



第2章 ArcGIS的Python开发技术

主讲：张宝一

Email: zhangbaoyi.csu@qq.com



教学目标

- 熟悉Python开发ArcGIS的方式
- 熟悉ArcPy的类、函数及功能
- 掌握ArcPy管理地图和图层的方法
- 掌握ArcPy要素操作的方法
- 掌握ArcPy调用空间分析的方法

教学重点

- Arcpy要素操作
- Arcpy地理工具



教学内容

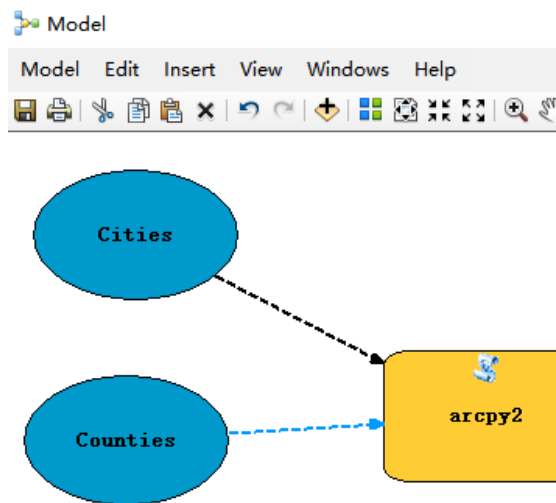
- 1. 面向ArcGIS的Python基础
- 2. 管理地理图文档和图层
- 3. 访问空间数据
- 4. 要素管理
- 5. 地理处理程序

1. 面向ArcGIS的Python基础

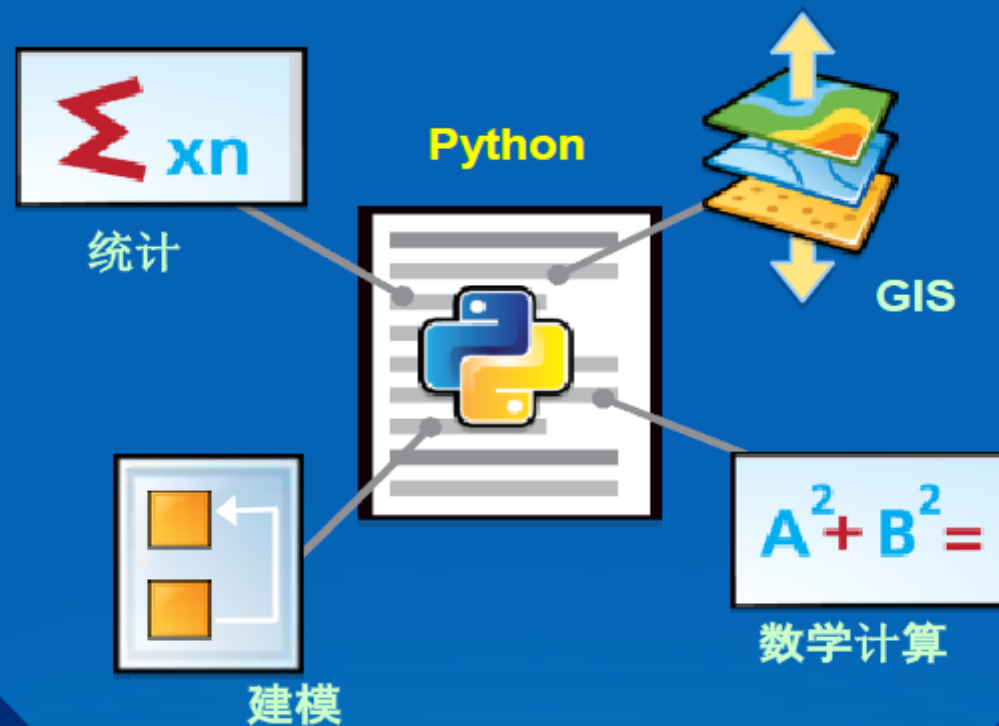


- 脚本语言(scripting languages)又被称为扩建的语言, 或者动态语言, 是一种编程语言, 用来控制软件应用程序, 脚本通常以文本(如ASCII)保存, 只在被调用时进行解释或编译。
- 脚本语言是为了缩短传统的编写-编译-链接-运行(edit-compile-link-run)过程而创建的计算机编程语言。
- 你知道哪些脚本语言?
 - (1)作为批处理语言或工作控制语言, 如Windows的批处理文件和Unix的shell脚本;
 - (2)作为通用的编程语言存在, 如Perl、Python、Ruby等。
 - (3)网页中的嵌入式脚本语言, 如HTML、JavaScript、ASP、JSP、PHP
 - (4)脚本语言在系统应用程序中嵌入使用, 如Office办公软件, 提供的宏和VBA

- PythonWin命令窗口执行
- 工具箱执行
- Python脚本程序
- 模型调用



功能集成



□ 导入模块

- `import arcpy`
- `from arcpy import mapping`
- `from arcpy import *`

□ 注释

- 单行注释 `#这是一个注释`
- 块注释

```
'''
```

```
文件: arcpy_demo.py
```

```
作者:
```

```
时间: 2022/10/26/
```

```
'''
```

□ 区分大小写

□ 变量名称规则同其它高级语言

□ 变量类型动态确定,无需先定义再使用

□ 命名规则

- 小驼峰命名法 `lowerCamel`
- 大驼峰命名法 `UpperCamel`
- 匈牙利命名法 `strName`
- 下划线命名法 `snake_case`

Python where we can, C++
where we must.

□常用的数据类型

- String、Number、Boolean、List、Dictionary、Object

□String常用操作

- 连接 (+)、赋值 (=)、关系运算 (==、>、<、>=、<=)
- 包含 (in)、取子串 (如[i:j]、[]、[i:]、[:j])
- String模块中的其它函数：split、find、rfind、join、lower、upper、lstrip、rstrip、replace
- 举例

```
Python
>>> from arcpy import *
>>> str = "Hello";
>>> str = str + " ArcPy";
>>> print str;
Hello ArcPy
>>> if str>"arcpy":
...     print "OK";
... else:
...     print "NO";
...
NO
>>> print str[2:4];
ll
>>> print "ArcPy" in str;
True
>>> |
```

□数值操作

- +、-、*、/、%、**、abs等

□List类型：类似于数组

- 例如：fcList=["road","jmd","sx"]
- 功能函数：sort、append、insert、remove、pop、reverse等

□Tuple类型：静态类型，定义后不能修改

- 例如：fcTuple=("Hydrants", "Water", "Valves")

□Dictionary类型：key-value结构

- fcDict={"Hydrants":0, "Water":1, "Valves":2}

□ 选择结构

- if / elif / else

□ 循环结构

- for <var> in <sequence>:
 <body>
- while <condition>:
 <body>
- 示例：求1-100的和

□ 函数定义

- `def 函数名([形参列表])`
 `body`

□ 类定义

- `class 类名:`
 `def __init__(self):`
 #构造函数
 方法/函数定义

□ 类示例

```
import math
class point:
    def __init__(self,x,y):
        #构造函数
        self.x = x;
        self.y = y;
    def distance(pt):
        #求两点距离
        return sqrt((self.x-pt.x)**2+ ((self.y-pt.y)**2;
```

- ArcPy modules : 模块为通常包含函数和类的 Python 文件
- ArcPy classes : 类为创建程序提供框架, 用来创建对象, 即通常所称的实例
- ArcPy functions : 函数是用于执行某项特定任务并能够纳入更大的程序的已定义功能, 包括工具函数和非工具函数
- ArcPy在线文档: <https://desktop.arcgis.com/zh-cn/arcmap/latest/analyze/arcpy/what-is-arcpy-.htm>



□(1)ArcPy函数

- ArcPy 中，所有地理处理工具均以函数形式提供，还提供多个函数以更好地支持使用 Python 的地理处理工作流。
- 函数可用于列出某些数据集、检索数据集的属性、在将表添加到地理数据库之前验证表名称或执行其他许多有用的地理处理任务。
- 函数只能从 ArcPy 获得，而不能作为 ArcGIS 应用程序中的工具
- 函数接受参数（可能需要也可能不需要）并返回某些结果,工具函数会始终返回 Result 对象，并提供地理处理消息支持。
- 函数列表：<https://desktop.arcgis.com/zh-cn/arcmap/latest/analyze/arcpy-functions/alphabetical-list-of-arcpy-functions.htm>

□(1)ArcPy函数

```
import arcpy
import os
arcpy.env.workspace = "c:/data"
out_workspace = "c:/data/results/"
clip_features = "c:/data/testarea/boundary.shp"
```

```
# Loop through a list of feature classes in the workspace
for fc in arcpy.ListFeatureClasses():
    output = os.path.join(out_workspace, fc)
    arcpy.Clip_analysis(fc, clip_features, output, 0.1)
```



□(2)ArcPy类

- ArcPy 类(如 **SpatialReference** 和 **Extent** 类)通常用作地理处理工具参数设置的快捷方式;否则, 这些参数会使用更加复杂的字符串。
- 类实例化之后, 便可使用其属性和方法。
- 类包含一个或多个方法, 其中构造函数用于初始化类的新实例。
- 示例:

```
import arcpy  
prjFile = "d:/myproject.prj"  
spatialRef = arcpy.SpatialReference(prjFile)  
print(spatialRef.name)  
print(spatialRef.type)
```

```
import arcpy  
pointA = arcpy.Point(2.0, 4.5)  
pointB = arcpy.Point(3.0, 7.0)
```

□(3)ArcPy模块

- 模块为通常包含函数和类的 Python 文件。
- ArcPy 由一系列模块支持，包括
 - 数据访问模块 (arcpy.da)
 - 制图模块 (arcpy.mapping)
 - ArcGIS Spatial Analyst extension 模块 (arcpy.sa)
 - ArcGIS Network Analyst extension 模块 (arcpy.na)

□(4)ArcPy环境设置

- 地理处理环境设置可视为影响工具执行结果的附加参数。这些参数包括感兴趣区域、输出数据集的坐标系以及新栅格数据集的像元大小等
- 环境设置通过env类获得或设置。
- 示例：
 - >>> arcpy.env.workspace = "c:/data/Portland.gdb"
 - >>> arcpy.Buffer_analysis("streets", "streetBuf", "500 METERS")

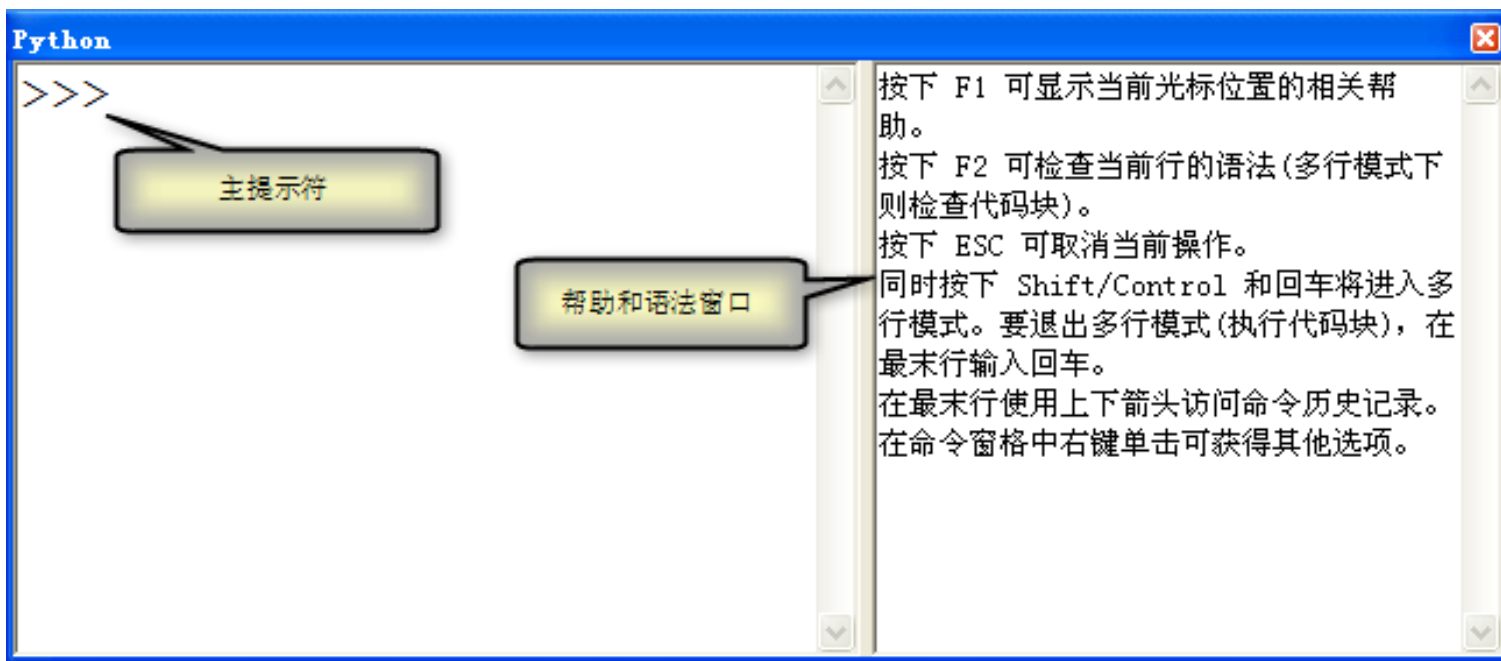
□(5)ArcPy运行工具

- 从技术角度讲，地理处理工具是可通过 arcpy 访问的函数,即可以像 Python函数那样访问这些工具。
- 为了避免引起混淆，总是会在工具函数和非工具函数之间加以区分
- 执行地理处理工具时，工具的结果会返回到 Result 对象中。
- 示例：
 - >>> arcpy.env.workspace = "c:/data/Portland.gdb"
 - >>> result = arcpy.Buffer_analysis("streets", "streetBuf",
 - >>> print result



Python窗口

- Python窗口为用户提供了高效、便捷地使用地理处理工具和 ArcGIS 中的 Python 函数的交互操作界面。
- 在该窗口中运行的 Python 命令包括**单行代码、带逻辑的复杂块**。
- 在该窗口中可通过自定义或第三方Python模块和库实现其他功能



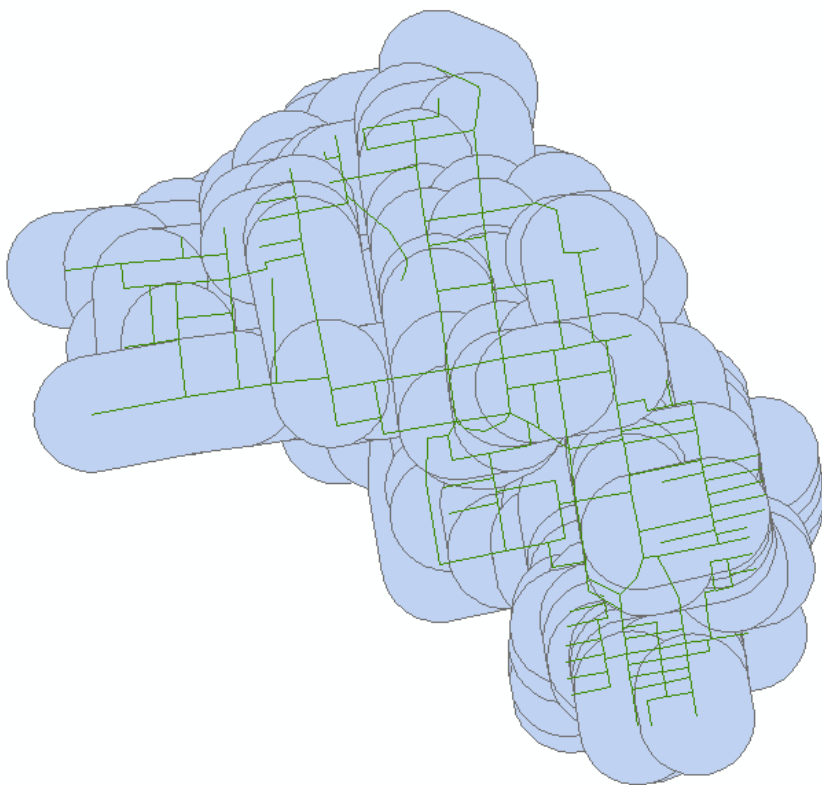
□ Python窗口操作步骤:

- 打开地图文档
- 打开Python窗口
- import arcpy
- 设置工具空间
 - 例: `arcpy.env.workspace = "C:/data/"`
- 调用工具箱中的工具或执行python脚本
 - 例: `arcpy.Buffer_analysis("road","road_buffer",100);`

1.4 ArcGIS Python窗口



Python窗口操作示例：

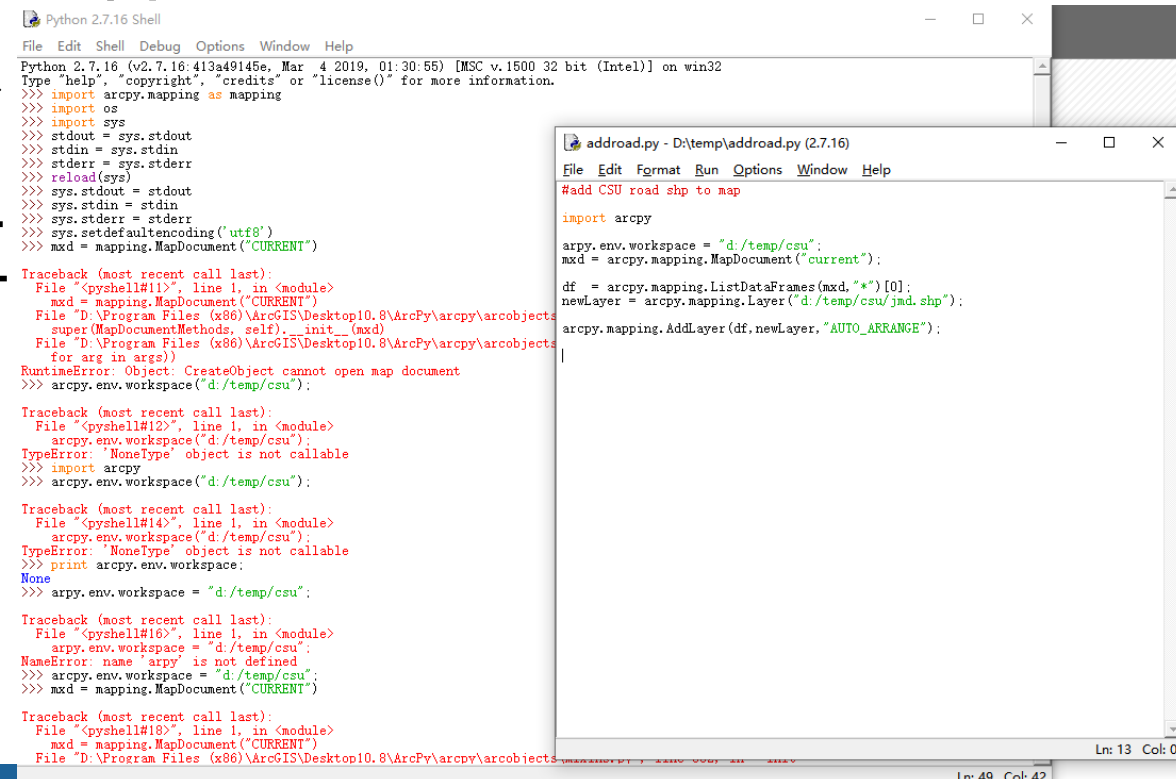


Python

```
>>> import arcpy
>>> arcpy.env.workspace = "d:/temp/csu";
>>> result = arcpy.Buffer_analysis("road","road_buffer",0.001);
>>> |
```

Executing: Buffer road d:/temp/CSU
\\road_buffer.shp ".001 Unknown" FULL ROUND
NONE # PLANAR
Start Time: Wed Oct 26 16:27:05 2022
Succeeded at Wed Oct 26 16:27:05 2022
(Elapsed Time: 0.13 seconds)

- ❑ IDLE（编辑器集成开发环境）既有交互式窗口，可用于执行单行 Python 代码
- ❑ IDLE帮助用户输入、编辑、检查语法以及调试 Python 代码
- ❑ IDLE可以保存、打开和执行脚本文件
 - 注意：不是所有的脚本文件都可以在IDLE中直接执行，一些与Map对象交互操作的脚本只能在ArcMap工具里执行



1.5 Python IDLE

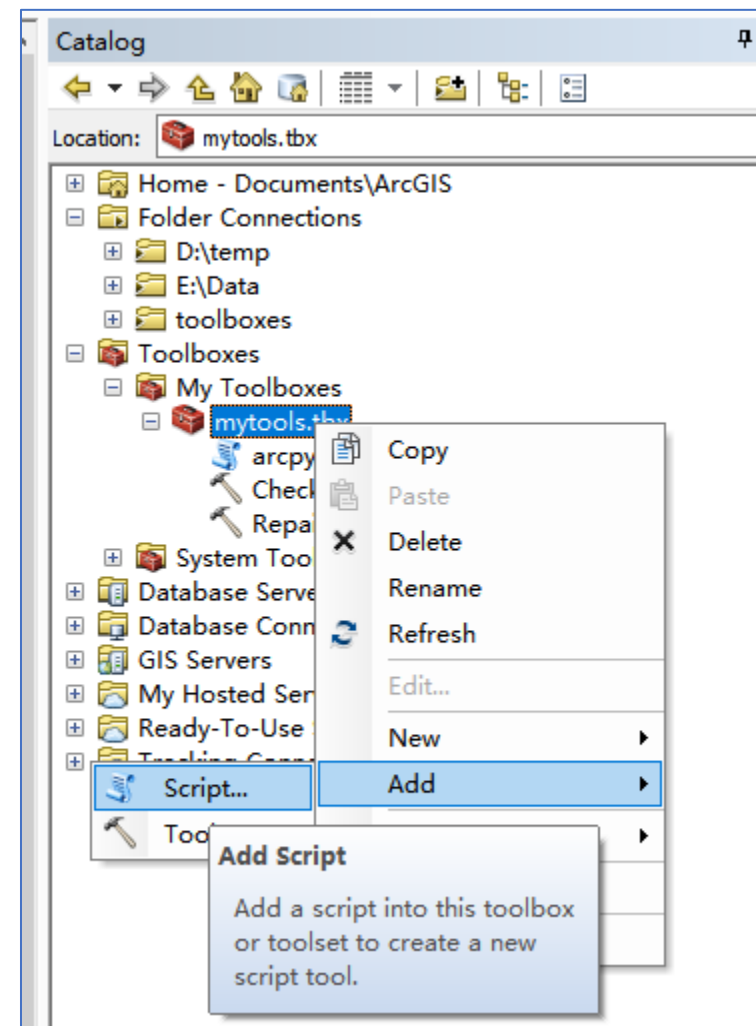
脚本程序文件示例:

在IDLE中编辑py文件

```
import arcpy  
arcpy.env.workspace = "d:/temp/csu";  
mxd = arcpy.mapping.MapDocument("current");  
df = arcpy.mapping.ListDataFrames(mxd,"*")[0];  
newLayer = arcpy.mapping.Layer("d:/temp/csu/jmd.shp");  
arcpy.mapping.AddLayer(df,newLayer,"AUTO_ARRANGE");
```

在ArcMap的工具箱里添加script

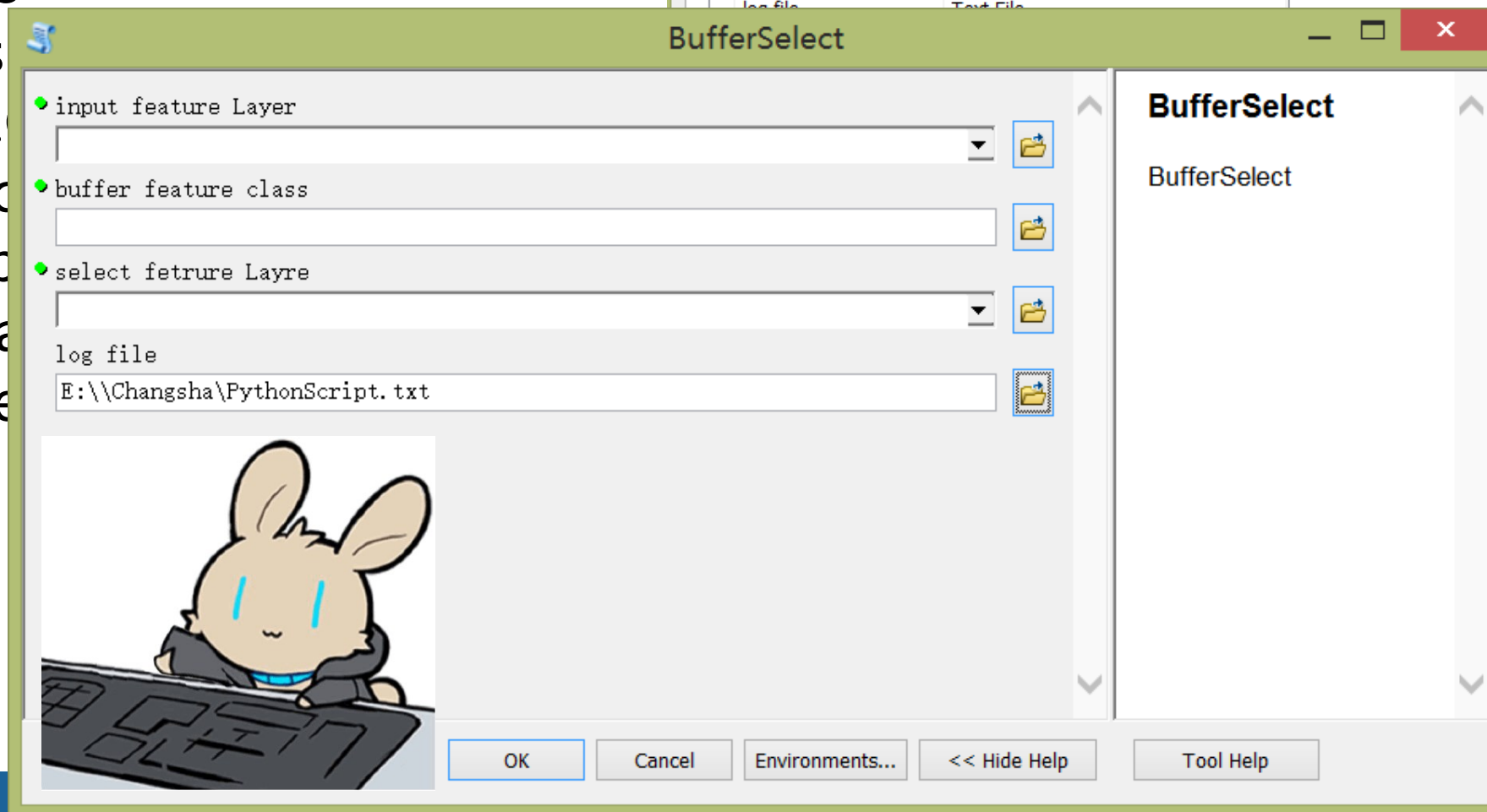
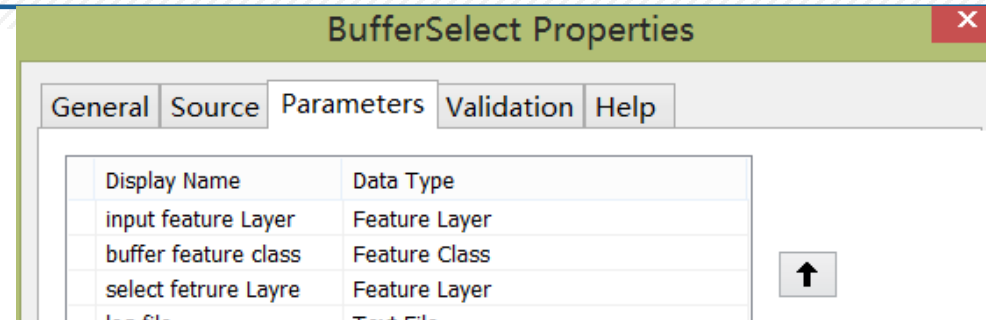
将脚本文件指向保存的脚本文件



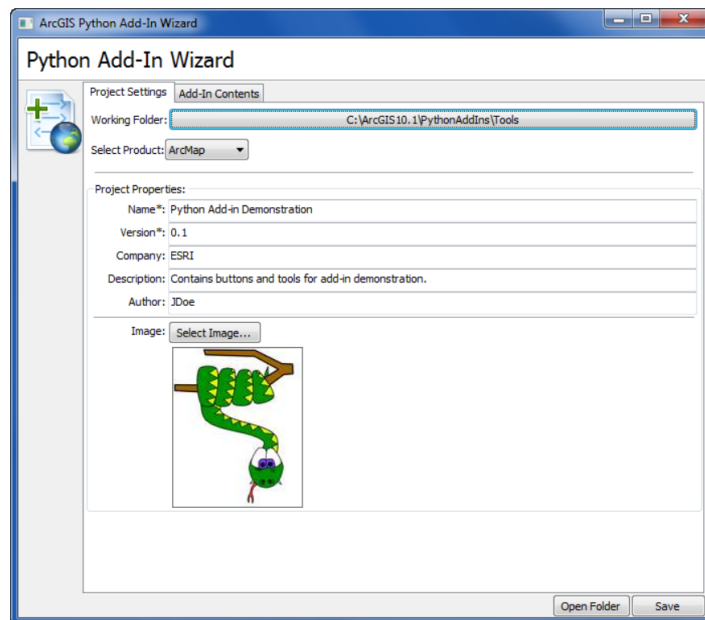
创建用户地理工具：

在IDLE中编辑py文件

- import arcpy, os
- arcpy.env.workspace = ...
- railway = arcpy.SearchCursor(...)
- railwayBuf = arcpy.Buffer_analysis(railway, railwayBuf, 100)
- residents = arcpy.SearchCursor(...)
- arcpy.Buffer_analysis(residents, residentsBuf, 100)
- arcpy.SelectLayerByLocation(railwayBuf, 'NEWPOINT', residentsBuf, 'TRUE', 'DISSOLVE', 'NODATA')



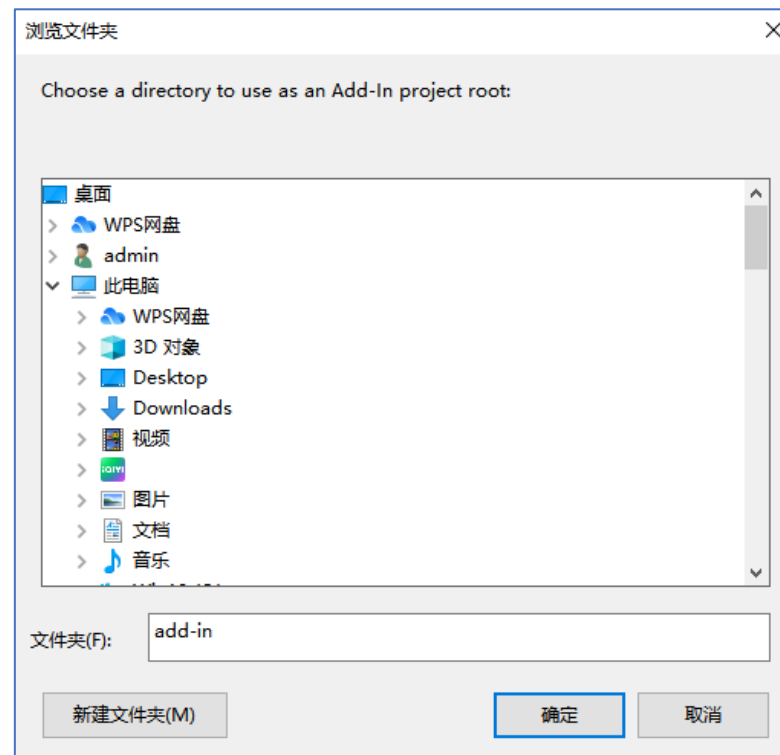
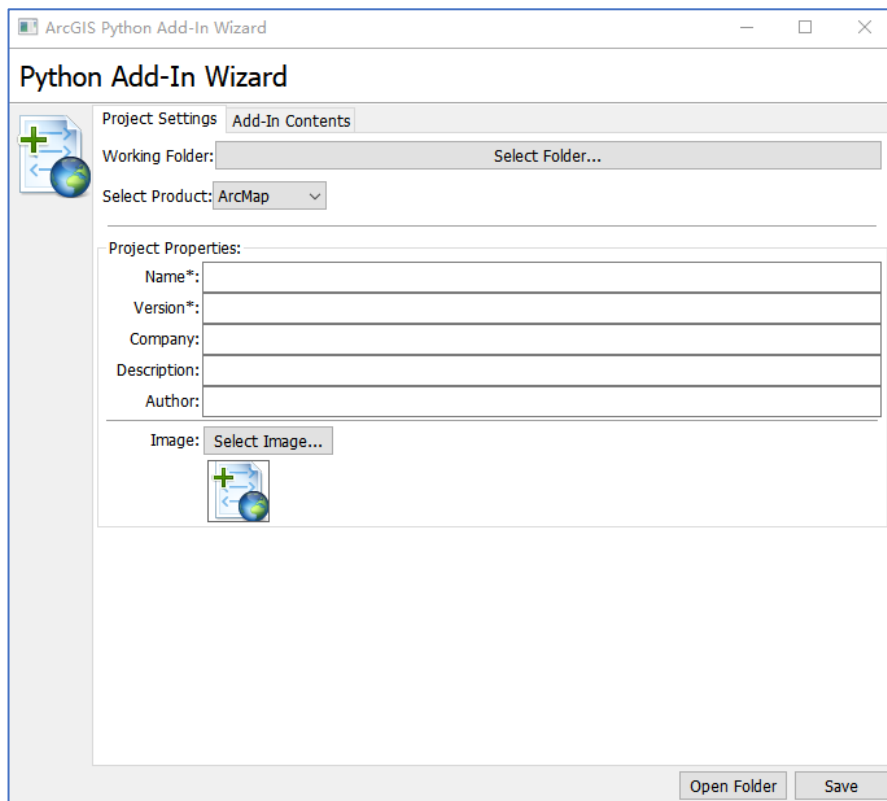
- Add-in(加载项或插件)是自定义项（如工具栏上的工具集合），可以插入到 ArcGIS Desktop 应用程序中，以提供用于完成自定义任务的补充功能
- Add-in向导下载地址
 - <https://www.arcgis.com/home/item.html?id=5f3aefe77f6b4f61ad3e4c62f30bff3b>



- 在线教程
- <https://desktop.arcgis.com/zh-cn/arcmap/latest/analyze/python-addins/creating-an-add-in-project.htm>

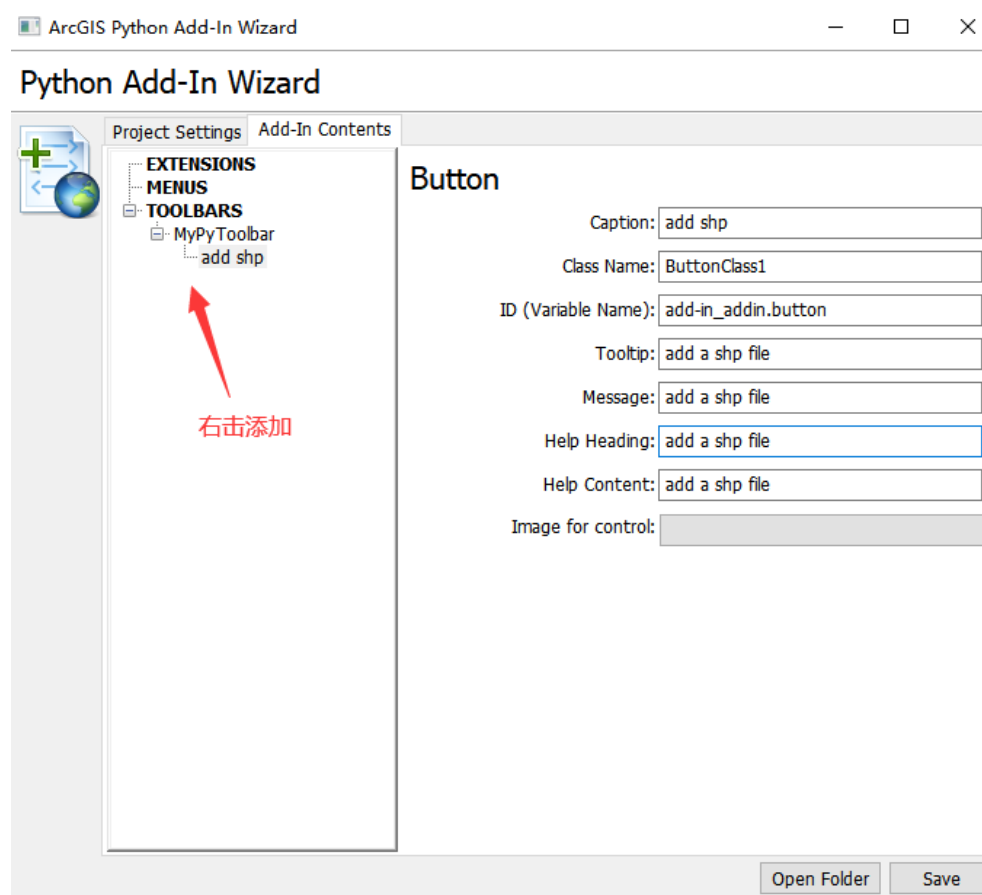
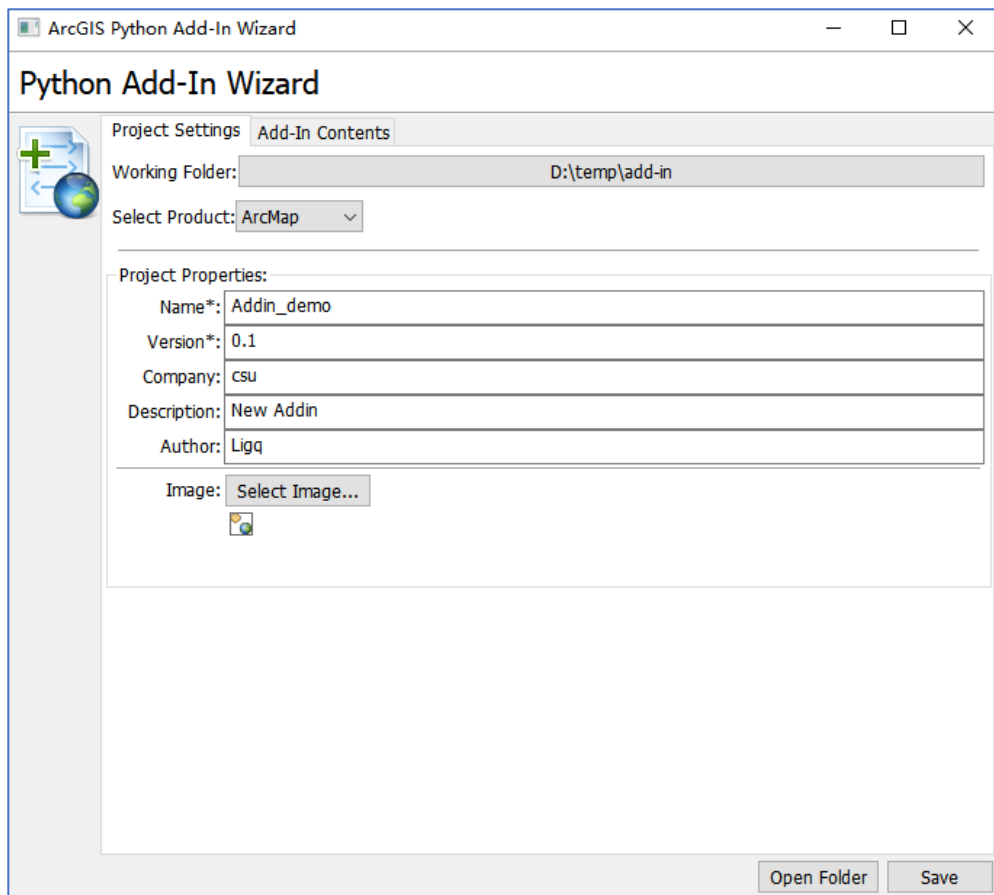
创建add-in

- (1) 执行addin_assistant.exe, 在弹出对话框中, 选择add-in存储目录 (必须为空), 并录入add-in信息



创建add-in

- (2)在Add-in Contents页框中，添加工具栏、按钮等，点save按钮



□ 创建add-in

脚本中不要用中文，包括注释！！！！

- (3)编辑add-in脚本。 打开add-in目录下的install文件夹下的.py文件， 添加需要脚本。

- 示例代码：

```
import arcpy
import pythonaddins
```

```
class ButtonClass1(object):
```

```
    """Implementation for add-in_addin.button (Button)"""
```

```
    def __init__(self):
```

```
        self.enabled = True
```

```
        self.checked = False
```

```
    def onClick(self):
```

```
        mxd = arcpy.mapping.MapDocument("current");
```

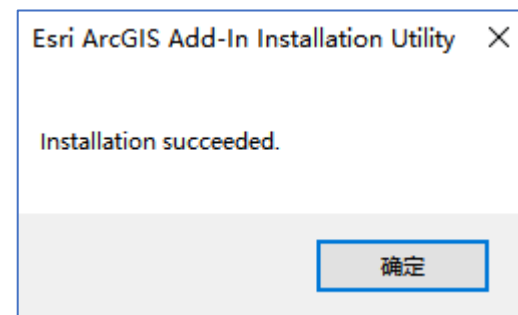
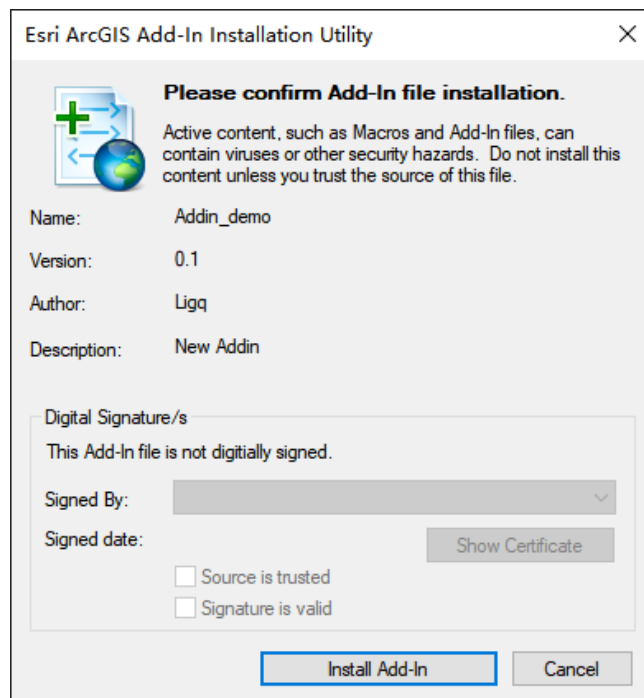
```
        df = arcpy.mapping.ListDataFrames(mxd,"*")[0];
```

```
        newLayer = arcpy.mapping.Layer("d:/temp/csu/jmd.shp");
```

```
        arcpy.mapping.AddLayer(df,newLayer,"AUTO_ARRANGE");
```

安装add-in

- 执行makeaddin.py文件, 生成add-in安装文件. esriaddin
- 双击. esriaddin, 安装生成的add-in,在弹出对象话框中点"Install Add-in", 完成安装



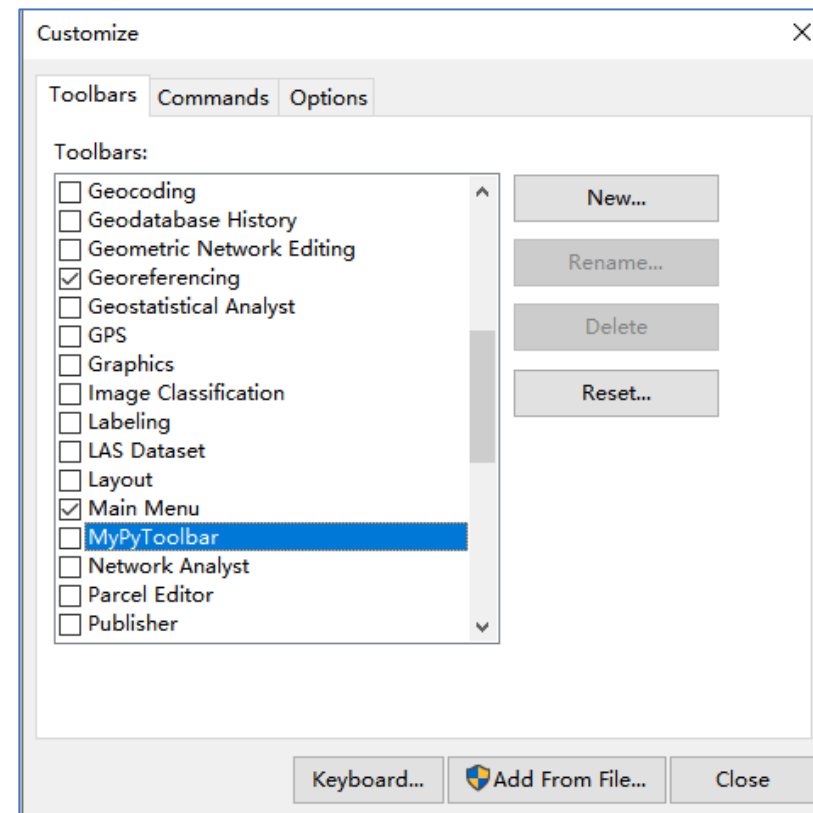
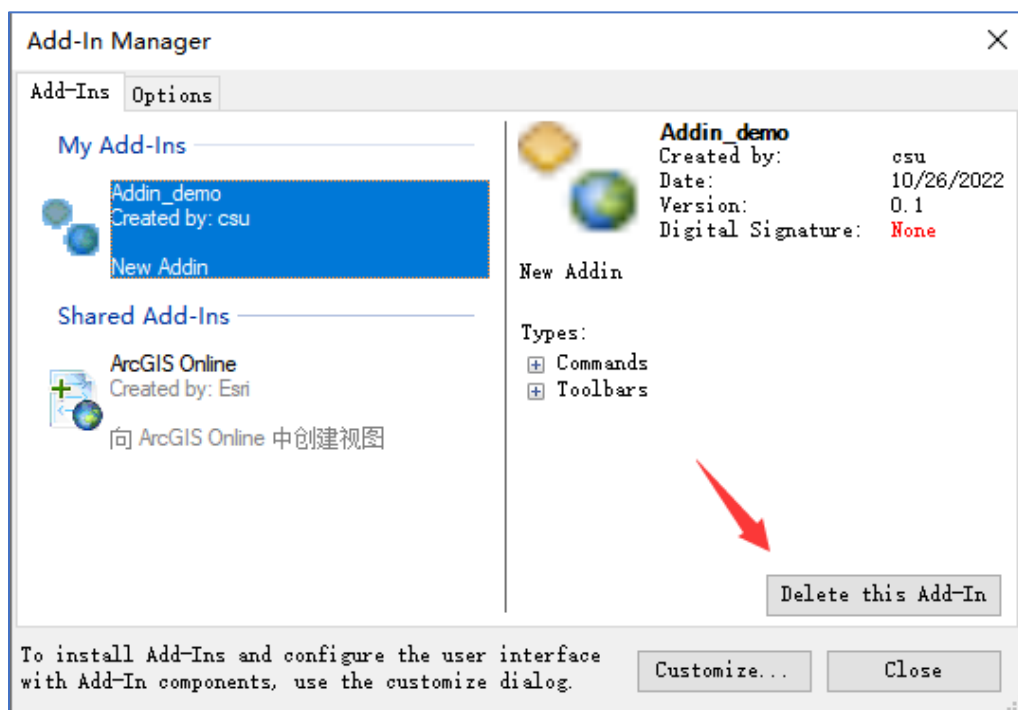
1.6 用python开发Add-in (加载项)

向ArcMap添加add-in

- 打开Customize -> Toolbars -> Customize, 选中add-in

卸载add-in

- 打开Customize -> Add-in Manger



1.6 用python开发Add-in (加载项)



□ 注意

- 存储add-in目录名称、项目名称、工具栏名称、按钮名称等信息中，不要使用空格、- 等字符，否则会产生错误.
- 在修改代码时，需要重新生成安装文件。即先删除原.esriaddin文件,然后**重新**双击makeaddin.py生成.esriaddin文件

□ Add-in调试方法

- Add-in程序调试不像其它IDE可以设置断点、跟踪调试
- 在可能出错的地方添加print语句，输出变量值等。ArcGIS在python窗口中输出打印结果
- 添加 try ... except

try:

... ..

except Exception as e:

print e

□ 实例

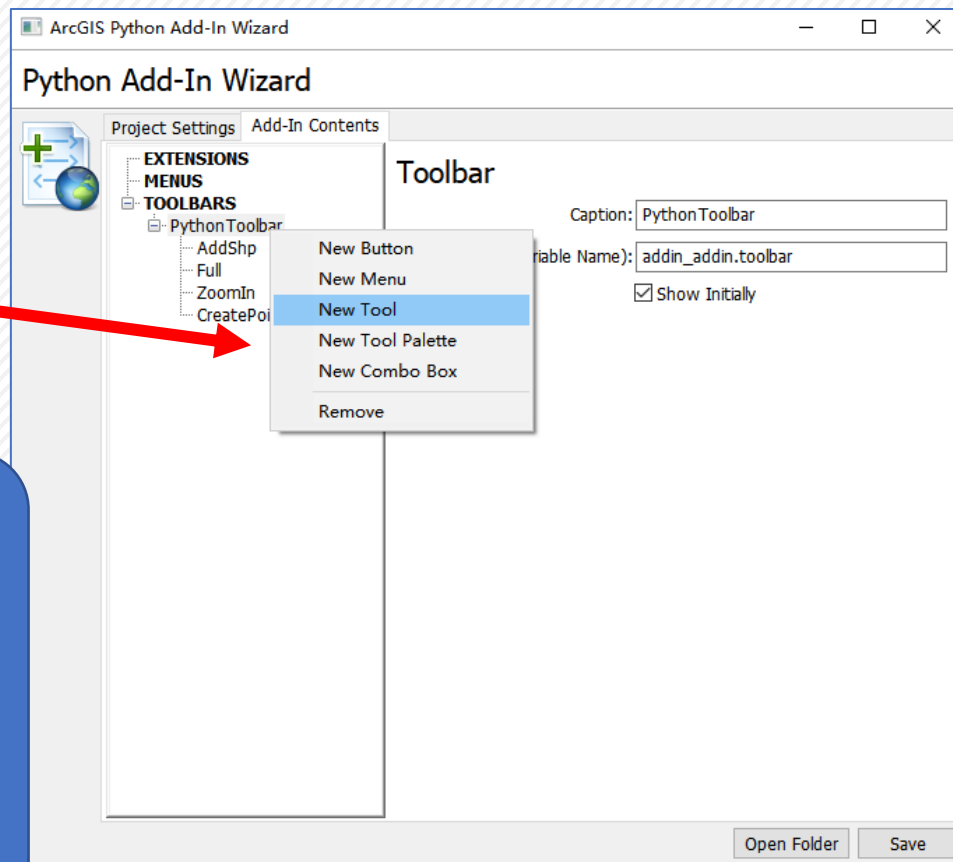
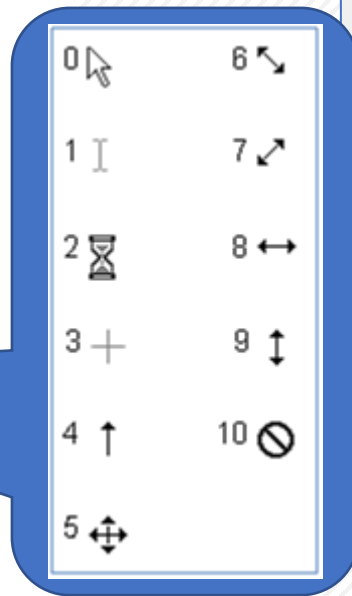
- 制作工具栏MyPyToolbar, 添加以下按钮或工具
 - (1)按钮(Button) AddShape: 点击添加指定的shp文件
 - (2)按钮(Button)Full: 显示全图
 - (3)工具(Tool)ZoomIn: 拉框放大
 - (4)工具(Tool)CreateRandomPoints: 在绘制的矩形框中随机生成200个点
 - (5)工具(Tool)SelectPoints: 在地图上绘制一条线, 选中线附近的点



实例

- 添加/设置按钮或工具
- 注意

- Button和Tool有区别
- 需要与地图交互绘制图形时, 必须使用Tool
- Tool的属性
 - cursor: 鼠标形状
 - shape: 要绘制的图形类型
 - 包括 **Line, Circle, Rectangle**



实例

- 点击向导对话框中的save按钮，在Install文件夹里生成py脚本

```
import arcpy
import pythonaddins

class BtnAddShape(object):
    """Implementation for addin_addin.button_addshp (Button)"""
    def __init__(self):
        self.enabled = True
        self.checked = False
    def onClick(self):
        pass

class BtnFull(object):
    """Implementation for addin_addin.button_full (Button)"""
    def __init__(self):
        self.enabled = True
        self.checked = False
    def onClick(self):
        pass

class ToolCreatePoints(object):
    """Implementation for addin_addin.tool_create_points (Tool)"""
    def __init__(self):
        self.enabled = True
        self.shape = "NONE" # Can set to "Line", "Circle" or "Rectangle" for interactive shape drawing and to activate the onLine/Polygon/Circle event

    def onRectangle(self, rectangle_geometry):
        pass

class ToolSelect(object):
    """Implementation for addin_addin.tool_select (Tool)"""
    def __init__(self):
        self.enabled = True
        self.shape = "NONE" # Can set to "Line", "Circle" or "Rectangle" for interactive shape drawing and to activate the onLine/Polygon/Circle event

    def onRectangle(self, rectangle_geometry):
        pass

class ToolZoomIn(object):
    """Implementation for addin_addin.tool_zoomin (Tool)"""
    def __init__(self):
        self.enabled = True
        self.shape = "NONE" # Can set to "Line", "Circle" or "Rectangle" for interactive shape drawing and to activate the onLine/Polygon/Circle event sinks.

    def onRectangle(self, rectangle_geometry):
        pass
```

```
class ToolZoomIn(object):
    """Implementation for addin_addin.tool_zoomin (Tool)"""
    def __init__(self):
        self.enabled = True
        self.shape = "NONE" # Can set to "Line", "Circle" or "Rectangle" for interactive shape drawing and to activate the onLine/Polygon/Circle event

    def onMouseDown(self, x, y, button, shift):
        pass

    def onMouseDownMap(self, x, y, button, shift):
        pass

    def onMouseUp(self, x, y, button, shift):
        pass

    def onMouseUpMap(self, x, y, button, shift):
        pass

    def onMouseMove(self, x, y, button, shift):
        pass

    def onMouseMoveMap(self, x, y, button, shift):
        pass

    def onDoubleClick(self):
        pass

    def onKeyDown(self, keycode, shift):
        pass

    def onKeyUp(self, keycode, shift):
        pass

    def deactivate(self):
        pass

    def onCircle(self, circle_geometry):
        pass

    def onLine(self, line_geometry):
        pass

    def onRectangle(self, rectangle_geometry):
        pass
```

□ 实例

- BtnAddShp类的onClick事件中添加以下代码:

```
def onClick(self):
```

```
    mxd = arcpy.mapping.MapDocument( "current" );
```

```
    df = arcpy.mapping.ListDataFrames(mxd,"*")[0];
```

```
    newLayer = arcpy.mapping.Layer( "d:/temp/csu/jmd.shp" );
```

```
    arcpy.mapping.AddLayer(df,newLayer, "AUTO_ARRANGE" );
```

完整代码从GitHub下载.

□ 实例

- BtnFull类的onClick事件中添加以下代码：

```
def onClick(self):
```

```
    mxd = arcpy.mapping.MapDocument( "current" );
```

```
    df = arcpy.mapping.ListDataFrames(mxd, "*")[0];
```

```
    lyr = arcpy.mapping.ListLayers(mxd, "*", df)[0];
```

```
    df.extent = lyr.getExtent();
```

□ 实例

- ToolZoomIn类的代码:

```
class ToolZoomIn(object):
```

```
    """Implementation for addin_addin.tool_zoomin (Tool)"""
```

```
    def __init__(self):
```

```
        self.enabled = True
```

```
        self.cursor = 3
```

```
        self.shape = "Rectangle" # Can set to "Line", "Circle" or "Rectangle"
```

```
    def onRectangle(self, rectangle_geometry):
```

```
        mxd = arcpy.mapping.MapDocument( "current" );
```

```
        df = arcpy.mapping.ListDataFrames(mxd,"*")[0];
```

```
        df.extent = rectangle_geometry;
```


□ 实例

- ToolCreatePoints类的代码:

```
class ToolCreatePoints(object):
```

```
    def __init__(self):
```

```
        self.enabled = True
```

```
        self.checked = False
```

```
        self.cursor = 3
```

```
        self.shape = "Rectangle" # Can set to "Line", "Circle" or "Rectangle"
```

```
    def onRectangle(self, rectangle_geometry):
```

```
        arcpy.env.workspace = "d:/temp/";
```

```
        arcpy.env.overwriteOutput = True;
```

```
        ext = rectangle_geometry;
```

```
        if arcpy.Exists( "points" ):
```

```
            arcpy.Delete_management( "points" );
```

```
        arcpy.CreateRandomPoints_management( "d:/temp","points.shp" , "" ,ext,200);
```

```
        arcpy.RefreshTOC();
```

实例

ToolSelect类的代码:

```
class ToolSelect (object):
```

```
    def __init__ (self):
```

```
        self.enabled = True
```

```
        self.checked = False
```

```
        self.cursor = 3
```

```
        self.shape = "Line"
```

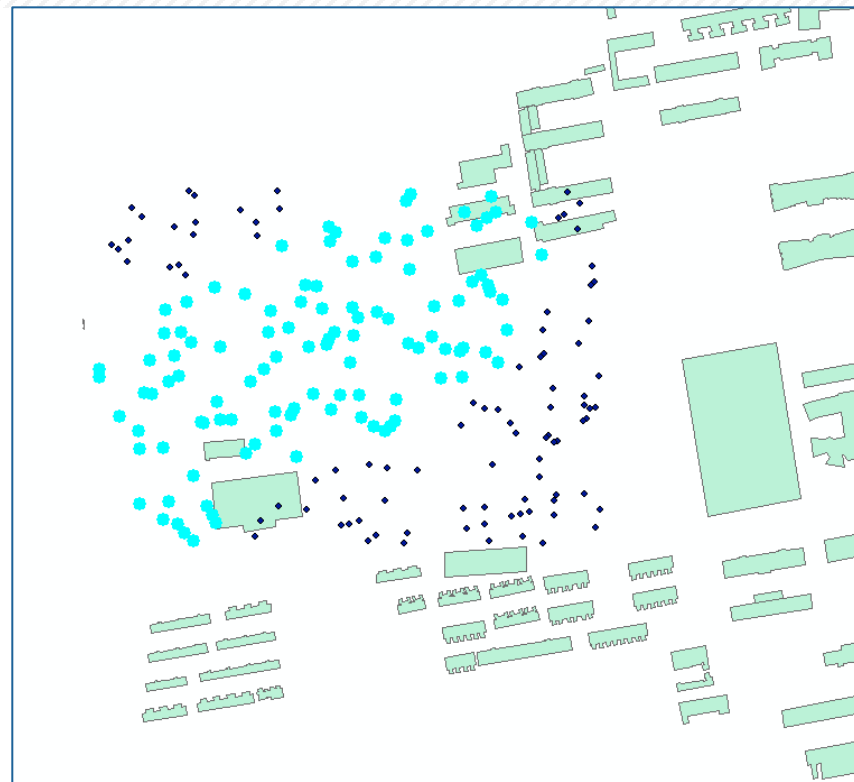
```
    def onLine (self, line_geometry):
```

```
        newLayer = arcpy.MakeFeatureLayer_management( "points",'lyr' );
```

```
        arcpy.SelectLayerByLocation_management( "lyr","WITHIN_A_DISTANCE",line_geometry,0.001);
```

```
        matchcount = arcpy.GetCount_management( "lyr" ).getOutput(0);
```

```
        print "The matched points count: " +matchcount;
```



□ 2.1 地图文档管理

- (1) 引用当前地图或地图文件
- (2) 修改地图范围(缩放)

□ 2.2 地图图层管理

- (1) 添加图层
- (2) 获取地图图层列表
- (3) 移除图层
- (4) 修改图层数据源

□ 2.3 地图输出

□(1)引用当前地图或地图文件

- 当前地图: `arcpy.mapping.MapDocument("current");`
- 地图文件: `arcpy.mapping.MapDocument("d:/temp/demo.mxd");`
 - `Import os`
 - `os.startfile(mypath)` #打开地图文件

□(2)修改地图范围(缩放)

```
mxd = arcpy.mapping.MapDocument("current");  
df = arcpy.mapping.ListDataFrames(mxd,"*")[0];  
df.extent = 指定范围;
```

■ 示例：将地图缩放到指定的范围

- `rect = arcpy.Extent(0.0,0.0,100,100,None,None,None,None);`
- `df.extent=rect;`

□(1)添加图层

```
mxd = arcpy.mapping.MapDocument("current");  
df = mxd.activeDataFrame;  
newLayer = arcpy.mapping.Layer("d:/temp/csu/jmd.shp");  
arcpy.mapping.AddLayer(df,newLayer,"AUTO_ARRANGE");
```

□(2)获取地图图层列表

```
mxd = arcpy.mapping.MapDocument("current");  
df = mxd.activeDataFrame;  
layers = arcpy.mapping.ListLayers(mxd,'*',df);  
for lyr in layers:  
    print lyr.name
```



□(3)移除图层

```
mxd = arcpy.mapping.MapDocument("current");
```

```
df = mxd.activeDataFrame;
```

```
layers = arcpy.mapping.ListLayers(mxd,'*',df);
```

```
for lyr in layers:
```

```
    if lyr.name == 'points':
```

```
        layer = lyr;
```

```
arcpy.mapping.RemoveLayer(df,lyr);
```

■ 获取指定名称图层的方法:

```
lyr = arcpy.mapping.ListLayers('图层名', '*',df)[0];
```

或

```
lyr = arcpy.mapping.ListLayers(mxd,'图层名',df)[0]
```

□(4)修改图层数据源

- 方法1：统改mxd文件的工作区

```
mxd = arcpy.mapping.MapDocument( "current" );  
mxd.replaceWorkspaces(r "d:\csu.mdb" ,  
                       "ACCESS_WORKSPACE" ,  
                       r"D:\csu3.mdb" ,  
                       "ACCESS_WORKSPACE" ,  
                       True);
```

```
mxd.save();
```

#原工作区

#原工作区类型

#新工作区

#新工作区类型

□(4)修改图层数据源

■ 方法2：逐层修改数据源

```
mxd = arcpy.mapping.MapDocument( "current" );  
df = mxd.activeDataFrame;  
layers = arcpy.mapping.ListLayers(mxd,'*',df);  
#获取所有缺失数据源的图层列表  
brknList = arcpy.mapping.ListBrokenDataSources(mxd);  
for lyr in layers:  
    if lyr.name in [brn.name for brn in brknList]:  
        lyr.replaceDataSource(r "d:\csu.mdb",  
                              "ACCESS_WORKSPACE",  
                              lyr.name  
                              );  
mxd.save();
```

#新数据源
#新数据源工作区类型
#新数据源名称



□(1)打印地图

```
mxd = arcpy.mapping.MapDocument( "current" );  
df = mxd.activeDataFrame;  
arcpy.mapping.PrintMap(mxd,"",df);
```

□(2)输出为PDF

```
mxd = arcpy.mapping.MapDocument( "current" );  
df = mxd.activeDataFrame;  
arcpy.mapping.ExportToPDF(mxd,"d:/temp/demo.pdf");
```

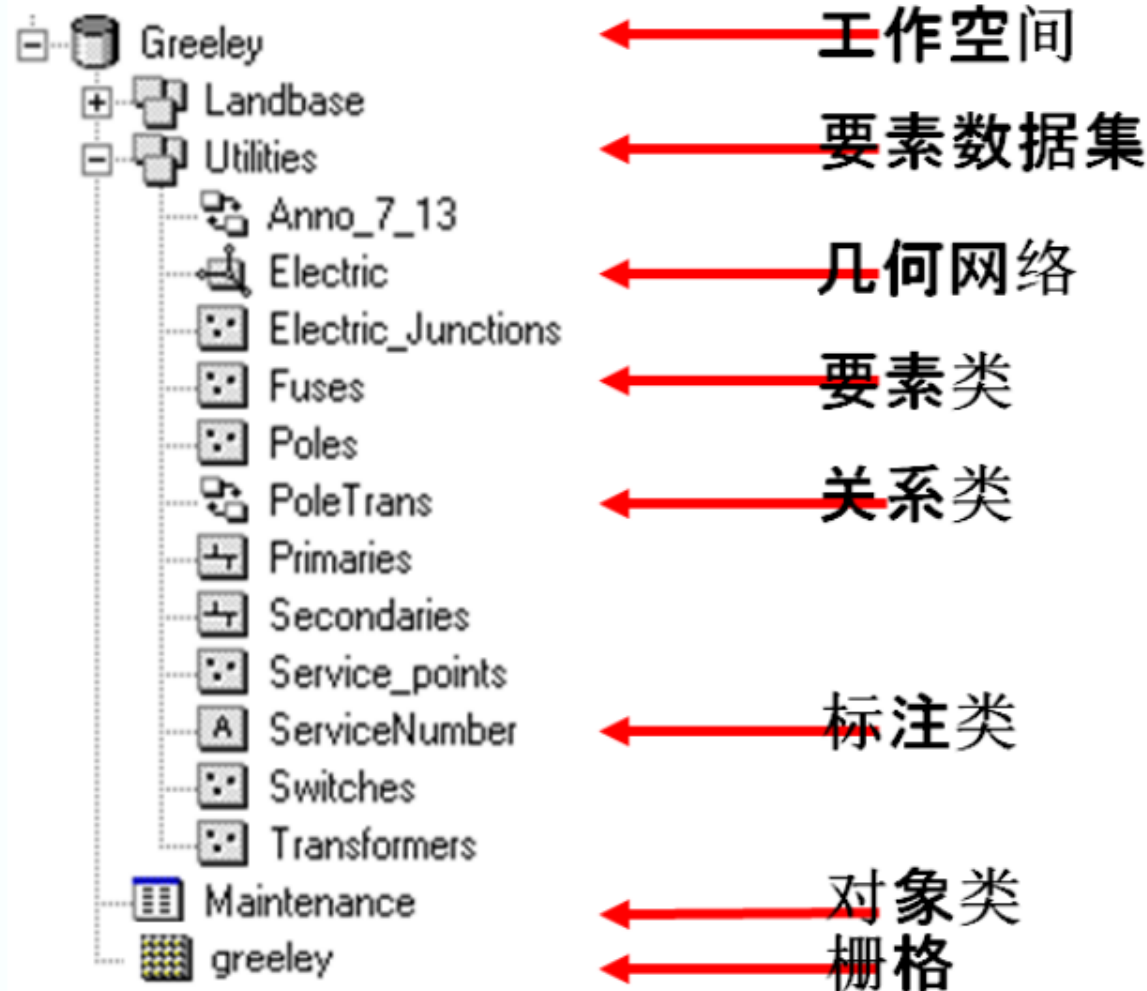
□(3)输出为其它格式

ExportToJPEG、ExportToBMP、ExportToEPS、ExportToGIF

- 3.1 ArcGIS常用的空间数据类型
- 3.2 列出数据
- 3.3 判断数据是否存在
- 3.4 数据描述
- 3.5 数据字段

空间数据类型

空间数据类型	文件后缀名
SHP文件	.shp,.dbf,.shx等
文件地理数据库	.gdb
个人地理数据库	.mdb
企业地理数据库	.sde
CAD文件	.dwg, .dxf
EXCEL文件	.xls, .xlsx
文本文件	.txt, .csv



3.2 列出数据



□(1)列出指定目录下的SHP文件列表

```
>>> import arcpy
>>> from arcpy import env
>>> env.workspace = "d:/temp/csu" ;
>>> fcList = arcpy.ListFeatureClasses();
>>> print fcList;
[u'jmd.shp, u'road.shp']
```

□(2)列出个人空间数据库的数据列表

```
>>> env.workspace = "d:/temp/csu.mdb" ;
>>> fcList = arcpy.ListFeatureClasses();
>>> print fcList;
[u'jmd.shp, u'road.shp']
```



□(1)判断shp文件是否存在示例

```
>>> arcpy.env.workspace = "d:/temp/csu" ;
>>> if arcpy.Exists( "road.shp" ):
...     lyr = arcpy.mapping.Layer( "road.shp" );
...     arcpy.mapping.AddLayer(df,lyr, "AUTO_ARRANGE" );
... else:
...     print "The road feature class not exists";
```

□(2)判断个人空间数据库中是否存在要素类示例

```
>>> arcpy.env.workspace = "d:/temp/csu.mdb" ;
>>> if arcpy.Exists( "road" ):
...     lyr = arcpy.mapping.Layer( "road" );
...     arcpy.mapping.AddLayer(df,lyr, "AUTO_ARRANGE" );
... else:
...     print "The road feature class not exists";
```

- 数据描述 `arcpy.Describe` 函数返回的 `Describe` 对象包含多个属性:
 - 数据类型
 - 字段
 - 索引
 - 其他属性
- `Describe` 对象的属性是 **动态** 的，根据所描述数据类型的不同，属性内容也会不同。支持的数据类型：
 - 常用要素类：gdb、mdb、shp、sde、cad等
 - 表和数据集属性组中的属性

□(1)ShapeFile要素类属性

- 要素类类型属性 : featureType
- 是否启用m值 : hasM
- 是否启用z值 : hasZ
- 是否有空间索引 : hasSpatialIndex
- Shape字段名称 : shapeFieldName
- 几何形状类型 : shapeType
- 示例1
 - `import arcpy`
 - `desc = arcpy.Describe("d:/temp/csu/road.shp");`
 - `print "要素类型: " + desc.featureType + " \n几何类型: " + desc.shapeType + "\有索引:" + str(desc.hasSpatialIndex);`

□(2)个人空间数据库中要素类属性

■ 示例

- import arcpy
- arcpy.env.workspace = "d:/temp/csu/csu.mdb" ;
- desc = arcpy.Describe("road");
- print "要素类型: " +desc.featureType + " \n几何类型: " +desc.shapeType+"\n有索引:"+str(desc.hasSpatialIndex);

□(2)GDB数据库中要素类属性

■ 示例

- import arcpy
- arcpy.env.workspace = "d:/temp/csu/csu.gdb" ;
- desc = arcpy.Describe("road");
- print "要素类型: " +desc.featureType + " \n几何类型: " +desc.shapeType+"\n有索引:"+str(desc.hasSpatialIndex);

- 请写出arcpy程序代码，打印给定目录(如d:\csu)下所有shapefile的名称、要素类型和几何类型。

```
import arcpy
arcpy.env.workspace = "d:/csu" ;
shps = arcpy.ListFeatureClasses();
for shp in shps:
    desc = arcpy.Describe(shp);
    print "文件: \t" + shp+ "\n要素类型:\t" + desc.featureType+ " \n几何类型:\t" + desc.shapeType;
```



▣ **arcpy.ListFields函数** 列出指定数据集中的要素类、shapefile 或表中的字段

■ 示例1

- **import** arcpy
- fields = arcpy.ListFields("d:/temp/csu/road.shp")
- **for** field **in** fields:
- **print**("字段{0}: 类型{1},长度{2}" .format(field.name, field.type, field.length))

■ 示例2：输出字段列表

- **import** arcpy
- featureclass = "d:/temp/csu/road.shp"
- field_names = [f.name for f in arcpy.ListFields(featureclass)]

■ arcpy.Field类说明见：

- <https://desktop.arcgis.com/zh-cn/arcmap/latest/analyze/arcpy-classes/field.htm>

- 请写出arcpy程序代码，打印给定目录(如d:\temp\csu)下所有shapefile的名称及其字段名称、类型。如果字段为文本类型，则显示字段长度。

```
import arcpy
arcpy.env.workspace = "d:/temp/csu ";
shps = arcpy.ListFeatureClasses();
for shp in shps:
    fields = arcpy.ListFields(shp);
    print "\n{0}的字段列表:".format(shp);
    for field in fields:
        str = "字段[{0}]\t{1}\t".format(field.name,field.type);
        if field.type == 'String':
            str += "({0})".format(field.length);
    print str;
```



□4.1 要素类管理

□4.2 要素管理

□4.3 要素查询

□4.4 要素选择

□(1)创建要素类

■ 用法

```
arcpy.CreateFeatureclass_management(  
    out_path,          #要素类存储路径  
    out_name,          #要素类名称  
    geometry_type,     #要素类几何类型: POINT、MULTIPOINT、POLYGON、POLYLINE、MULTIPATCH  
    template,          #使用模板定义要素, 可选项  
    has_m,             #是否有m值, 可选项  
    has_z,             #是否有z值, 可选项  
    spatial_reference   #指定空间参考系, 可选项  
)
```

■ 示例

```
arcpy.CreateFeatureclass_management("d:/temp","test.shp","POINT");
```

□(1)创建要素类

■ 示例2（独立脚本）

```
import arcpy
```

```
arcpy.env.workspace = "C:/data"
```

```
out_path = "C:/output"
```

```
out_name = "habitatareas.shp"
```

```
geometry_type = "POLYGON"
```

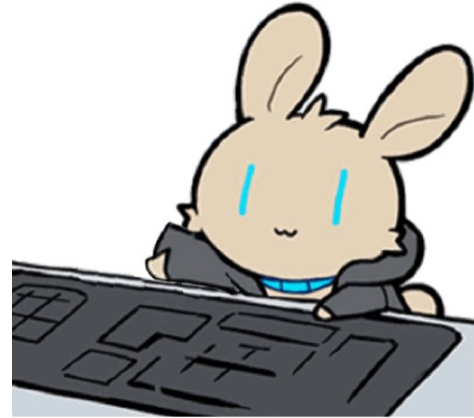
```
template = "study_quads.shp"
```

```
has_m = "DISABLED"
```

```
has_z = "DISABLED"
```

```
spatial_ref = arcpy.Describe( "C:/data/studyarea.shp" ).spatialReference
```

```
arcpy.CreateFeatureclass_management(out_path, out_name, geometry_type,  
                                     template, has_m, has_z, spatial_ref)
```



□(2)删除要素类

■ 用法

- `arcpy.Delete_management(featureClass_path);`

■ 示例

```
featureClass_path = "d:/temp/points.shp";  
if arcpy.Exists(featureClass_path ):  
    arcpy.Delete_management(featureClass_path );
```


□(3) 添加字段

■ 用法

```
arcpy.management.AddField(  
    in_table,           #表名称  
    field_name,         #字段名称  
    field_type,         #字段类型: TEXT,FLOAT,DOUBLE,SHORT,LONG,DATE,...  
    field_precision,    #数值类型精度, 可选项  
    field_scale,        #数值类型的小数位数, 可选项  
    field_length,       #字符型的长度, 可选项  
    field_alias,        #字段别名, 可选项  
    field_is_nullable,  #是否允许空值, 可选项  
    field_is_required,  #是否必填, 可选项  
    field_domain        #值域, 可选项  
)
```

□(3) 添加字段

■ 示例

```
import arcpy  
featureClass_path = "d:/temp/csu/ test.shp"  
arcpy.management.AddField(featureClass_path , "name" , "STRING" ,20);
```



□(3)删除字段

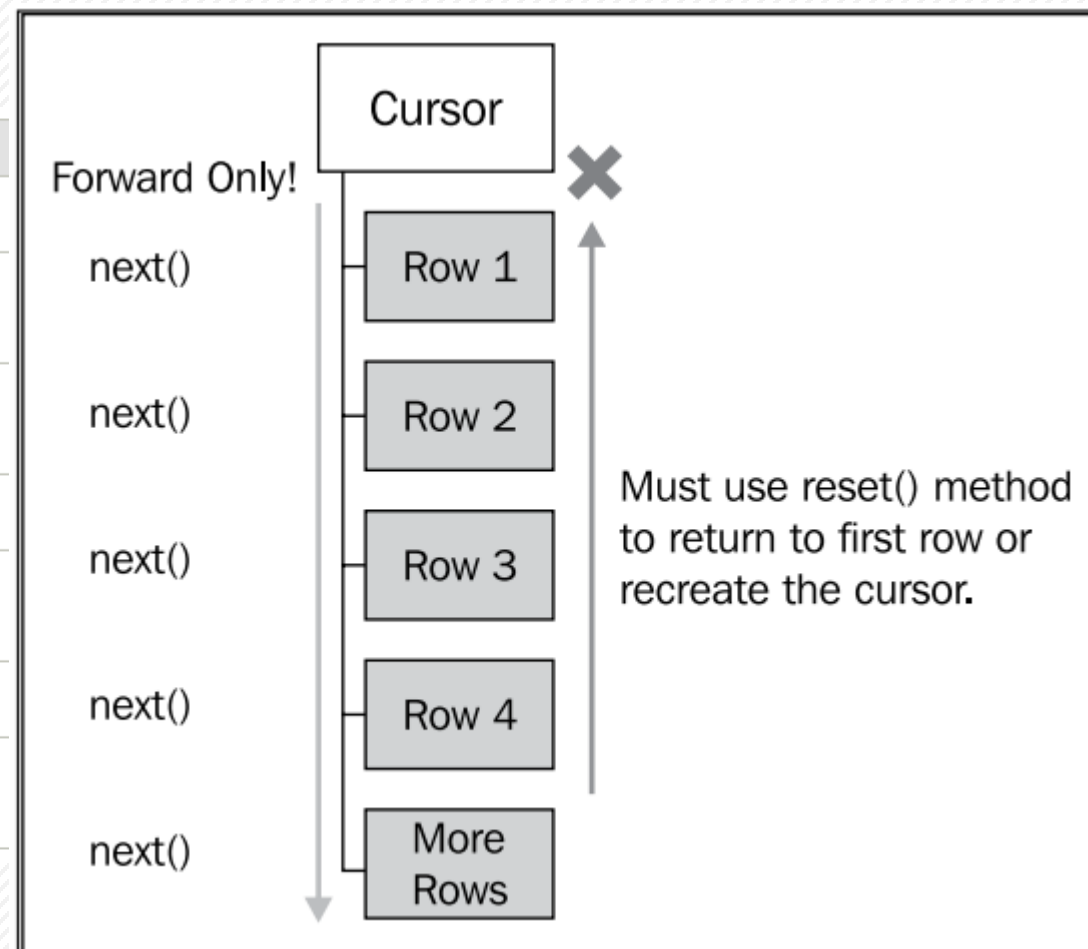
■ 用法

```
arcpy.management.DeleteField(  
    in_table,          #要素类/数据表  
    drop_field         #要删除的字段  
)
```

■ 示例

- `arcpy.management.DeleteField("d:/temp/test.shp","name");`

Cursor type		Method
arcpy.da.SearchCursor	查询光标	next ()
		reset ()
arcpy.da.InsertCursor	插入光标	insertRow ()
arcpy.da.UpdateCursor	更新光标	updateRow ()
		deleteRow ()
		next ()
		reset ()



□(1)添加要素

■ 方法

- 使用arcpy.da.InsertCursor插入新游标（新要素）
- 向新游标插入一行数据

■ 示例：

- import arcpy
- row_values = [('Anderson', (1409934.4442000017, 1076766.8192000017)),
('Andrews', (752000.2489000037, 1128929.8114))]
- cursor = arcpy.da.InsertCursor('d:/temp/test.shp' , ['NAME', '**SHAPE@XY**'])
- for row in row_values:
- cursor.insertRow(row)
- del cursor



□(2)几何数据处理

- SHAPE@XY : 一组要素的质心 x,y 坐标。
- SHAPE@XYZ : 一组要素的质心 x,y,z 坐标。
- SHAPE@TRUECENTROID : 一组要素的质心 x,y 坐标
- SHAPE@X : 要素的双精度 x 坐标。
- SHAPE@Y : 要素的双精度 y 坐标。
- SHAPE@Z : 要素的双精度 z 坐标。
- SHAPE@M : 要素的双精度 m 值。
- SHAPE@JSON : 表示几何的 Esri JSON 字符串。
- SHAPE@WKB : OGC 几何的熟知二进制 (WKB) 表示
- SHAPE@WKT : OGC 几何的熟知文本 (WKT) 表示
- SHAPE@ : 要素的几何对象。

□(2)几何数据处理

- arcpy定义了Point、Polyline、Polygon、Geometry、Multipoint等类，用于管理几何数据。
- 详见：<https://desktop.arcgis.com/zh-cn/arcmap/latest/analyze/arcpy-classes/polyline.htm>
- 示例：
 - import arcpy
 - feature_info = [[[1, 2], [2, 4], [3, 7]],
• [[6, 8], [5, 7], [7, 2], [9, 5]]]
 - features = []
 - for feature in feature_info:
 - array = arcpy.Array([arcpy.Point(*coords) for coords in feature])
 - **array.append(array[0])**
 - features.append(arcpy.Polygon(array))
 - arcpy.CopyFeatures_management(features, "c:/geometry/polygons.shp")



□(2)几何数据处理

■ 示例1

- `cur = arcpy.da.InsertCursor("d:/temp/test.shp", ("SHAPE@WKT" , "name"));`
- `cur.insertRow(["POINT(15 15)" , "test_wkt"]);`

■ 示例2

- `array = arcpy.Array([arcpy.Point(459111.6681, 5010433.1285),`
- `arcpy.Point(472516.3818, 5001431.0808),`
- `arcpy.Point(477710.8185, 4986587.1063)])`
- `polyline = arcpy.Polyline(array)`
- `cursor = arcpy.da.InsertCursor('d:/temp/lines.shp', ['SHAPE@'])`
- `cursor.insertRow([polyline])`



□(3)编辑要素

■ 方法

UpdateCursor (

in_table,

field_names,

where_clause,

spatial_reference,

explode_to_points,

sql_clause

)

#要素类或表

#字段列表, 如["field1", "field2"...]

#限制记录的条件, 默认None, 可选项

#空间参考, 默认None, 可选项

#是否要将要素分解为点集, 默认False, 可选项

#SQL的前/后缀, 默认None, 可选项

■ 示例

- with arcpy.da.UpdateCursor("d:/temp/test.shp", "name") as cur:
- ... for row in cur:
- ... row[0]="point_"+row[0];
- ... cur.updateRow(row);

□(3)编辑要素

■ 示例

- ... with arcpy.da.UpdateCursor ("d:/temp/test.shp",["name","FID"]) as cur:
- ... for row in cur:
- ... row[0] = "point_"+str(row[1]);
- ... cur.updateRow(row);
- ... del cur;



□游标和行的使用参见：

- <https://desktop.arcgis.com/zh-cn/arcmap/latest/analyze/arcpy-classes/cursor.htm>
- <https://desktop.arcgis.com/zh-cn/arcmap/latest/analyze/arcpy-classes/row.htm>

□(3)删除要素

■ 方法

- 使用arcpy.da.UpdateCursor访问数据以后，在其中使用deleteRow函数删除当前数据

■ 示例

- with arcpy.da.UpdateCursor("d:/temp/test.shp", "OID@") as cur:
- ... for row in cur:
- ... if row[0]==5:
- ... cur.deleteRow();

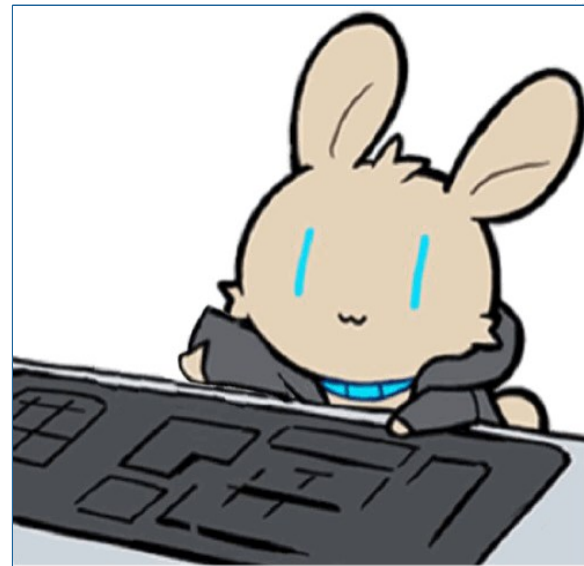
- with arcpy.da.UpdateCursor("d:/temp/test.shp", "OID@", where_clause="fid>6") as cur:
- ... for row in cur:
- ... cur.deleteRow();



- 使用InsertCursor函数，在(0,0)到(80,80)范围内随机生成10个点，存入points.shp，再将这些点按序生成线，存入line.shp文件。

```
import arcpy
import random
arcpy.env.workspace = "d:/temp/"
if arcpy.Exists("d:/temp/points.shp"):
    arcpy.Delete_management("d:/temp/points.shp");
if arcpy.Exists("d:/temp/line.shp"):
    arcpy.Delete_management("d:/temp/line.shp");
arcpy.CreateFeatureclass_management("d:/temp/", "points.shp", "POINT");
arcpy.CreateFeatureclass_management("d:/temp/", "line.shp", "POLYLINE");
cursor=arcpy.da.InsertCursor("d:/temp/points.shp", ['SHAPE@XY']);
arr = [];
for i in range(0,10):
    x = random.random()*80;
    y = random.random()*80;
    cursor.insertRow([(x,y)]);
    arr.append(arcpy.Point(x,y));
```

```
cursor =
arcpy.da.InsertCursor("d:/temp/line.shp", ['SHAPE@']);
arr = arcpy.Array(arr);
polyline = arcpy.Polyline(arr);
cursor.insertRow([polyline]);
```



4.3 查询要素



□ 根据属性查询要素方法

- `Cursor = arcpy.da.SearchCursor (`
 - `in_table,` #要素类或表
 - `field_names,` #字段列表, 如["field1", "field2"...]
 - `where_clause,` #查询记录的条件, 默认None, 可选项
 - `spatial_reference,` #空间参考, 默认None, 可选项
 - `explode_to_points,` #是否要将要素分解为点集, 默认False, 可选项
 - `sql_clause` #SQL的前/后缀, 默认None, 可选项



■ 示例:

```
import arcpy
cursor = arcpy.da.SearchCursor("d:/temp/test.shp",["name","SHAPE@WKT'],'fid>3');
for row in cursor:
...   print(u'Name:{0},WKT:{1}'.format(row[0],row[1]));
del cursor
```

□(1)选择要素SelectLayerByLocation方法

```
arcpy.management.SelectLayerByLocation(  
    in_layer,                #输入图层  
    overlap_type,            #空间关系, 可选项  
    select_features,         #输入要素, 可选项  
    search_distance,         #搜索距离目标的距离, 可选项  
    selection_type,          #选择结果是否用于输入, 可选项  
    invert_spatial_relationship #是否使用反转的空间关系 (Boolean),可选项  
)
```

□(1)选择要素SelectLayerByLocation方法的参数overlap_type:

- INTERSECT : 空间相交
- WITHIN_A_DISTANCE : 选择要素的指定距离内 (欧氏), 需用search_distance 指定距离
- COMPLETELY_CONTAINS : 完全包含选择要素
- CONTAINS_CLEMENTINI : 同CONTAINS, 但要素完全位于要素的边界上不会选择
- WITHIN : 要素位于选择要素之中, 包括在边界上
- COMPLETELY_WITHIN : 要素完全位于或包含在选择要素之内
- WITHIN_CLEMENTINI : 同CONTAINS_CLEMENTINI
- ARE_IDENTICAL_TO : 输入图层中的要素与选择要素相同 (在几何形状上)
- BOUNDARY_TOUCHES : 输入图层中要素的边界与选择要素接触
- SHARE_A_LINE_SEGMENT_WITH : 要素共线
- CROSSED_BY_THE_OUTLINE_OF : 要素与选择要素的轮廓交叉
- HAVE_THEIR_CENTER_IN : 要素的中心落在选择要素内

□(1)选择要素SelectLayerByLocation方法示例

- 在地图中绘制一个矩形框，选择其包含的所有点要素。
- ```
def onRectangle(self, rectangle_geometry):
```
- ```
    arcpy.env.workspace = "d:/CSU/" ;
```
- ```
 print str(rectangle_geometry);
```
- ```
    layer = arcpy.MakeFeatureLayer_management("points", 'lyr');
```
- ```
 pg = extent_to_polygon(rectangle_geometry);
```
- ```
    arcpy.SelectLayerByLocation_management("lyr", "INTERSECT", pg);
```
- ```
 matchcount = arcpy.GetCount_management("lyr").getOutput(0);
```
- ```
    print "The matched points count: "+matchcount;
```
- ```
 with arcpy.da.SearchCursor(layer, ["OID@", "SHAPE@WK"]) as cursor:
```
- ```
        for row in cursor:
```
- ```
 if str(row[0]) in arcpy.Describe("lyr").FIDSet:
```
- ```
                print u"FID:{0},SHAPE:{1}".format(row[0], row[1]);
```
- ```
 del cursor;
```
- ```
    pass
```



□(1)选择要素SelectLayerByLocation方法示例

- 将矩形框转换为一个多边形对象。

```
def extent_to_polygon(extent):  
    """  
    Args:  
        extent (arcpy.Extent)  
    Returns:  
        arcpy.Polygon  
    """  
    array = arcpy.Array()  
    array.add(extent.lowerLeft)  
    array.add(extent.lowerRight)  
    array.add(extent.upperRight)  
    array.add(extent.upperLeft)  
    array.add(extent.lowerLeft)  
    return arcpy.Polygon(array)
```

□(2)选择要素SelectLayerByAttribute方法:

- arcpy.management.SelectLayerByAttribute(
 in_layer_or_view, #图层或视图
 selection_type, #选择类型, 可选项
 where_clause #SQL表达式, 可选项
)

- selection_type参数
 - NEW_SELECTION : 生成新选择集, **默认设置**。
 - ADD_TO_SELECTION : 向存在的选择集添加到当前选择内容, 否则创建新的选择集
 - REMOVE_FROM_SELECTION: 从当前选择集中移除, 若不存在选择集, 忽略。
 - SUBSET_SELECTION : 在已有选择集中再次组合选择
 - SWITCH_SELECTION : 清空当前选择集, 把未选择的要素加入选择集
 - CLEAR_SELECTION : 清空当前选择集

□选择实例

- import arcpy
- arcpy.env.workspace = 'c:/data/mexico.gdb'
- *# Select all cities that overlap the chihuahua polygon*
- chihuahua_cities = arcpy.management.SelectLayerByLocation('cities', 'INTERSECT',
■ 'chihuahua', 0,
■ 'NEW_SELECTION')
- *# Within selected features, further select only those cities with a
■ # population > 10,000*
- arcpy.management.SelectLayerByAttribute(chihuahua_cities, 'SUBSET_SELECTION',
■ '"population" > 10000')
- # Write the selected features to a new feature class
- arcpy.management.CopyFeatures(chihuahua_cities, 'chihuahua_10000plus')



- 5.1 Spatial Analyst模块
- 5.2 地理处理工具
- 5.3 自定义地理处理工具

5.1 Spatial Analyst模块



- Spatial Analyst 模块 `arcpy.sa` 是用于分析栅格和矢量数据的模块，可访问 Spatial Analyst 工具箱中提供的所有地理处理工具以及其他函数和类。
- sa模块包含
 - 类：用于定义Spatial Analyst 工具的参数
 - 函数：用于处理和创建栅格或栅格对象
 - 地理处理函数： 对处理结果进行管理
 - 运算符： sa的运算符

5.1 Spatial Analyst模块



□sa示例：反距离加权插值

```
import arcpy
from arcpy import env
from arcpy.sa import *
env.workspace = "C:/sapyexamples/data"
outIDW = Idw("ozone_pts.shp", "ozone", 2000, 2, RadiusVariable(10, 150000))
outIDW.save("C:/sapyexamples/output/idwout.tif")
```



- 在 ArcPy 中，地理处理工具以两种方法进行组织。
 - (1) 所有工具都可以作为 arcpy 命名空间中的函数使用
 - 例如，`arcpy.GetCount_management(in_features)`
 - (2) 在与工具箱别名相匹配的模块中使用，工具箱的别名和工具名称紧随 arcpy 之后，中间用句点分隔。
 - 例如，在 Python 中，交集取反工具标识为 `arcpy.analysis.SymDiff`。

- (1)提取工具集：允许通过查询（SQL 表达式）或空间和属性提取操作来选择要素类或表中的要素和属性。输出要素和属性将存储于要素类或表中。

■ 裁剪(clip)

• 用法

```
arcpy.analysis.Clip(
```

```
    in_features,
```

```
#输入
```

```
    clip_features,
```

```
#裁剪要素
```

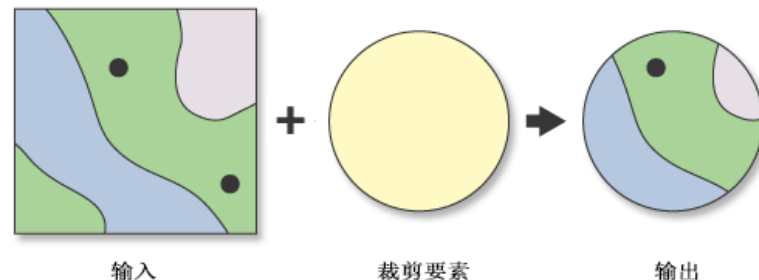
```
    out_feature_class,
```

```
#输出
```

```
    cluster_tolerance
```

```
#tolerance, 可选项
```

```
)
```



• 示例:

```
arcpy.env.workspace = "C:/data "  
_in = "in.shp"  
_clip = "clip.shp"  
_out = "out.shp"  
arcpy.analysis.Clip(_in, _clip, _out);
```


□ (1) 提取工具集

■ 选择(Select)

• 用法

```
arcpy.analysis.Select(  
    in_features,          #输入  
    out_feature_class,    #输出  
    where_clause           #查询条件, 可选项  
)
```

■ 示例

```
import arcpy  
arcpy.env.workspace = "C:/data"  
arcpy.analysis.Select("in.shp", "out.shp", '"CLASS" = \'4\'')
```

□ (1) 提取工具集

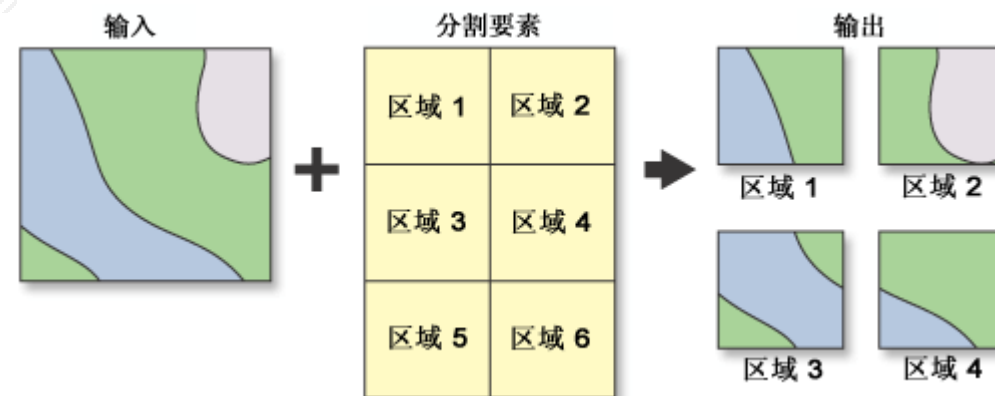
■ 分割(Split)

• 用法

```
arcpy.analysis.Split(  
    in_features,           #输入  
    split_features,        #分割要素  
    split_field,           #分割字段  
    out_workspace          #输出工作空间  
)
```

■ 示例

- `arcpy.env.workspace = "c:/data"`
- `arcpy.analysis.Split("in.shp", "split.shp", "Zone","C:/output/", "1 Meters")`



□(2)叠加分析工具集：包含的工具用于叠加多个要素类以合并、擦除、修改或更新空间要素，从而生成新要素类。

■ 联合/合并 (Union)

- `arcpy.analysis.Union(["in1.shp", "in2.shp", "in3.shp"], "out.shp", "NO_FID", 0.0003)`

■ 相交 (Intersect)

- `arcpy.analysis.Intersect(["in1.shp", "in2.shp", "in3.shp"], "out.shp", "ALL")`

■ 擦除

- `arcpy.analysis.Erase(in_features, erase_features, out_feature_class, {cluster_tolerance})`

■ 交集取反/对称差 (SymDiff)

- `arcpy.analysis.SymDiff(in_features, update_features, out_feature_class, {join_attributes}, {cluster_tolerance})`

□更多工具参见：<https://pro.arcgis.com/zh-cn/pro-app/latest/tool-reference/analysis/an-overview-of-the-analysis-toolbox.htm>

□ arcpy命名空间中的函数调用处理工具

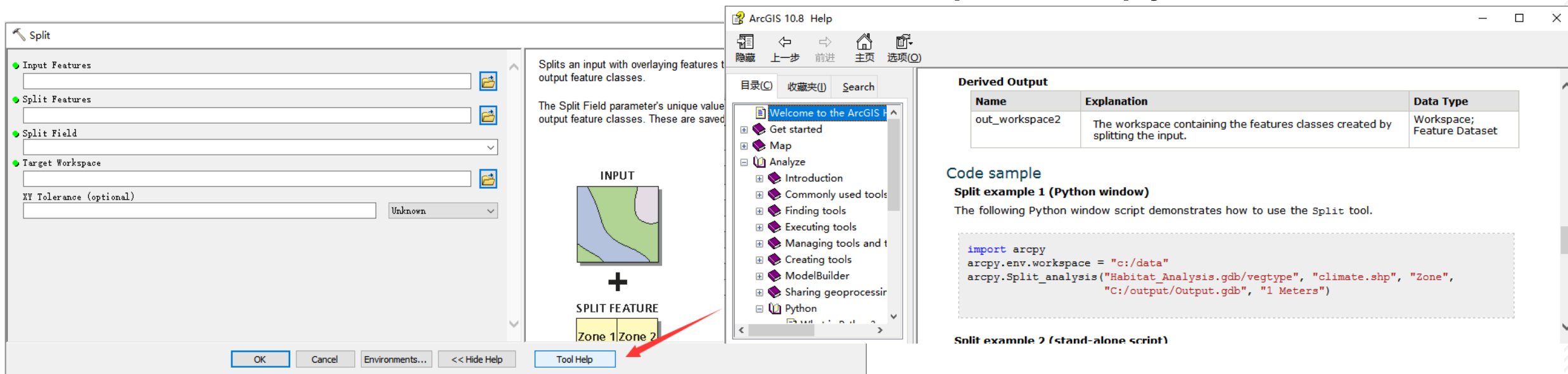
```
import arcpy
```

```
from arcpy import env
```

```
env.workspace = "C:/data"
```

```
arcpy.Clip_analysis("in.shp", "clip.shp", "output.shp")
```

□ 在ArcToolBox里打开工具，点击“Tool Help”查看python用法



The screenshot displays the ArcGIS 10.8 Split tool interface and its associated help window. The Split tool dialog box on the left includes fields for Input Features, Split Features, Split Field, and Target Workspace, along with an XY Tolerance (optional) dropdown. The help window on the right provides a description of the tool, a diagram illustrating the input and split features, and a code sample for using the Split tool.

Split tool description: Splits an input with overlaying features to output feature classes. The Split Field parameter's unique value output feature classes. These are saved.

Diagram: INPUT (a map with green and blue areas) + SPLIT FEATURE (a map with yellow and green areas) = Zone 1 Zone 2

Derived Output:

Name	Explanation	Data Type
out_workspace2	The workspace containing the features classes created by splitting the input.	Workspace; Feature Dataset

Code sample:

Split example 1 (Python window)

The following Python window script demonstrates how to use the Split tool.

```
import arcpy
arcpy.env.workspace = "c:/data"
arcpy.Split_analysis("Habitat_Analysis.gdb/vegtype", "climate.shp", "Zone", "C:/output/Output.gdb", "1 Meters")
```

Split example 2 (stand-alone script)

- 为ArcMap创建Python开发的Add-in ToolBar，添加相应按钮和工具，实现shp文件的创建，通过地图交互添加点要素，修改点要素字段值；图上点选要素，输出要素信息；将地图输出为PDF文件。

□ 参考代码

■ 创建要素类的代码

```
class BtnNewFeature(object):
```

```
    """Implementation for MyAddIn_addin.button_newfeatureclass (Button)"""
```

```
    def __init__(self):
```

```
        self.enabled = True
```

```
        self.checked = False
```

```
    def onClick(self):
```

```
        if arcpy.Exists("d:/temp/points.shp"):
```

```
            arcpy.Delete_management("d:/temp/points.shp");
```

```
            arcpy.CreateFeatureclass_management("d:/temp","points.shp","POINT");
```

```
            arcpy.AddField_management("d:/temp/points.shp","name","TEXT",20);
```

□ 参考代码

■ 绘制点工具代码

```
class ToolDrawPoint(object):
    """Implementation for MyAddIn_addin.tool_drawpoint (Tool)"""
    def __init__(self):
        self.enabled = True
        self.cursor = 3;
        self.checked = False;
        self.shape = "None" # Can set to "Line", "Circle" or "Rectangle"
        self.drawing = True;

    def onMouseDownMap(self, x, y, button, shift):
        if self.drawing == True:
            with arcpy.da.InsertCursor("d:/temp/points.shp",['SHAPE@XY']) as dc:
                dc.insertRow([(x,y)]);
                print "One point ({0},{1}) has been added.".format(x,y);
```

□ 参考代码

- 标识 (Identify) 点工具代码
- class ToolIdentify(object):
- def onMouseDownMap(self, x, y, button, shift):
- mxd = arcpy.mapping.MapDocument("current");
- df = mxd.activeDataFrame;
- lyr = arcpy.mapping.ListLayers("points", "*", df)[0];
- pt = arcpy.Point(x, y);
- geom = arcpy.PointGeometry(pt)
- arcpy.SelectLayerByLocation_management("points", "WITHIN_A_DISTANCE" ,
- geom, 2.5);
- matchcount = arcpy.GetCount_management("points").getOutput(0);
- print "The matched points count: "+matchcount;
- with arcpy.da.SearchCursor(lyr, ["OID@", "SHAPE@WKTEXT"]) as cursor:
- for row in cursor:
- if str(row[0]) in arcpy.Describe("points").FIDSet:
- print u"FID:{0},SHAPE:{1}".format(row[0], row[1]);
- del cursor;
- pass

- 1.面向ArcGIS的Python基础
- 2.管理地图文档和图层
- 3.访问空间数据
- 4.要素管理
- 5.地理处理工具

开发实例

- 1. 基于POI的城市功能划分
- 2. 基于管点的管线自动构建
- 3. Add-in交互操作实例

1. 基于POI的城市功能划分



□ 本实例的开发目的

- 本实例使用POI点密度的不同，将城市区域划分不同的功能区域
- 本实例使用两种方法
 - 空间DBSCAN聚类
 - 网格剖分法

■ 本实例数据来源

- 工作区为长沙市岳麓区部分区域
- POI数据来自互联网
- POI点数

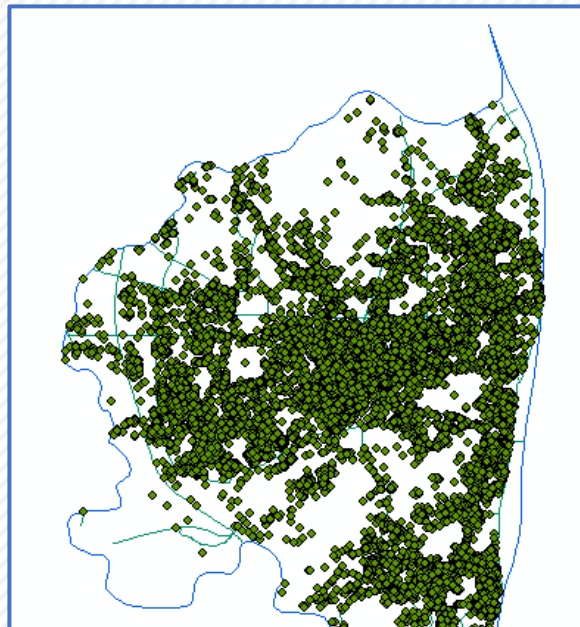
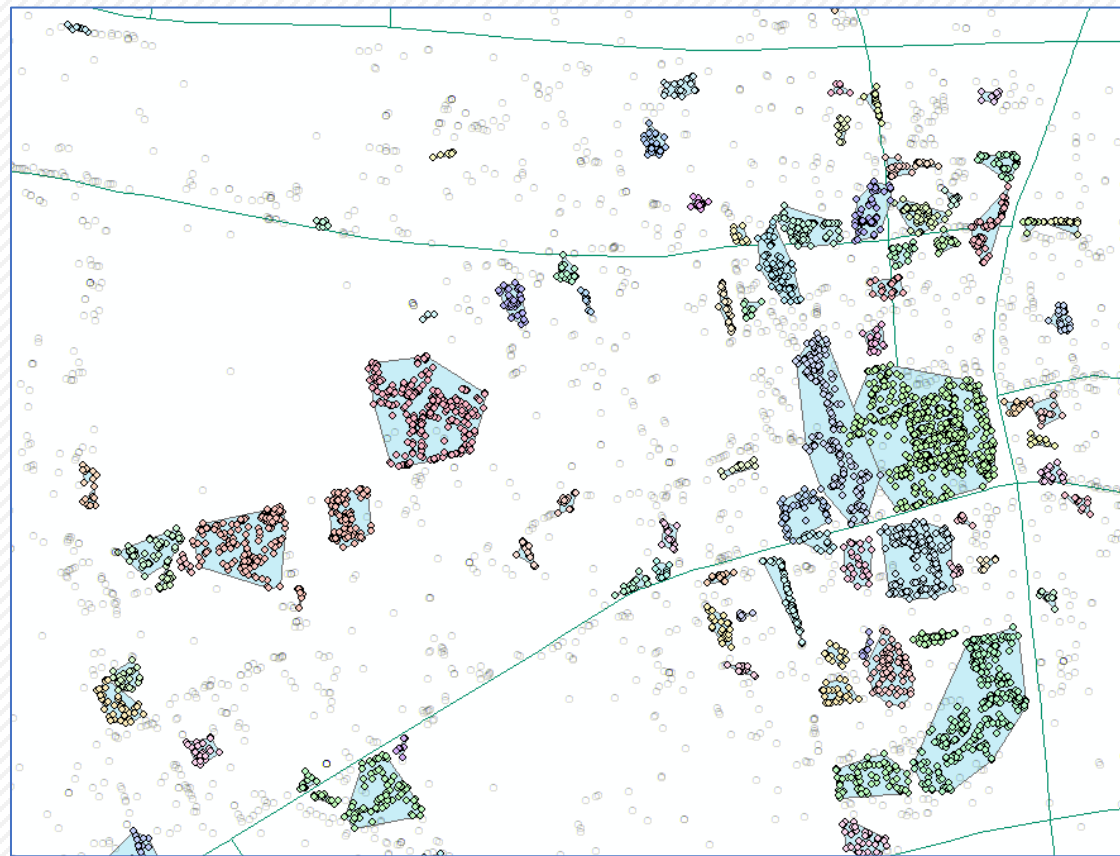
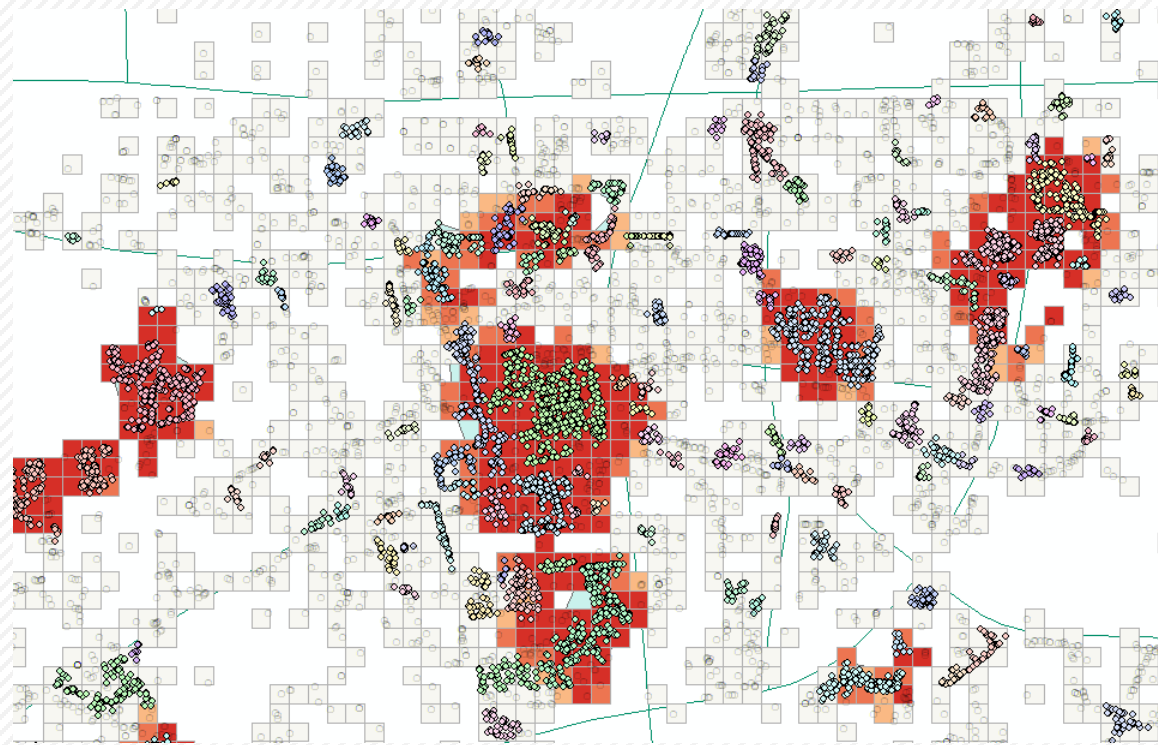


Table								
poi_yuelu								
FID	Shape *	id	catalog1	catalog2	catalog3	lon	lat	
0	Point	143557	商务住宅:住宅区:住宅小区			112.925545	28.194021	
1	Point	148608	政府机构及社会团体	公检法机构	社会治安机构	112.899117	28.243055	
2	Point	91915	商务住宅:住宅区:宿舍			112.866287	28.199429	
3	Point	91923	科教文化服务	学校	小学	112.858482	28.213705	
4	Point	91928	体育休闲服务	运动场馆	综合体育馆	112.861618	28.198032	
5	Point	91930	科教文化服务	学校	高等院校	112.862976	28.200842	
6	Point	92032	公司企业	公司	网络科技	112.870522	28.231594	
7	Point	92068	科教文化服务	科教文化场所	科教文化场所	112.869606	28.202366	
8	Point	91784	公司企业	公司	网络科技	112.871811	28.236467	
9	Point	15962	体育休闲服务	体育休闲服务场	体育休闲服务场	112.95874	28.232634	
10	Point	15968	生活服务	美容美发店	美容美发店	112.914902	28.206495	
11	Point	11959	餐饮服务	中餐厅	中餐厅	112.943596	28.17075	
12	Point	11997	科教文化服务	培训机构	培训机构	112.928802	28.148424	
13	Point	12068	餐饮服务	中餐厅	中餐厅	112.943314	28.149937	
14	Point	92281	科教文化服务	传媒机构	传媒机构	112.868462	28.222641	

- ❑ (1) 读取POI坐标，存入数组X，格式为[[x0,y0],[x1,y1]...]
- ❑ (2) 将X转换为numpy数组
- ❑ (3) 使用sklearn.DBSCAN
- ❑ (4) 将聚类id写入poi字段cid
- ❑ (5) 根据poi的cid生成各类簇的最小约束多边形MBG



- (1)生成格网 (鱼网)
- (2)使用空间联接方法计算每个格网中的poi数量
- (3)使用自然分类法, 将poi数量划分为若干个组
- (4)为每个格网赋分组号
- (5)在arcmap中渲染
- (6)利用arccgis自带的hotarea分析方法, 并对比



- (1)创建管线points.shp和lines.shp
- (2)读取管点excel文件，创建管点要素
- (3)读取管线excel文件，根据起止点编号，获取起止点几何图形，生成管线要素。

	A	B	C	D	E	F	G	H	I	J	K	L
1	ExpNo	MapNo	X	Y	SurfH	WDeep	PCode	Subsid	Feature	Mtype	Msize	Wdepth
2	WS304	WS304	12861	11 3402	128.284	146.25	142.15	WS	检修井	圆井	半径0.35米	4.1
3	WS001	WS001	12861	43 3402	128.291	145.83	141.12	WS	检修井	圆井	半径0.35米	4.71
4	WS022	WS022	12861	44 3400	127.341	133.07	129.19	WS	检修井	圆井	半径0.35米	3.88
5	WS023	WS023	12861	81 3400	127.401	134.42	130.5	WS	检修井	圆井	半径0.35米	3.92
6	WS024	WS024	12861	3.4 3400	127.611	136.65	132.8	WS	检修井	圆井	半径0.35米	3.85
7	WS025	WS025	12861	61 3400	127.761	137.97	133.92	WS	检修井	圆井	半径0.35米	4.05
8	YS073	YS073	12861	99 3400	127.561	139.39	135.59	WS	检修井	圆井	半径0.35米	
9	WS026	WS026	12861	49 3400	127.191	140.51	136.81	WS	检修井			
10	WS028	WS028	12861	1.6 3400	127.281	142.38	138.63	WS	检修井			
11	YS075	YS075	12861	79 3400	127.311	142.91	139.03	WS	检修井			
12	WS030	WS030	12861	04 3400	127.351	143.4	139.53	WS	检修井			
13	WS031	WS031	12861	85 3400	127.071	143.7	140.22	WS	检修井			
14	WS032	WS032	12861	3.9 3400	127.631	145.25	142.02	WS	检修井			
15	WS034	WS034	12862	7.78 3400	127.281	146.49	143.22	WS	检修井			
16	WS035	WS035	12862	1.57 3400	127.141	146.93	143.58	WS	检修井			
17	YS076	YS076	12862	1.59 3400	127.981	147.55	144.15	WS	检修井			
18	WS036	WS036	12862	1.65 3400	127.411	147.85	144.47	WS	检修井			

A	B	C	D	E	F	G	H	I	J	K
QID	S Point	E Point	S Deep	E Deep	S H	E H	DType	Material	DSize	Memo
40	WS116	WS117	3.56	3.41	138.84	138.57	直埋	混凝土	Φ400mm	支管
41	WS107	WS115	4.1	3.42	138.74	138.52	直埋	混凝土	Φ400mm	
42	WS115	WS114	3.96	3.9	138.52	138.46	直埋	混凝土	Φ800mm	
43	WS114	WS113	3.9	3.85	138.46	138.29	直埋	混凝土	Φ800mm	
44	WS113	WS112	3.85	2.98	138.29	138.1	直埋	混凝土	Φ800mm	
45	WS112	YS164	2.98	3.81	138.1	138.07	直埋	混凝土	Φ800mm	
46	YS164		3.81		138.07		直埋	混凝土	Φ800mm	终点
47	WS048	WS049	3.02	2.77	139.66	139.47	直埋	混凝土	Φ600mm	
48	WS049	WS050	2.77	2.81	139.47	139.23	直埋	混凝土	Φ600mm	
49	WS050	WS051	2.81	2.77	139.23	139	直埋	混凝土	Φ600mm	
50	WS051	WS052	2.77	2.55	139	138.72	直埋	混凝土	Φ600mm	
51	WS052	WS053	2.55	2.28	138.72	138.47	直埋	混凝土	Φ600mm	
52	WS053	YS078	2.28	2.23	138.47	138.26	直埋	混凝土	Φ600mm	
53	YS078	YS080	2.23	1.99	138.26	137.85	直埋	混凝土	Φ600mm	
54	YS080	WS054	1.99	1.9	137.85	137.77	直埋	混凝土	Φ600mm	
55	WS054	WS081	1.9	2.02	137.77	137.52	直埋	混凝土	Φ600mm	

- 为ArcMap创建Add-in工具栏，在其中添加按钮或工具，实现以下功能：
 - (1) 添加类名为 “BtnNewFeatureClass” 的按钮，标题为 “New Feature Class” ， 点击该按钮实现：
 - ① 检查c:\data目录下是否存在Points.shp文件，如果存在则删除之；
 - ② 创建点状Points.shp文件，并添加 “name” 字段（类型Text长度20）。
 - (2)添加类名为 “ToolDrawPoint” 的工具，标题为 “Draw Point” ， 点击该工具实现：
 - ① 在地图上点击，向Points.shp文件中添加一个点要素。
 - (3)添加类名为 “BtnEditFeature” 的按钮，标题为 “Edit Feature” ， 点击该按钮实现：
 - ① 将Points.shp中的所有要素的name值更改为 “Point_” +fid值，例如fid=0的点name值为 “Point_0” 。
 - (4)添加类名为 “ToolIdentify” 的工具，标题为 “Identify” ， 点击该工具实现：
 - ① 在地图上点选Points.shp的点要素，弹窗显示点的fid和name值。
 - (5)添加类名为 “ToolRemove” 的工具，标题为 “Remove Poins ”， 点击该工具实现：
 - ① 在图上绘制矩形框，删除在矩形框中的点要素



中南大學
CENTRAL SOUTH UNIVERSITY



再见!

THANKS FOR ALL