Lekcia 4

Dávkové spracovanie pomocou ArcPy Listing

Cvičenie 3

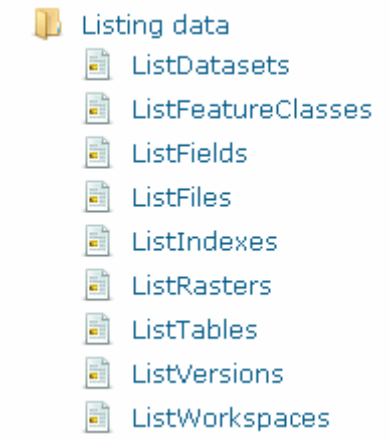
Funkcia **ListFeatureClasses**

- vytvorí zoznam tried objektov v pracovnom prostredí (v adresári alebo v geodatabáze)

```
Python
>>> fclist = arcpy.ListFeatureClasses()
>>> print fclist
[u'ClimateZones', u'Elev_LT_250', u'MajorRoads',
 u'Slope_LT_40', u'Vegetation', u'StudyArea']

>>>
```

- celkovo 12 listovacích funkcií

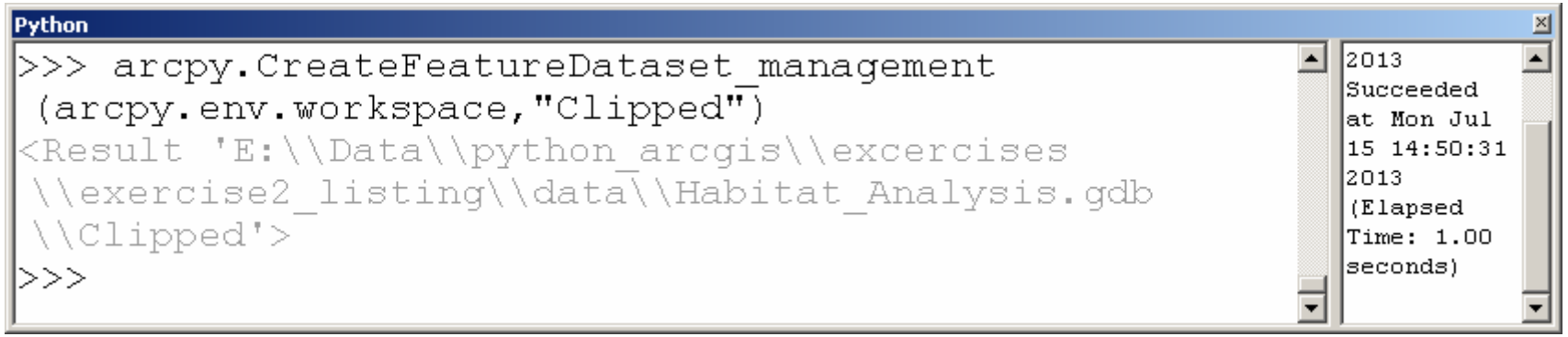


Nástroj
CreateFeatureDataset_management

- vytvorí nový Feature Dataset
- parametre: output dataset path, output name

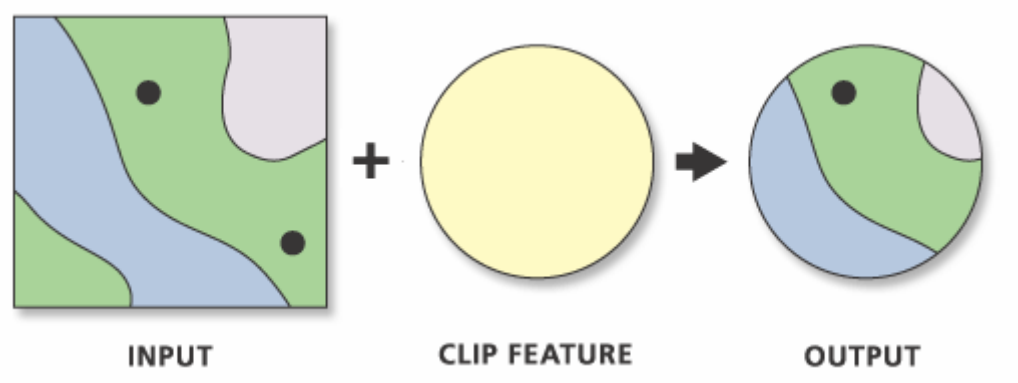
Nástroj
CreateFeatureDataset_management

- vytvorí nový Feature Dataset
- parametre: output dataset path, output name



Funkcia **Clip_analysis**

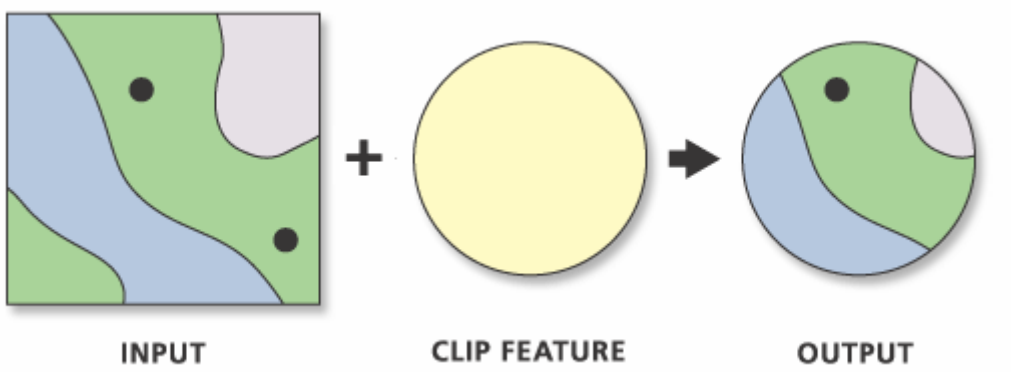
- parametre: input, clip features, output



- dávkové orezanie – pomocou listovacích funkcií Python-u (cyklus **for**)

Funkcia **Clip_analysis**

- parametre: input, clip features, output



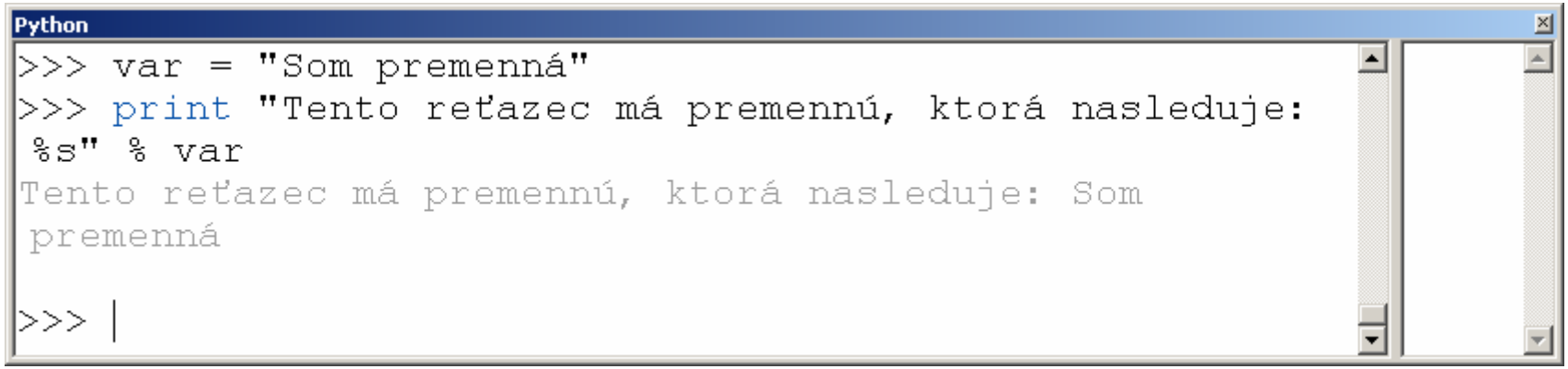
- dávkové orezanie – pomocou listovacích funkcií Python-u (cyklus **for**)

```
Python
>>> for fc in fclist:
...     if fc != "Study Area":
...         arcpy.Clip_analysis(fc,"Study
 Area","Clipped/%s_clip" % fc)
...
>>>
```

```
Time:
Mon Jul
15
15:03:24
2013
Reading
Features
...
Cracking
```

Formátovanie reťazcov v ArcPy

- v ArcPy sa používa zástupný znak pre premennú **%s**
- tento reťazec v Python-e volá funkciu **str()**, ktorá (nasilu) konvertuje objekt na textový reťazec, ktorý sa dá „ľahko vytlačiť“
- podobne pre konverziu na *(signed) integer* sa používa zástupný znak **%i**

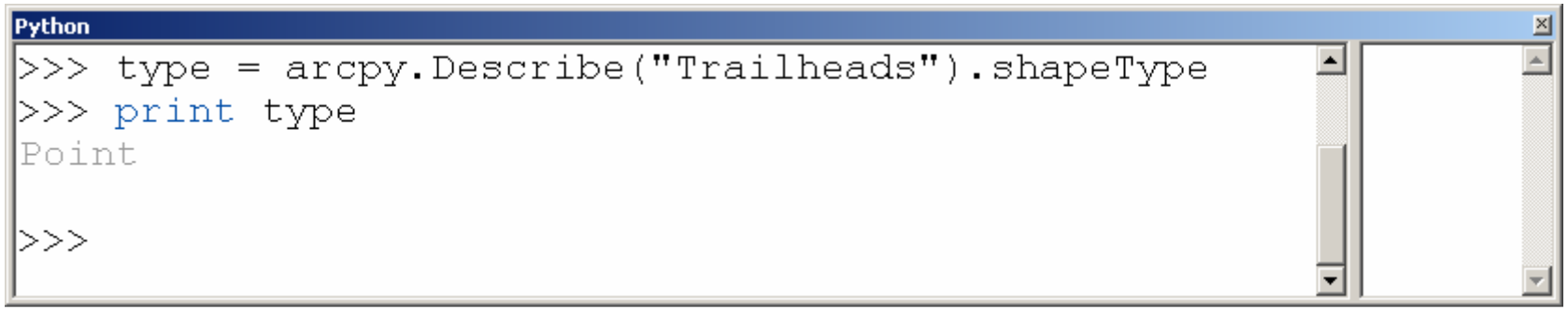


ArcPy kurzory

Cvičenie 4

Funkcia **Describe**

- funkcia **Describe()** vracia vlastnosti určitých druhov dát ako sú tabuľky, triedy objektov, geodatabázy, coverage, layer súbory, relačné triedy, pracovné prostredia (workspace) a sady údajov (dataset), takisto ako geoprocesných objektov ako sú FeatureLayers a TableViews
- vlastnosti: typ údajov, typ geometrie (bod, línia, polygón), názov poľa OID, názov poľa shape, ... atď.



Výpis názvov atribútov

- **arcpy.Describe().fields** – vracia pole objektov fields (stĺpce tabuľky)
- **arcpy.ListFields()** – detto
- tieto objekty majú vlastnosti name, type, length, precision, scale, aliasName, domain...atď.

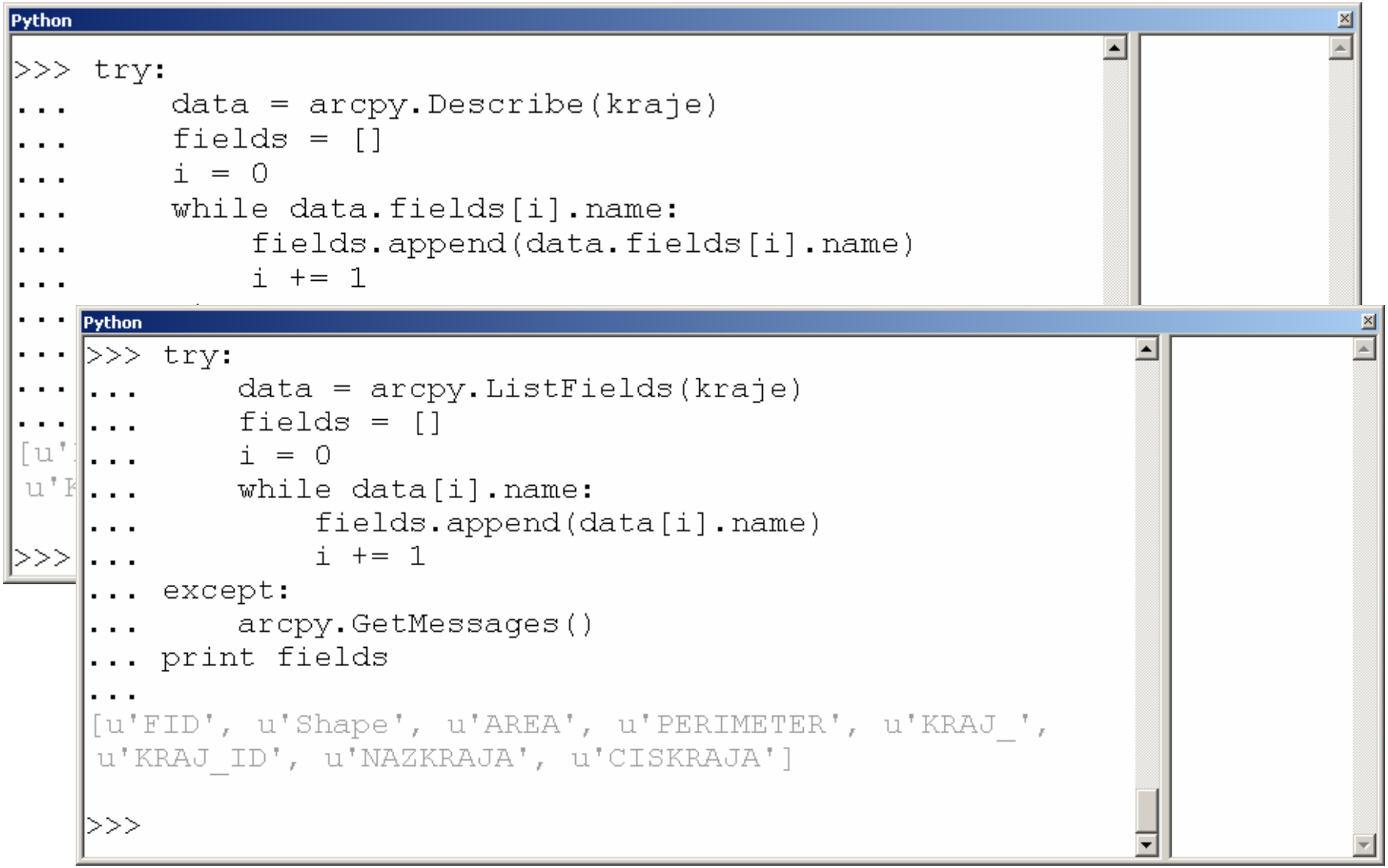
Výpis názvov atribútov

Python

```
>>> try:
...     data = arcpy.Describe(kraje)
...     fields = []
...     i = 0
...     while data.fields[i].name:
...         fields.append(data.fields[i].name)
...         i += 1
... except:
...     arcpy.GetMessages()
... print fields
...
[u'FID', u'Shape', u'AREA', u'PERIMETER', u'KRAJ_',
 u'KRAJ_ID', u'NAZKRAJA', u'CISKRAJA']

>>>
```

Výpis názvov atribútov

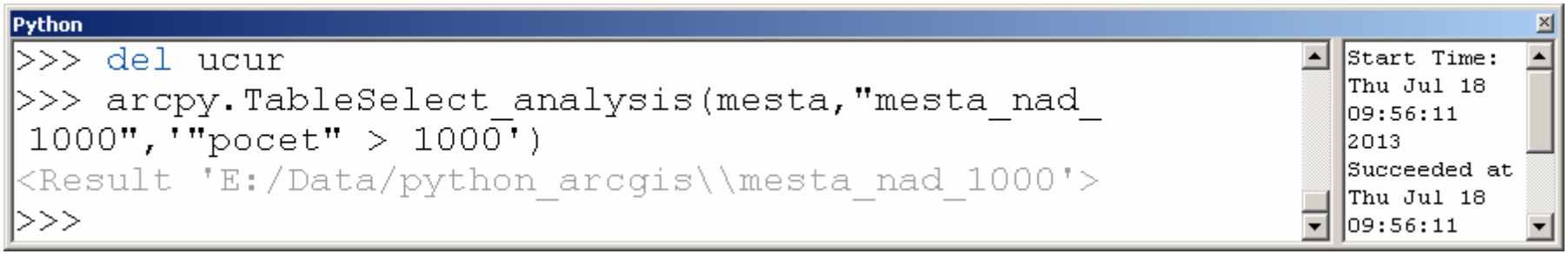


Atribútové dopyty

- nástroje **Select_analysis**, **TableSelect_analysis**
- parametre: input table / feature class
  output table / feature class
  where clause
- where pravidlo musí byť uzavreté v ’… ’
- názov poľa v shapefile ”...”, v personálnej gdb [...]

Atribútové dopyty

- nástroje **Select_analysis**, **TableSelect_analysis**
- parametre: input table / feature class
  output table / feature class
  where clause
- where pravidlo musí byť uzavreté v '… '
- názov poľa v shapefile "...", v personálnej gdb [...]



ArcPy kurzory

- používajú sa na prístup k záznamom tabuľky:
- prechádzajú cez jednotlivé záznamy
- získavajú hodnoty atribútov z tabuliek, tried objektov, rastrov
- získavajú geometriu objektov
- funkcie:

  SearchCursor – *read-only* prístup

  UpdateCursor – aktualizuje alebo vymazáva hodnoty atribútov alebo geometriu (*write* prístup)

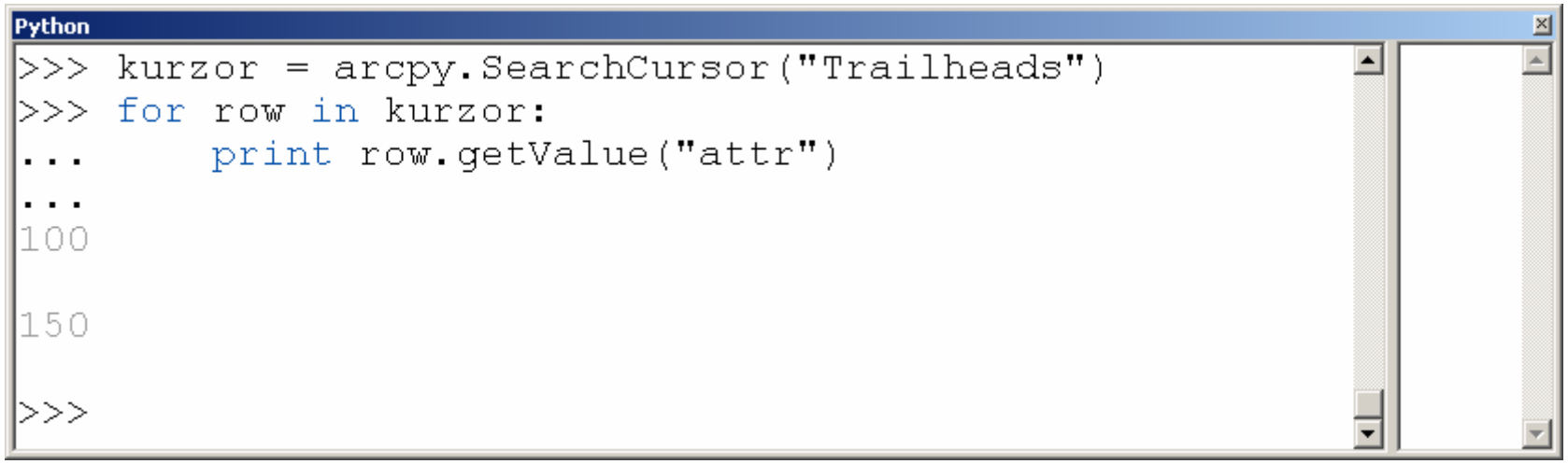
  InsertCursor – pridáva nové záznamy do tabuľky , zapisuje hodnoty atribútov a geometriu

Funkcie **getValue**, **setValue**

- používajú sa na získanie alebo zapísanie hodnoty atribútov
- parametre: fieldname, object (setValue)

SearchCursor

- **SearchCursor**



- parametre: dataset, {where clause}, {fields}...

SearchCursor

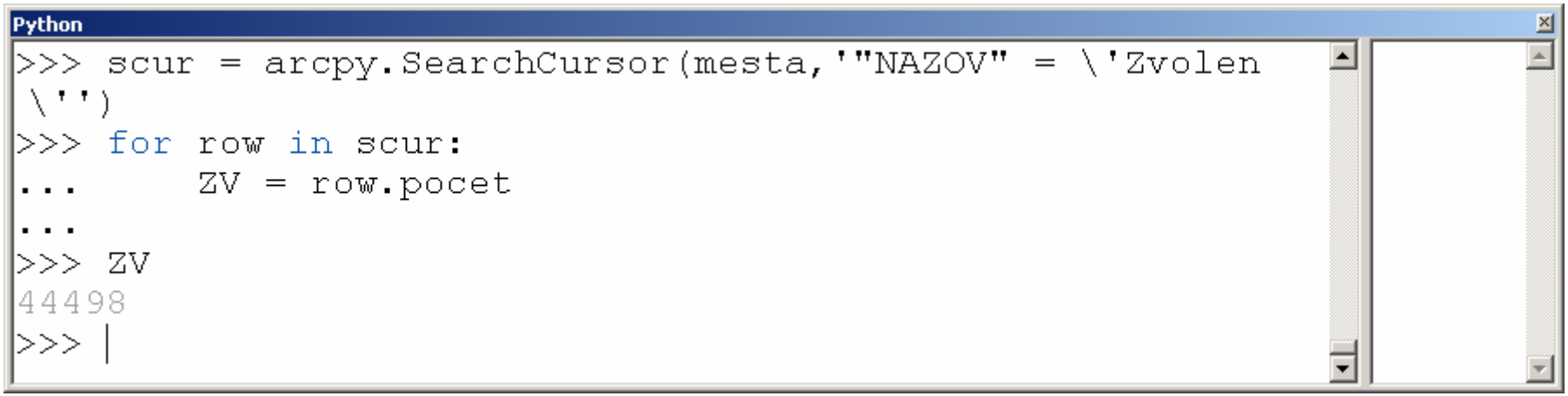
- **SearchCursor**

```
Python
>>> scur = arcpy.SearchCursor(mesta,'"NAZOV" = \'Zvolen
 \'')
>>> for row in scur:
...     ZV = row.getValue("pocet")
...
>>> ZV
44498
>>>
```

```
Python
>>> scur = arcpy.SearchCursor(mesta)
>>> for row in scur:
...     pocet = row.getValue("pocet")
...     if pocet > ZV:
...         print "%s - %i" % (row.getValue
 ("NAZOV"),pocet)
...
Banská Bystrica - 85052
```

SearchCursor

- **SearchCursor**



UpdateCursor

- **UpdateCursor**

Python

```
>>> arcpy.AddField_management("Trailheads","Log","FLOAT")
<Result 'Trailheads'>
>>> kurzor = arcpy.UpdateCursor("Trailheads")
>>> for row in kurzor:
...     log = math.log(row.getValue("attr"))
...     row.setValue("Log",log)
...     kurzor.updateRow(row)
...
```

Table

Trailheads

FID	Shape	attr	Log
0	Point	100	4.605
1	Point	150	5.010

1 (0 out of 2 Selected)

Trailheads

UpdateCursor

- **UpdateCursor**

```
Python
>>> arcpy.TableSelect_analysis(mesta,"mesta_nad_
 1000",'"ludia" > 1000')
Runtime error <class 'arcgisscripting.ExecuteError'>: ERROR
 999999: Error executing function. An invalid SQL statement
 was used. An invalid SQL statement was used. Failed to
 execute (TableSelect).
>>> arcpy.AddField_management(mesta,"pocet","LONG")
<Result 'E:\\Data\\python_arcgis\\mesta_sk_kr.shp'>
>>>
```

```
Succeeded at Thu Jul 18 09:37:01 2013 (Elapsed Time: 0.00 seconds)
```

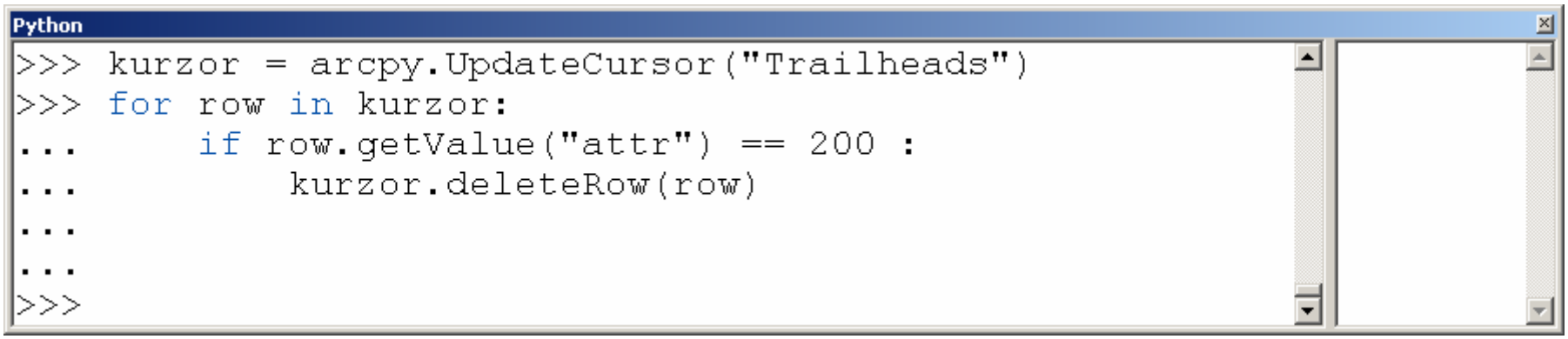
```
Python
>>> ucur = arcpy.UpdateCursor(mesta)
>>> for row in ucur:
...     pocet = row.getValue("ludia")
...     row.setValue("pocet",pocet)
...     ucur.updateRow(row)
...
>>>
```

Metódy kurzorov

- všetky kurzory majú metódu **next**, ktorá získava nasledujúci záznam
- metódy kurzora **Update**:
- **updateRow** – aktualizuje príslušný riadok
- **deleteRow** – odstráni riadok z tabuľky
- metódy kurzora **Insert**:
- **newRow** – vytvorí prázdny riadok (prvý krok)
- **insertRow** – vloží riadok do tabuľky (druhý krok)

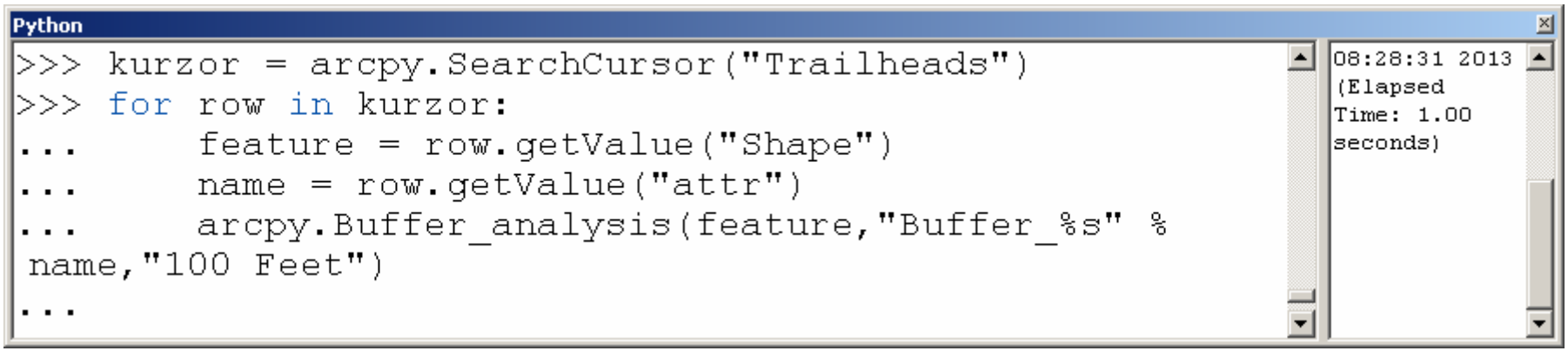
UpdateCursor

- vymazanie objektu (záznamu z tabuľky)



Čítanie geometrie

- geometrické pole (obyčajne Shape) vracia geometrický objekt, ktorý sa dá použiť pri geoprocessingu namiesto triedy objektov

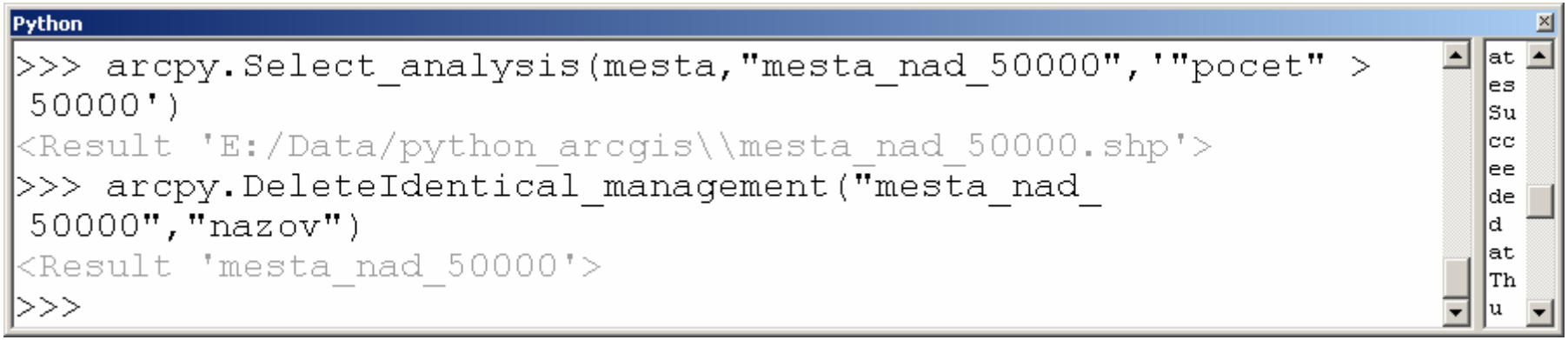


Práca s geometrickými objektmi

- geometrický objekt má vlastnosti: type, area, length, centroid, extent, first point, last point...atď.
- geometrický objekt má metódu **getPart**, ktorá vracia pole bodov objektu pre časť geometrie alebo pole obsahujúce polia, každé pre jednu časť

Práca s geometrickými objektmi

- geometrický objekt má vlastnosti: type, area, length, centroid, extent, first point, last point...atď.
- geometrický objekt má metódu **getPart**, ktorá vracia pole bodov objektu pre časť geometrie alebo pole obsahujúce polia, každé pre jednu časť



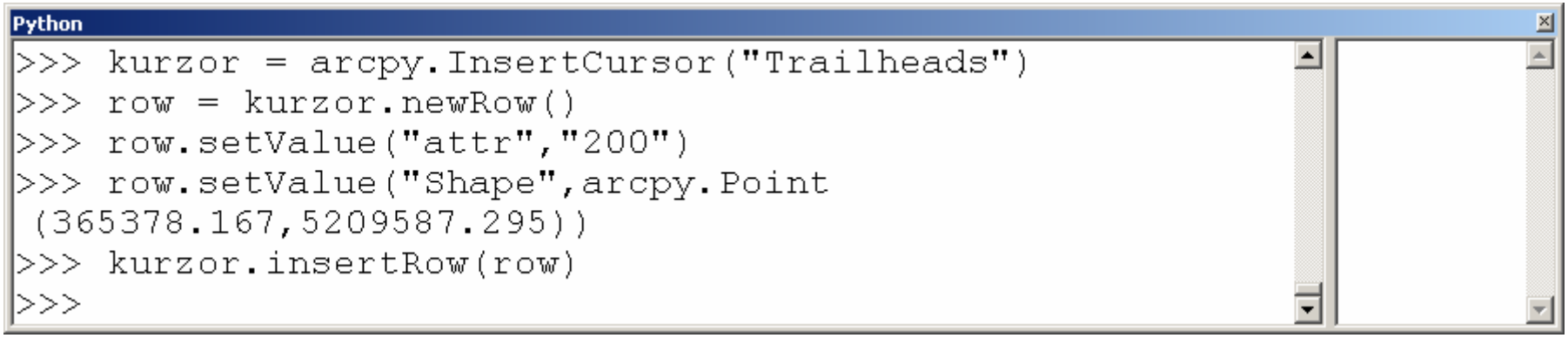
Práca s geometrickými objektmi

Python

```
>>> scur = arcpy.SearchCursor("mesta_nad_50000")
>>> mestaShapename = arcpy.Describe("mesta_nad_
 50000").shapeFieldName
>>> mestaShapename
u'Shape'
>>> pole = []
>>> for row in scur:
...     objekt = row.getValue(mestaShapename)
...     geom = objekt.getPart()
...     pole.append(geom)
...
>>> pole
[<Point (-417485.923709, -1228783.89083, #, #)>, <Point (-
 430523.172864, -1190623.72062, #, #)>, <Point (-501096.996457,
 -1326122.0112, #, #)>, <Point (-331005.180124, -1199269.91181,
```

InsertCursor

- **InsertCursor**
- vkladá hodnoty do atribútových polí
- vkladá geometriu do geometrického poľa



- funkcia **Point** sa používa na tvorbu bodových objektov (voliteľné parametre X, Y, M, Z, ID)

Zápis hodnôt do zoznamu

- vytvorenie prázdneho zoznamu:

  zoznam = []

  zoznam = list()
- zápis hodnôt do poľa:

  zoznam.append()

Zápis hodnôt do zoznamu

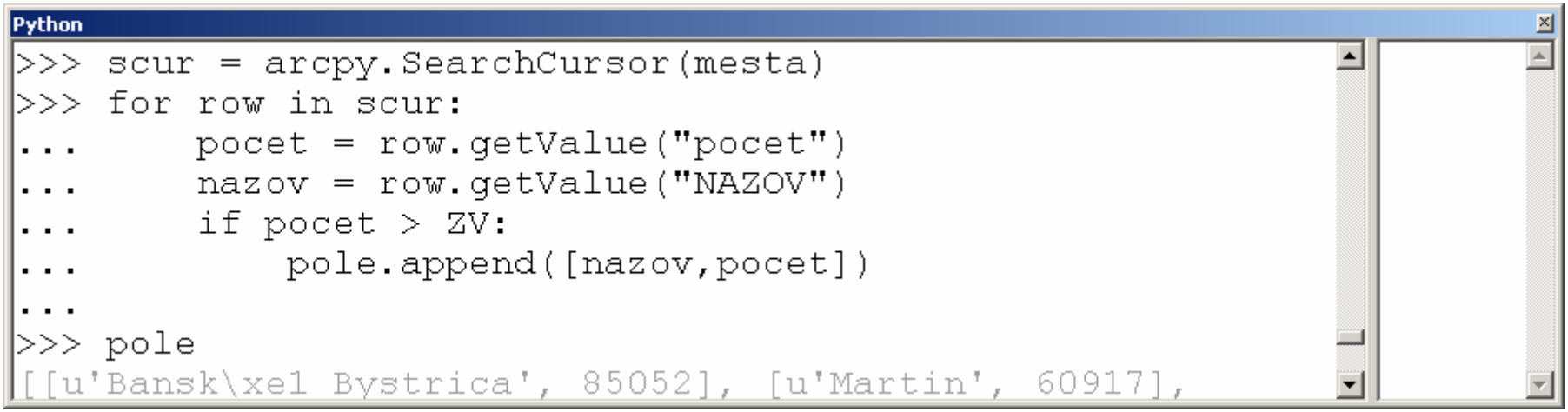
- vytvorenie prázdneho zoznamu:

  zoznam = []

  zoznam = list()

- zápis hodnôt do poľa:

  zoznam.append()

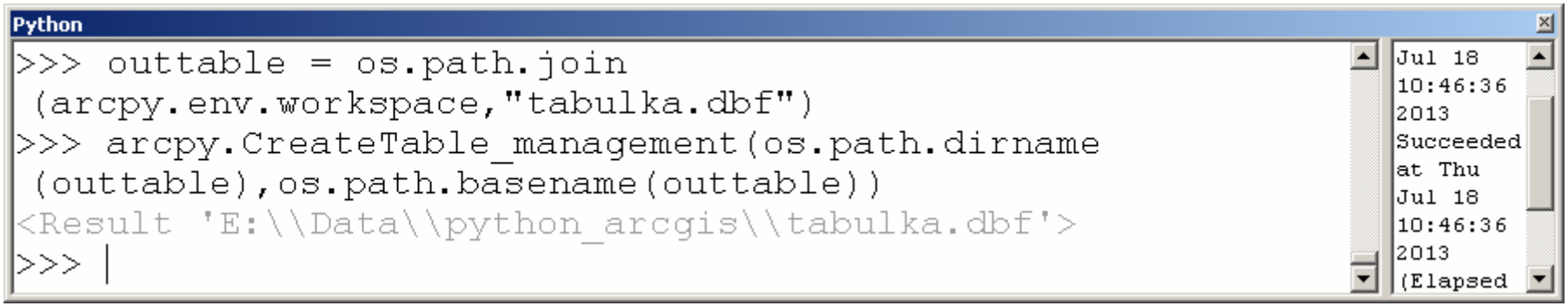


Vytvorenie tabuľky

- nástroj **CreateTable_management**
- parametre: output path, output name

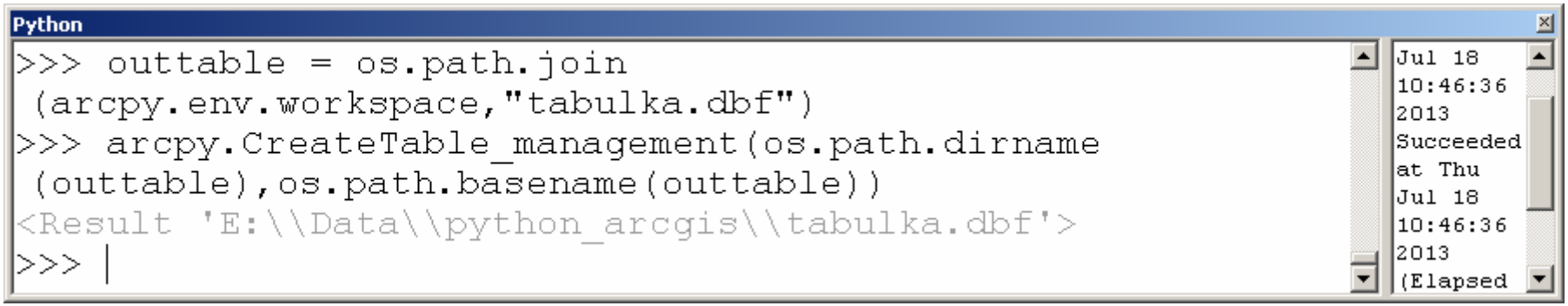
Vytvorenie tabuľky

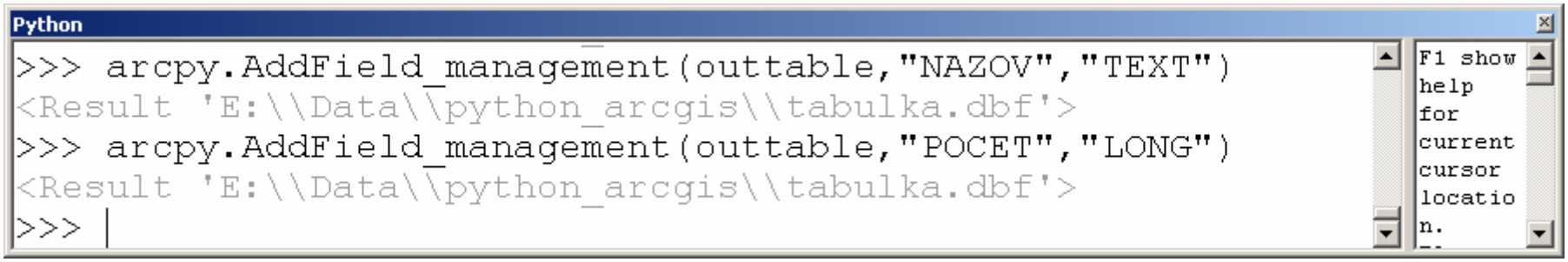
- nástroj **CreateTable_management**
- parametre: output path, output name



Vytvorenie tabuľky

- nástroj **CreateTable_management**
- parametre: output path, output name





Vyplnenie tabuľky hodnotami

Python

```
>>> icur = arcpy.InsertCursor(outtable)
>>> for arow in pole:
...     row = icur.newRow()
...     row.setValue("NAZOV",arow[0])
...     row.setValue("POCET",arow[1])
...     icur.insertRow(row)
...
>>>
```

Table

tabulka

OID	Field1	NAZOV	POCET
0	0	Banská Bystrica	85052
1	0	Martin	60917
2	0	Nitra	87569
3	0	Nitra	87569
4	0	Nitra	87569
5	0	Poprad	55303
6	0	Poprad	55303

1 (0 out of 14 Selected)

mesta_nad_1000 tabulka

Vymazanie identických záznamov

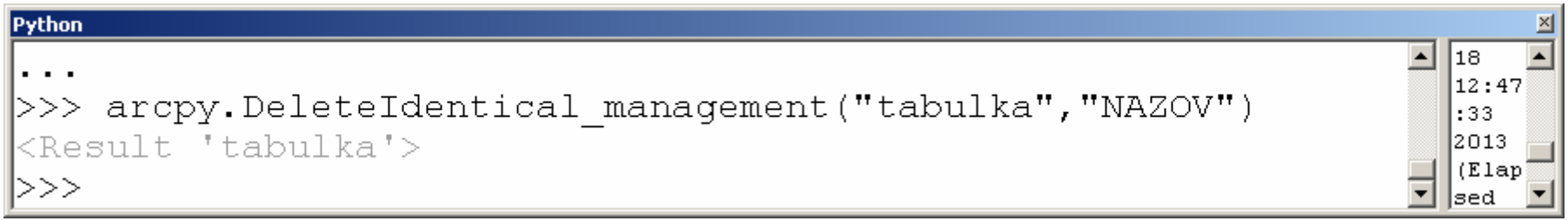
- nástroj **DeleteIdentical_management**
- parametre: input table / feature class
  field(s)
  {x,y tolerance}, {z tolerance}

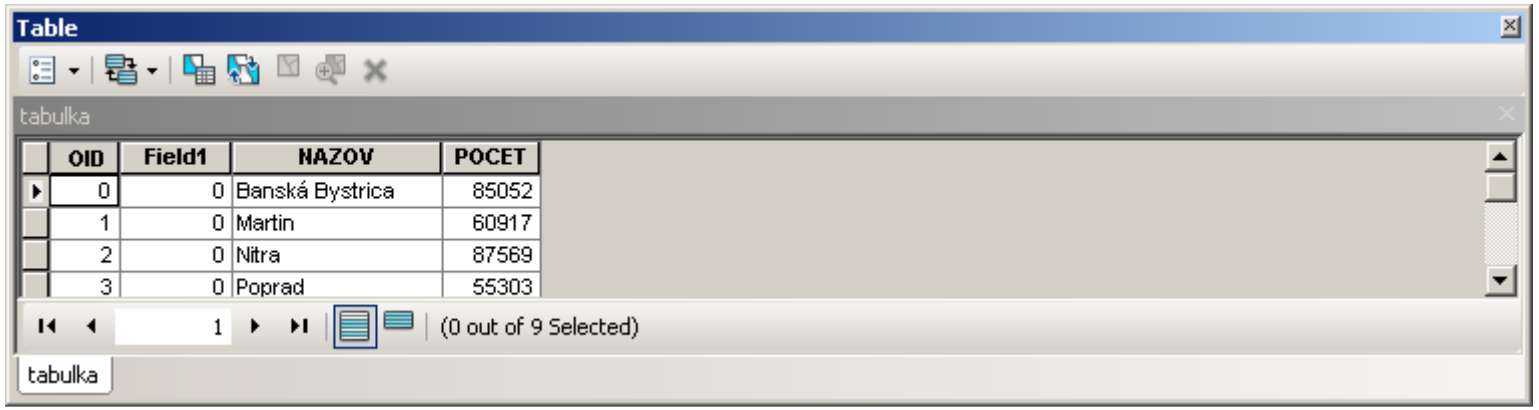
Vymazanie identických záznamov

- nástroj **DeleteIdentical_management**
- parametre: input table / feature class

  field(s)

  {x,y tolerance}, {z tolerance}



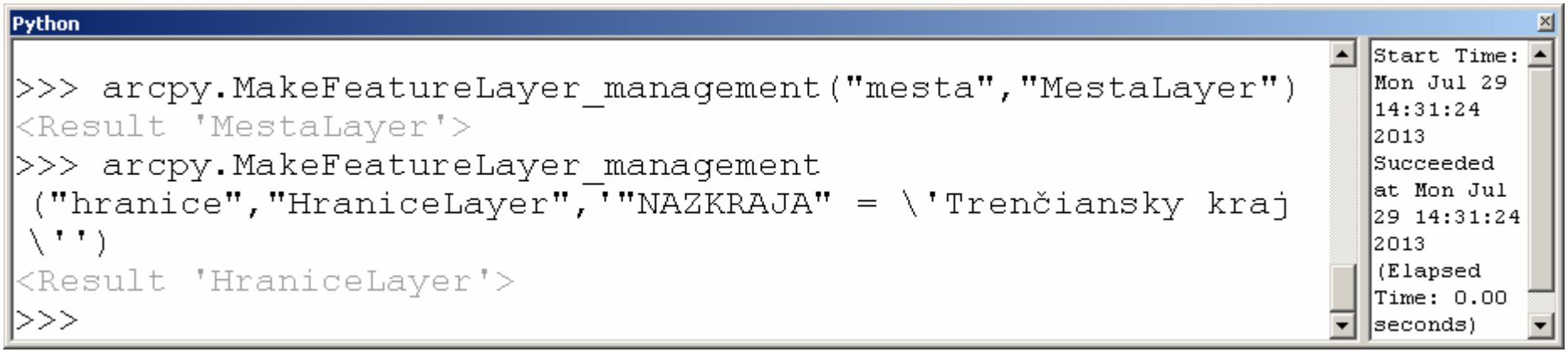


Vytvorenie dočasnej vrstvy

- nástroj **MakeFeatureLayer_management()**
- parametre: input features

  output layer

  {where clause}
- výstupná vrstva sa dá použiť ako vstup do všetkých nástrojov, ktoré akceptujú feature layer ako vstup

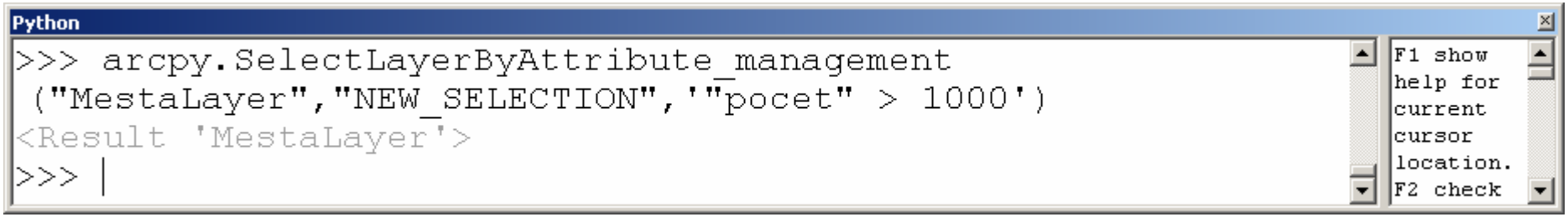


Atribútové dopyty

- nástroj **SelectLayerByAttribute_management()**
- parametre: input layer / table view

  {selection type}

  {where clause}
- na rozdiel od **Select_analysis** nevytvára novú vrstvu, iba robí výber (pridáva k výberu, maže z výberu, vyberá z výberu, robí obrátený výber...)



Priestorové dopyty

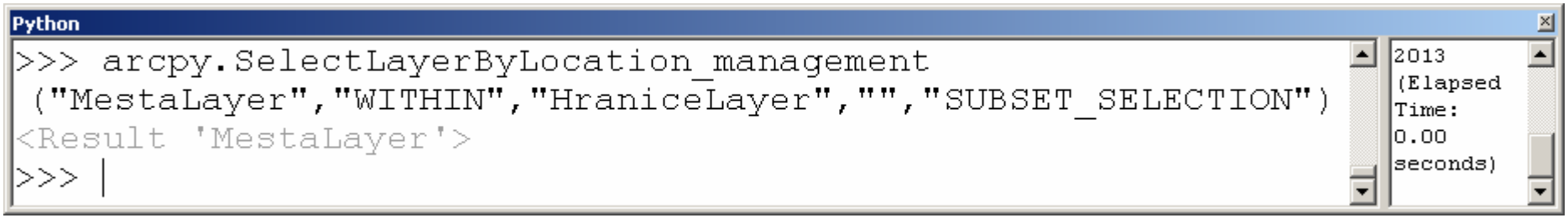
- nástroj **SelectLayerByLocation_management()**
- parametre: input layer

  {overlap type} INTERSECT, CONTAINS, WITHIN…

  {selecting features}

  {search distance}

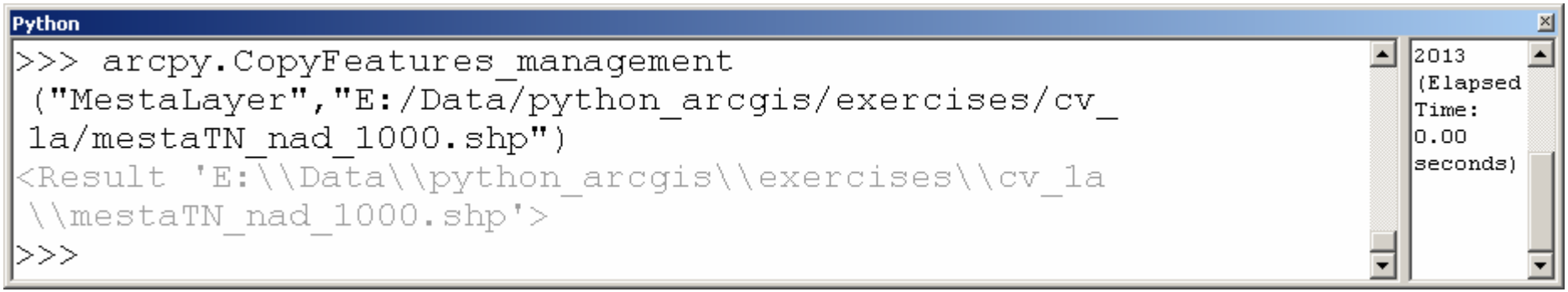
  {selection type}



Uloženie výberu do novej vrstvy

- nástroj **CopyFeatures_management()**
- parametre: input features

  output feature class



Čistenie kódu

- príkazy **try**, **except**, **finally**
- použijeme ich vtedy, ak chceme zachytiť výnimky v programe (napr. vstupná trieda objektov nemá žiadne objekty)
- najprv sa vykonajú príkazy za kľúčovým slovom **try:**
- ak sa neobjaví výnimka, dokončí sa vykonanie príkazy try
- ak sa objaví výnimka, zvyšok príkazov sa preskočí a výnimka sa porovnáva s výnimkami za kľúčovým slovom **except:** (môže ich byť aj viac)

Čistenie kódu

- ak sa výnimka zhoduje, vykonajú sa príkazy za except a pokračuje sa ďalej za príkazom try
- ak sa výnimka nezhoduje (nezachytená výnimka), pokračuje sa vo vykonávaní vonkajšieho príkazu try alebo sa program zastaví
- za kľúčové slovo **finally:** môžeme umiestniť príkazy, ktoré chceme aby sa vykonali bez ohľadu na to, či sa výnimka objaví alebo nie (napr. chceme vymazať kurzor, pretože kurzory niekedy majú výhradný prístup k dátam)

Čistenie kódu

- kľúčové slovo **except:** zachytáva výnimky až po najvyššiu úroveň (systémové)
- príkaz **raise** umožňuje tvorbu vlastných výnimiek (musia byť inštanciami alebo podtriedami triedy Exception)
- ak sa zadá príkaz raise bez argumentov, znovu sa získa výnimka, ktorá je práve aktívna, a vypíše sa chyba (bez tohto by sa len vykonali príkazy za except)

Čistenie kódu