Scripting with Arc GIS 10: Introducing Python geo-processing

A GIS Lecture tutorial

Prof. Yuji MURAYAMA, Ph.D
Kondwani Godwin MUNTHALI, MSc.

2012

Introduction - definitions

- What are we talking about? What is a script?
  - program written for a “software environment” that automates the execution of tasks which could alternatively be executed one by one by a human operator
    - i.e. Scripts are used to automate time-consuming or complex workflows
  - It is a form of programming language that is “interpreted” rather than “compiled”
    - An interpreter executes one command line at a time while a compiler compiles all the lines of commands into an executable file before executing them. Ok! Enough of jargons, google “compiler vs interpreter” if you want to learn more

Introduction – who can script?

- Did you say scripting is a form of programming!? I am no programmer!
  - Relax!! Scripting isn’t just for professional computer programmers.
    - there are much more powerful and complicated tools for professional programmers to use than scripts (with all due respect to scripts anyway)
  - In fact you don’t need to be visionary or adventurous to try scripting, you just need to be the type of person who wants to save some time. SIMPLE!

Introduction…ok!! but why should I script really?

- I guess the importance of scripts ought to have been obvious in the definition but anyway, imagine:
  - You have to perform a series of data management tasks on a regular basis
  - You have to perform a series of geo-processing tasks on several computers or several data sets
  - You want to consolidate and organize the output you get from the computer
  - You want to run tasks when you're not there to interact with the computer
  - You need to ensure the exact same actions are repeated each time a task is run
- Convinced? May be not! Lets try the fun part of it. May be not that too because it's still only the fanatics who think scripting is fun anyway!

Introduction… anyway what are we trying to achieve?

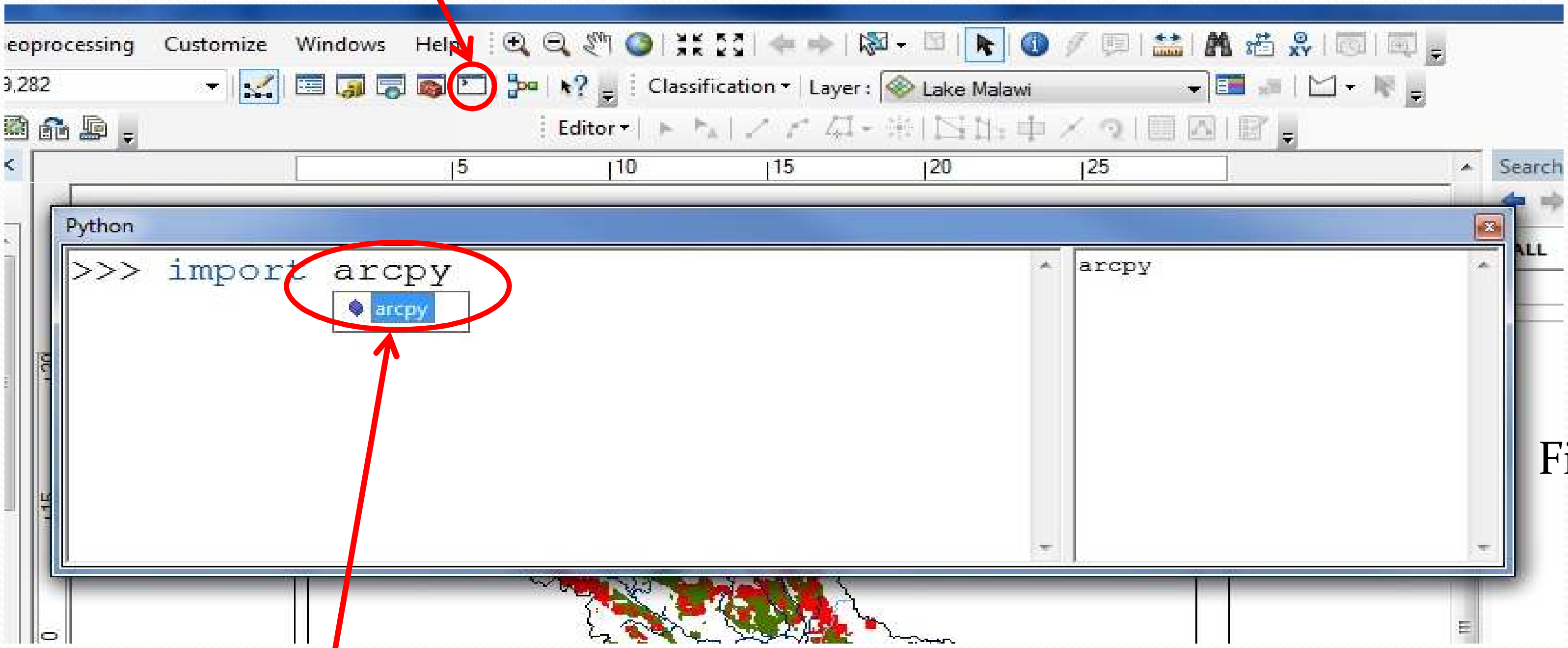
- Objective
  - The aim of this lecture tutorial is to learn how to build and use python scripts for geo-processing in ArcGIS 10

What should we know about python?

- Python is free, cross platform, open source, stable, mature, simple, and powerful
- In ArcGIS?
  - Python is pervasive in ArcGIS 10.
  - It replaces Visual Basic for Applications (VBA) in the field calculator (although you may continue to use VBScript).

What should we know about python?

- In ArcGIS? ...Cont'd
  - The Python Window replaces the Command Line window.



```
>>> import arcpy
```

Fig 1

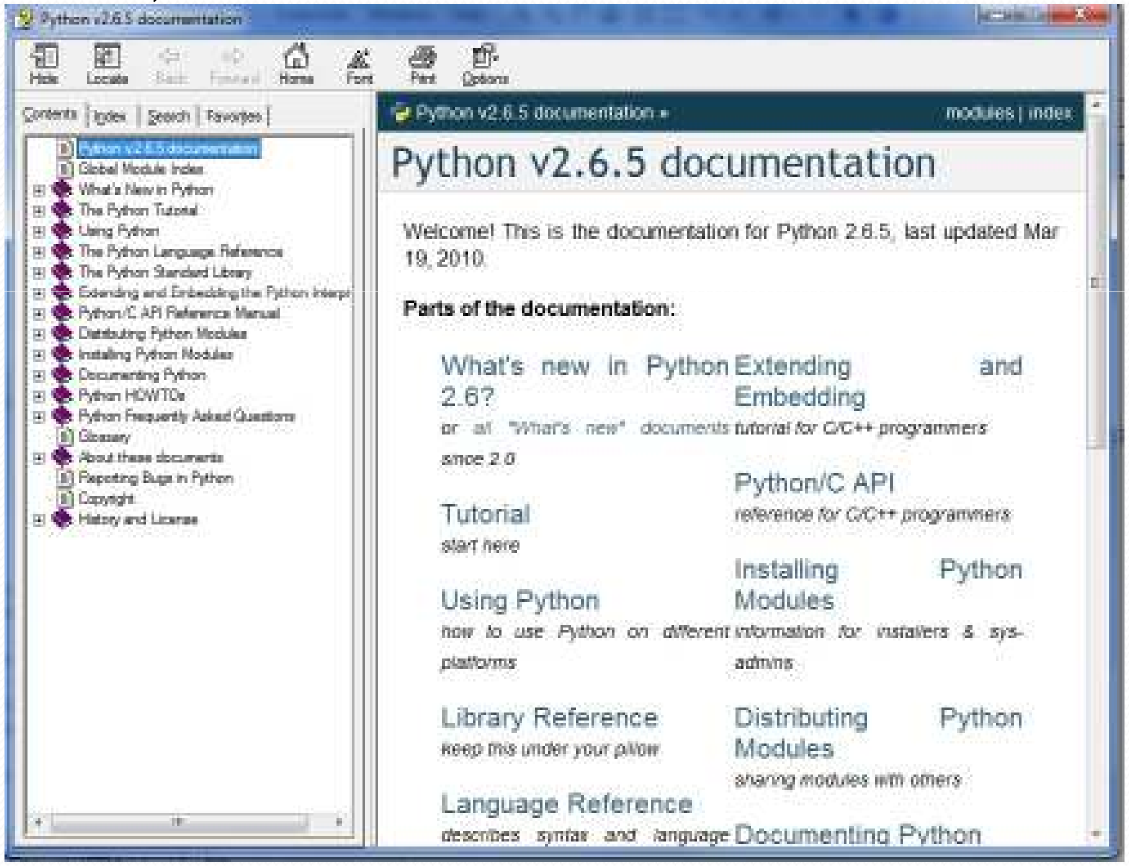
  - All geo-processing tools are available as Python through the "ArcPy" module that ships with ArcGIS 10

What should we know about python?

- In ArcGIS python geo-processing
  - The function names use the ArcGIS toolset and tool name (e.g., CreateFeatureClass_management).
  - ArcPy honors the ArcGIS licensing system, so tool access is identical in desktop applications and Python.
    - I.e. if a particular geo-processing module is not licensed under desktop application it will not also be licensed in ArcPy (Python) and vice versa
  - ArcPy provides access to geo-processing environment settings.
  - Custom user script tools may be loaded into the ArcPy geo-processing environment and accessed and run just like the system tools that are installed with ArcGIS.

What should we know about python?

- Hold it there!! Lets not get ahead of ourselves
  - Python Manuals? (Start>…>ArcGIS>Python 2.6>Python Manuals).


Python v2.6.5 documentation
Hide
Locate
Back
Forward
Home
Font
Print
Options
Contents
Index
Search
Favorites
Python v2.6.5 documentation
Global Module Index
What's New in Python
The Python Tutorial
Using Python
The Python Language Reference
The Python Standard Library
Extending and Embedding the Python Interp
Python/C API Reference Manual
Distributing Python Modules
Installing Python Modules
Documenting Python
Python HOWTOs
Python Frequently Asked Questions
Glossary
About these documents
Reporting Bugs in Python
Copyright
History and License
Python v2.6.5 documentation »
modules | index

Python v2.6.5 documentation

Welcome! This is the documentation for Python 2.6.5, last updated Mar 19, 2010.

Parts of the documentation:

What's new in Python 2.6?
or all "What's new" documents since 2.0

Tutorial
start here

Using Python
how to use Python on different platforms

Library Reference
keep this under your pillow

Language Reference
describes syntax and language

Extending and Embedding
tutorial for C/C++ programmers

Python/C API
reference for C/C++ programmers

Installing Python Modules
information for installers & sys-admins

Distributing Python Modules
sharing modules with others

Documenting Python

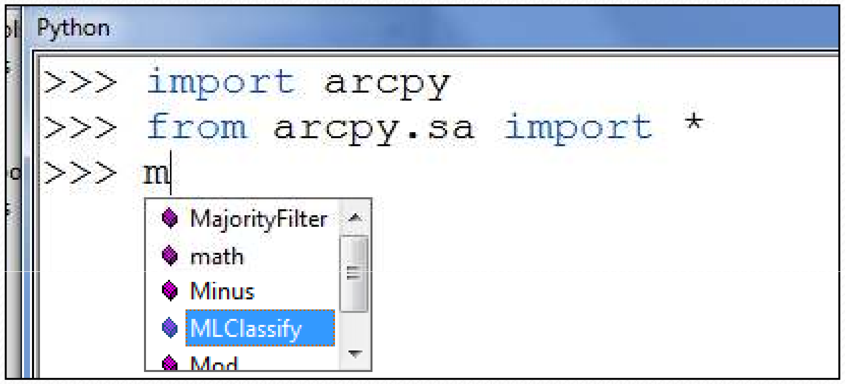
Fig. 2 Python Introduction Help screen

Lets get our hands dirty then, shall we?

- We will use “Image Classification” geo-processing tool as our example
  - **Task:** Imagine you are to classify satellite images for 10 different areas.
  - **Data required:** Some satellite image (You can download one for free here: https://earthexplorer.usgs.gov/) – we will use one for Johannesburg city for 2010
  - **Assumptions:**
    - This tutorial assumes you have already prepared a signature file for the RS image you will be using

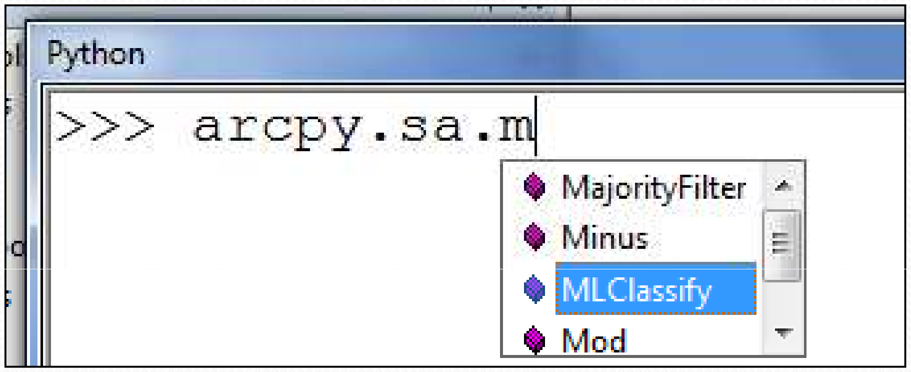
Getting our hands dirty…

- **Inputs:** your satellite image and signature (.gsg) file
- Accessing the tools
  - You can either “import” or use the “dot notation”.
- At the end of each command press ‘return key’ (Enter)



```
>>> import arcpy
>>> from arcpy.sa import *
>>> m
```

a



```
>>> arcpy.sa.m
```

b

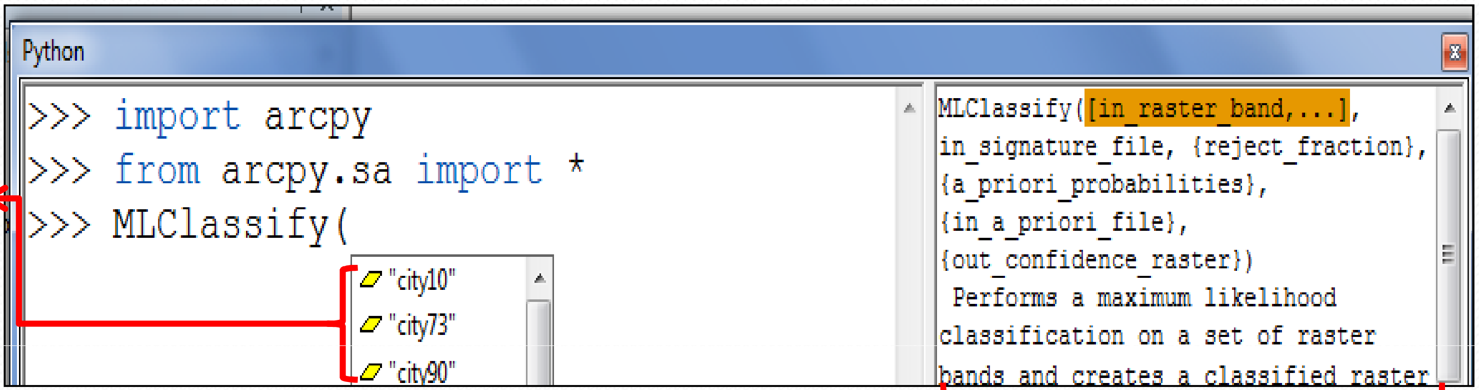
Fig. 3 Accessing the Maximum Likelihood classier (MLClassfiy) through: a) “import”; and b) “dot notation”

Note:
✓In Fig. 3a we have used the “dot notation” to access the spatial analyst (sa) from arcpy (arcpy.sa) and then imported all the tools in the spatial analyst (*)
✓Again, as you type the tools in either approach the “python window” will drop down all options available under the letter (s) you have typed in
✓While the “dot notation” can be quick for simple one-to-one commands the “import” utility is handy when the process is complex and requires a number of functions

Getting our hands dirty…

- From either approach in Fig. 3, complete typing 'mlclassify (' (or select, by double-clicking, 'MLClassify' from the drop-down list and type "(")
  - Make sure you have something as in Fig. 4a below

Drop-down list of all classifiable raster data sets open in your ArcMap 10. If you cannot see your satellite image load it into your Arcmap and try again



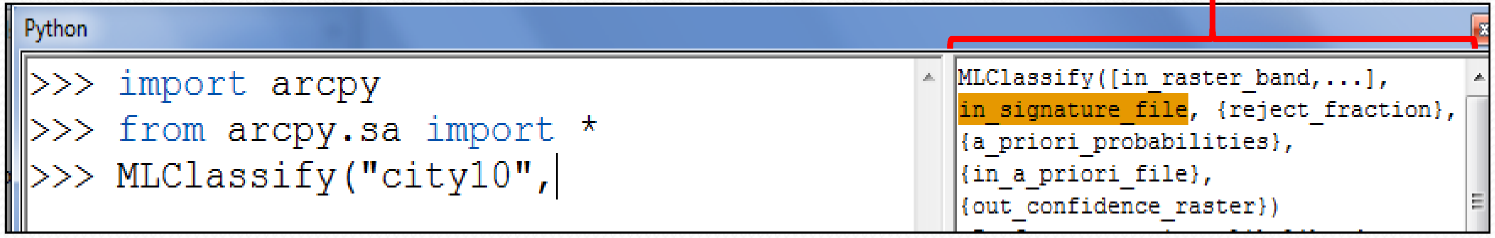
```
>>> import arcpy
>>> from arcpy.sa import *
>>> MLClassify(
```

```
MLClassify([in_raster_band,...],
in_signature_file, {reject_fraction},
{a_priori_probabilities},
{in_a_priori_file},
{out_confidence_raster})
 Performs a maximum likelihood
classification on a set of raster
bands and creates a classified raster
```

Fig. 4a Entering the parameters for the 'MLClassify 'tool

- Select your image from the drop-down list and type comma (,)
  - Make sure you have something as in Fig. 4b below

Documentation section describes the parameters of the function you are using. The highlight in this section indicates the parameter being entered (in Fig 4a we are inputting the image to be classified, in Fig. 4b it's the signature file)



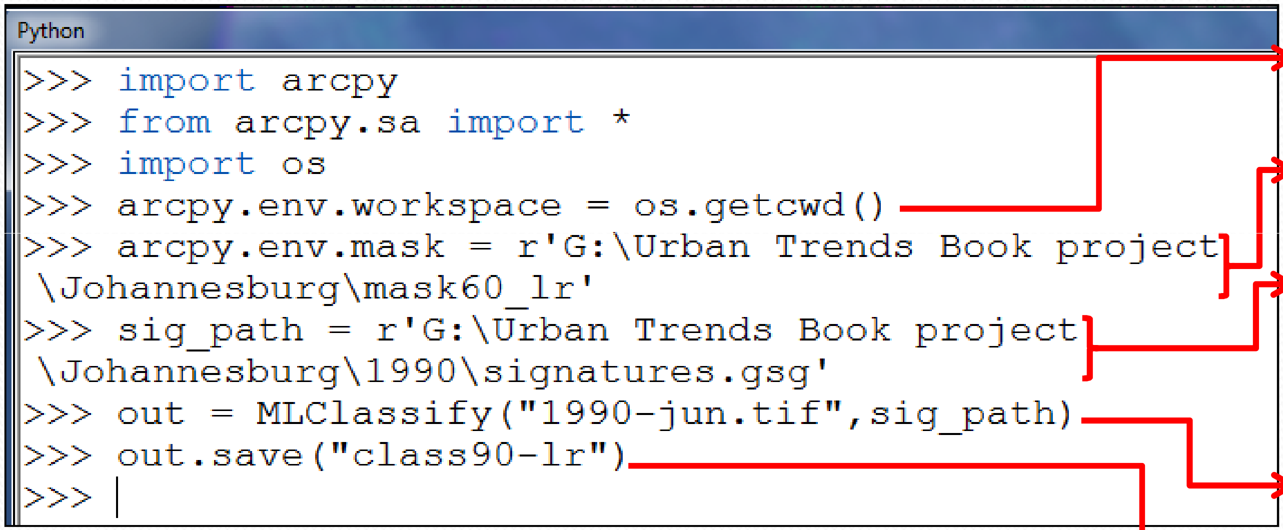
```
>>> import arcpy
>>> from arcpy.sa import *
>>> MLClassify("city10",
```

```
MLClassify([in_raster_band,...],
in_signature_file, {reject_fraction},
{a_priori_probabilities},
{in_a_priori_file},
{out_confidence_raster})
```

Fig. 4b Entering the parameters to run the 'MLClassify 'tool

Getting our hands dirty…

- Now before we actually run the classifier we need to introduce a few housekeeping issues: setting the workspace, mask etc.
- Study the ArcMap 10 documentation for the Maximum Likelihood tool and make sure you understand the parameters
  - Finish entering the parameters as in Fig.5 making relevant changes to the command parameters (eg path to signature file and mask etc)



```
>>> import arcpy
>>> from arcpy.sa import *
>>> import os
>>> arcpy.env.workspace = os.getcwd()
>>> arcpy.env.mask = r'G:\Urban Trends Book project
 \Johannesburg\mask60_lr'
>>> sig_path = r'G:\Urban Trends Book project
 \Johannesburg\1990\signatures.gsg'
>>> out = MLClassify("1990-jun.tif",sig_path)
>>> out.save("class90-lr")
>>> |
```

Sets the workspace to your current working directory

Sets 'mask60_lr' raster file to be the mask

Sets 'signature.gsg' file as a signature variable

'out' is a temporary variable to store the result of classifying the '1990-jun.tif' file using the signature file 'signatures.gsg' that we stored in the 'sig_path' variable

This now saves the classification result permanently in our workspace with a file name 'class90-lr'

Fig. 5a Entering the parameters to run the 'MLClassify 'tool

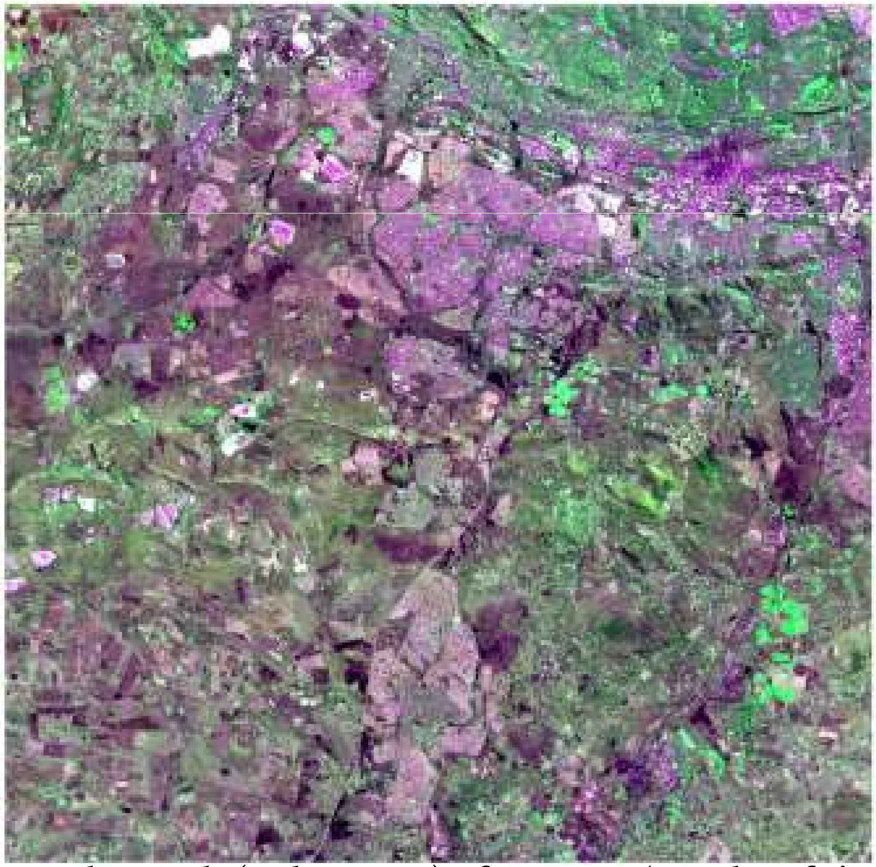
Note:
✓ 'os' was imported to access its tools for us to set the workspace to the current working directory
✓ Use of 'r' before specifying an absolute path to a file
✓ You can copy and paste the file path from windows

Result…

Legend

🟫	Builtup
🟩	Crop Lands
⬜	Other lands

Input satellite image in true colour



MLClassify output

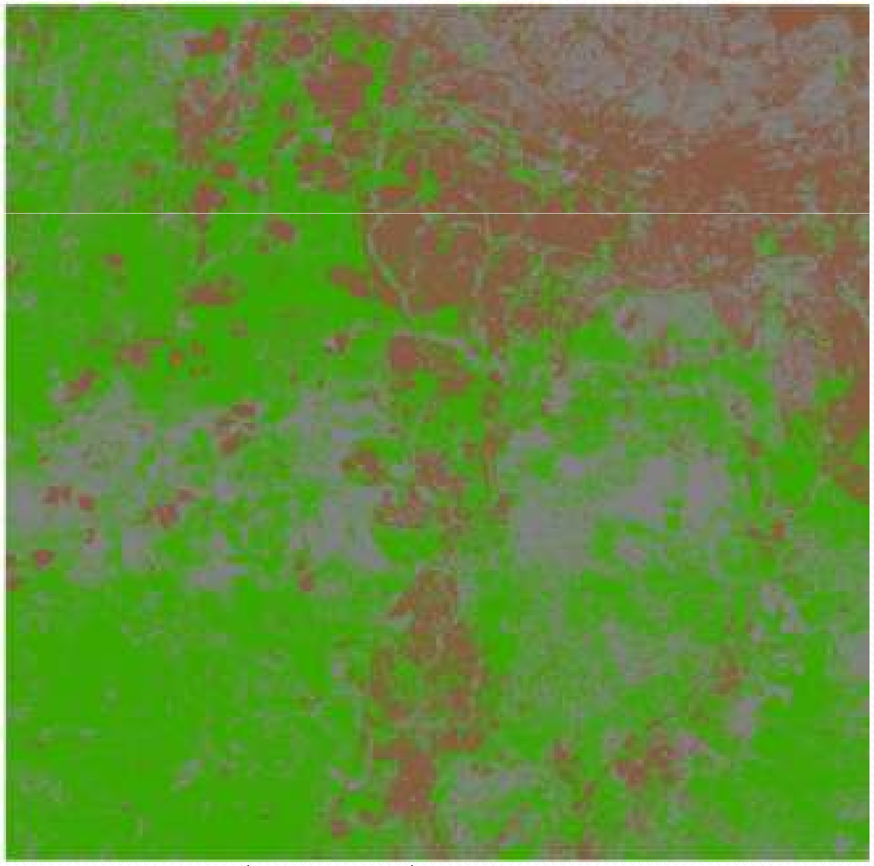


Fig. 5b Result (right image) of running 'MLClassify' tool on the input (left image)

This far….

- We have successfully managed to call and use the maximum likelihood classifier in ArcPy
- We have been introduced to a few housekeeping jobs
- We know where to go to seek python help in ArcGIS
- We know 2 ways of calling an ArcPy command

What next?

- Well, image classification in ArcGIS 10 does not end here
- In fact there are a series of post classification steps to be taken to have a final better classified output
  - These include: filtering, smoothing, and generalization
- Accessing and using these tools is as has been demonstrated with the 'MLClassify' tool
  - If you are up for it take the challenge to do the post classification
- We will be moving on to the essence of this tutorial though: scripting

Creating own scripting tool

- Reminder: the idea is to make repetitive tasks easier
- And our task is to classify 10 different satellite images
  - If you took the challenge in the previous slide you will have realized that it is a hectic task to classify one image let alone to do it repetitively
- Further advantages of a scripting tool
  - Portability (email .tbx and script)
  - Built in dialogs
  - Output to ArcMap data frame
  - Filtering to prevent errors
  - Toolbox, toolbar or context menu

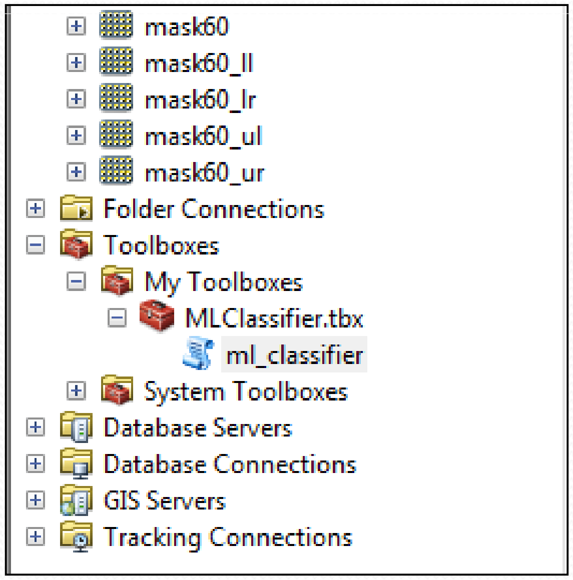
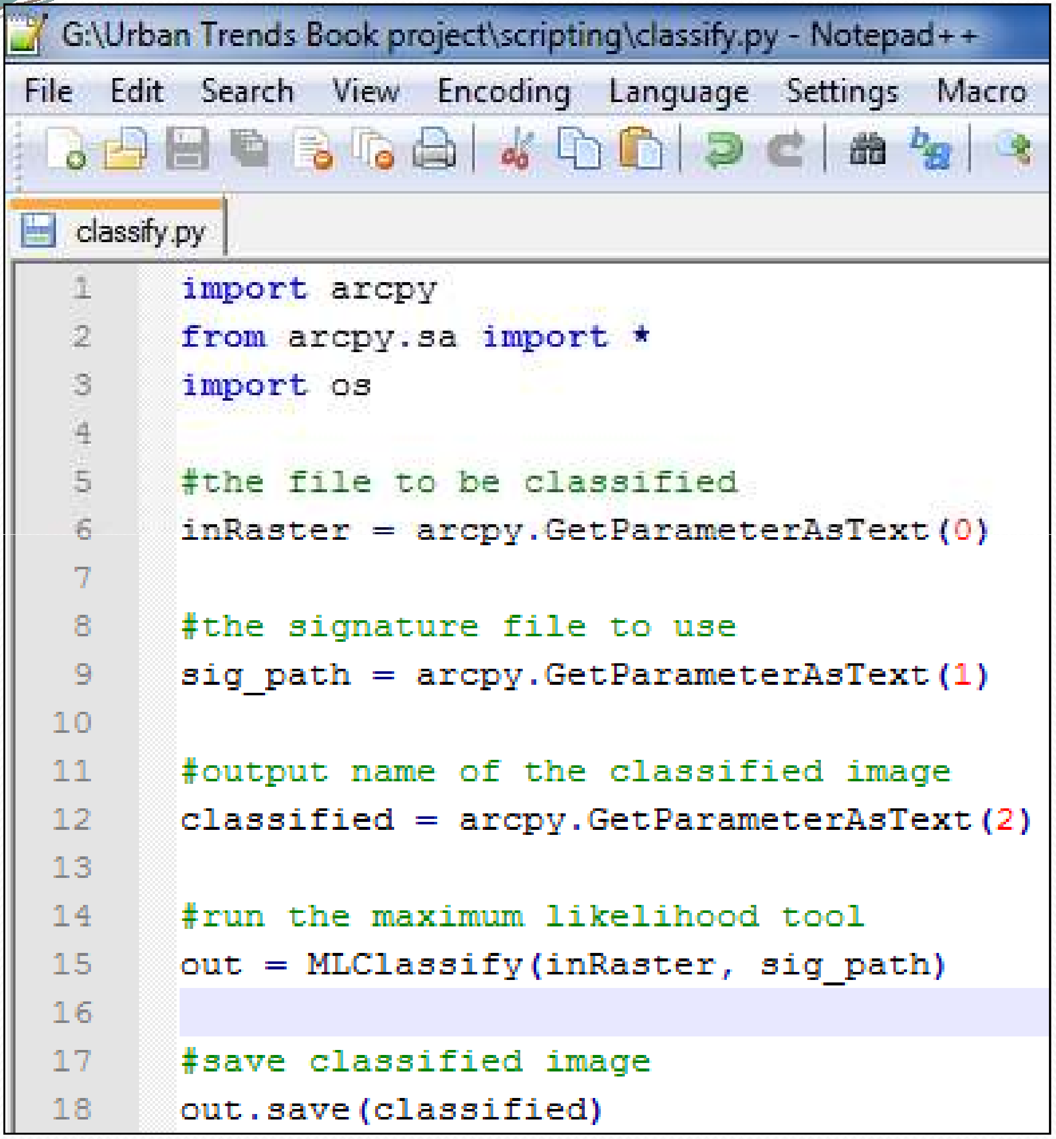


Fig. 6 MLClassifier toolset along sing side Arc system tools

Creating own scripting tool: step 1

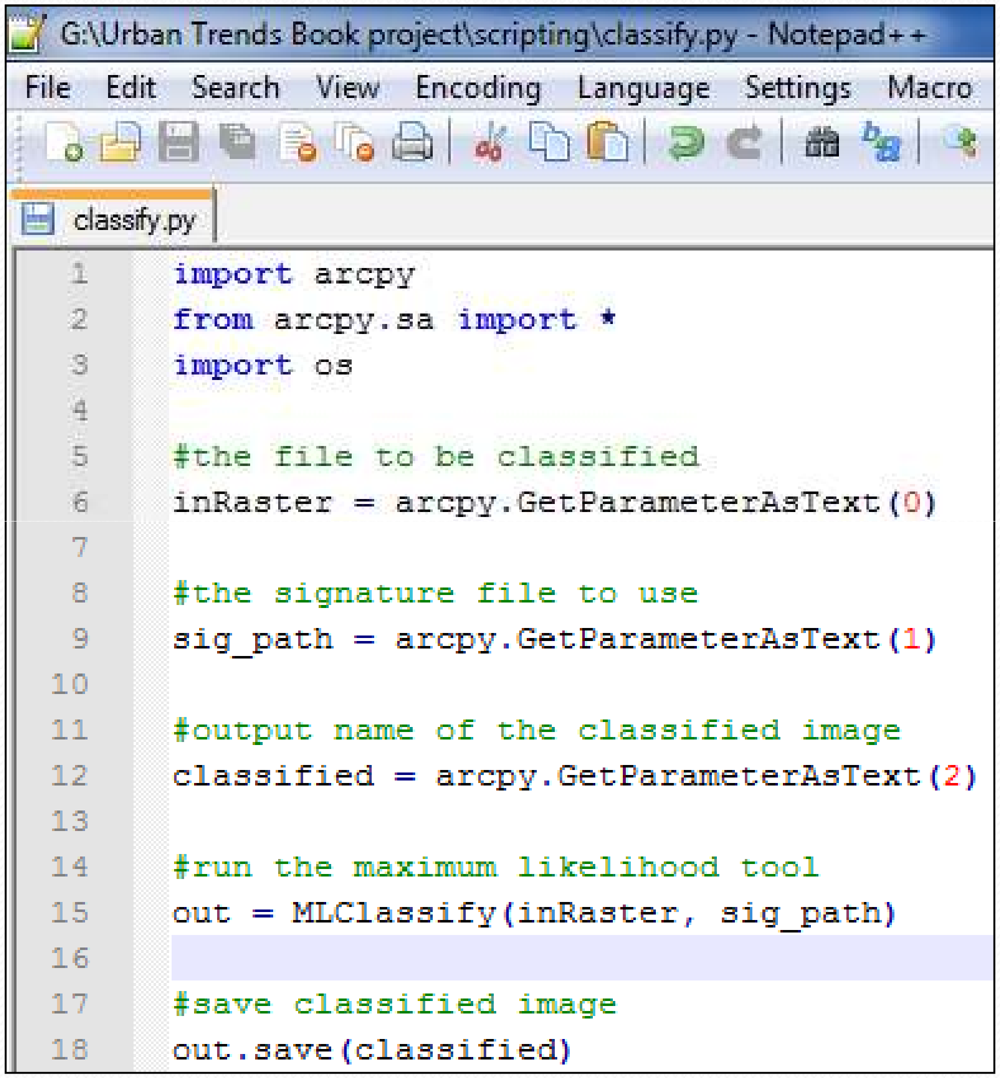

G:\Urban Trends Book project\scripting\classify.py - Notepad++
File Edit Search View Encoding Language Settings Macro
classify.py

```
import arcpy
from arcpy.sa import *
import os

#the file to be classified
inRaster = arcpy.GetParameterAsText(0)

#the signature file to use
sig_path = arcpy.GetParameterAsText(1)

#output name of the classified image
classified = arcpy.GetParameterAsText(2)

#run the maximum likelihood tool
out = MLClassify(inRaster, sig_path)

#save classified image
out.save(classified)
```

Fig. 7 Screenshot of notepad++ with our python script

- Open a text editor of your liking and type in the contents as shown in Fig. 7 (I used notepad)
- Save the file as .py extension (e.g. classify.py)
  - Make sure the file has been saved as, for example, 'classify.py' and **NOT** 'classify.py.txt'

Creating own scripting tool: step 1


G:\Urban Trends Book project\scripting\classify.py - Notepad++
File Edit Search View Encoding Language Settings Macro
classify.py

```
 1  import arcpy
 2  from arcpy.sa import *
 3  import os
 4
 5  #the file to be classified
 6  inRaster = arcpy.GetParameterAsText(0)
 7
 8  #the signature file to use
 9  sig_path = arcpy.GetParameterAsText(1)
10
11  #output name of the classified image
12  classified = arcpy.GetParameterAsText(2)
13
14  #run the maximum likelihood tool
15  out = MLClassify(inRaster, sig_path)
16
17  #save classified image
18  out.save(classified)
```

Line 3, the line 'import os' is not necessary
The workspace and mask commands in Fig. 5 have been dropped for reasons we will explain later

Line 9 replaces ***r' G:\Urban Trends Book project\Johannesburg\1990\signatures.gsg'*** in Fig. 5

Line 12 and 18 combines to replace ***out.save("class90_lr")*** in Fig. 5

Line 15 replaces ***out = MLClassify(1990-jun.tif, sig_path)*** in Fig. 5

Creating own scripting tool: step 2

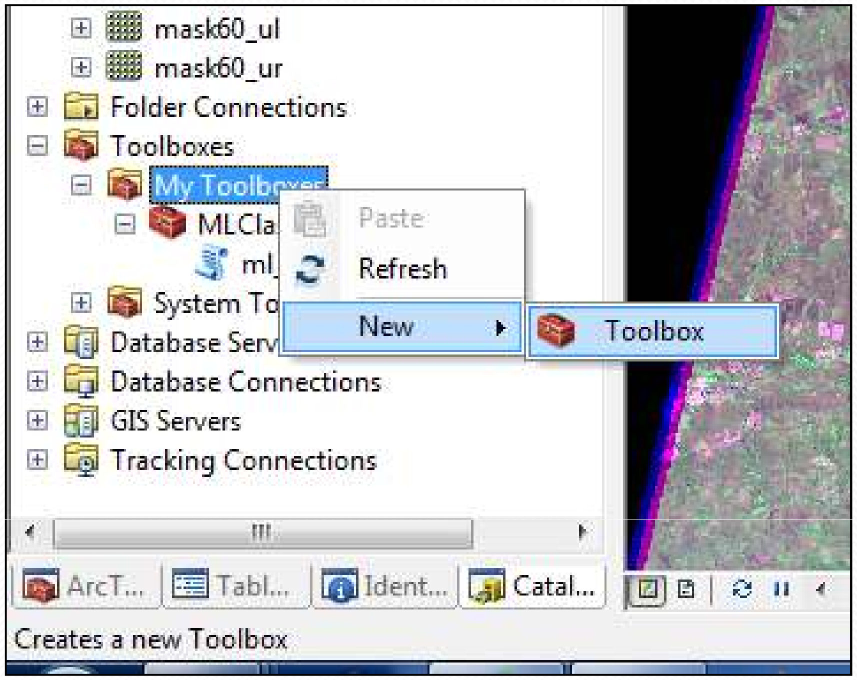


Fig. 8 Screenshot of Toolboxes in Arc Catalog

- In ArcMap navigate to the Toolboxes in Catalog
- Right click **My Toolboxes** and navigate to and click **New>Toolbox** (Fig. 8)
- Give the toolbox an appropriate name
- Right click your new toolbox and navigate to and click **Add>Script** (Fig. 9)

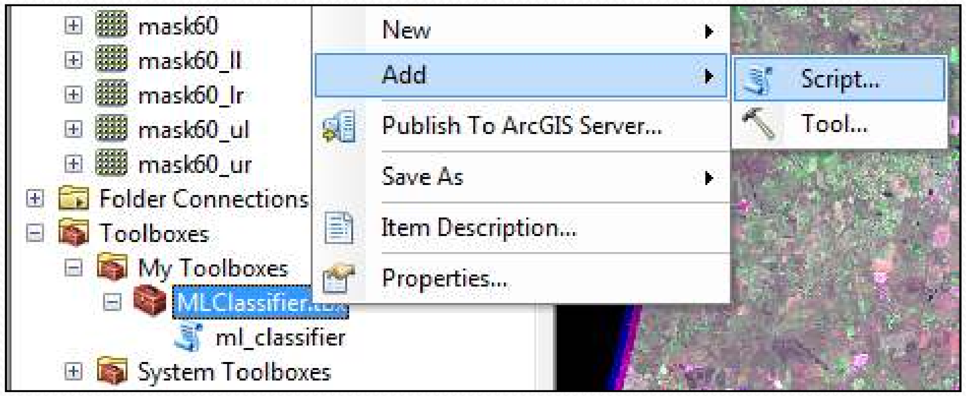
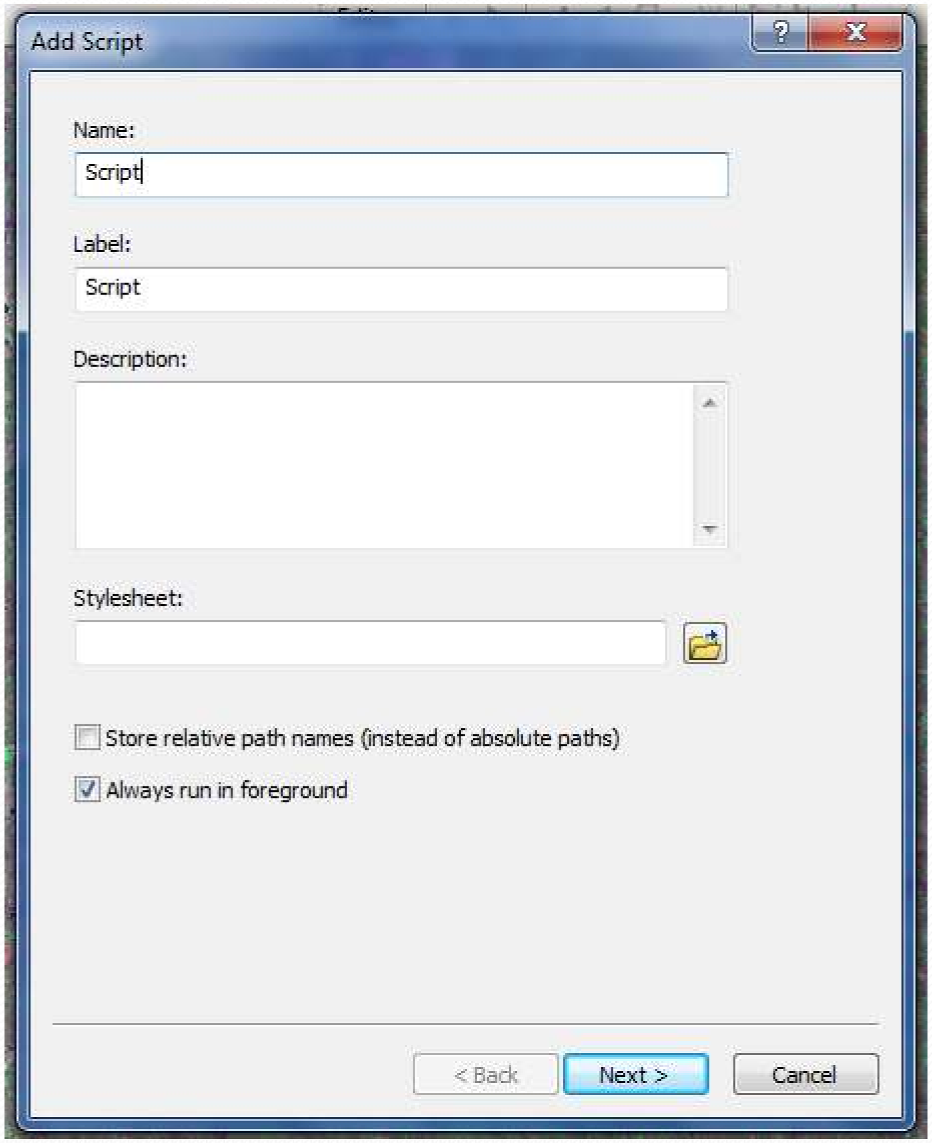


Fig. 9 Screenshot of Toolboxes in Arc Catalog

Creating own scripting tool: step 2


Add Script

Name: Script

Label: Script

Description:

Stylesheet:

☐ Store relative path names (instead of absolute paths)

☑ Always run in foreground

Fig. 10 Screenshot of Toolboxes in Arc Catalog

- You should have a screen similar to Fig. 10
- You can leave the options as are, though I changed the name to ‘ml_classifier’
- Click **Next>**

Creating own scripting tool: step 2

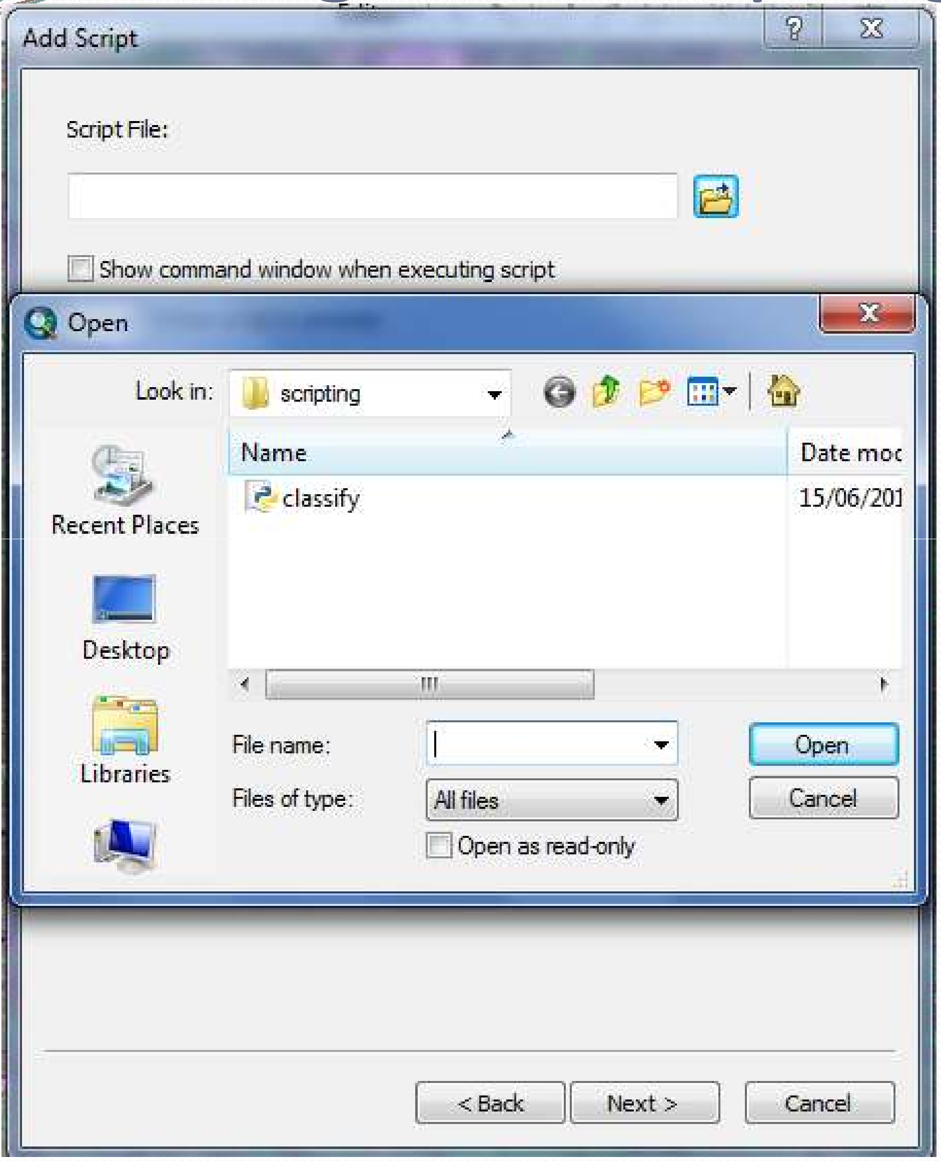
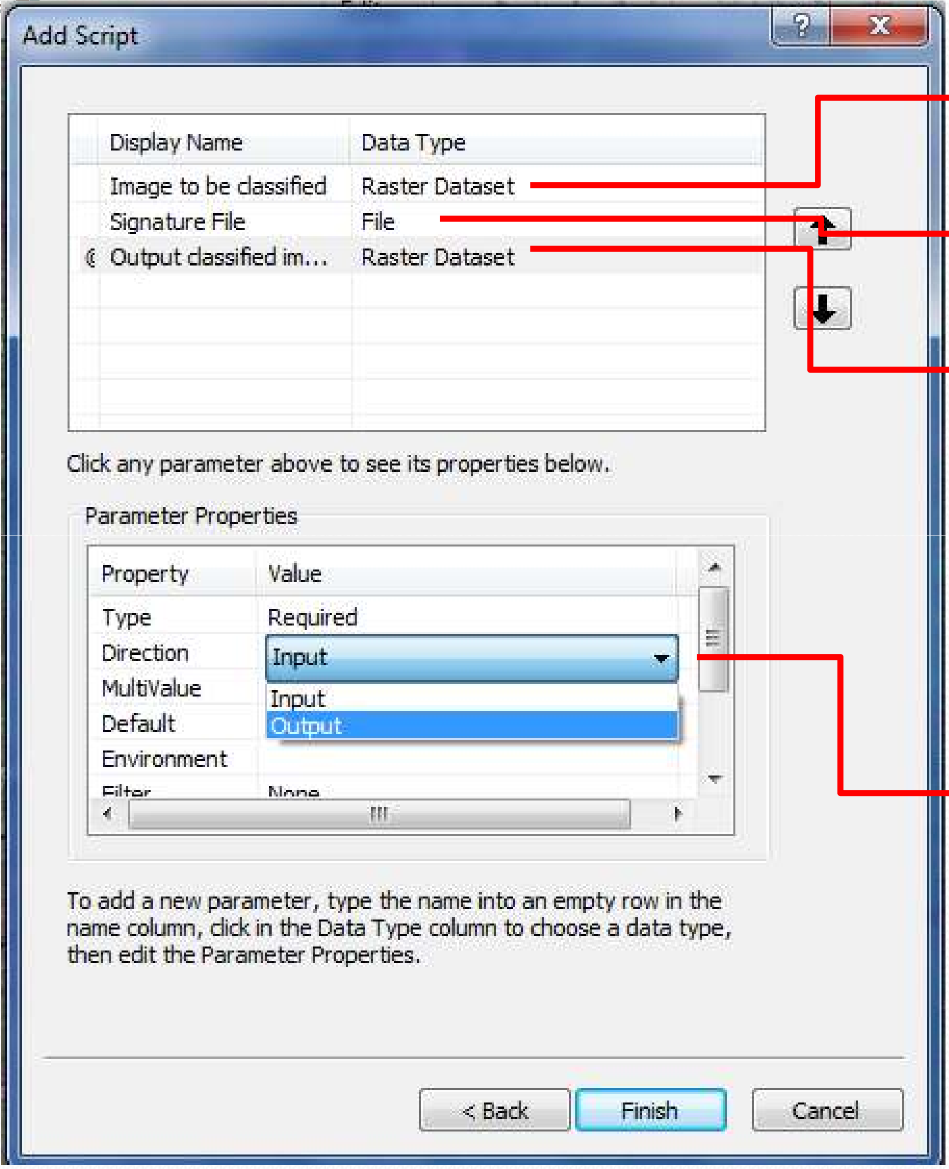


Fig. 11 Adding a script to toolbox

- On 'Script File:' click the folder option ( )
- You should have a screen similar to Fig. 11
- Navigate to where you saved your .py file
- Leave the rest of the options as are
- Click **Next>**

Creating own scripting tool: step 2



Add Script

Display Name	Data Type
Image to be classified	Raster Dataset
Signature File	File
Output classified im...	Raster Dataset

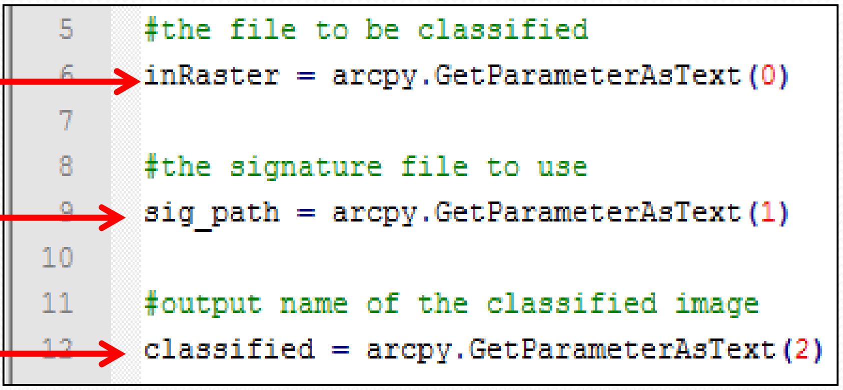
Click any parameter above to see its properties below.

Parameter Properties

Property	Value
Type	Required
Direction	Input
MultiValue	Input / Output
Default	
Environment	
Filter	None

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.

Fig. 12 Adding a script to toolbox



```
#the file to be classified
inRaster = arcpy.GetParameterAsText(0)

#the signature file to use
sig_path = arcpy.GetParameterAsText(1)

#output name of the classified image
classified = arcpy.GetParameterAsText(2)
```

- Enter the details as shown in Fig. 12 maintaining the order as they appear in the .py file
- ArcGIS parameter labeling starts from 0 in the order they are entered and labeled in Fig. 12
- Under ‘Parameter Properties’ choose ‘input’ for the ‘Direction’ property for all parameters except for the output classified file which must be ‘output’
- Click **Finish>**
- Double click your script

Creating own scripting tool: step 2

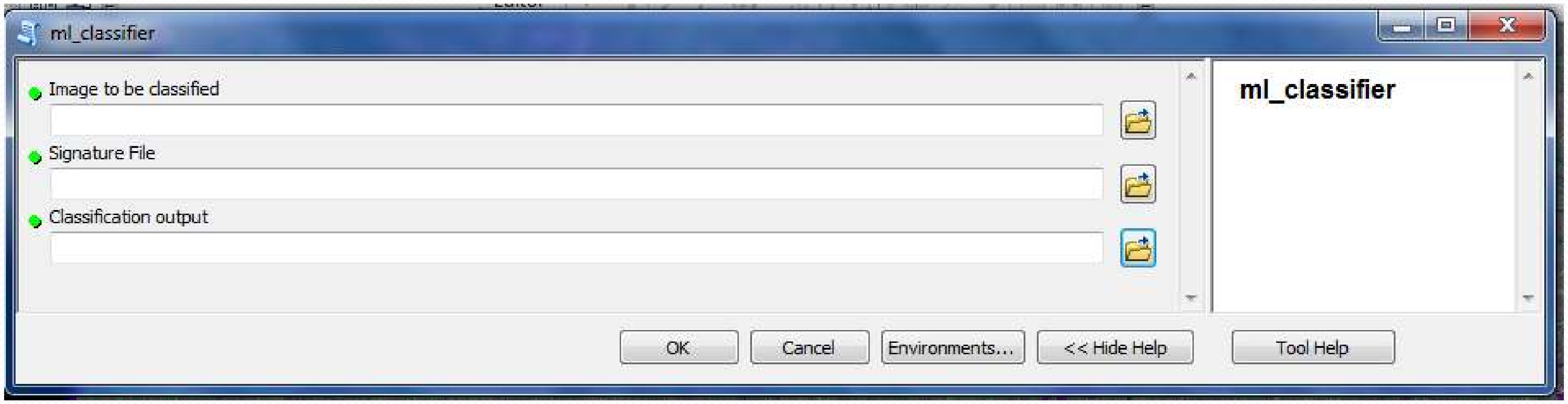
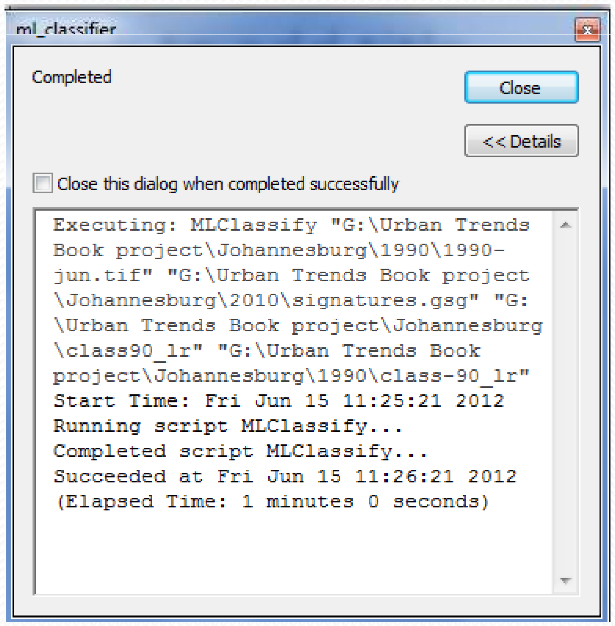


Fig. 13 An ml_classifier tool

- If you have something like Fig. 13 above, your scripting tool is ready for use
- Click on 'Environments…'
  - We did not bother specifying the mask in our script like we did with the command line because that option is eventually provided under the 'Environments…' options
- Run your tool by entering the relevant input in the toolset.
- If you get Fig. 14 after running your tool, you have just successfully built your script. CONGRATULATIONS!!!
- You should equally get a result similar to Fig. 5b



```
Executing: MLClassify "G:\Urban Trends
Book project\Johannesburg\1990\1990-
jun.tif" "G:\Urban Trends Book project
\Johannesburg\2010\signatures.gsg" "G:
\Urban Trends Book project\Johannesburg
\class90_lr" "G:\Urban Trends Book
project\Johannesburg\1990\class-90_lr"
Start Time: Fri Jun 15 11:25:21 2012
Running script MLClassify...
Completed script MLClassify...
Succeeded at Fri Jun 15 11:26:21 2012
(Elapsed Time: 1 minutes 0 seconds)
```

Fig. 14

Recap…

- We have been able to run ArcPy as a Script tool in ArcMap
  - Accessed and run just like the system tools that are installed with ArcGIS 10.
- We have been able to run ArcPy in the Python Window in ArcMap (Fig. 5)
  - Managing to access geo-processing environment settings.

How else can python be run?

- Stand alone python – from an IDE, from the command line, or as a scheduled task
- As geo-processing service in ArcGIS server

What other modules are available in ArcPy?

- Apart from the spatial analyst module (arcpy.sa) which we have just introduced ArcPy also provide
  - Mapping module (arcpy.mapping)
  - Geostatistical analyst module (arcpy.ga)

References

- http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/What_is_ArcPy/000v000000v7000000/
- http://www.esri.com/library/fliers/pdfs/python-in-arcgis10.pdf
- http://technet.microsoft.com/en-us/scriptcenter/dd940112.aspx
- http://searchwindevelopment.techtarget.com/definition/scripting-language