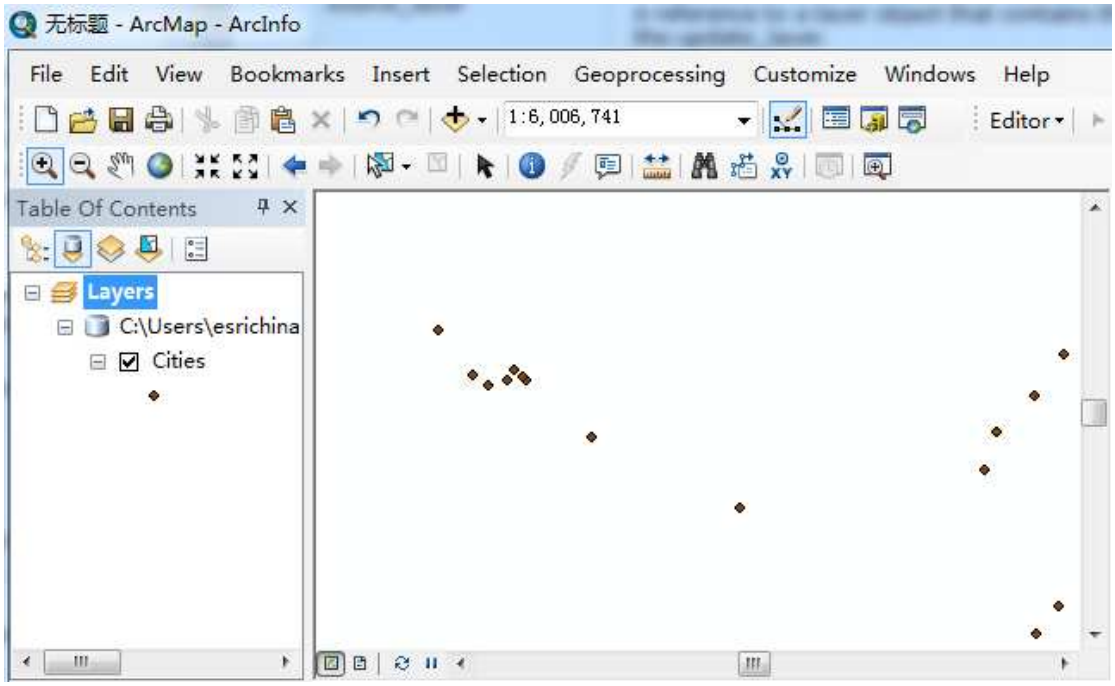
例 4：利用 Mapping Module 的功能修改图层符号

```
>>> import arcpy, arcpy.mapping
>>> mxd = arcpy.mapping.MapDocument("current")
>>> df = arcpy.mapping.ListDataFrames(mxd)[0]
>>> updatelyr = arcpy.mapping.ListLayers(mxd)[0]
>>> sourcelyr = arcpy.mapping.Layer(r'C:\heli.lyr')
>>>
arcpy.mapping.UpdateLayer(df,updatelyr,sourcelyr,True)
```

```
>>>
```

利用 mapping module 中的 MapDocument 类可以取得对指定 mxd 文件的引用，这里使用“current”关键字，表示取得对当前 ArcMap 中已经打开的地图文档的引用；ListDataFrames 函数会以 List 类型返回指定 mxd 中 DataFrame 的集合，由于此处 mxd 中只有一个 DataFrame ,所以用索引可直接取到唯一的 DataFrame；updatelyr 是对当前地图文档中第一个图层的引用，此处是一点图层，sourcelyr 中保存着需要设置的图层符号；UpdateLayer 函数对图层 lyr 属性做出修改，True 参数表示只修改图层的符号。

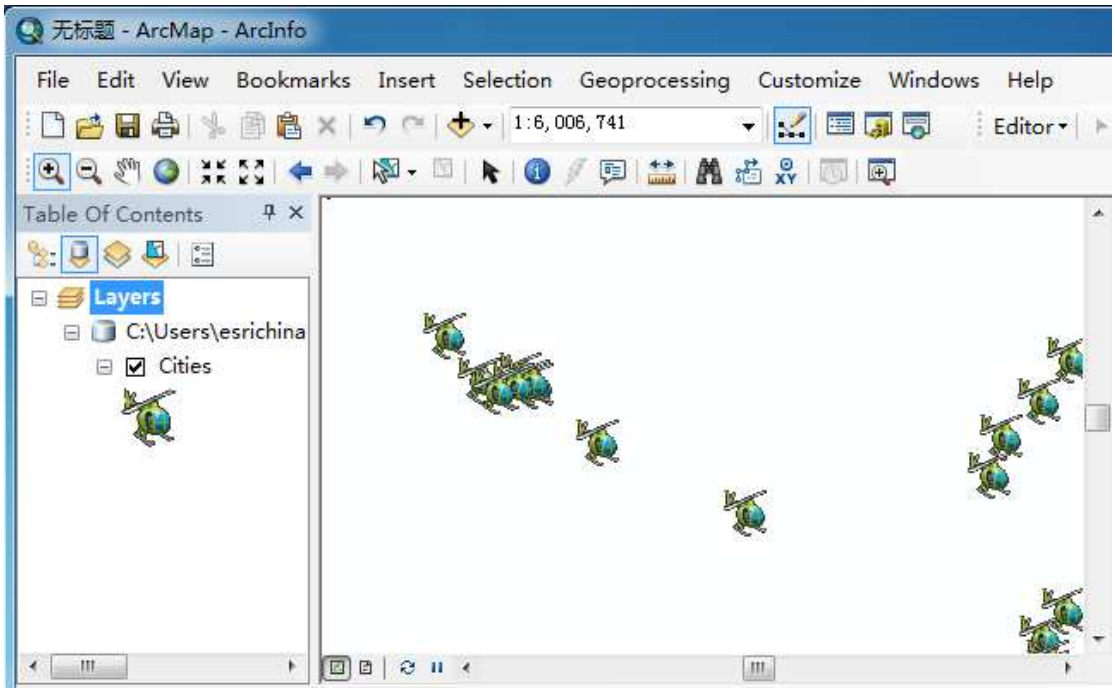
输入前：



可是执行上述代码后，发现图层符号并没有变化。实际上该图层符号已经被改变了，为了让变化显示出来，还需要执行以下语句：

```
>>> arcpy.RefreshContents()
>>> arcpy.RefreshGraphics()
>>>
```

执行后：



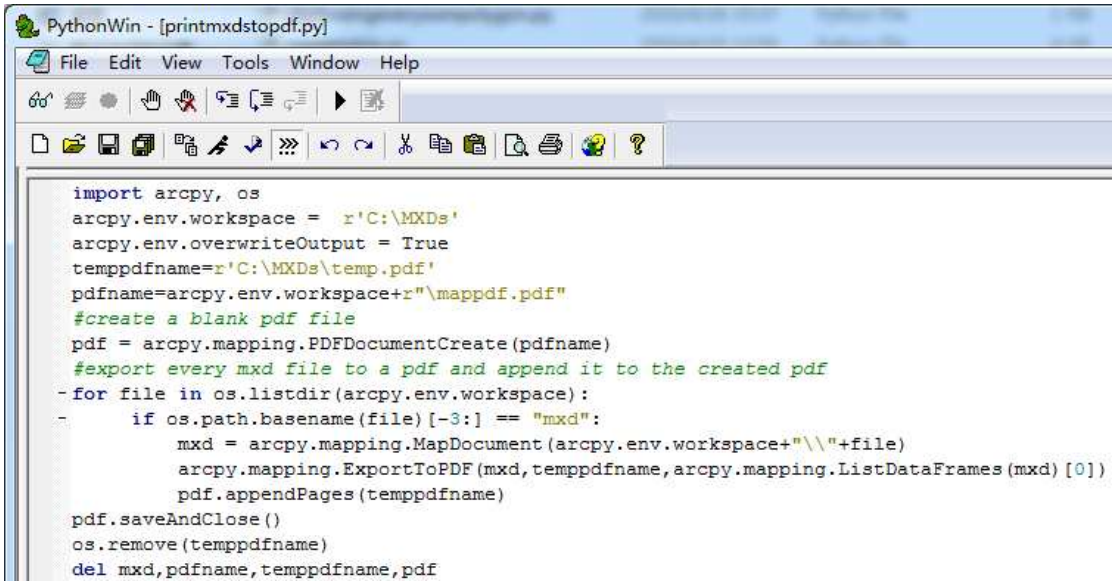
Tip7：内容刷新

arcpy.RefreshContents() 用来刷新 ArcMap 中 TOC 的内容；arcpy.RefreshGraphics()用来刷新 MapView 或 LayoutView 中的内容。

Tip8：保存地图文档

可以用 MapDocument 类的 save/saveACopy 方法来保存对 mxd 所做的修改。

例 5：地图文档打印




```
import arcpy, os
arcpy.env.workspace =  r'C:\MXDs'
arcpy.env.overwriteOutput = True
temppdfname=r'C:\MXDs\temp.pdf'
pdfname=arcpy.env.workspace+r"\mappdf.pdf"
#create a blank pdf file
pdf = arcpy.mapping.PDFDocumentCreate(pdfname)
#export every mxd file to a pdf and append it to the created pdf
for file in os.listdir(arcpy.env.workspace):
    if os.path.basename(file)[-3:] == "mxd":
        mxd = arcpy.mapping.MapDocument(arcpy.env.workspace+"\\"+file)
        arcpy.mapping.ExportToPDF(mxd,temppdfname,arcpy.mapping.ListDataFrames(mxd)[0])
        pdf.appendPages(temppdfname)
pdf.saveAndClose()
os.remove(temppdfname)
del mxd,pdfname,temppdfname,pdf
```

本例中 C:\MXDs 文件夹中存放有若干 mxd 文件，上述 Python 代码可直接将文件夹中包含的所有地图文档打印成一个 PDF 文件，每个 mxd 一页。

Tip8：Python Window 的 Save 和 Load

图中代码保存成了.py 文件，在 Python Window 中可通过右键，Load 将.py 文件中的代码直接载入，然后进行调试或执行；也可将 Python Window 中的现有工作 save 成.py 文件，保存以后使用。

总结：ArcGIS 10 的 Python 脚本中引入了新的 Geoprocessor——ArcPy，它对原有的 arcgisscripting 做了很多增强和改进，与 Python 的结合也更加紧密；新增的三个 Python Module 能够极大提高生产效率，充分发挥出脚本快速、自动化的特点。不论你对 Python 脚本熟悉与否，我们都有理由期待 ArcGIS 10 中带来的全新 Python 体验。