

Lekcia 4

Dávkové spracovanie pomocou ArcPy Listing

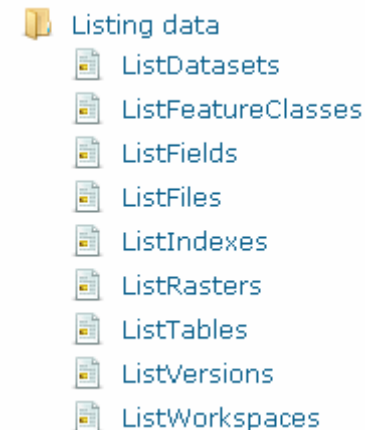
Cvičenie 3

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

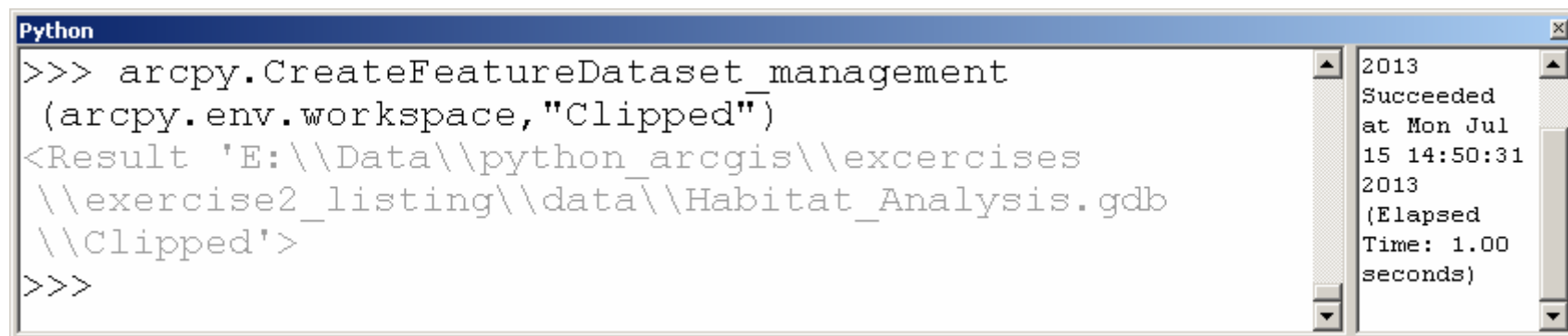
CreateFeatureDataset_management

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

Nástroj

CreateFeatureDataset_management

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

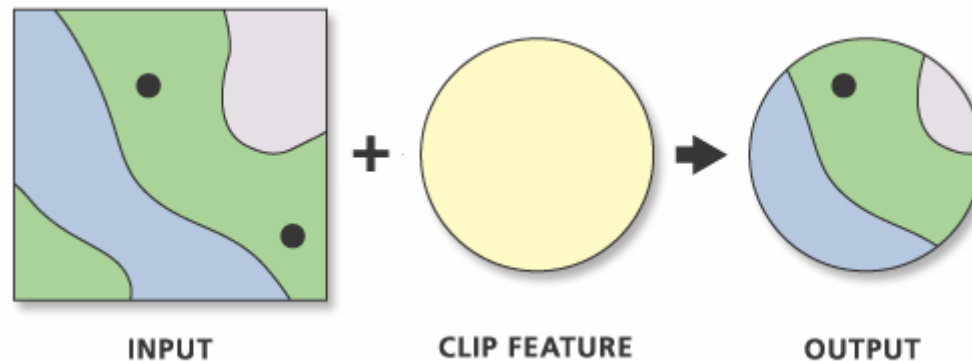


```
Python
>>> arcpy.CreateFeatureDataset_management
(arcpy.env.workspace, "Clipped")
<Result 'E:\\Data\\python_arcgis\\exercises
\\exercise2_listing\\data\\Habitat_Analysis.gdb
\\Clipped'>
>>>
```

2013
Succeeded
at Mon Jul
15 14:50:31
2013
(Elapsed
Time: 1.00
seconds)

Funkcia `Clip_analysis`

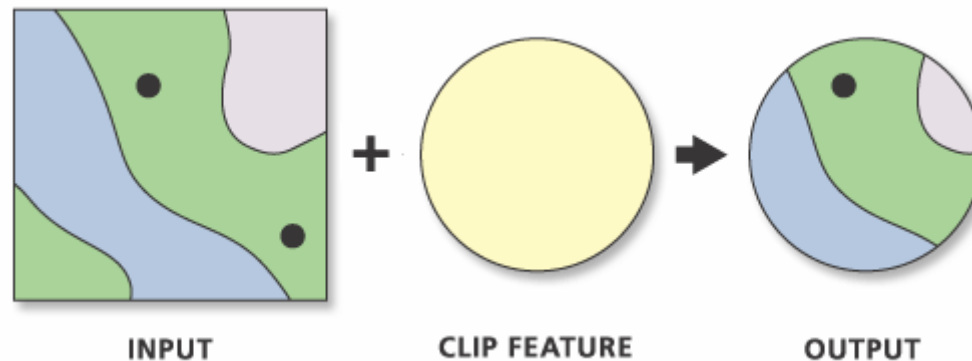
- parametre: input, clip features, output



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

Funkcia **Clip_analysis**

- parametre: input, clip features, output



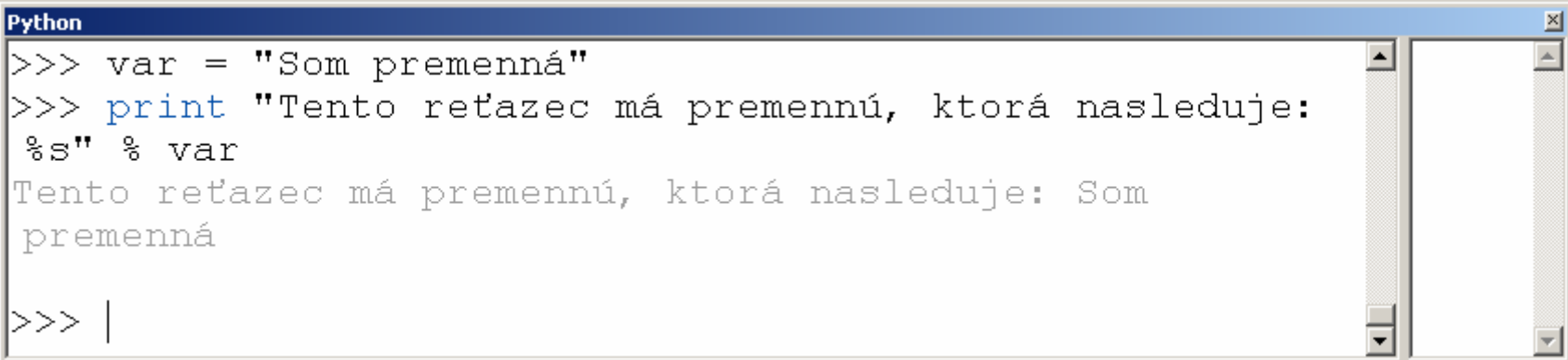
- dávkové orezanie – pomocou listovacích funkcií Python-u (cyklus **for**)

```
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`



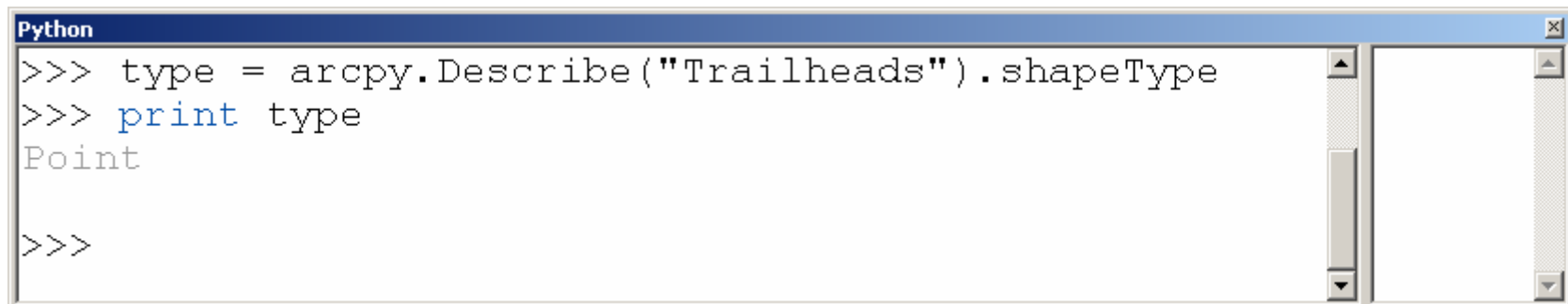
```
Python
>>> var = "Som premenná"
>>> print "Tento reťazec má premennú, ktorá nasleduje:
%s" % var
Tento reťazec má premennú, ktorá nasleduje: Som
premenná
>>> |
```


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 geoprocených objektov ako sú FeatureLayers a TableViews
- vlastnosti: typ údajov, typ geometrie (bod, línia, polygón), názov poľa OID, názov poľa shape, ... atď.



```
Python
>>> type = arcpy.Describe("Trailheads").shapeType
>>> print type
Point
>>>
```

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

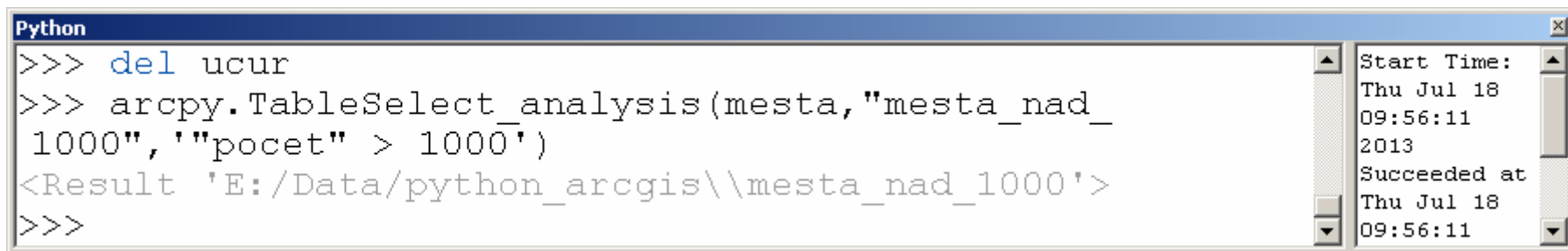
```
Python
>>> try:
...     data = arcpy.Describe(kraje)
...     fields = []
...     i = 0
...     while data.fields[i].name:
...         fields.append(data.fields[i].name)
...         i += 1
...
Python
>>> try:
...     data = arcpy.ListFields(kraje)
...     fields = []
...     i = 0
...     while data[i].name:
...         fields.append(data[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']
>>>
```

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 [...]



```
Python
>>> del ucur
>>> arcpy.TableSelect_analysis(mesta, "mesta_nad_
1000", '"pocet" > 1000')
<Result 'E:/Data/python_arcgis\\mesta_nad_1000'>
>>>
```

Start Time:
Thu Jul 18
09:56:11
2013
Succeeded at
Thu Jul 18
09:56:11

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)

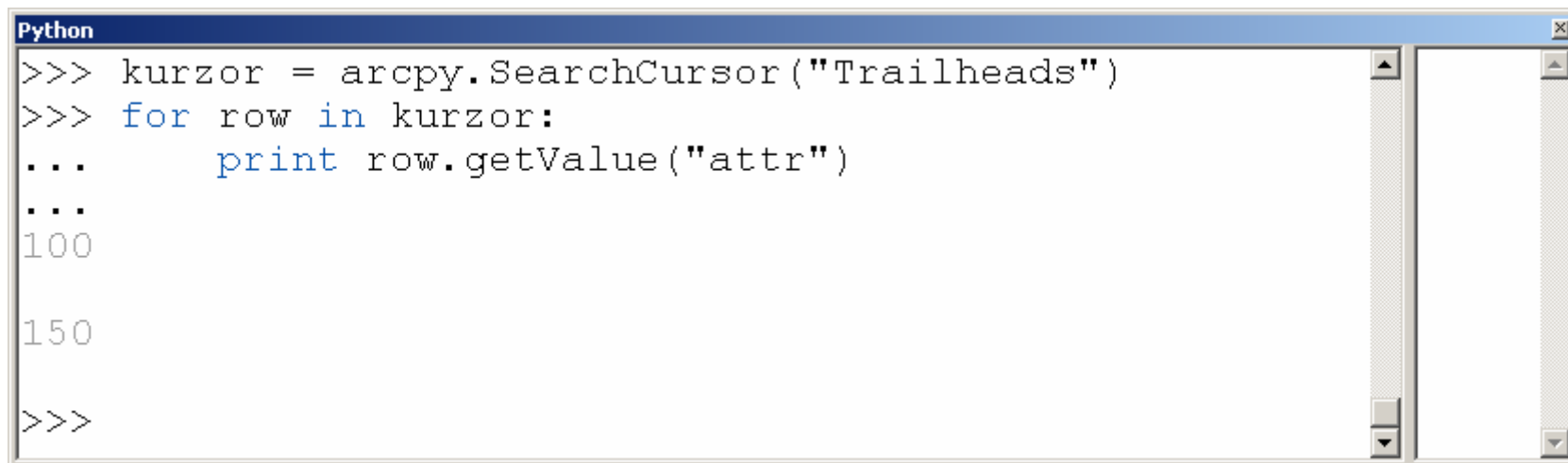
InsertCursor – pridáva nové záznamy do tabuľky , zapisuje hodnoty atribútov a geometriu

Funkcie `getValue`, `setValue`

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

SearchCursor

- SearchCursor

A screenshot of a Python interpreter window titled "Python". The window contains the following code and output:

```
>>> kurzor = arcpy.SearchCursor("Trailheads")
>>> for row in kurzor:
...     print row.getValue("attr")
...
100
150
>>>
```

The window has a standard Windows-style title bar with a close button (X) in the top right corner. The code is displayed in a monospaced font, and the output shows the values "100" and "150" printed on separate lines. The prompt ">>>" is shown at the beginning and end of the code block.

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

SearchCursor

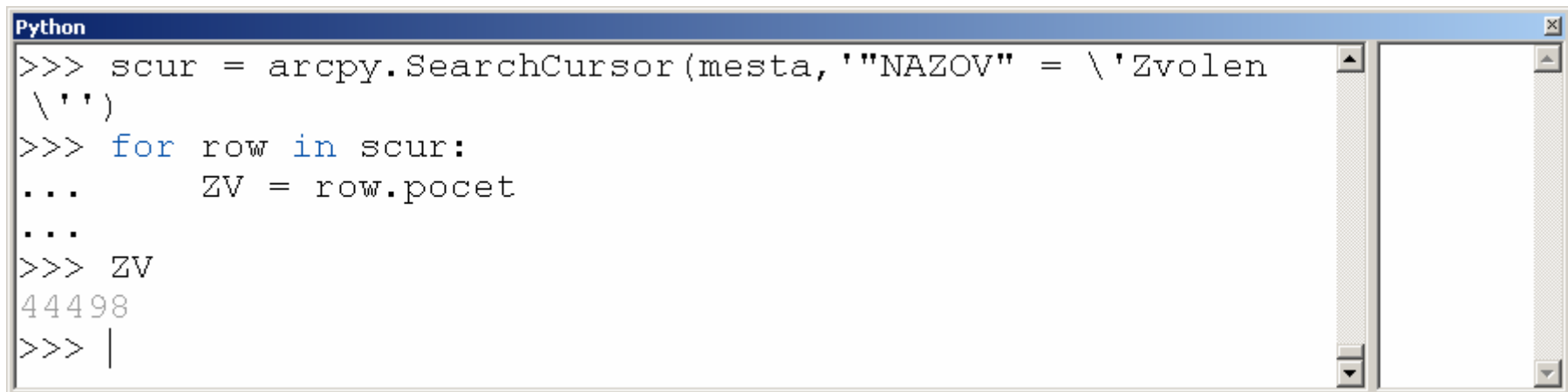
- SearchCursor

```
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

A screenshot of a Python shell window titled "Python". The window contains the following code and output:

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

The window has a blue title bar and a scrollable text area on the right.

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 'arcpy.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_arctis\\mesta_sk_kr.shp'>
>>>
```

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

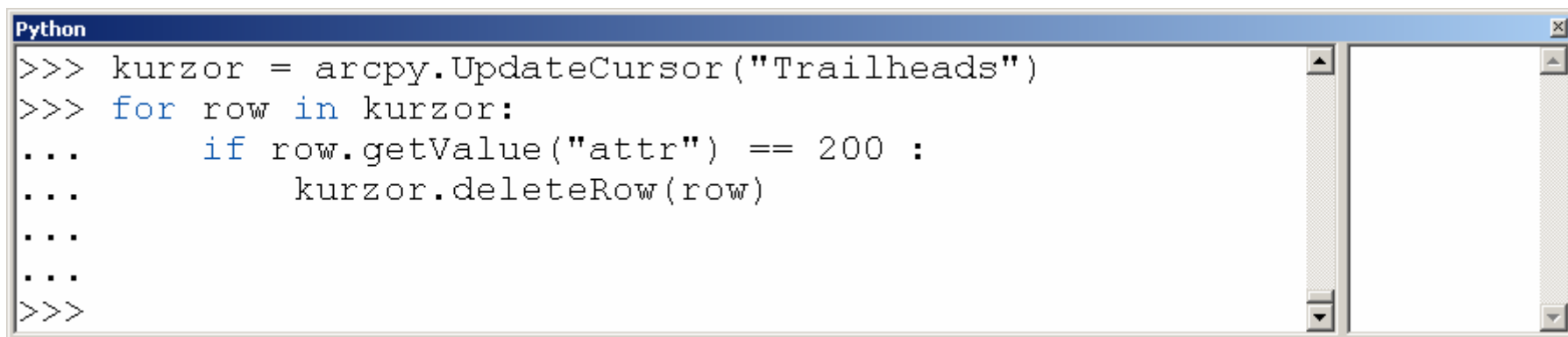
```
Python
>>> ucur = arcpy.UpdateCursor(mesta)
>>> for row in ucur:
...     pocet = row.getValue("ludia")
...     row.setValue("pocet",pocet)
...     ucur.updateRow(row)
...
>>>
```

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

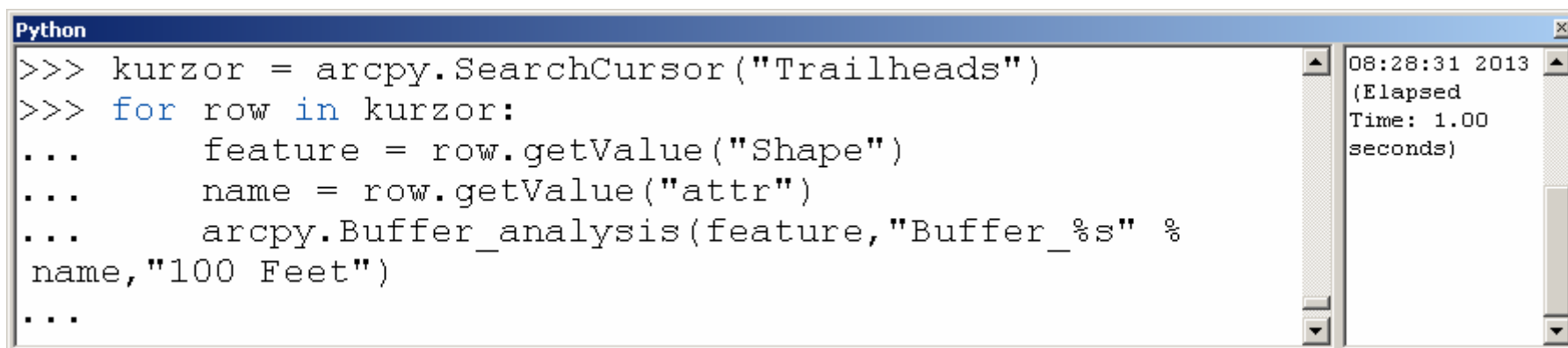
- vymazanie objektu (záznamu z tabuľky)



```
Python
>>> kurzor = arcpy.UpdateCursor("Trailheads")
>>> for row in kurzor:
...     if row.getValue("attr") == 200 :
...         kurzor.deleteRow(row)
...
...
>>>
```


Čítanie geometrie

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



```
Python
>>> kurzor = arcpy.SearchCursor("Trailheads")
>>> for row in kurzor:
...     feature = row.getValue("Shape")
...     name = row.getValue("attr")
...     arcpy.Buffer_analysis(feature, "Buffer_%s" %
name, "100 Feet")
...
08:28:31 2013
(Elapsed
Time: 1.00
seconds)
```

Práca s geometrickými objektmi

- geometrický objekt má vlastnosti: type, area, length, centroid, extent, first point, last point...atd'.
- 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

```
Python
>>> arcpy.Select_analysis(mesta,"mesta_nad_50000", '"pocet" >
50000')
<Result 'E:/Data/python_arcgis\\mesta_nad_50000.shp'>
>>> arcpy.DeleteIdentical_management("mesta_nad_
50000", "nazov")
<Result 'mesta_nad_50000'>
>>>
```

Práce s geometrickými objekty

```
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

```
Python
>>> kurzor = arcpy.InsertCursor("Trailheads")
>>> row = kurzor.newRow()
>>> row.setValue("attr", "200")
>>> row.setValue("Shape", arcpy.Point
(365378.167, 5209587.295))
>>> kurzor.insertRow(row)
>>>
```

- 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

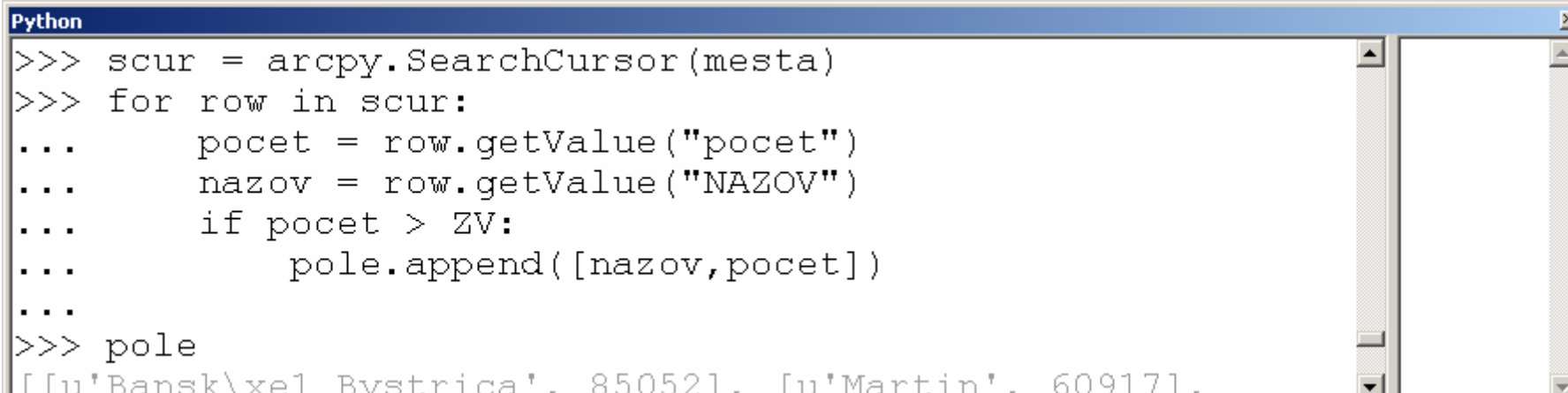
- vytvorenie prázdneho zoznamu:

```
zoznam = []
```

```
zoznam = list()
```

- zápis hodnôt do poľa:

```
zoznam.append()
```



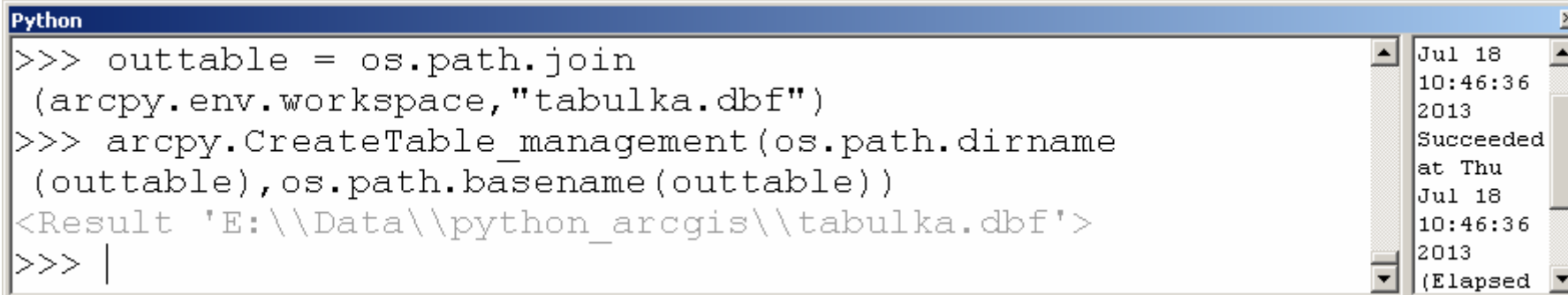
```
Python
>>> scur = arcpy.SearchCursor(mesta)
>>> for row in scur:
...     pocet = row.getValue("pocet")
...     nazov = row.getValue("NAZOV")
...     if pocet > ZV:
...         pole.append([nazov,pocet])
...
>>> pole
[[u'Bansk\xel Bystrica', 85052], [u'Martin', 60917],
```

Vytvorenie tabuľky

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

Vytvorenie tabuľky

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



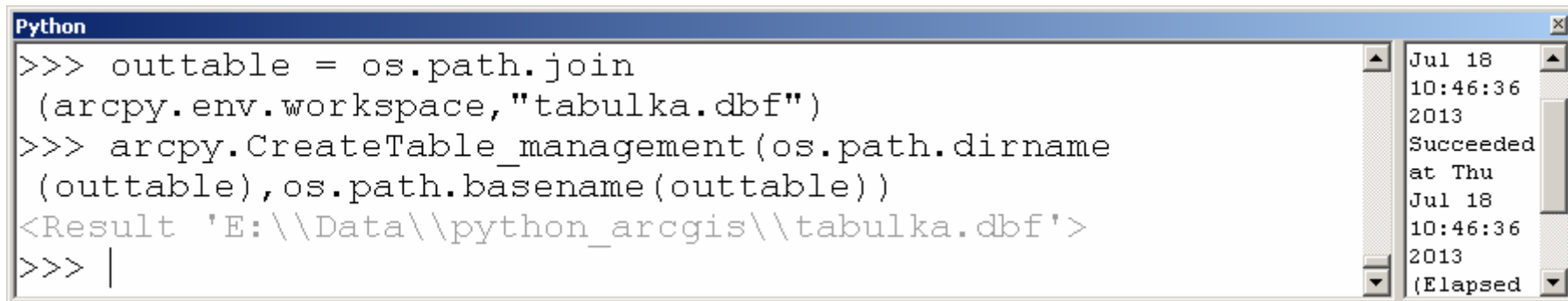
```
Python
>>> outtable = os.path.join
      (arcpy.env.workspace, "tabulka.dbf")
>>> arcpy.CreateTable_management(os.path.dirname
      (outtable), os.path.basename(outtable))
<Result 'E:\\Data\\python_arctgis\\tabulka.dbf'>
>>> |
```

Jul 18
10:46:36
2013
Succeeded
at Thu
Jul 18
10:46:36
2013
(Elapsed

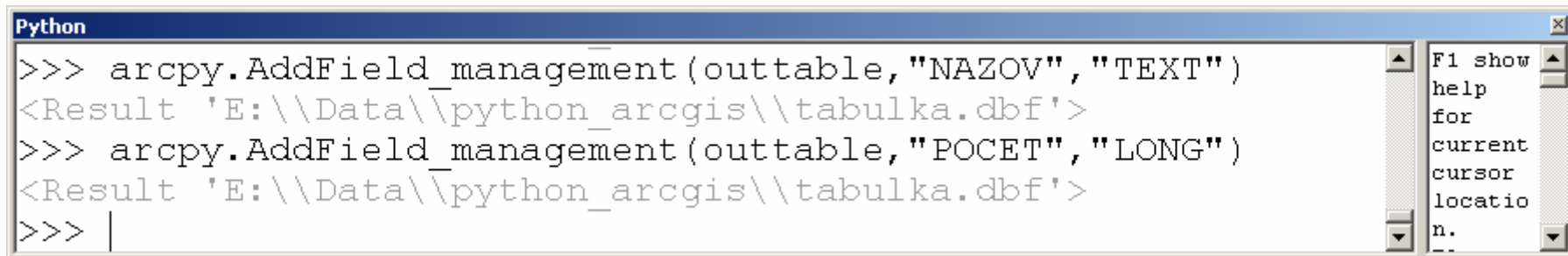
Vytvorenie tabuľky

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

```
Python
>>> outtable = os.path.join
( arcpy.env.workspace, "tabulka.dbf" )
>>> arcpy.CreateTable_management( os.path.dirname
( outtable ), os.path.basename( outtable ) )
<Result 'E:\\Data\\python_arcgis\\tabulka.dbf'>
>>> |
```



```
Python
>>> arcpy.AddField_management( outtable, "NAZOV", "TEXT" )
<Result 'E:\\Data\\python_arcgis\\tabulka.dbf'>
>>> arcpy.AddField_management( outtable, "POCET", "LONG" )
<Result 'E:\\Data\\python_arcgis\\tabulka.dbf'>
>>> |
```



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)
...
>>> |
```

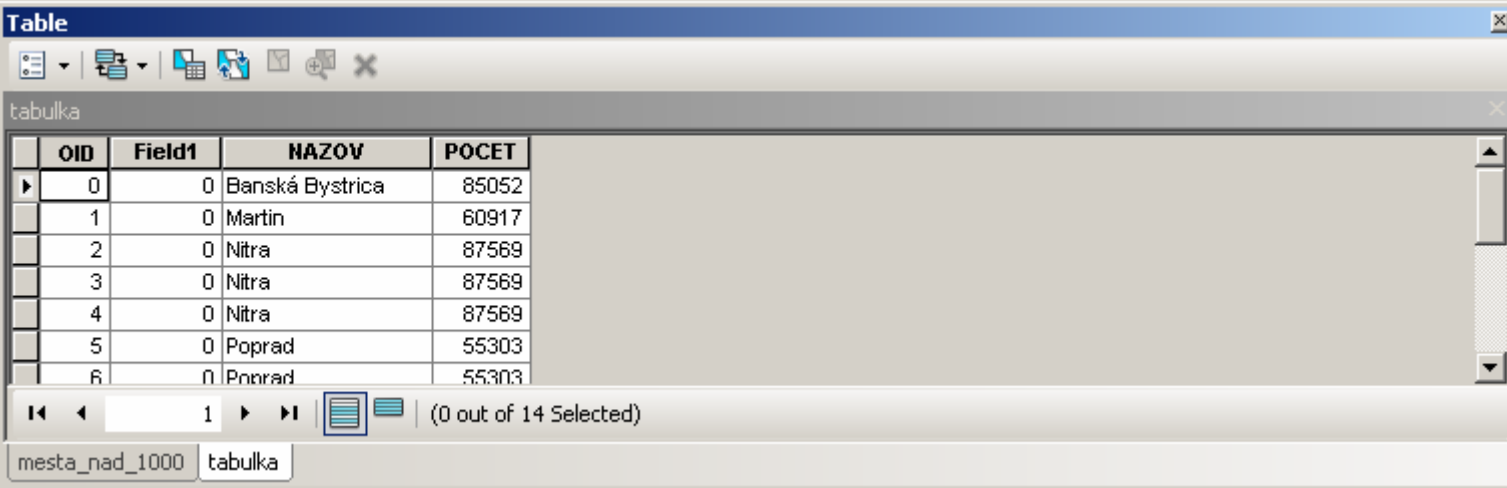


Table window showing a data table with the following columns: **OID**, **Field1**, **NAZOV**, and **POCET**. The table contains 7 rows of data.

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

Navigation: 1 | (0 out of 14 Selected)

mesta_nad_1000 tabulka

Vymazanie identických záznamov

- nástroj **DeletIdentical_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)

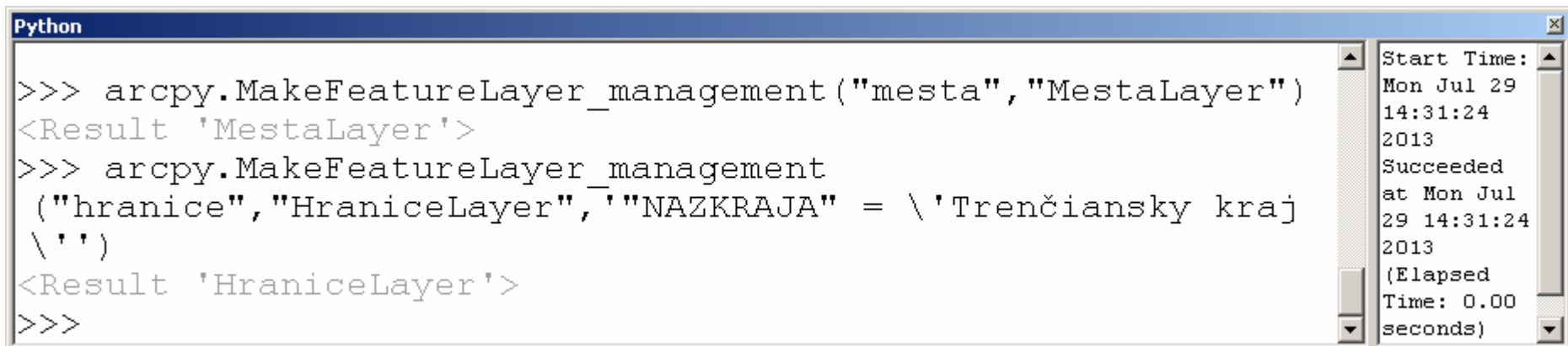
{x,y tolerance}, {z tolerance}

```
Python
...
>>> arcpy.DeleteIdentical_management("tabulka", "NAZOV")
<Result 'tabulka'>
>>>
```

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

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

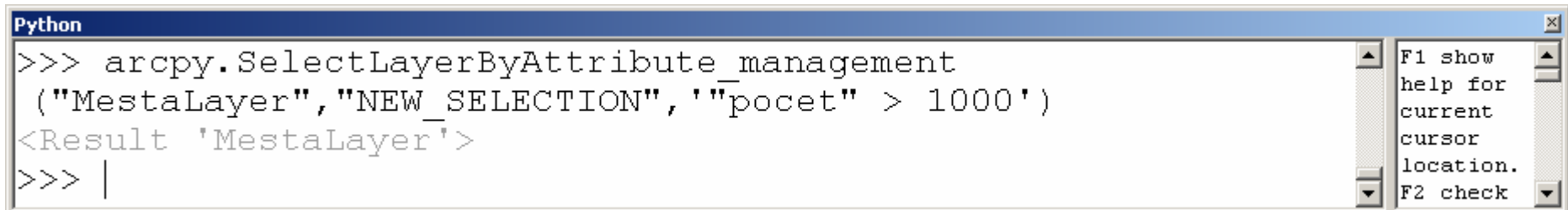


```
Python
>>> arcpy.MakeFeatureLayer_management("mesta", "MestaLayer")
<Result 'MestaLayer'>
>>> arcpy.MakeFeatureLayer_management
("hranice", "HraniceLayer", '"NAZKRAJA" = \'Trenčiansky kraj
\'')
<Result 'HraniceLayer'>
>>>
```

Start Time:
Mon Jul 29
14:31:24
2013
Succeeded
at Mon Jul
29 14:31:24
2013
(Elapsed
Time: 0.00
seconds)

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...)



```
Python
>>> arcpy.SelectLayerByAttribute_management
("MestaLayer", "NEW_SELECTION", '"pocet" > 1000')
<Result 'MestaLayer'>
>>> |
```

The screenshot shows a Python command prompt window with a blue title bar. The text in the window shows the execution of the `arcpy.SelectLayerByAttribute_management` function with three arguments: "MestaLayer", "NEW_SELECTION", and '"pocet" > 1000'. The output is a `<Result 'MestaLayer'>` object. On the right side of the window, there is a vertical scroll bar and a list of keyboard shortcuts: F1 show help for current cursor location, and F2 check.

Priestorové dopyty

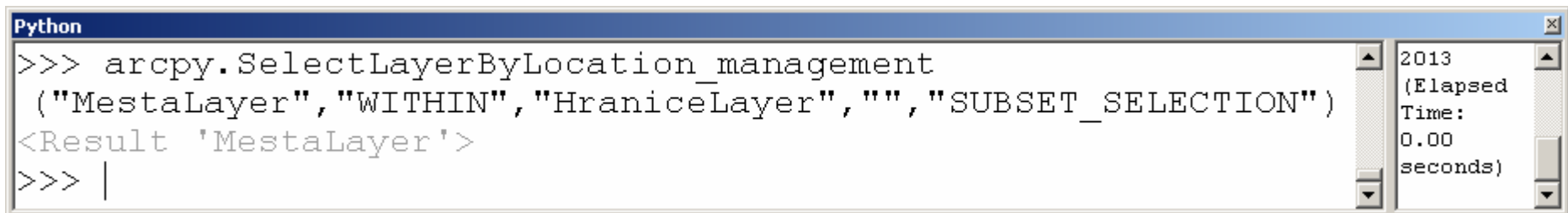
- nástroj **SelectLayerByLocation_management()**
- parametre: input layer

{overlap type} INTERSECT, CONTAINS,
WITHIN...

{selecting features}

{search distance}

{selection type}

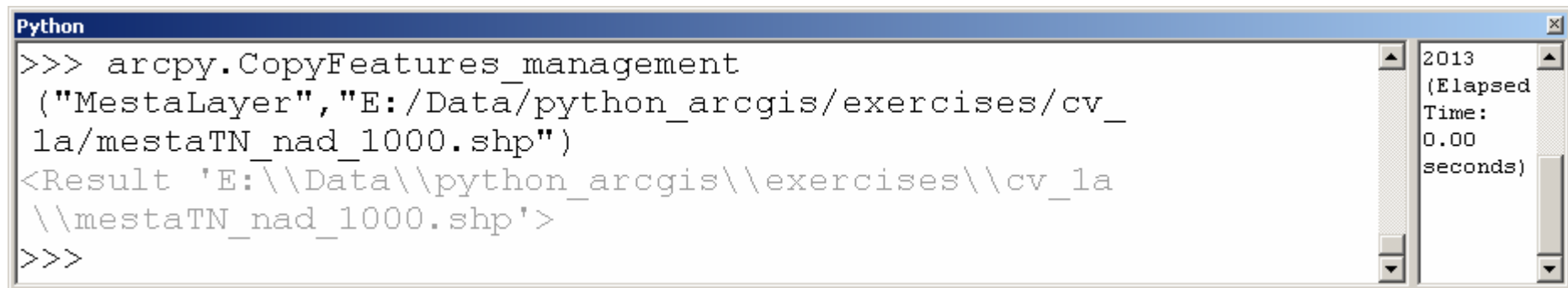


```
Python
>>> arcpy.SelectLayerByLocation_management
  ("MestaLayer", "WITHIN", "HraniceLayer", "", "SUBSET_SELECTION")
<Result 'MestaLayer'>
>>> |
```

2013
(Elapsed
Time:
0.00
seconds)

Uloženie výberu do novej vrstvy

- nástroj **CopyFeatures_management()**
- parametre: input features
output feature class



```
Python
>>> arcpy.CopyFeatures_management
("MestaLayer", "E:/Data/python_arcgis/exercises/cv_
la/mestaTN_nad_1000.shp")
<Result 'E:\\Data\\python_arcgis\\exercises\\cv_la
\\mestaTN_nad_1000.shp'>
>>>
```

2013
(Elapsed
Time:
0.00
seconds)

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

```
Python
>>> try:
...     raise NameError('Hi there')
... except NameError:
...     print 'Vynimka'
...
Vynimka

>>> try:
...     raise NameError('Hi there')
... except:
...     raise
...
Runtime error <type 'exceptions.NameError'>: Hi there
>>>
```