Putting Python Into Practice

2014 OKSCAUG Conference

Joel A. Foster

GIS Coordinator

Canadian County Assessor’s Office

Why Python?

- According to *Python Scripting for ArcGIS*...
  - Simple and easy to learn (relative)
  - It's free and open source
  - It's cross platform
  - It's interpreted (instead of compiled)
  - It's object-oriented

Zandbergen, Paul A. (2013). ***Python Scripting for ArcGIS*****. New York, NY: ESRI Press**

Why Python?

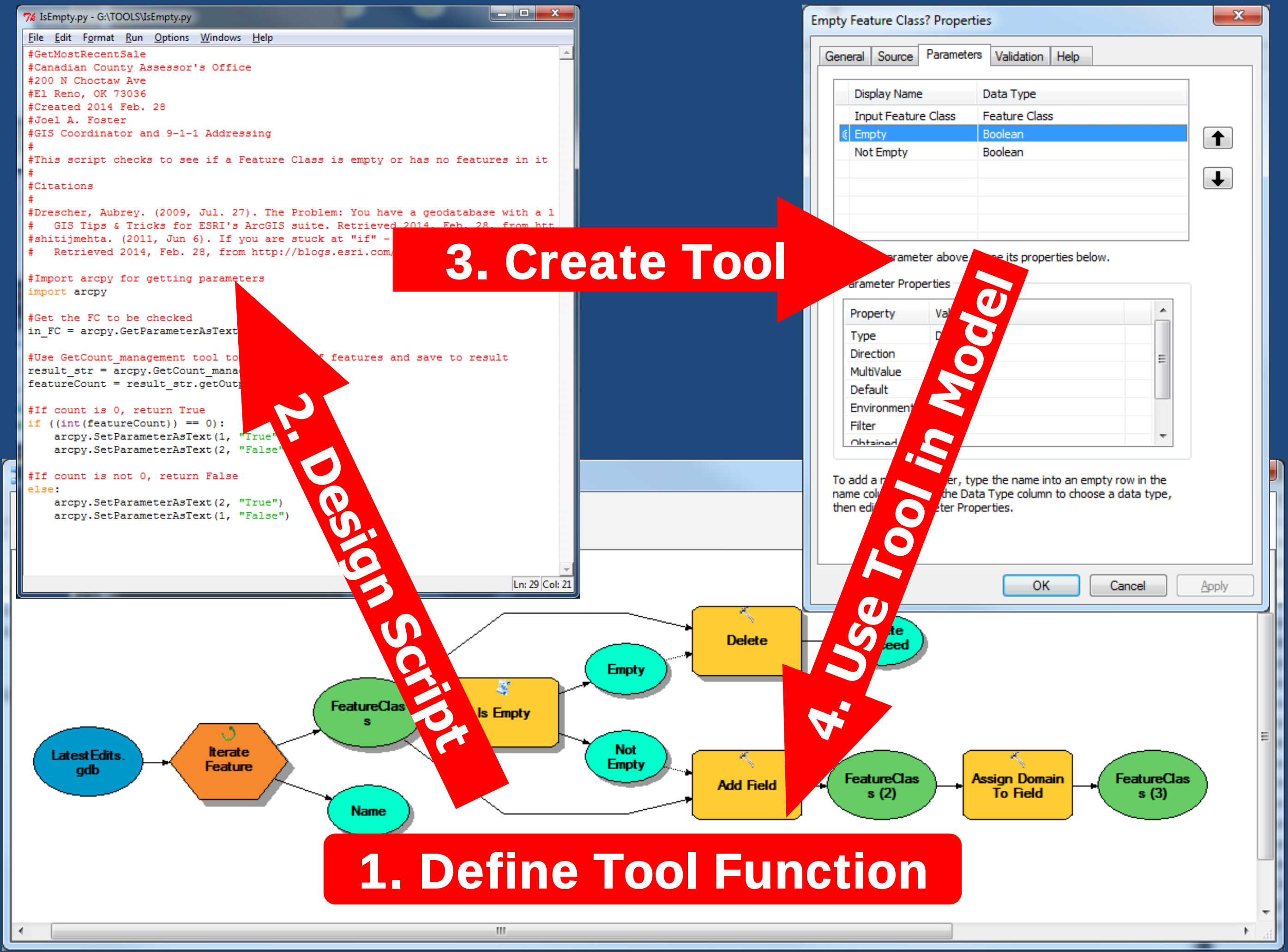
- Areas within ArcGIS that use Python:
  - Model Builder
  - Field Calculator
  - Label Expressions

Model Builder

- Using Python:
  - Allows for “If” or branching logic in models
  - Adds functions that ArcGIS tools do not cover

Model Builder

Creating Custom Tools



```
#GetMostRecentSale
#Canadian County Assessor's Office
#200 N Choctaw Ave
#El Reno, OK 73036
#Created 2014 Feb. 28
#Joel A. Foster
#GIS Coordinator and 9-1-1 Addressing
#
#This script checks to see if a Feature Class is empty or has no features in it
#
#Citations
#
#Drescher, Aubrey. (2009, Jul. 27). The Problem: You have a geodatabase with a l
#   GIS Tips & Tricks for ESRI's ArcGIS suite. Retrieved 2014, Feb. 28, from htt
#shitijmehta. (2011, Jun 6). If you are stuck at "if" -
#   Retrieved 2014, Feb. 28, from http://blogs.esri.com/

#Import arcpy for getting parameters
import arcpy

#Get the FC to be checked
in_FC = arcpy.GetParameterAsText

#Use GetCount_management tool to              f features and save to result
result_str = arcpy.GetCount_mana
featureCount = result_str.getOutp

#If count is 0, return True
if ((int(featureCount)) == 0):
    arcpy.SetParameterAsText(1, "True"
    arcpy.SetParameterAsText(2, "False"

#If count is not 0, return False
else:
    arcpy.SetParameterAsText(2, "True")
    arcpy.SetParameterAsText(1, "False")
```

Empty Feature Class? Properties

General | Source | Parameters | Validation | Help

Display Name	Data Type
Input Feature Class	Feature Class
Empty	Boolean
Not Empty	Boolean

rameter above   e its properties below.

arameter Properties

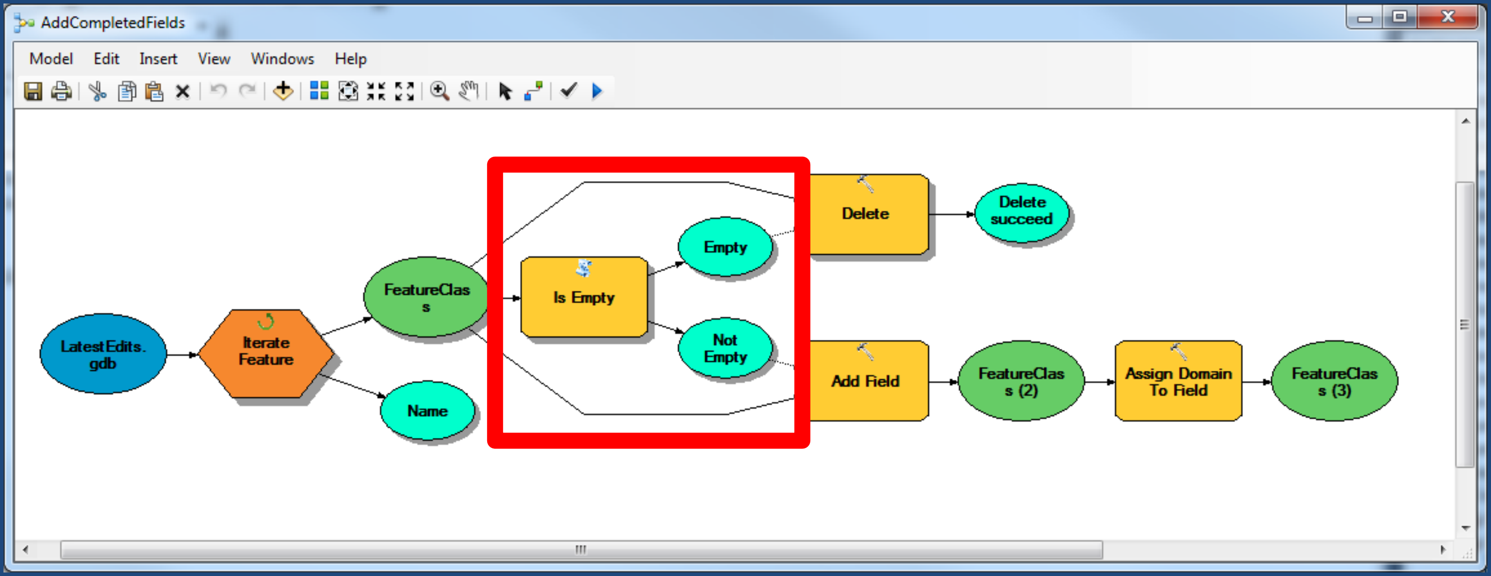
Property	Va
Type	D
Direction	
MultiValue	
Default	
Environment	
Filter	
Obtained	

To add a n   er, type the name into an empty row in the name colu   the Data Type column to choose a data type, then edi   eter Properties.

OK | Cancel | Apply

Model Builder

Creating Custom Tools



Example of Branching Logic in Model

Model Builder

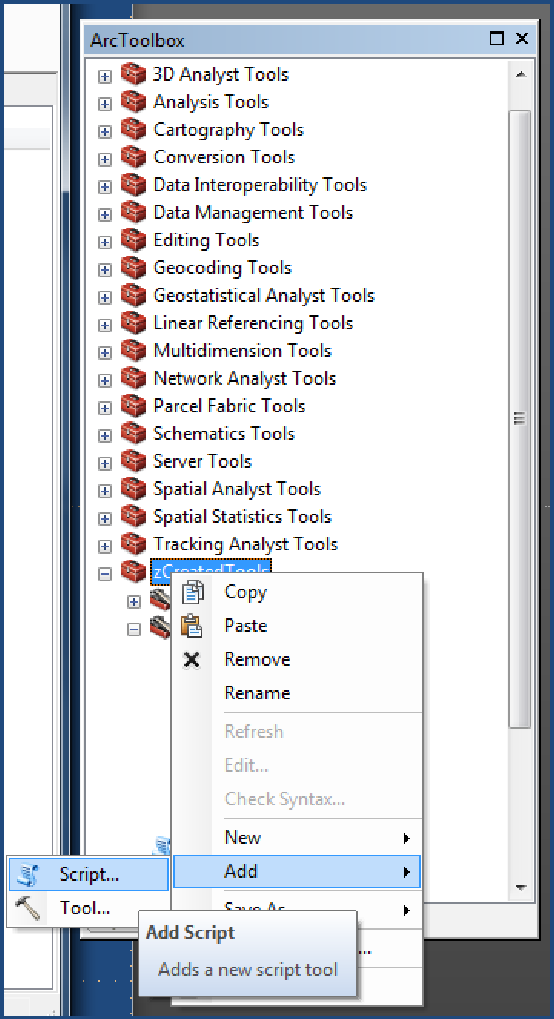
Creating Custom Tools

```python
#Import arcpy for getting parameters
import arcpy

#Get the FC to be checked
in_FC = arcpy.GetParameterAsText(0)

#Use GetCount_management tool to get count of features and save to result
result_str = arcpy.GetCount_management(in_FC)
featureCount = result_str.getOutput(0)

#If count is 0, return True
if ((int(featureCount)) == 0):
    arcpy.SetParameterAsText(1, "True")
    arcpy.SetParameterAsText(2, "False")

#If count is not 0, return False
else:
    arcpy.SetParameterAsText(2, "True")
    arcpy.SetParameterAsText(1, "False")
```

Model Builder

Creating Custom Tools

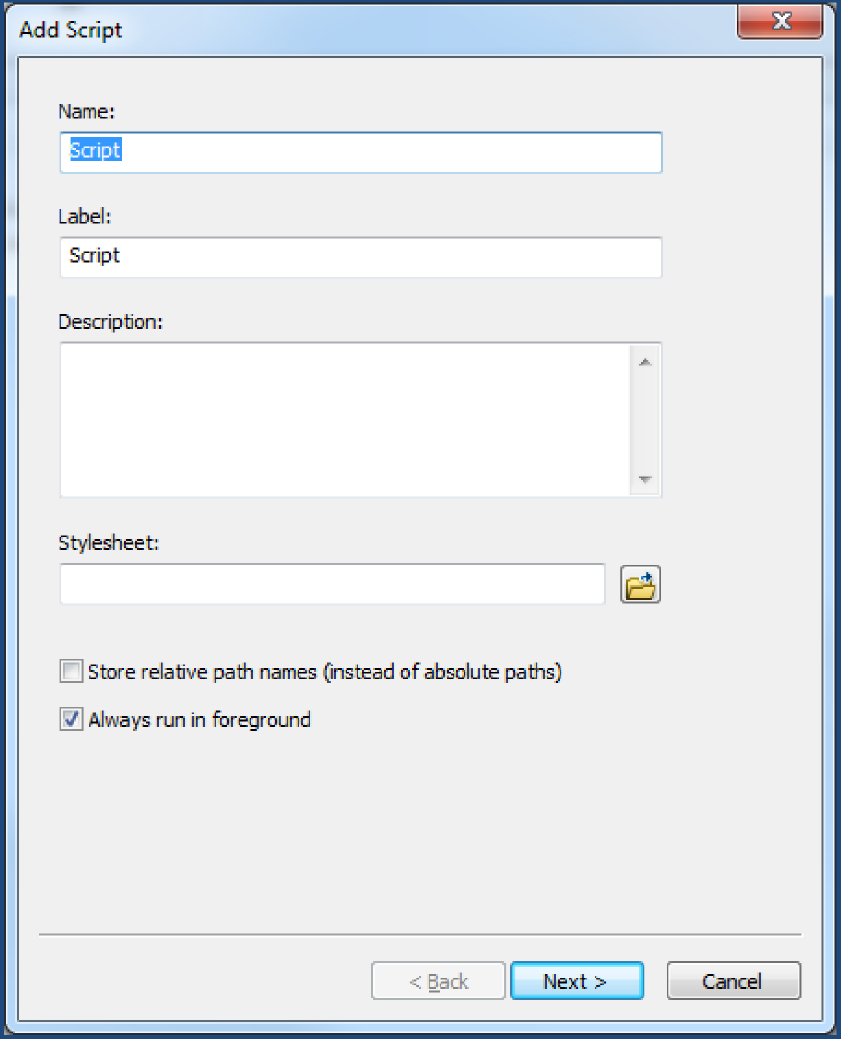
Create a new script in a toolbox



Model Builder

Creating Custom Tools

Add name and description



Add Script

Name: Script

Label: Script

Description:

Stylesheet:

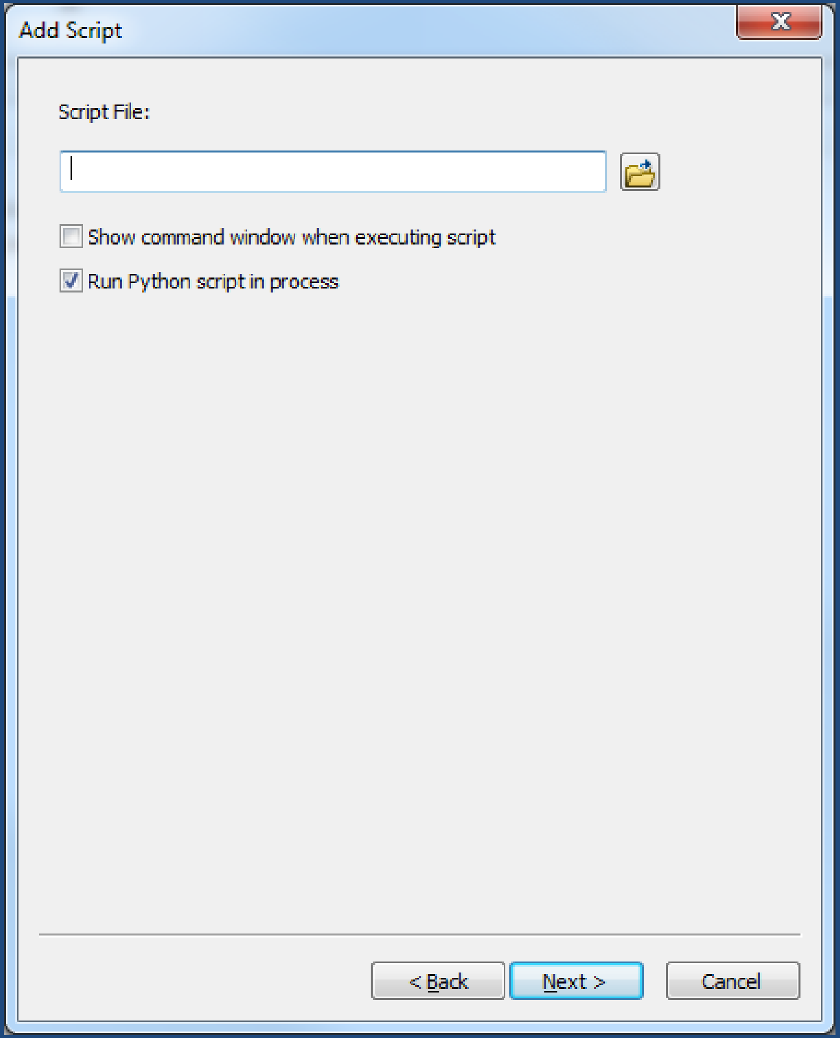
[ ] Store relative path names (instead of absolute paths)

[✓] Always run in foreground

Model Builder

Creating Custom Tools

Point to .py text file


Add Script

Script File:

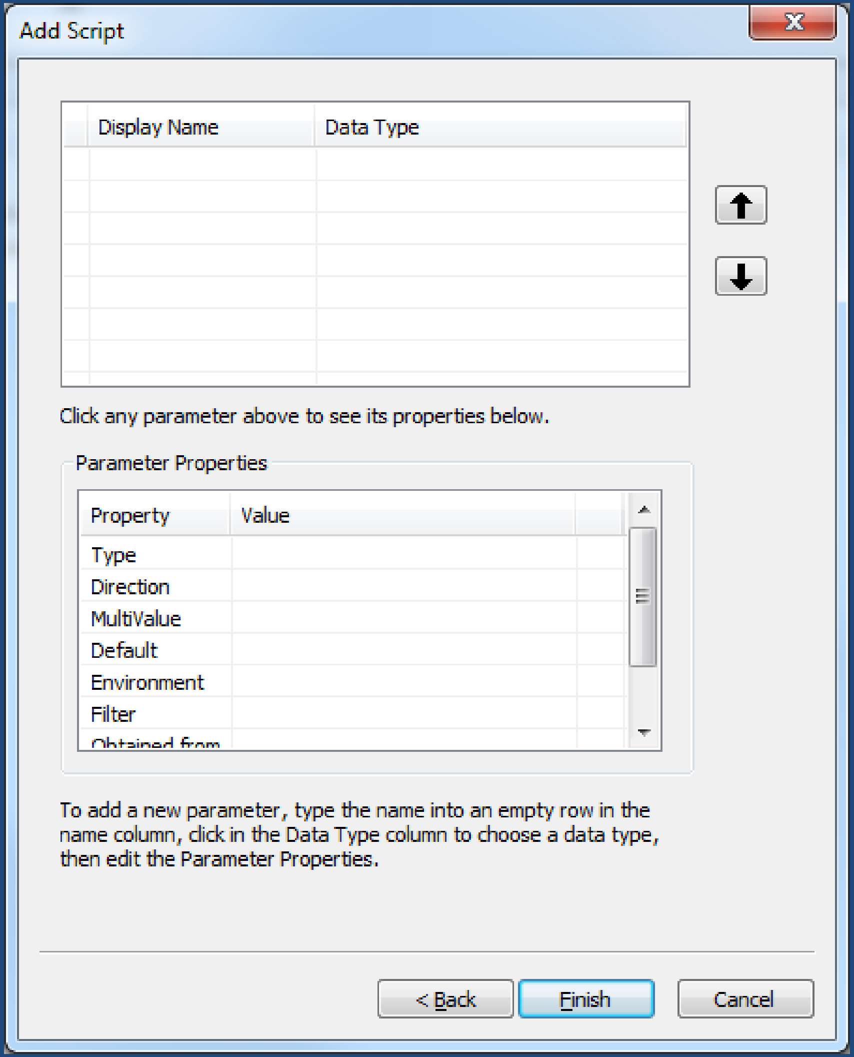
[ ] Show command window when executing script

[✓] Run Python script in process

Model Builder

Creating Custom Tools

Specify parameters



Add Script

Display Name	Data Type

Click any parameter above to see its properties below.

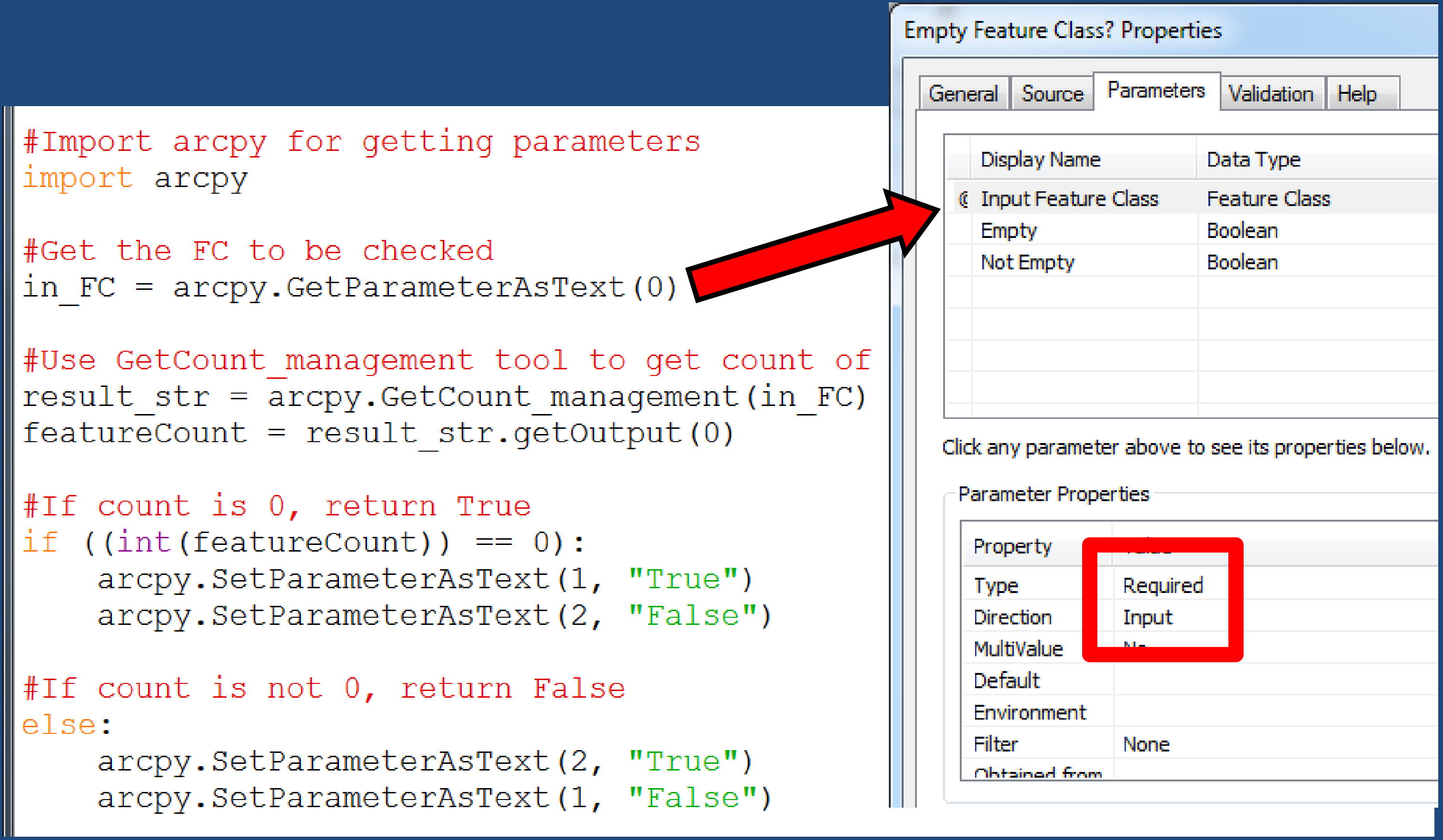
Parameter Properties

Property	Value
Type	
Direction	
MultiValue	
Default	
Environment	
Filter	
Obtained from	

To add a new parameter, type the name into an empty row in the name column, click in the Data Type column to choose a data type, then edit the Parameter Properties.

Model Builder

Creating Custom Tools



```
#Import arcpy for getting parameters
import arcpy

#Get the FC to be checked
in_FC = arcpy.GetParameterAsText(0)

#Use GetCount_management tool to get count of
result_str = arcpy.GetCount_management(in_FC)
featureCount = result_str.getOutput(0)

#If count is 0, return True
if ((int(featureCount)) == 0):
    arcpy.SetParameterAsText(1, "True")
    arcpy.SetParameterAsText(2, "False")

#If count is not 0, return False
else:
    arcpy.SetParameterAsText(2, "True")
    arcpy.SetParameterAsText(1, "False")
```

Empty Feature Class? Properties

General | Source | Parameters | Validation | Help

Display Name	Data Type
Input Feature Class	Feature Class
Empty	Boolean
Not Empty	Boolean

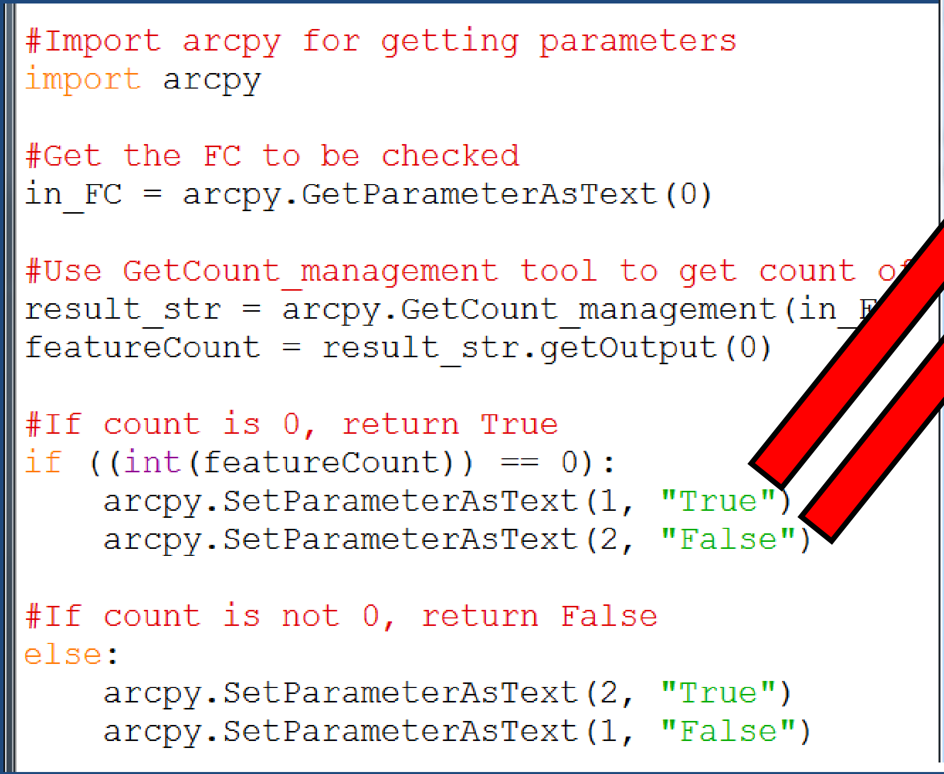
Click any parameter above to see its properties below.

Parameter Properties

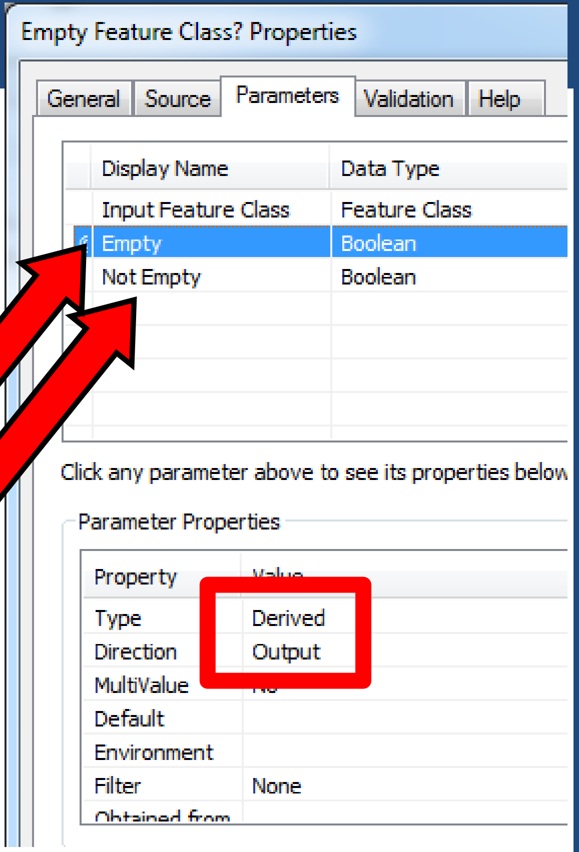
Property	Value
Type	Required
Direction	Input
MultiValue	No
Default	
Environment	
Filter	None
Obtained from	

Model Builder

Creating Custom Tools



```
#Import arcpy for getting parameters
import arcpy

#Get the FC to be checked
in_FC = arcpy.GetParameterAsText(0)

#Use GetCount_management tool to get count o
result_str = arcpy.GetCount_management(in_F
featureCount = result_str.getOutput(0)

#If count is 0, return True
if ((int(featureCount)) == 0):
    arcpy.SetParameterAsText(1, "True")
    arcpy.SetParameterAsText(2, "False")

#If count is not 0, return False
else:
    arcpy.SetParameterAsText(2, "True")
    arcpy.SetParameterAsText(1, "False")
```



Empty Feature Class? Properties

Display Name	Data Type
Input Feature Class	Feature Class
Empty	Boolean
Not Empty	Boolean

Click any parameter above to see its properties below

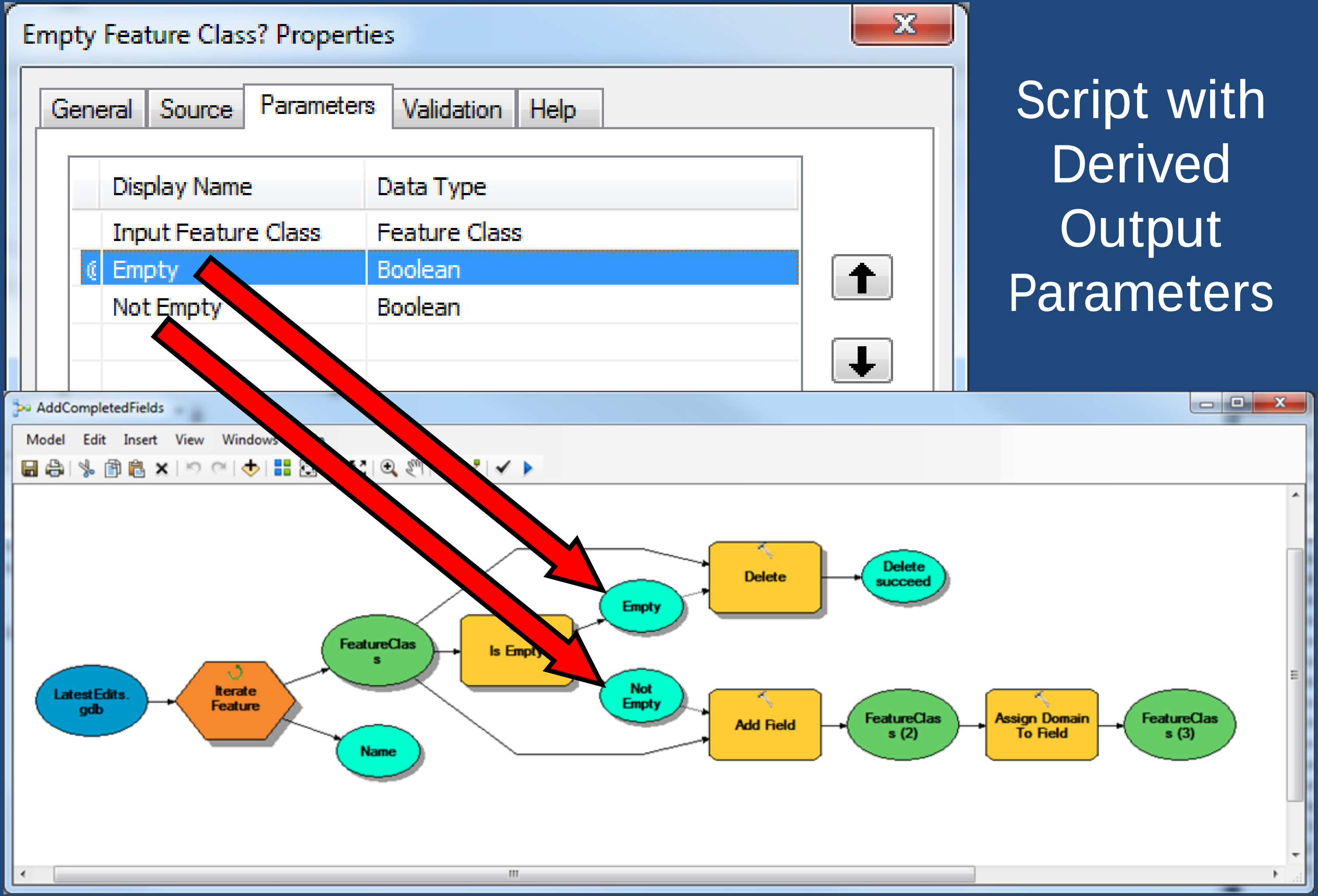
Parameter Properties

Property	Value
Type	Derived
Direction	Output
MultiValue	No
Default	
Environment	
Filter	None
Obtained from	

Derived Output parameters do not allow user input

Model Builder

Branching Logic



Empty Feature Class? Properties

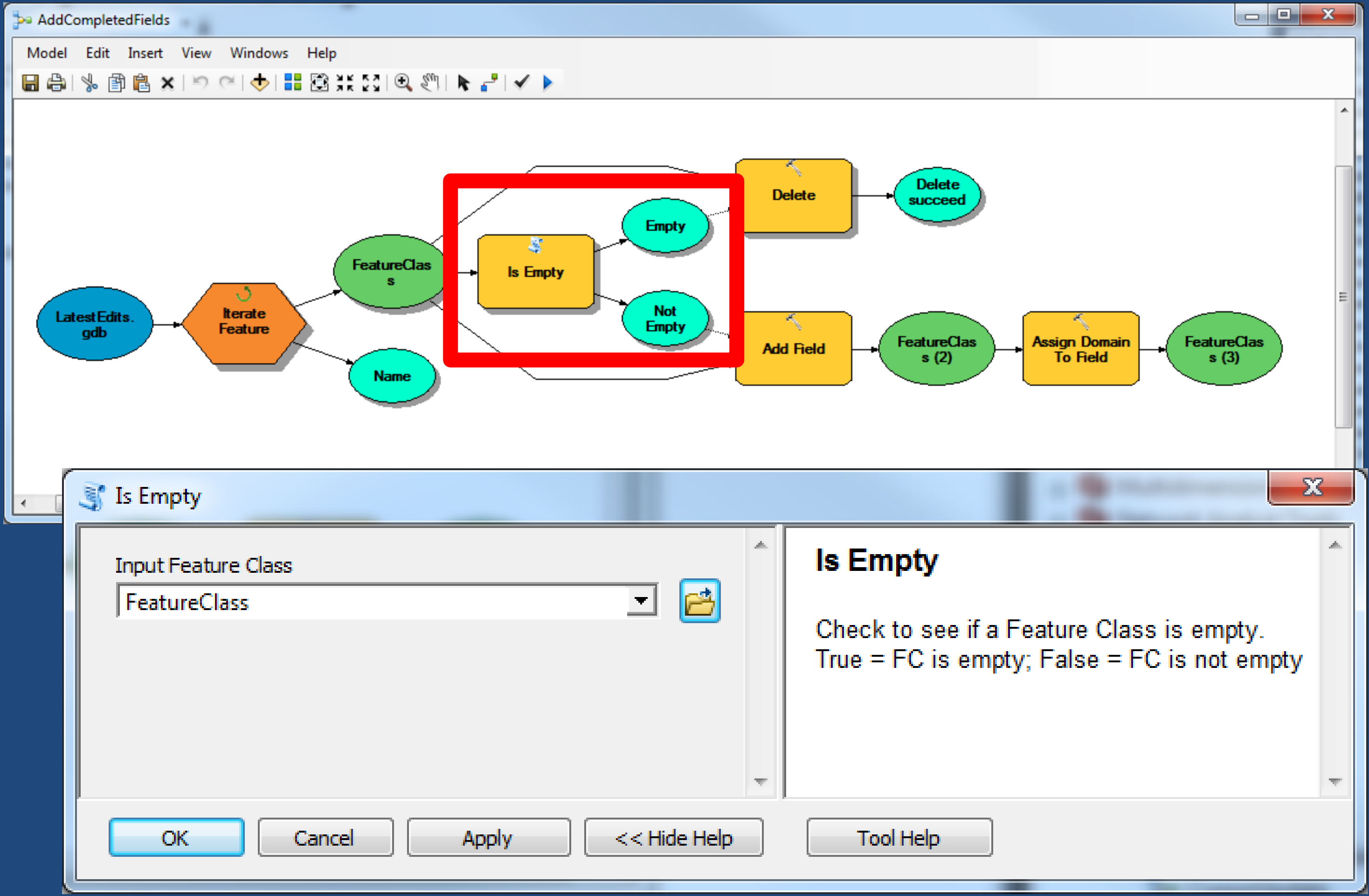
General | Source | Parameters | Validation | Help

Display Name	Data Type
Input Feature Class	Feature Class
Empty	Boolean
Not Empty	Boolean

Script with Derived Output Parameters

Model Builder

Branching Logic



Is Empty

Input Feature Class

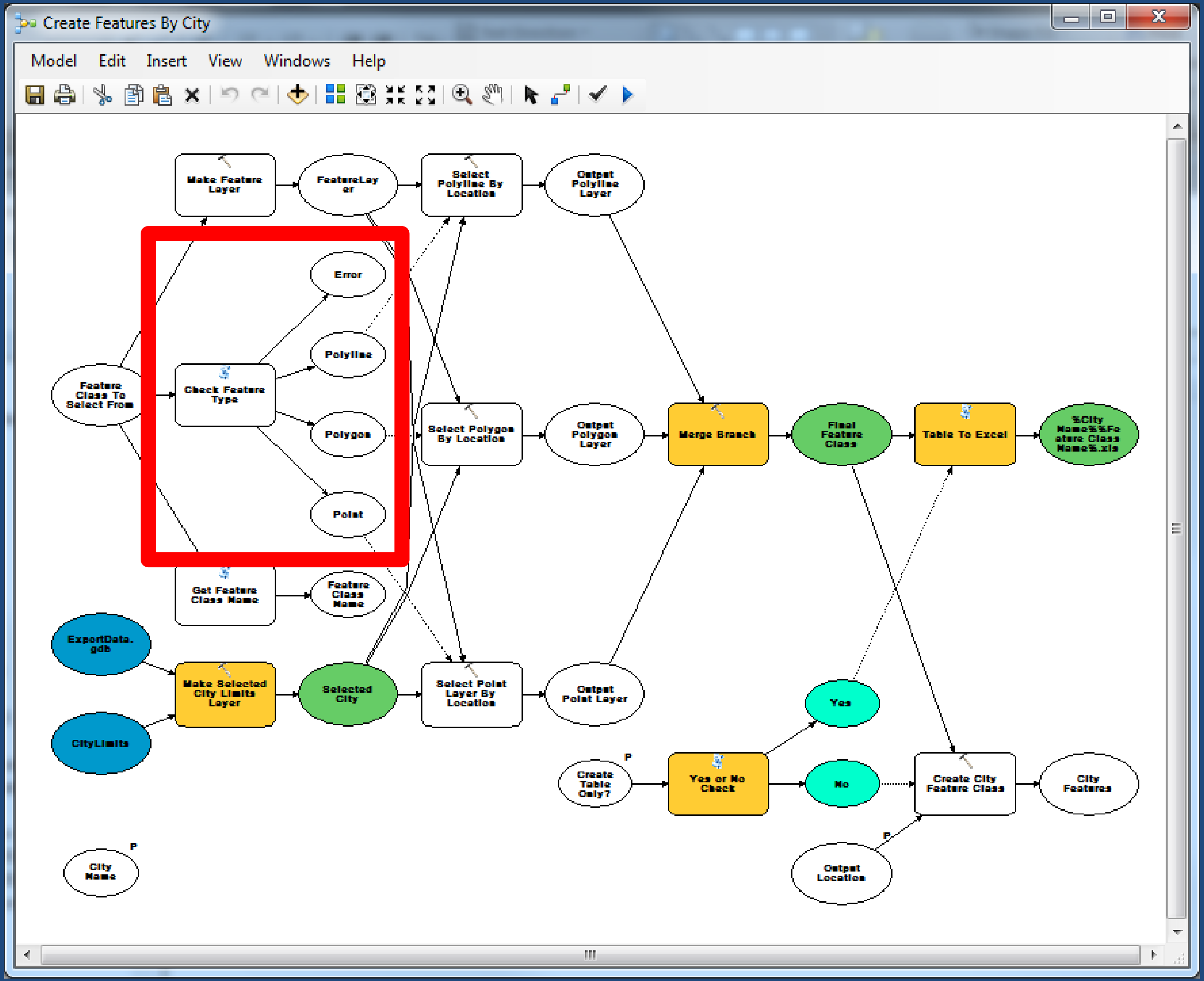
FeatureClass

Is Empty

Check to see if a Feature Class is empty.
True = FC is empty; False = FC is not empty

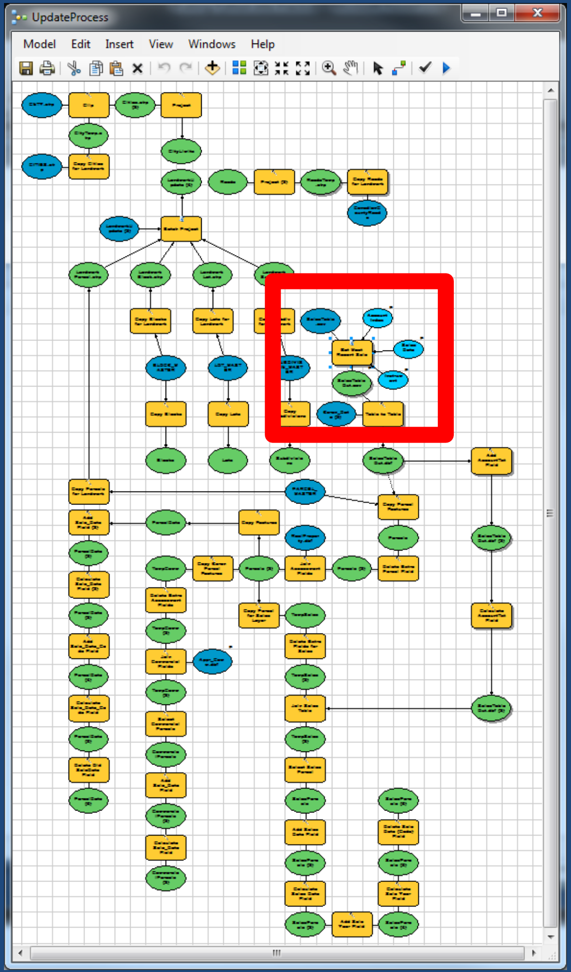
Model Builder

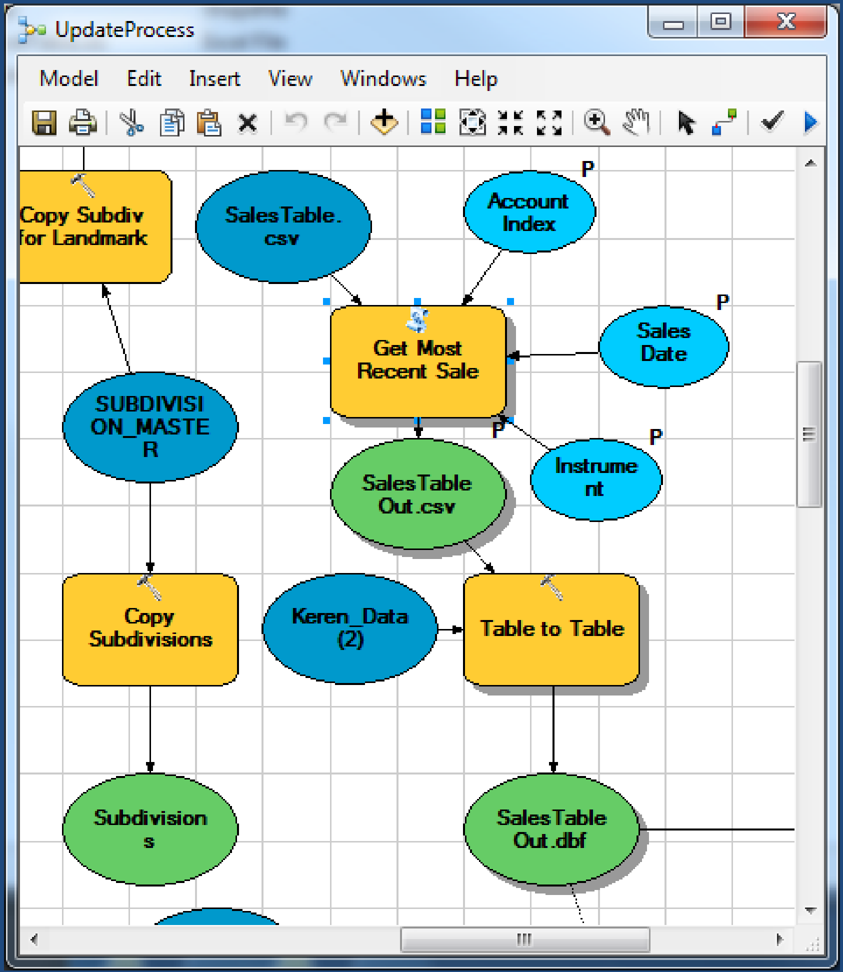
Branching Logic

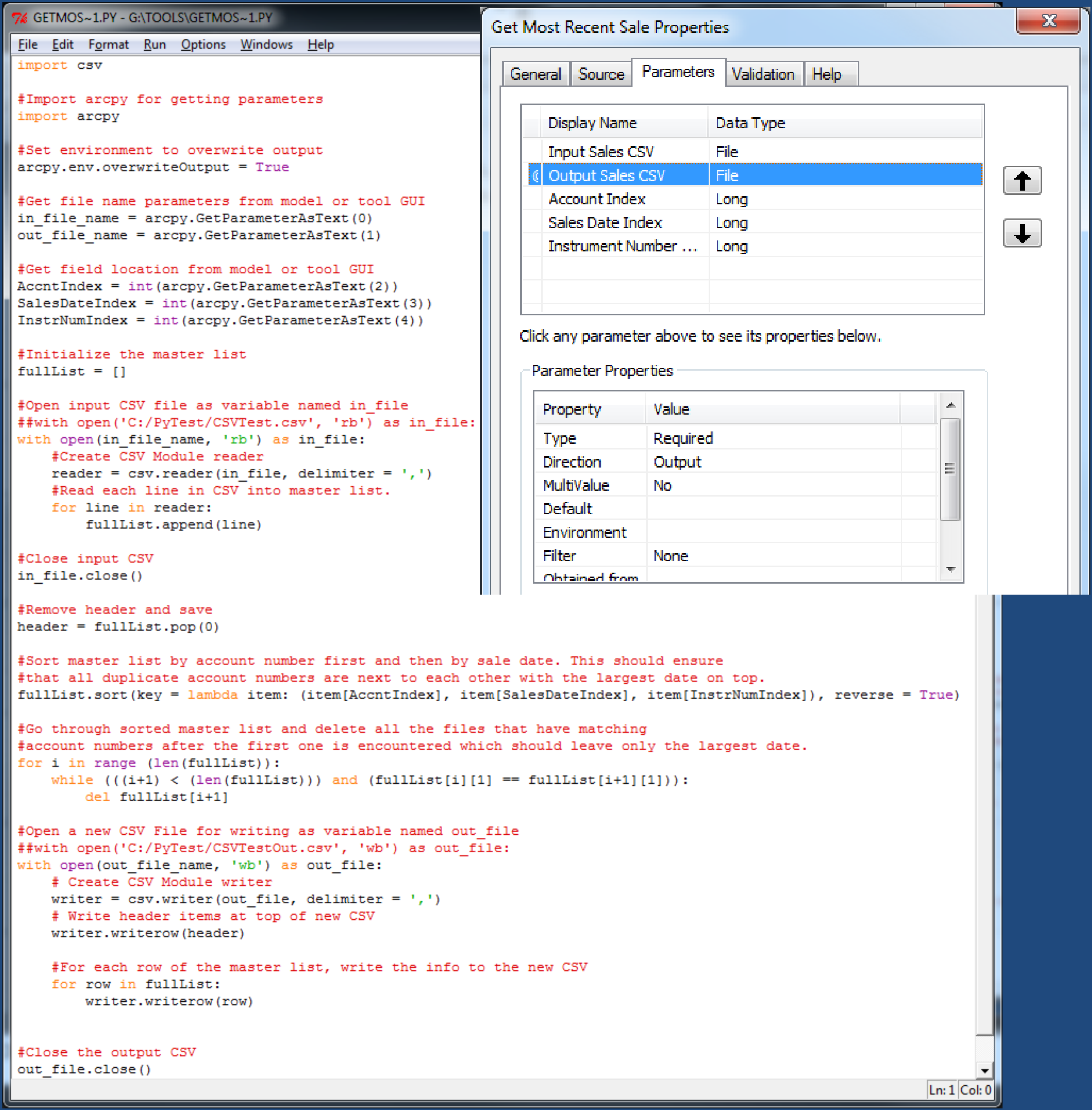


Model Builder

Creating New Tools






GETMOS~1.PY - G:\TOOLS\GETMOS~1.PY
File Edit Format Run Options Windows Help
Get Most Recent Sale Properties
General Source Parameters Validation Help
Ln: 1 Col: 0

```
import csv

#Import arcpy for getting parameters
import arcpy

#Set environment to overwrite output
arcpy.env.overwriteOutput = True

#Get file name parameters from model or tool GUI
in_file_name = arcpy.GetParameterAsText(0)
out_file_name = arcpy.GetParameterAsText(1)

#Get field location from model or tool GUI
AccntIndex = int(arcpy.GetParameterAsText(2))
SalesDateIndex = int(arcpy.GetParameterAsText(3))
InstrNumIndex = int(arcpy.GetParameterAsText(4))

#Initialize the master list
fullList = []

#Open input CSV file as variable named in_file
##with open('C:/PyTest/CSVTest.csv', 'rb') as in_file:
with open(in_file_name, 'rb') as in_file:
    #Create CSV Module reader
    reader = csv.reader(in_file, delimiter = ',')
    #Read each line in CSV into master list.
    for line in reader:
        fullList.append(line)

#Close input CSV
in_file.close()

#Remove header and save
header = fullList.pop(0)

#Sort master list by account number first and then by sale date. This should ensure
#that all duplicate account numbers are next to each other with the largest date on top.
fullList.sort(key = lambda item: (item[AccntIndex], item[SalesDateIndex], item[InstrNumIndex]), reverse = True)

#Go through sorted master list and delete all the files that have matching
#account numbers after the first one is encountered which should leave only the largest date.
for i in range (len(fullList)):
    while (((i+1) < (len(fullList))) and (fullList[i][1] == fullList[i+1][1])):
        del fullList[i+1]

#Open a new CSV File for writing as variable named out_file
##with open('C:/PyTest/CSVTestOut.csv', 'wb') as out_file:
with open(out_file_name, 'wb') as out_file:
    # Create CSV Module writer
    writer = csv.writer(out_file, delimiter = ',')
    # Write header items at top of new CSV
    writer.writerow(header)

    #For each row of the master list, write the info to the new CSV
    for row in fullList:
        writer.writerow(row)


#Close the output CSV
out_file.close()
```

Display Name	Data Type
Input Sales CSV	File
Output Sales CSV	File
Account Index	Long
Sales Date Index	Long
Instrument Number …	Long

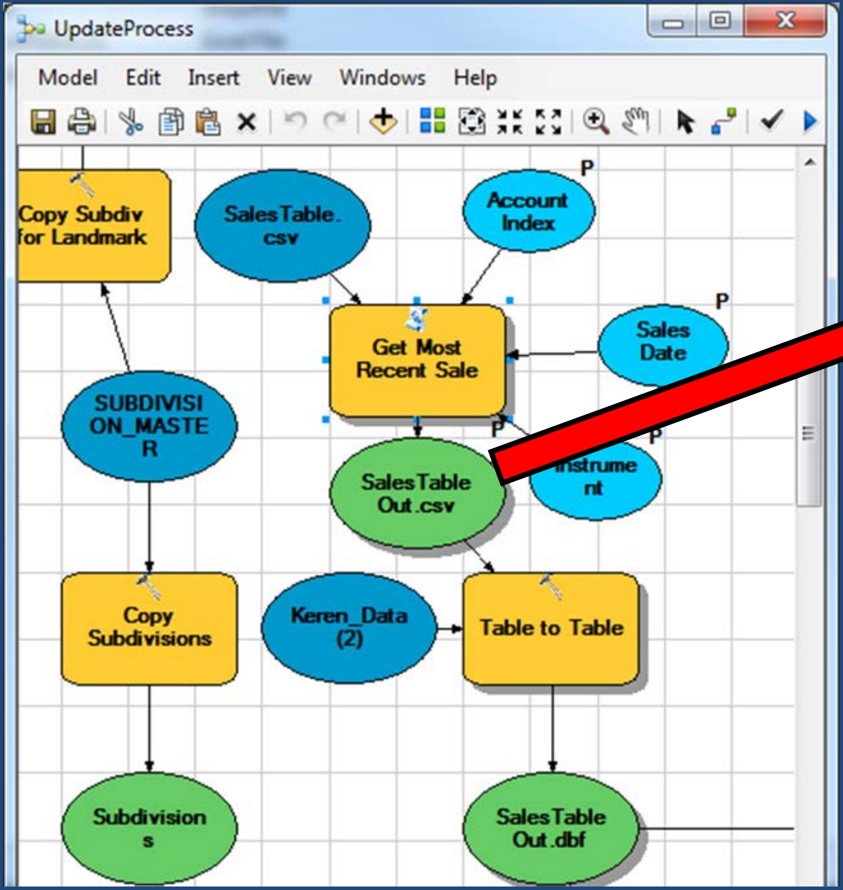
Click any parameter above to see its properties below.

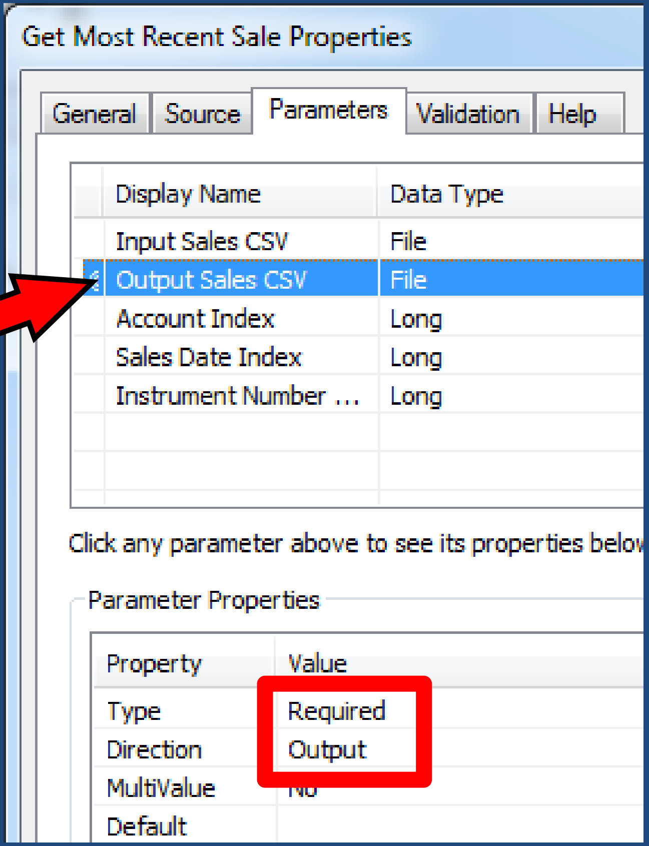
Parameter Properties

Property	Value
Type	Required
Direction	Output
MultiValue	No
Default	
Environment	
Filter	None
Obtained from	

Model Builder

Creating New Tools





Get Most Recent Sale Properties

General | Source | Parameters | Validation | Help

Display Name	Data Type
Input Sales CSV	File
Output Sales CSV	File
Account Index	Long
Sales Date Index	Long
Instrument Number …	Long

Click any parameter above to see its properties belo

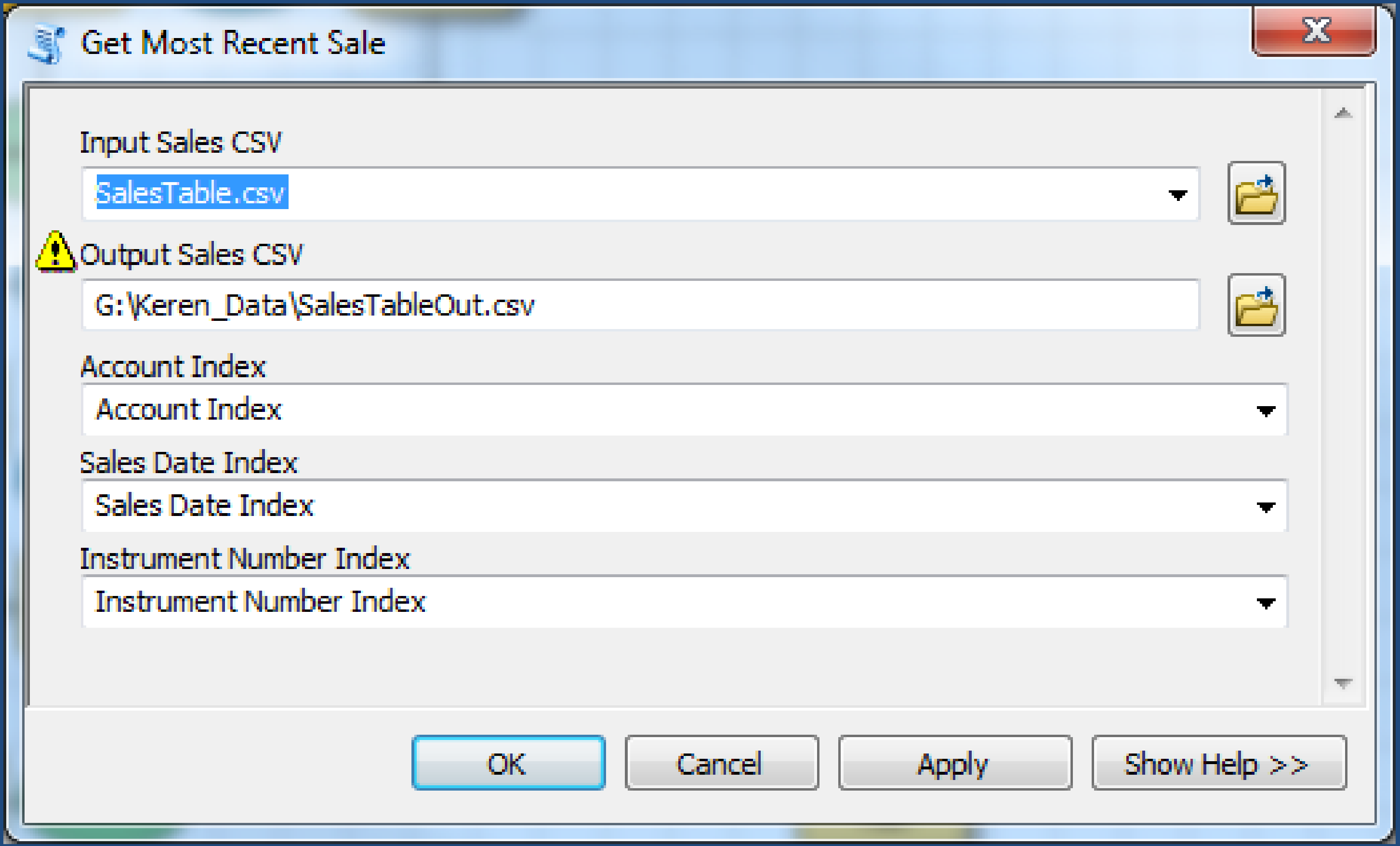
Parameter Properties

Property	Value
Type	Required
Direction	Output
MultiValue	No
Default	

Required/Optional Output Parameters allow user input

Model Builder

Creating New Tools


Get Most Recent Sale

Input Sales CSV
SalesTable.csv

Output Sales CSV
G:\Keren_Data\SalesTableOut.csv

Account Index
Account Index

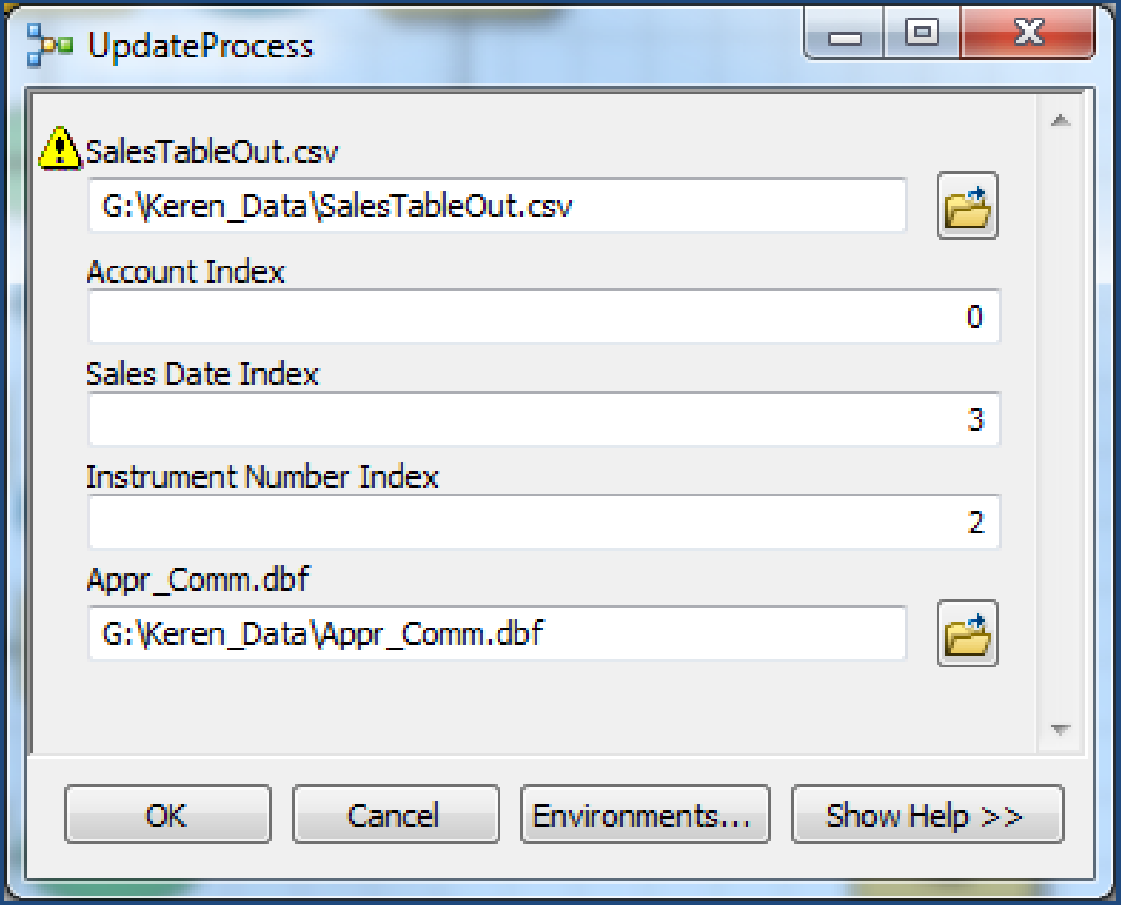
Sales Date Index
Sales Date Index

Instrument Number Index
Instrument Number Index

Standalone Tool Window

Model Builder

Creating New Tools



UpdateProcess

SalesTableOut.csv: G:\Keren_Data\SalesTableOut.csv

Account Index: 0

Sales Date Index: 3

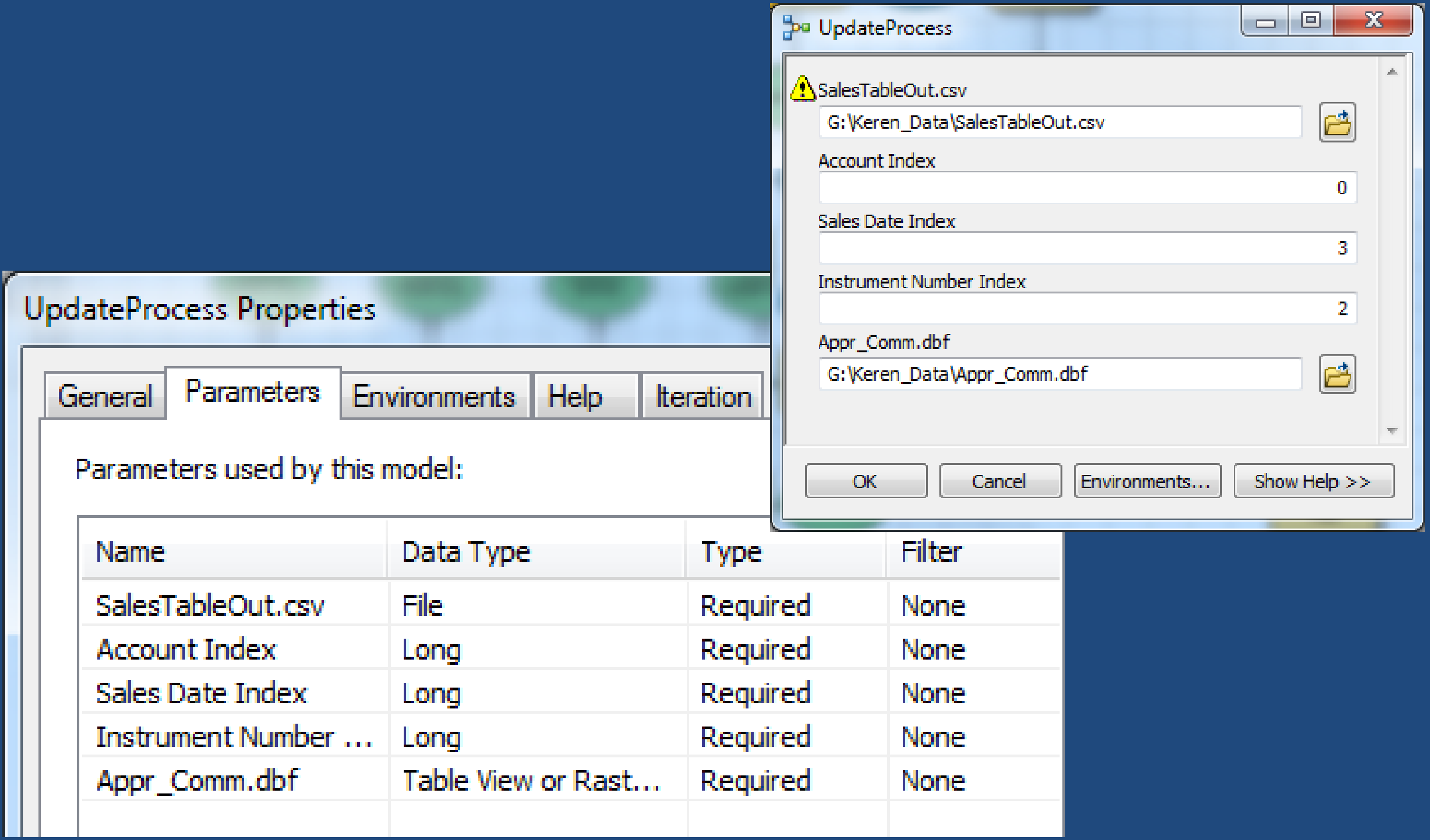
Instrument Number Index: 2

Appr_Comm.dbf: G:\Keren_Data\Appr_Comm.dbf

Model Window

Model Builder

Creating New Tools



UpdateProcess Properties

General | Parameters | Environments | Help | Iteration

Parameters used by this model:

Name	Data Type	Type	Filter
SalesTableOut.csv	File	Required	None
Account Index	Long	Required	None
Sales Date Index	Long	Required	None
Instrument Number ...	Long	Required	None
Appr_Comm.dbf	Table View or Rast...	Required	None

UpdateProcess

SalesTableOut.csv	G:\Keren_Data\SalesTableOut.csv
Account Index	0
Sales Date Index	3
Instrument Number Index	2
Appr_Comm.dbf	G:\Keren_Data\Appr_Comm.dbf

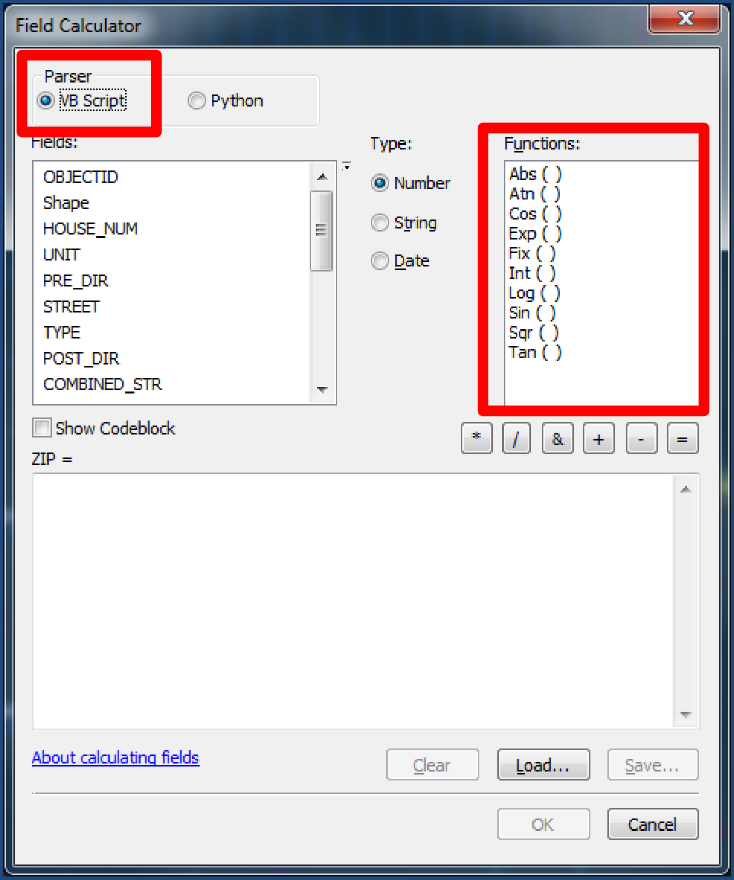
Model Properties

Field Calculator

- Using Python:
  - Allows for calculations to be based on the values in other fields
  - Allows you to deal with any exceptions in one calculation
  - Helps with complex string calculations
  - SAVES TIME!!!

Field Calculator

Word of Caution…
Make sure you have Python Checked



Field Calculator

Parser
[● VB Script] ○ Python

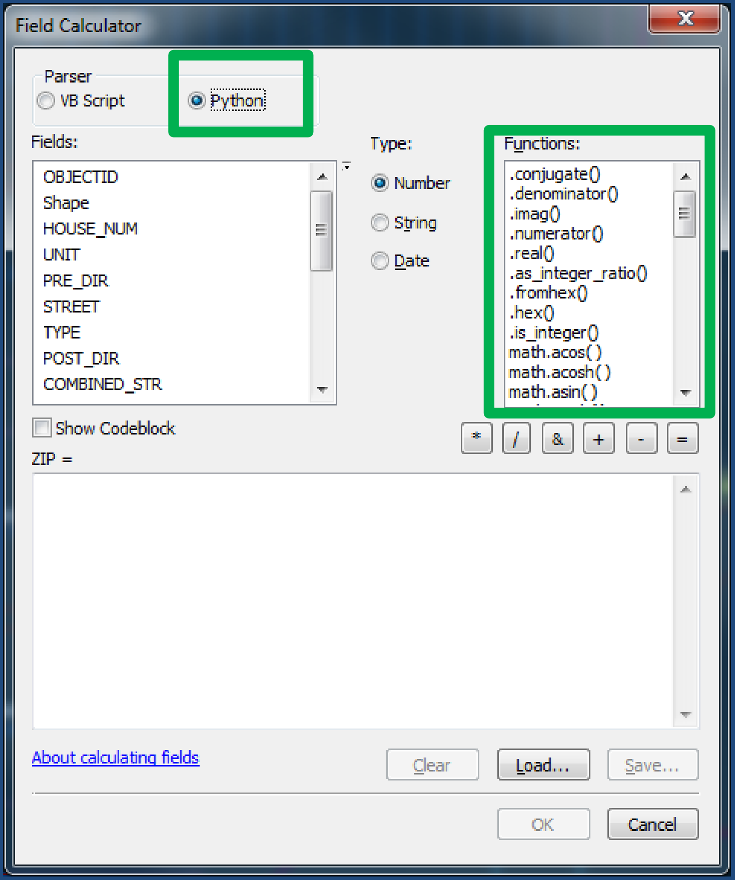
Fields:
OBJECTID
Shape
HOUSE_NUM
UNIT
PRE_DIR
STREET
TYPE
POST_DIR
COMBINED_STR

Type:
[● Number] ○ String ○ Date

Functions:
Abs ( )
Atn ( )
Cos ( )
Exp ( )
Fix ( )
Int ( )
Log ( )
Sin ( )
Sqr ( )
Tan ( )

☐ Show Codeblock

ZIP =



Field Calculator

Parser
○ VB Script [● Python]

Fields:
OBJECTID
Shape
HOUSE_NUM
UNIT
PRE_DIR
STREET
TYPE
POST_DIR
COMBINED_STR

Type:
[● Number] ○ String ○ Date

Functions:
.conjugate()
.denominator()
.imag()
.numerator()
.real()
.as_integer_ratio()
.fromhex()
.hex()
.is_integer()
math.acos( )
math.acosh( )
math.asin( )

☐ Show Codeblock

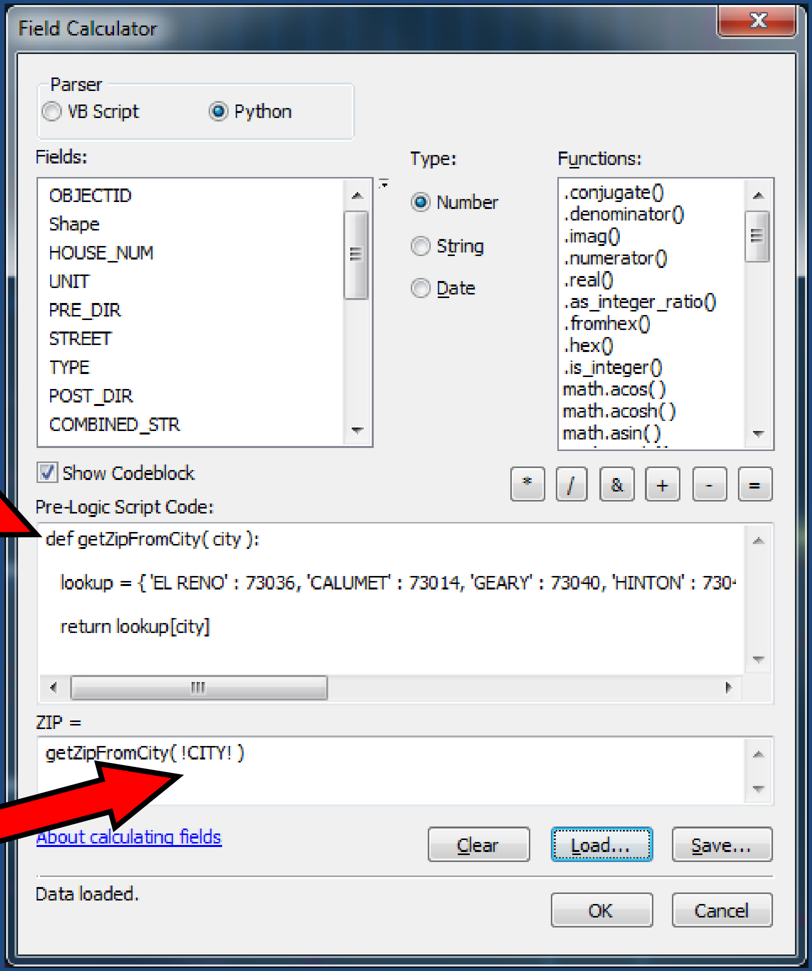
ZIP =

Field Calculator

Calculate One Field from Another

The ‘Codeblock’ is the script itself

Script is called in the bottom box with the field you want to send it



Parser: ○ VB Script [● Python]

Type: [● Number] ○ String ○ Date

[✓] Show Codeblock

Pre-Logic Script Code:

```
def getZipFromCity( city ):
  lookup = { 'EL RENO' : 73036, 'CALUMET' : 73014, 'GEARY' : 73040, 'HINTON' : 730
  return lookup[city]
```

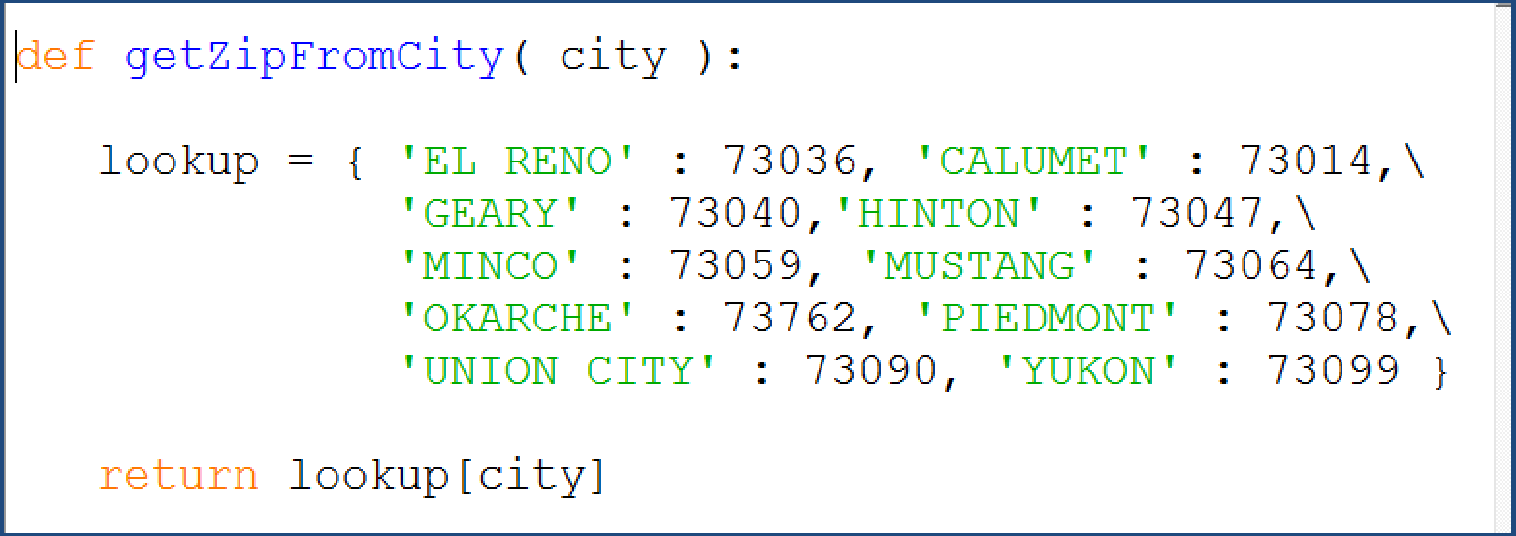
ZIP =

```
getZipFromCity( !CITY! )
```

Data loaded.

Field Calculator

Calculate One Field from Another

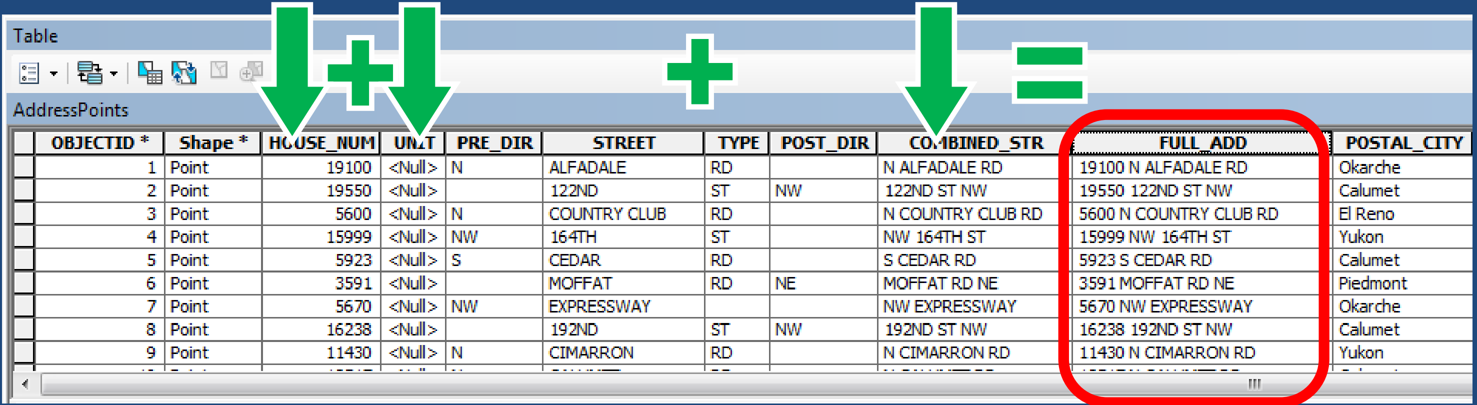


```
def getZipFromCity( city ):

    lookup = { 'EL RENO' : 73036, 'CALUMET' : 73014,\
               'GEARY' : 73040,'HINTON' : 73047,\
               'MINCO' : 73059, 'MUSTANG' : 73064,\
               'OKARCHE' : 73762, 'PIEDMONT' : 73078,\
               'UNION CITY' : 73090, 'YUKON' : 73099 }

    return lookup[city]
```

Field Calculator code in IDLE with ZIP code lookup dictionary

Field Calculator

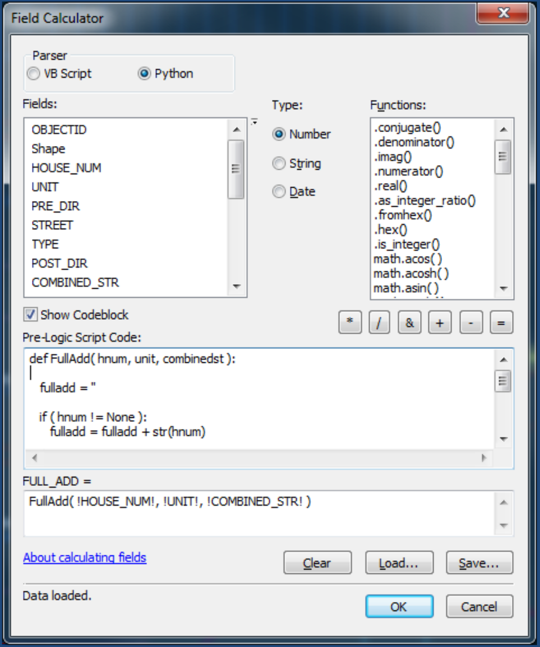
Combining Strings



Table

AddressPoints

OBJECTID *	Shape *	HOUSE_NUM	UNIT	PRE_DIR	STREET	TYPE	POST_DIR	COMBINED_STR	FULL_ADD	POSTAL_CITY
1	Point	19100	<Null>	N	ALFADALE	RD		N ALFADALE RD	19100 N ALFADALE RD	Okarche
2	Point	19550	<Null>		122ND	ST	NW	122ND ST NW	19550 122ND ST NW	Calumet
3	Point	5600	<Null>	N	COUNTRY CLUB	RD		N COUNTRY CLUB RD	5600 N COUNTRY CLUB RD	El Reno
4	Point	15999	<Null>	NW	164TH	ST		NW 164TH ST	15999 NW 164TH ST	Yukon
5	Point	5923	<Null>	S	CEDAR	RD		S CEDAR RD	5923 S CEDAR RD	Calumet
6	Point	3591	<Null>		MOFFAT	RD	NE	MOFFAT RD NE	3591 MOFFAT RD NE	Piedmont
7	Point	5670	<Null>	NW	EXPRESSWAY			NW EXPRESSWAY	5670 NW EXPRESSWAY	Okarche
8	Point	16238	<Null>		192ND	ST	NW	192ND ST NW	16238 192ND ST NW	Calumet
9	Point	11430	<Null>	N	CIMARRON	RD		N CIMARRON RD	11430 N CIMARRON RD	Yukon



Field Calculator

Parser: ( ) VB Script [✓ Python]

Fields: OBJECTID, Shape, HOUSE_NUM, UNIT, PRE_DIR, STREET, TYPE, POST_DIR, COMBINED_STR

Type: [✓ Number] ( ) String ( ) Date

Functions: .conjugate() .denominator() .imag() .numerator() .real() .as_integer_ratio() .fromhex() .hex() .is_integer() math.acos( ) math.acosh( ) math.asin( )

[✓] Show Codeblock

Pre-Logic Script Code:

```
def FullAdd( hnum, unit, combinedst ):

  fulladd = ''

  if ( hnum != None ):
    fulladd = fulladd + str(hnum)
```

FULL_ADD =

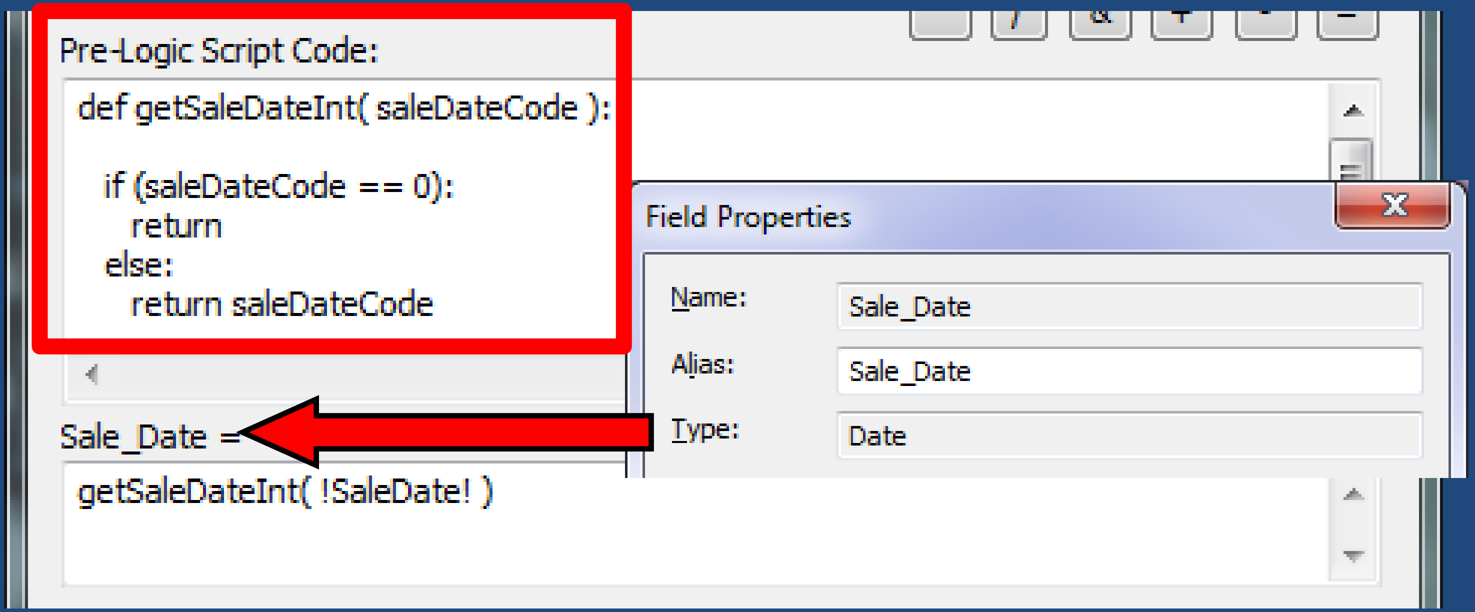
```
FullAdd( !HOUSE_NUM!, !UNIT!, !COMBINED_STR! )
```

Data loaded.

```
def FullAdd( hnum, unit, combinedst ):

    fulladd = ''

    if ( hnum != None ):
        fulladd = fulladd + str(hnum)

    if ( unit != None ):
        fulladd = fulladd + ' ' + unit

    if ( combinedst != None ):
        fulladd = fulladd + ' ' + combinedst

    return fulladd
```

Field Calculator

Error/Exception Handling



Pre-Logic Script Code:

```
def getSaleDateInt( saleDateCode ):

  if (saleDateCode == 0):
     return
  else:
     return saleDateCode
```

Sale_Date =

```
getSaleDateInt( !SaleDate! )
```

Field Properties

Name:	Sale_Date
Alias:	Sale_Date
Type:	Date

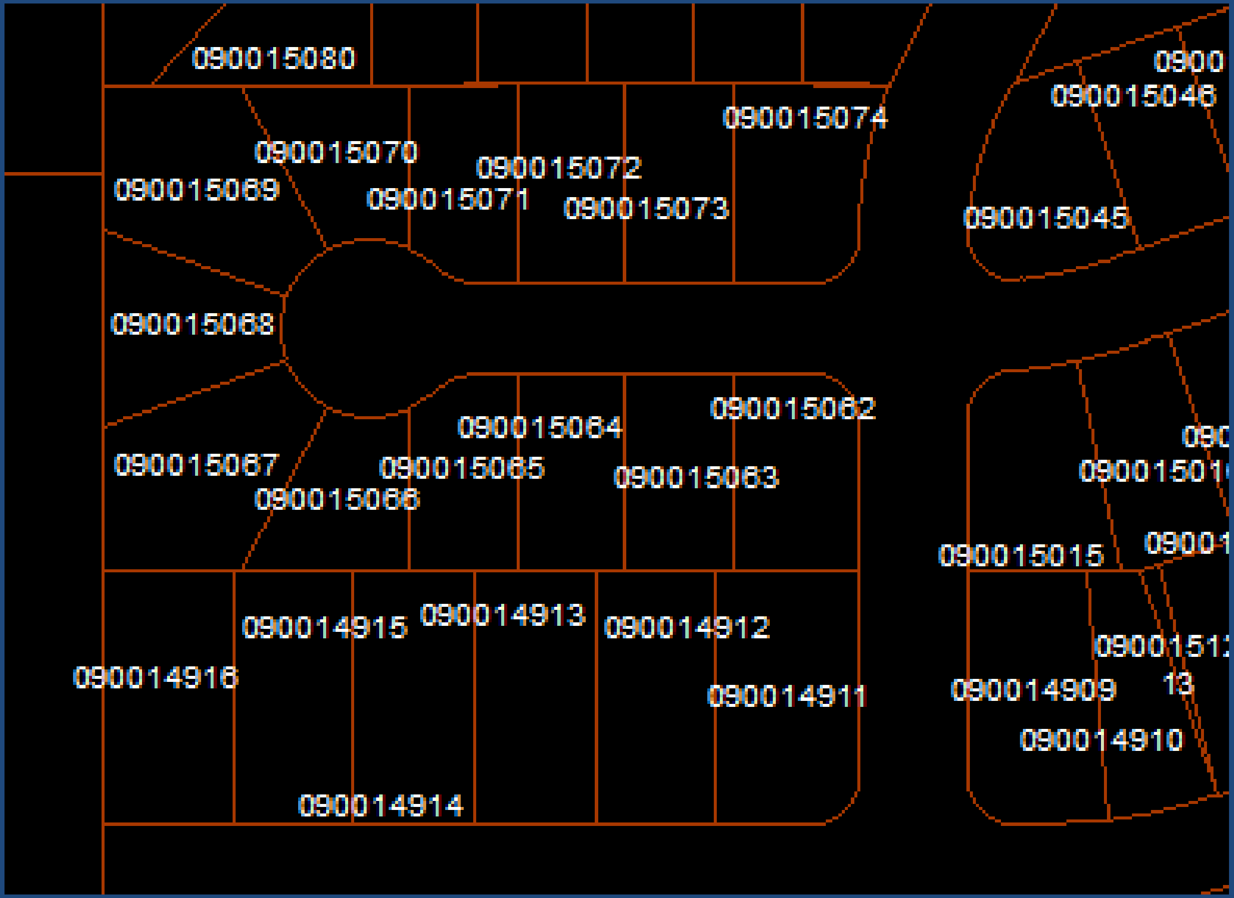
Note that Field Calculator will recognize the Field's type when you return a value

Label Expressions

- Using Python:
  - Allows for adjustments to labels without adjusting the data
  - Allows for labeling based on calculations and error checking

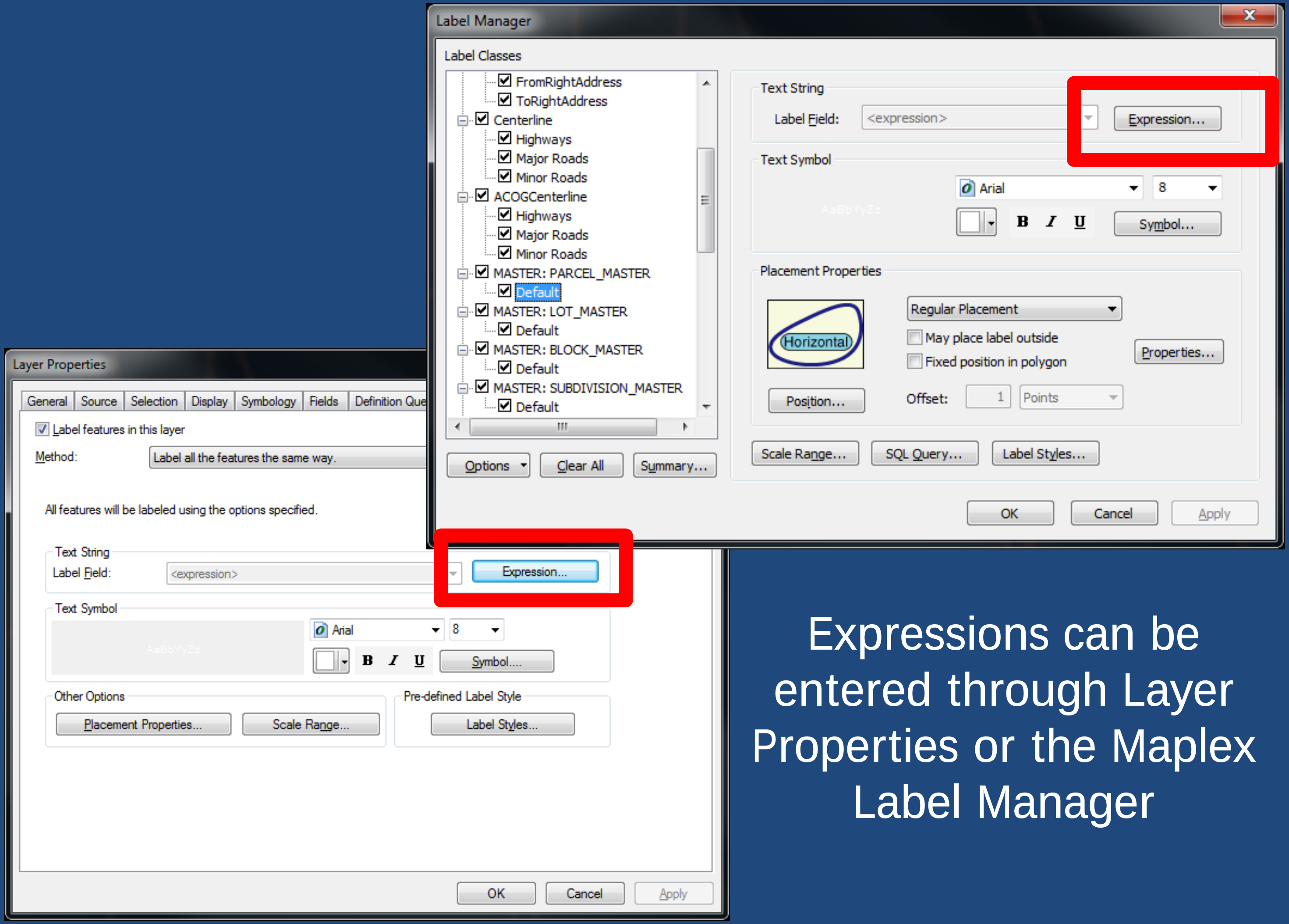
Label Expressions

Adjusting Labels without Adjusting Data



Label Expressions

Adjusting Labels without Adjusting Data


Label Manager

Label Classes
- ☑ FromRightAddress
- ☑ ToRightAddress
- ☑ Centerline
  - ☑ Highways
  - ☑ Major Roads
  - ☑ Minor Roads
- ☑ ACOGCenterline
  - ☑ Highways
  - ☑ Major Roads
  - ☑ Minor Roads
- ☑ MASTER: PARCEL_MASTER
  - ☑ Default
- ☑ MASTER: LOT_MASTER
  - ☑ Default
- ☑ MASTER: BLOCK_MASTER
  - ☑ Default
- ☑ MASTER: SUBDIVISION_MASTER
  - ☑ Default

Text String
Label Field: <expression>

Text Symbol
Arial 8

Placement Properties
Regular Placement
☐ May place label outside
☐ Fixed position in polygon
Offset: 1 Points

Layer Properties

☑ Label features in this layer
Method: Label all the features the same way.

All features will be labeled using the options specified.

Text String
Label Field: <expression>

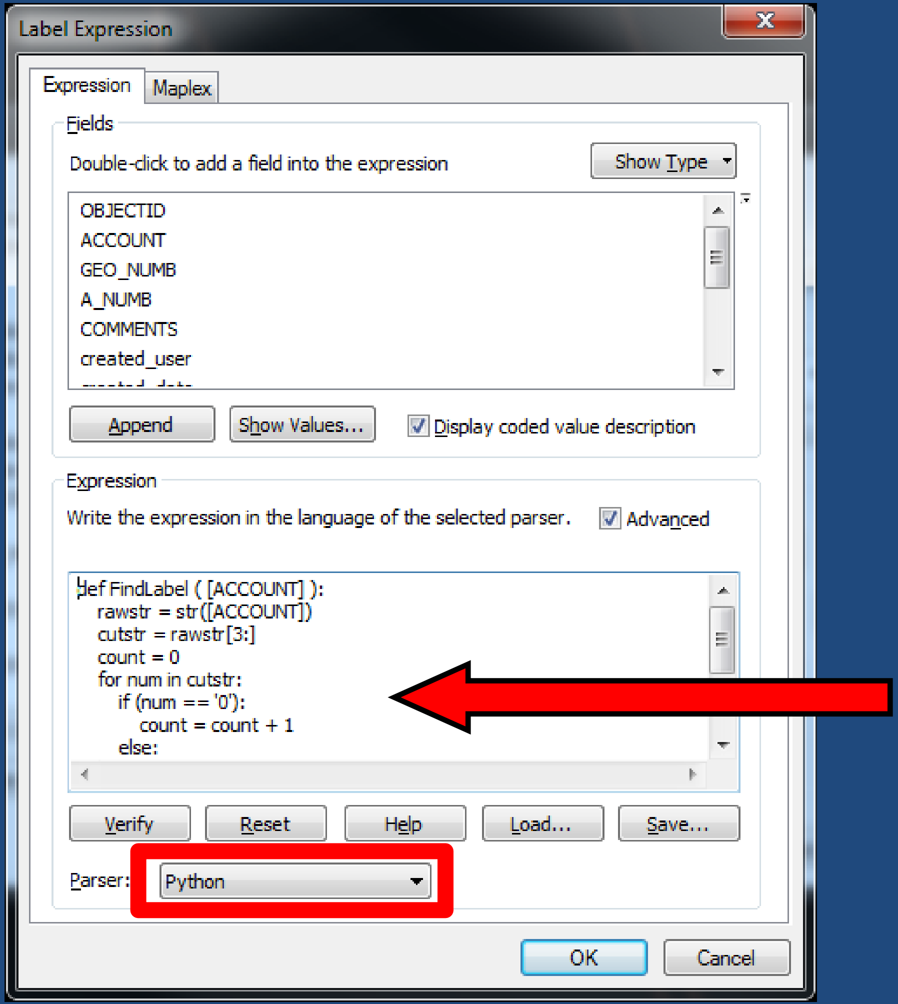
Text Symbol
Arial 8

Other Options
Pre-defined Label Style

Expressions can be entered through Layer Properties or the Maplex Label Manager

Label Expressions

Adjusting Labels without Adjusting Data


Label Expression
Expression
Maplex
Fields
Double-click to add a field into the expression
Show Type
Append
Show Values...
Display coded value description
Expression
Write the expression in the language of the selected parser.
Advanced
Verify
Reset
Help
Load...
Save...
OK
Cancel

Fields: OBJECTID, ACCOUNT, GEO_NUMB, A_NUMB, COMMENTS, created_user

```
def FindLabel ( [ACCOUNT] ):
  rawstr = str([ACCOUNT])
  cutstr = rawstr[3:]
  count = 0
  for num in cutstr:
    if (num == '0'):
      count = count + 1
    else:
```

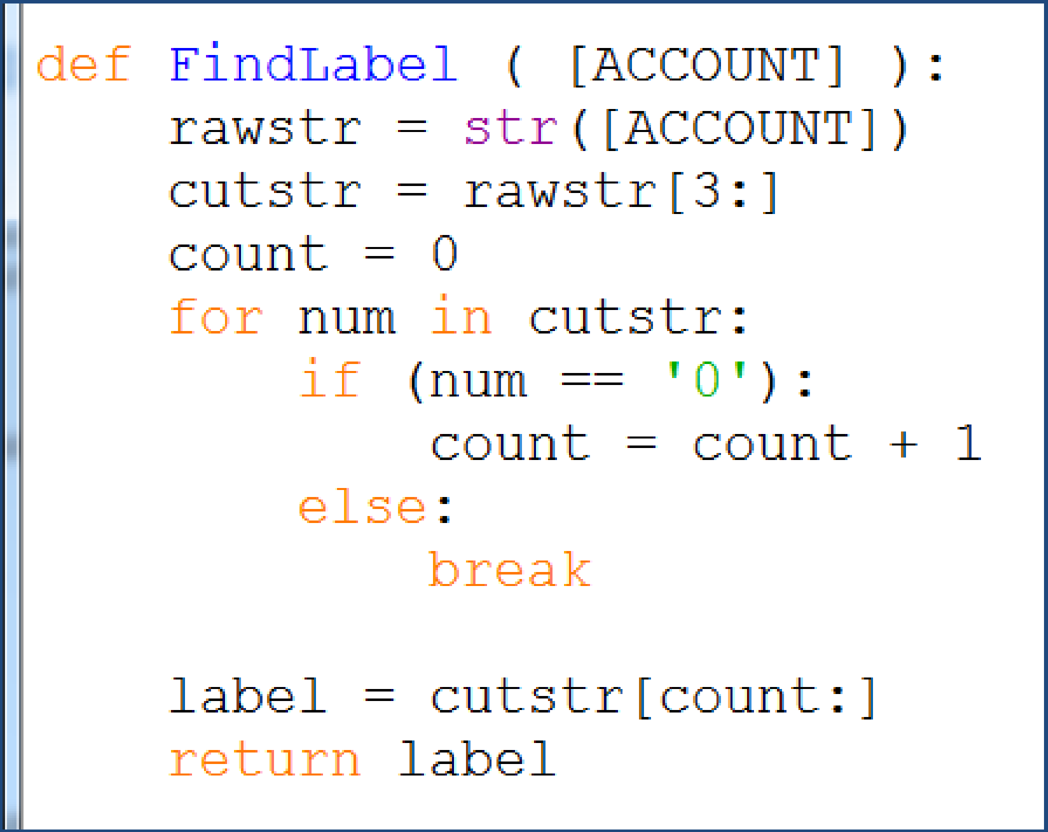
Parser: Python

Label Expression Dialog Window

Only one area for code, no calling of function like Field Calculator

Label Expressions

Adjusting Labels without Adjusting Data

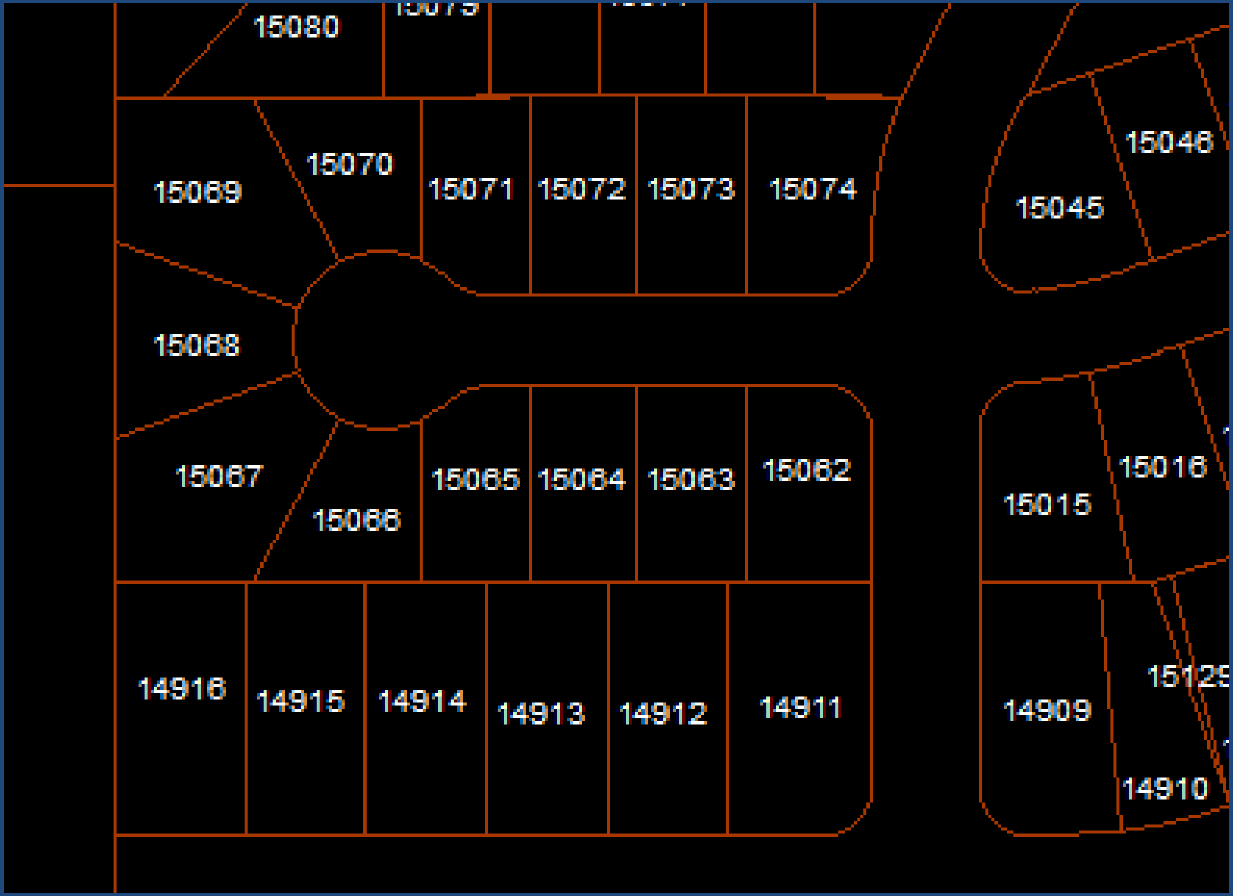


```
def FindLabel ( [ACCOUNT] ):
    rawstr = str([ACCOUNT])
    cutstr = rawstr[3:]
    count = 0
    for num in cutstr:
        if (num == '0'):
            count = count + 1
        else:
            break

    label = cutstr[count:]
    return label
```

Function must be called “FindLabel”
Field names must be in square brackets

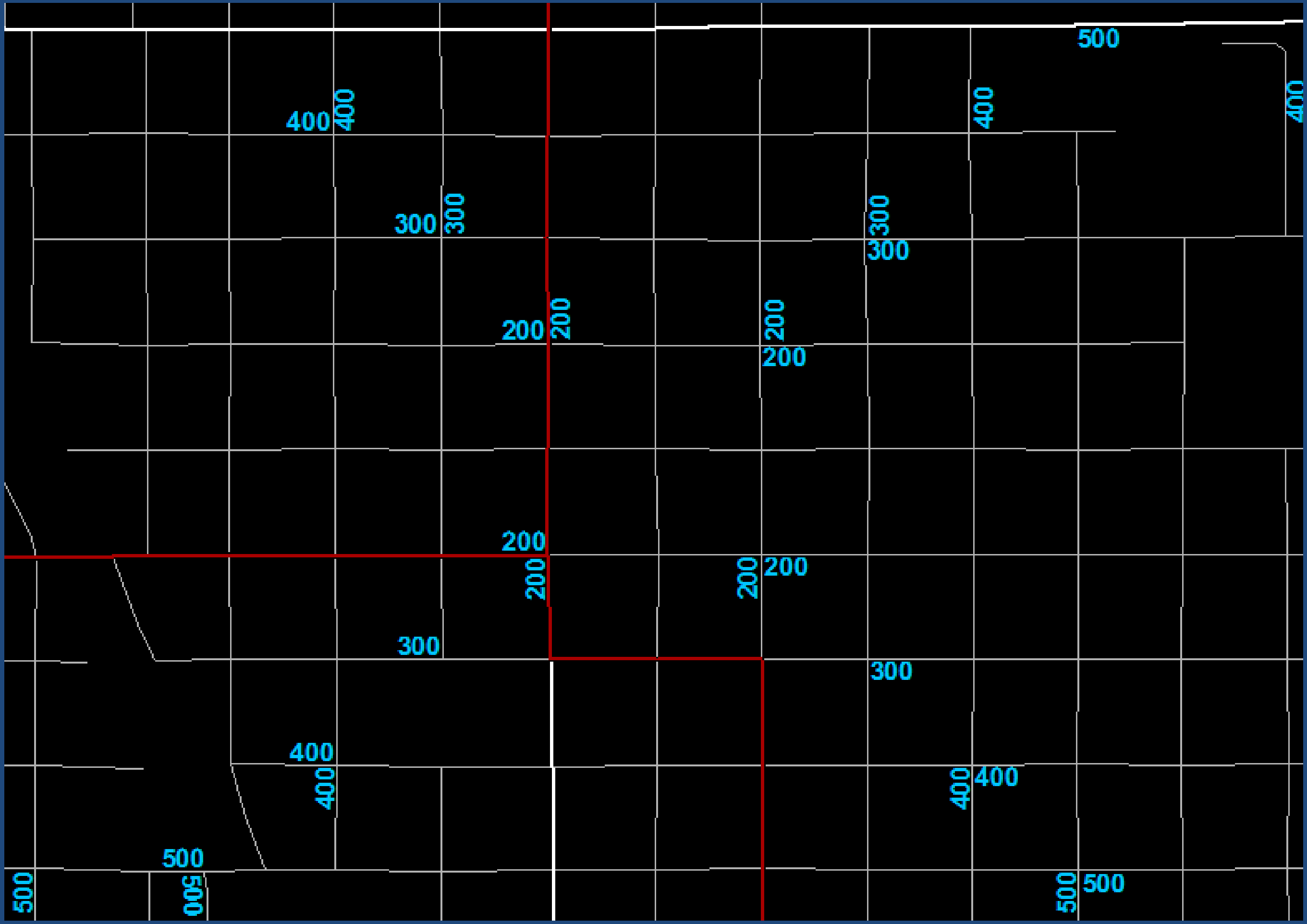
Label Expressions

Adjusting Labels without Adjusting Data



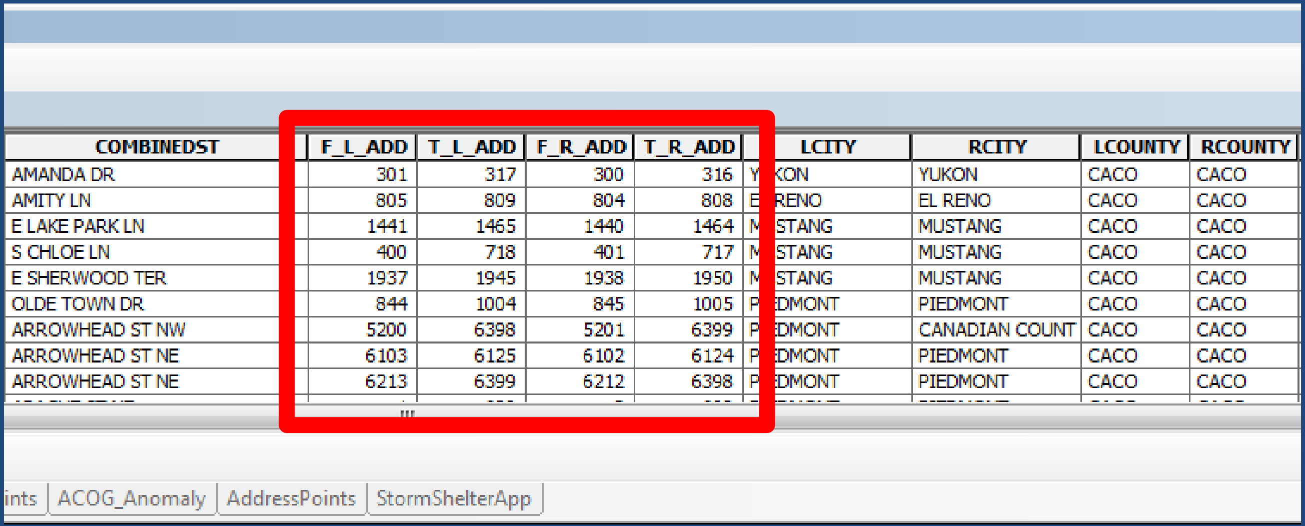
Label Expressions

Creating Dynamic Labeling



Label Expressions

Creating Dynamic Labeling

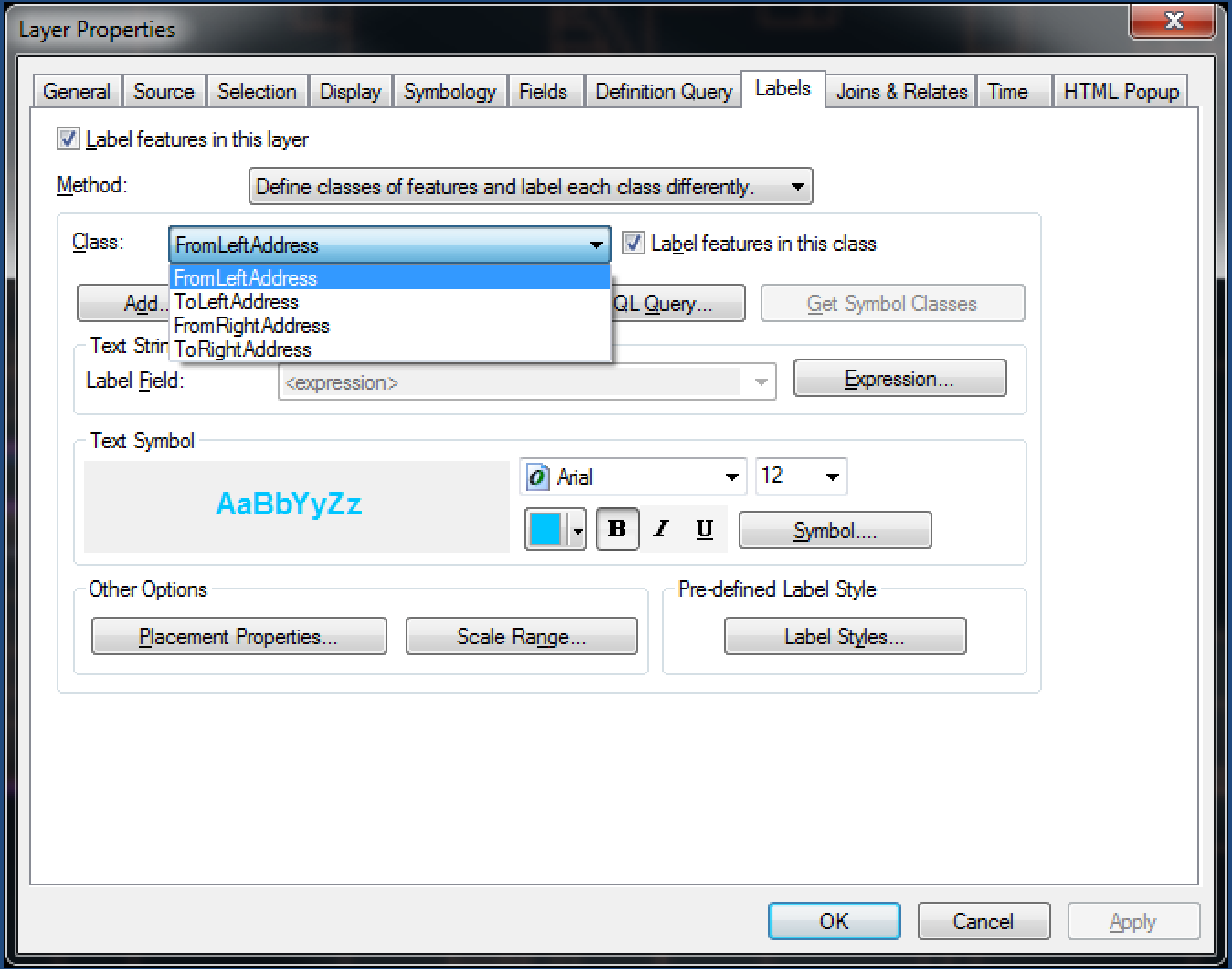


COMBINEDST	F_L_ADD	T_L_ADD	F_R_ADD	T_R_ADD	LCITY	RCITY	LCOUNTY	RCOUNTY
AMANDA DR	301	317	300	316	Y KON	YUKON	CACO	CACO
AMITY LN	805	809	804	808	E RENO	EL RENO	CACO	CACO
E LAKE PARK LN	1441	1465	1440	1464	M STANG	MUSTANG	CACO	CACO
S CHLOE LN	400	718	401	717	M STANG	MUSTANG	CACO	CACO
E SHERWOOD TER	1937	1945	1938	1950	M STANG	MUSTANG	CACO	CACO
OLDE TOWN DR	844	1004	845	1005	P EDMONT	PIEDMONT	CACO	CACO
ARROWHEAD ST NW	5200	6398	5201	6399	P EDMONT	CANADIAN COUNT	CACO	CACO
ARROWHEAD ST NE	6103	6125	6102	6124	P EDMONT	PIEDMONT	CACO	CACO
ARROWHEAD ST NE	6213	6399	6212	6398	P EDMONT	PIEDMONT	CACO	CACO

Python Label Expression finds lowest non-zero address range to create label

Label Expressions

Creating Dynamic Labeling



Layer Properties

☑ Label features in this layer

Method: Define classes of features and label each class differently.

Class: FromLeftAddress ☑ Label features in this class

FromLeftAddress
ToLeftAddress
FromRightAddress
ToRightAddress

Text String

Label Field: <expression>

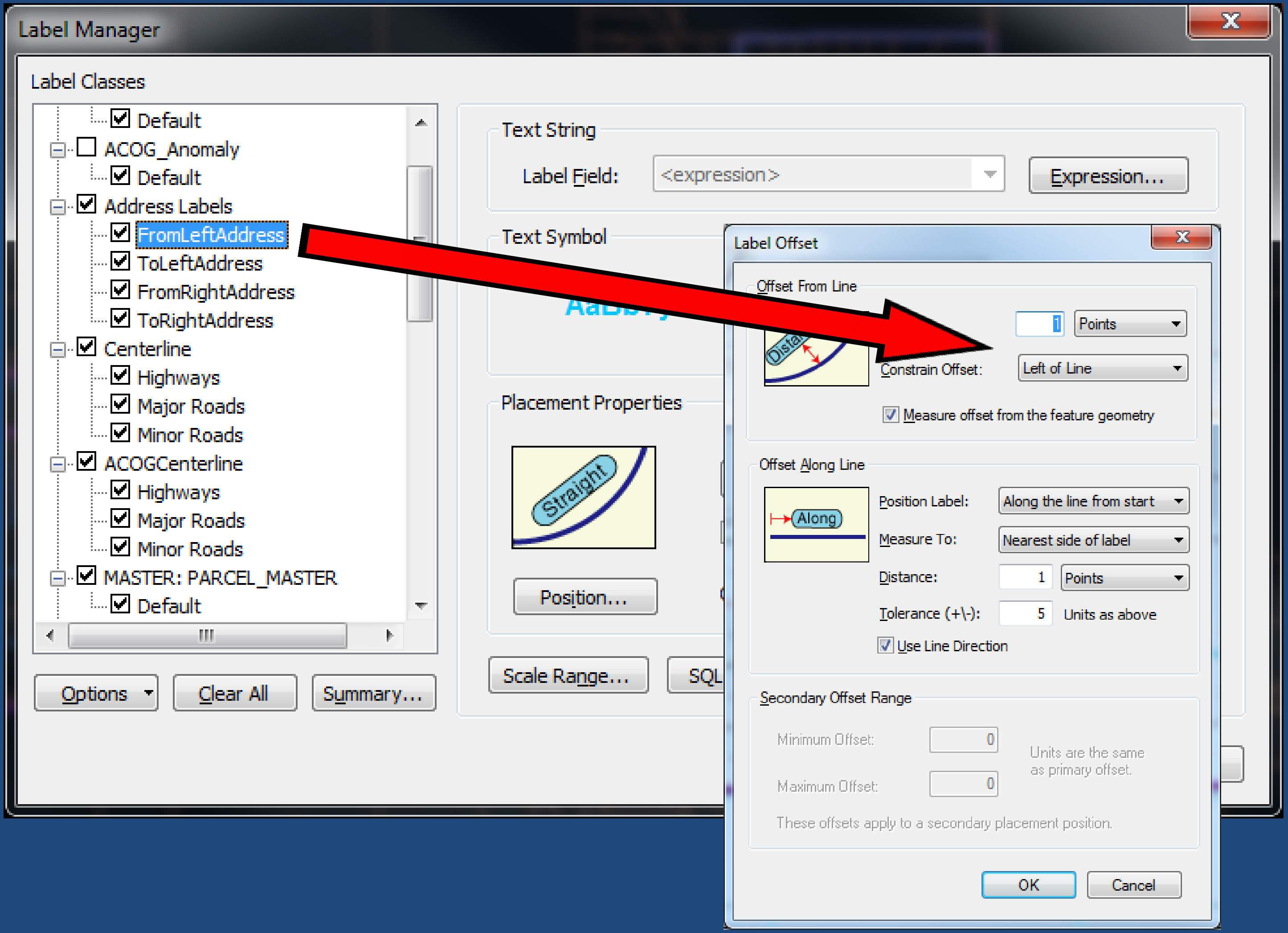
Text Symbol

AaBbYyZz

Arial 12

Label Expressions

Creating Dynamic Labeling



Label Manager

Label Classes

- ☑ Default
- ☐ ACOG_Anomaly
  - ☑ Default
- ☑ Address Labels
  - ☑ FromLeftAddress
  - ☑ ToLeftAddress
  - ☑ FromRightAddress
  - ☑ ToRightAddress
- ☑ Centerline
  - ☑ Highways
  - ☑ Major Roads
  - ☑ Minor Roads
- ☑ ACOGCenterline
  - ☑ Highways
  - ☑ Major Roads
  - ☑ Minor Roads
- ☑ MASTER: PARCEL_MASTER
  - ☑ Default

Text String

Label Field: <expression>

Label Offset

Offset From Line

- Offset: 1 Points
- Constrain Offset: Left of Line
- ☑ Measure offset from the feature geometry

Offset Along Line

- Position Label: Along the line from start
- Measure To: Nearest side of label
- Distance: 1 Points
- Tolerance (+\-): 5 Units as above
- ☑ Use Line Direction

Secondary Offset Range

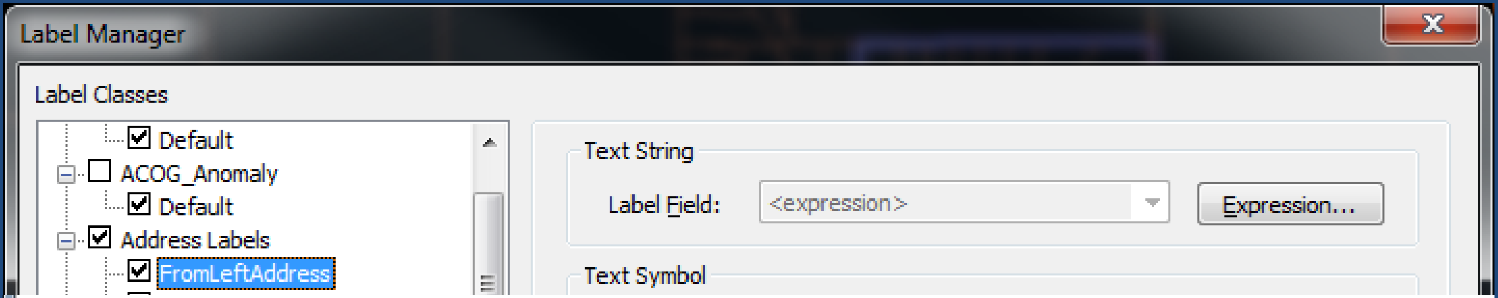
- Minimum Offset: 0
- Maximum Offset: 0

Units are the same as primary offset.

These offsets apply to a secondary placement position.

Label Expressions

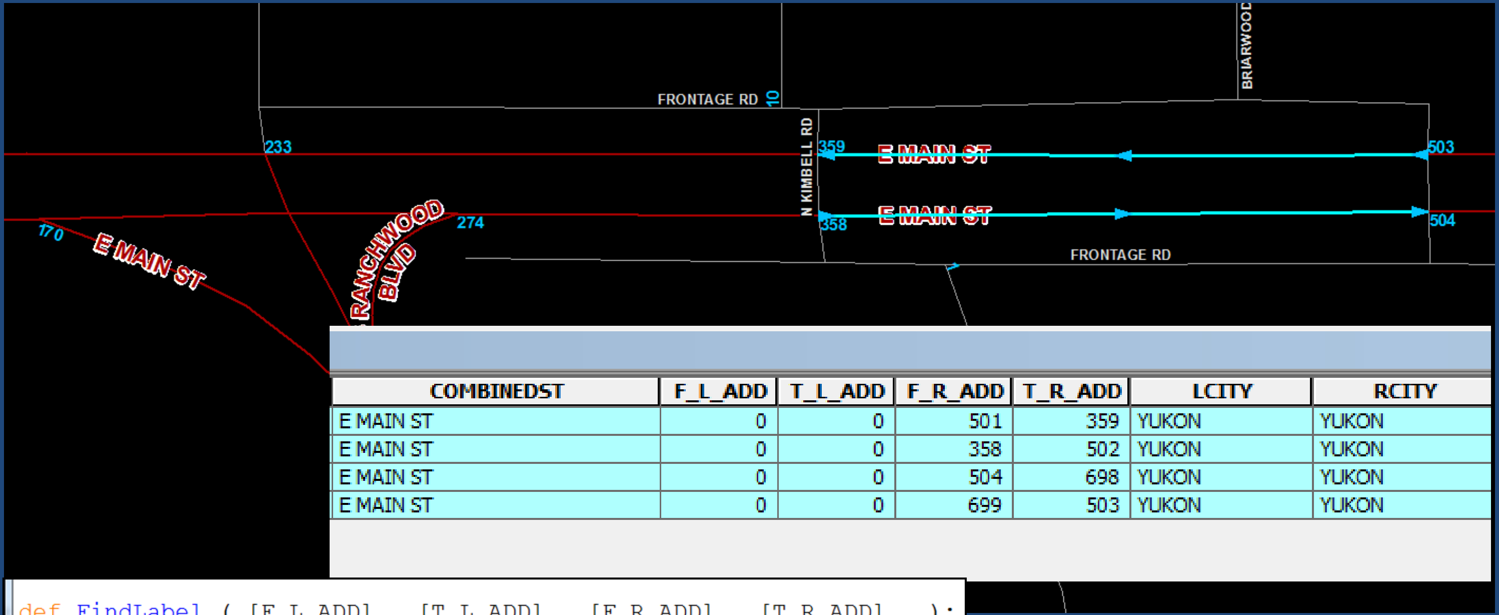
Creating Dynamic Labeling



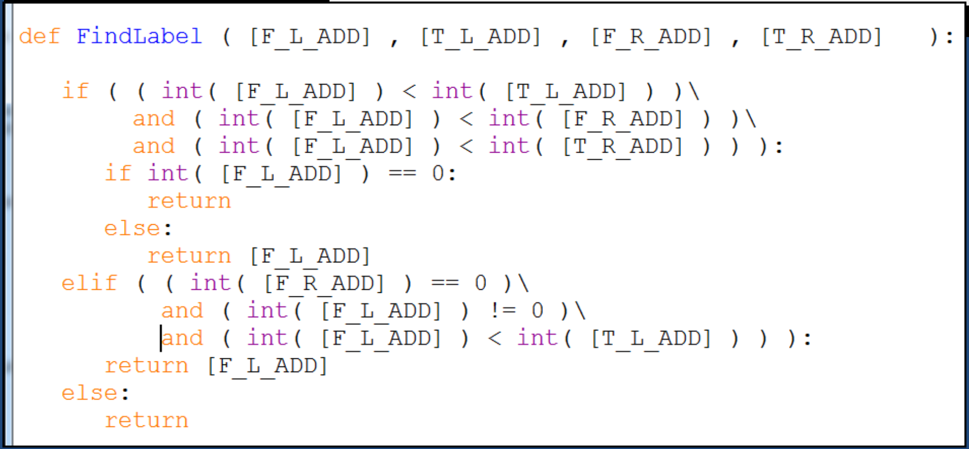
```
def FindLabel ( [F_L_ADD] , [T_L_ADD] , [F_R_ADD] , [T_R_ADD]   ):

   if ( ( int( [F_L_ADD] ) < int( [T_L_ADD] ) )\
        and ( int( [F_L_ADD] ) < int( [F_R_ADD] ) )\
        and ( int( [F_L_ADD] ) < int( [T_R_ADD] ) ) ):
      if int( [F_L_ADD] ) == 0:
         return
      else:
         return [F_L_ADD]
   elif ( ( int( [F_R_ADD] ) == 0 )\
          and ( int( [F_L_ADD] ) != 0 )\
          and ( int( [F_L_ADD] ) < int( [T_L_ADD] ) ) ):
      return [F_L_ADD]
   else:
      return
```

Label Expressions

Creating Dynamic Labeling



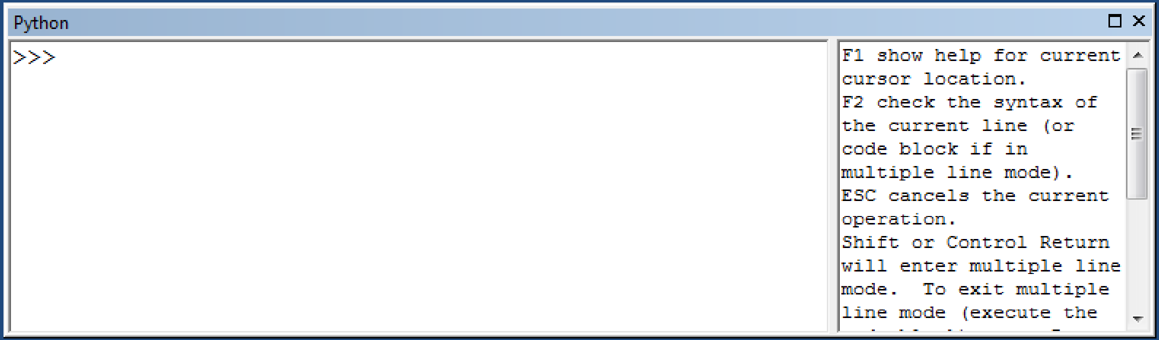
COMBINEDST	F_L_ADD	T_L_ADD	F_R_ADD	T_R_ADD	LCITY	RCITY
E MAIN ST	0	0	501	359	YUKON	YUKON
E MAIN ST	0	0	358	502	YUKON	YUKON
E MAIN ST	0	0	504	698	YUKON	YUKON
E MAIN ST	0	0	699	503	YUKON	YUKON



```
def FindLabel ( [F_L_ADD] , [T_L_ADD] , [F_R_ADD] , [T_R_ADD]   ):

   if ( ( int( [F_L_ADD] ) < int( [T_L_ADD] ) )\
        and ( int( [F_L_ADD] ) < int( [F_R_ADD] ) )\
        and ( int( [F_L_ADD] ) < int( [T_R_ADD] ) ) ):
      if int( [F_L_ADD] ) == 0:
         return
      else:
         return [F_L_ADD]
   elif ( ( int( [F_R_ADD] ) == 0 )\
          and ( int( [F_L_ADD] ) != 0 )\
          and ( int( [F_L_ADD] ) < int( [T_L_ADD] ) ) ):
      return [F_L_ADD]
   else:
      return
```

Other Uses for Python

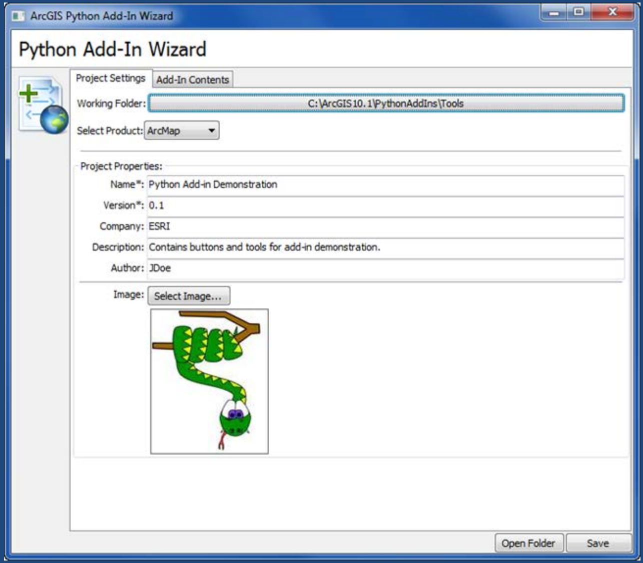
- Command Line



```
>>>
```

```
F1 show help for current
cursor location.
F2 check the syntax of
the current line (or
code block if in
multiple line mode).
ESC cancels the current
operation.
Shift or Control Return
will enter multiple line
mode.  To exit multiple
line mode (execute the
```

- Python Add-Ins



Working Folder: C:\ArcGIS10.1\PythonAddIns\Tools
Select Product: ArcMap
Project Properties:
Name*: Python Add-in Demonstration
Version*: 0.1
Company: ESRI
Description: Contains buttons and tools for add-in demonstration.
Author: JDoe
Image:

Questions

Joel Foster

GIS Coordinator

Canadian County Assessor’s Office

200 N Choctaw Ave.

El Reno, OK 73036

405-295-6331

fosterj@canadiancounty.org