# python ArcGIS

## Geoprocessing

# Access to the tools of ATB

# We can use arcpy to run every ArcToolBox tool.

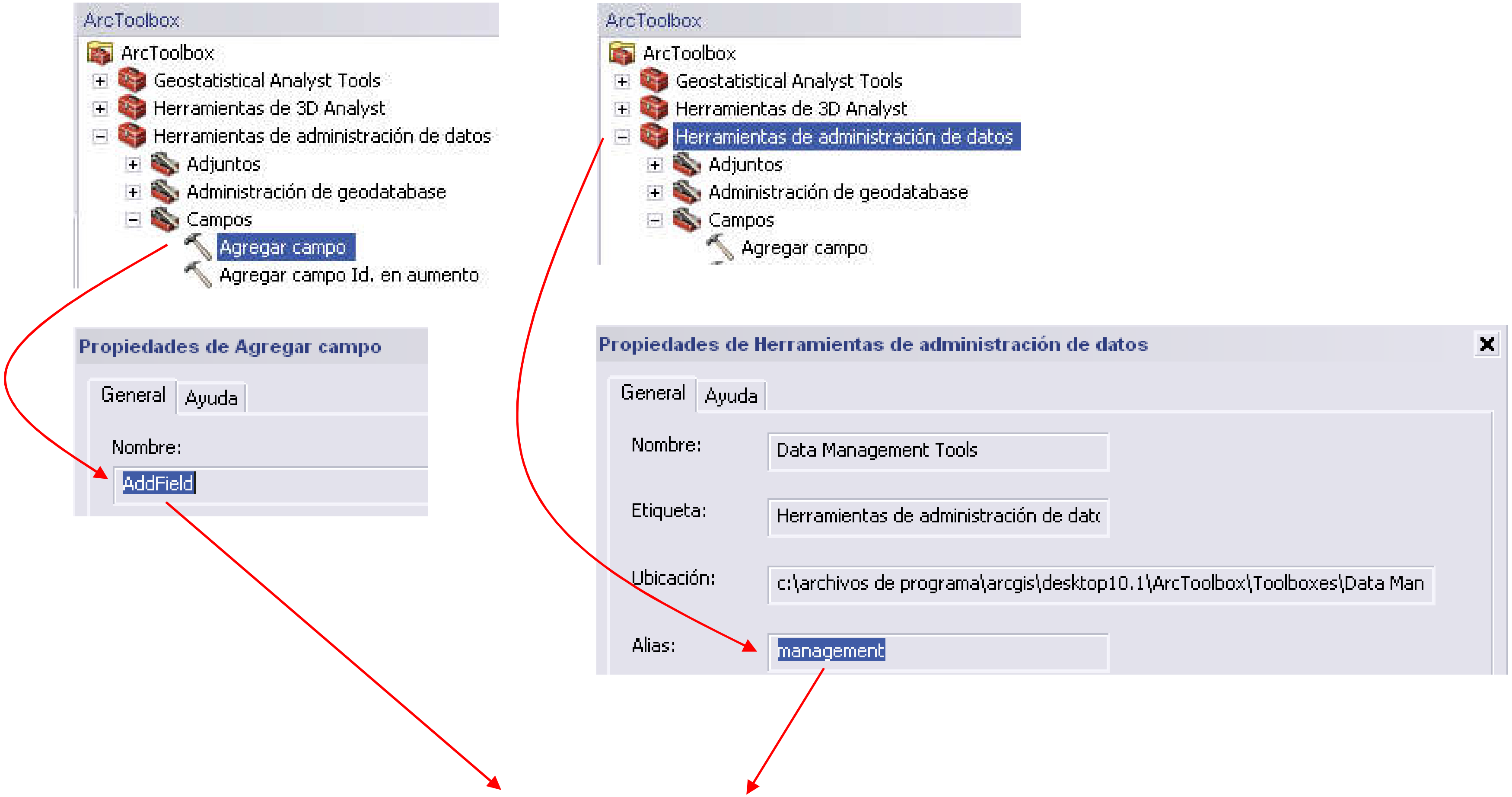
# The key is to write the the tool name in a suitable way. To build the name of a specific tool you must write the next parts:

- arcpy (module)
- tool name (field “Name” in tool properties).
- underscore character “_”
- name of the highest level tool box that contains the tool (field "Alias" int tool box properties).

```
arcpy.AddField_management (table, field name, field type)

arcpy.Buffer_analysis (input layer, output layer, distance)
```

# Parameters can be strings. You must be careful with its order and its number (have a look at help documentation).

# Access to the tools of ATB



ArcToolbox

- ArcToolbox
  - Geostatistical Analyst Tools
  - Herramientas de 3D Analyst
  - Herramientas de administración de datos
    - Adjuntos
    - Administración de geodatabase
    - Campos
      - Agregar campo
      - Agregar campo Id. en aumento

Propiedades de Agregar campo

Nombre: AddField

ArcToolbox

- ArcToolbox
  - Geostatistical Analyst Tools
  - Herramientas de 3D Analyst
  - Herramientas de administración de datos
    - Adjuntos
    - Administración de geodatabase
    - Campos
      - Agregar campo

Propiedades de Herramientas de administración de datos

Nombre: Data Management Tools

Etiqueta: Herramientas de administración de dat(

Ubicación: c:\archivos de programa\arcgis\desktop10.1\ArcToolbox\Toolboxes\Data Man

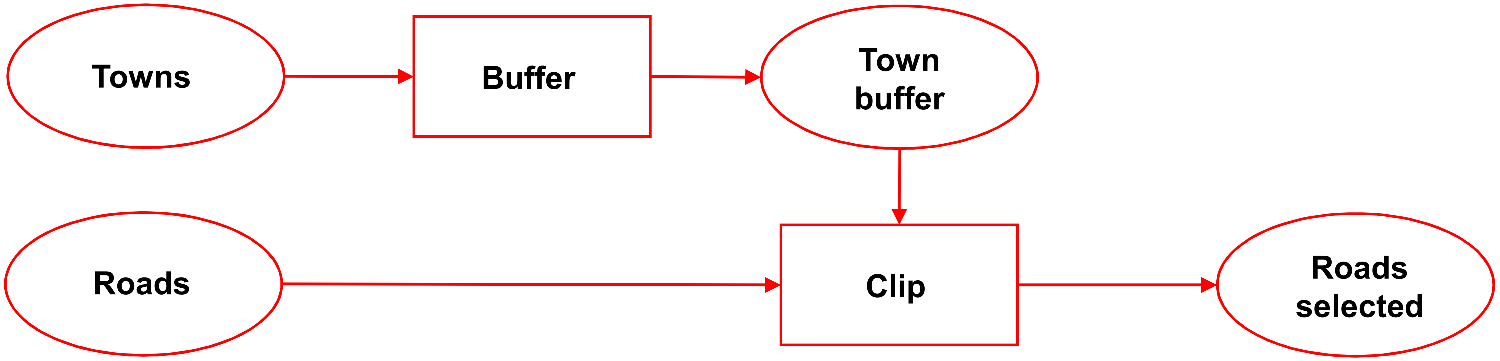
Alias: management

AddField_management

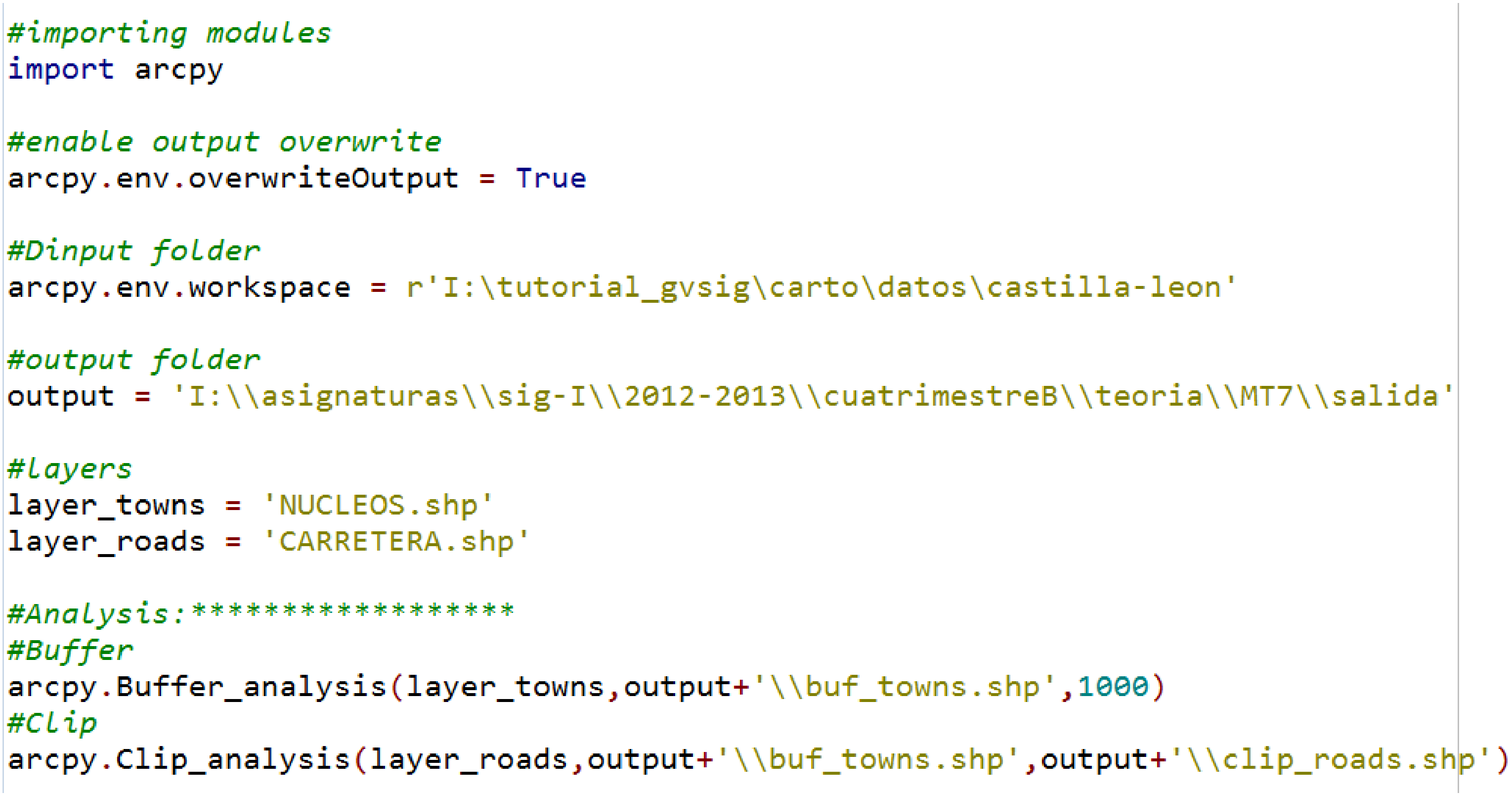
# Creating geoprocessing workflows

# As we do in Model builder, we can create tool workflows using python and arcpy. That way, the output from a tool is the input for the next tool.

# Example: get all roads up to 1 kilometer from every town.



# Creating geoprocessing workflows

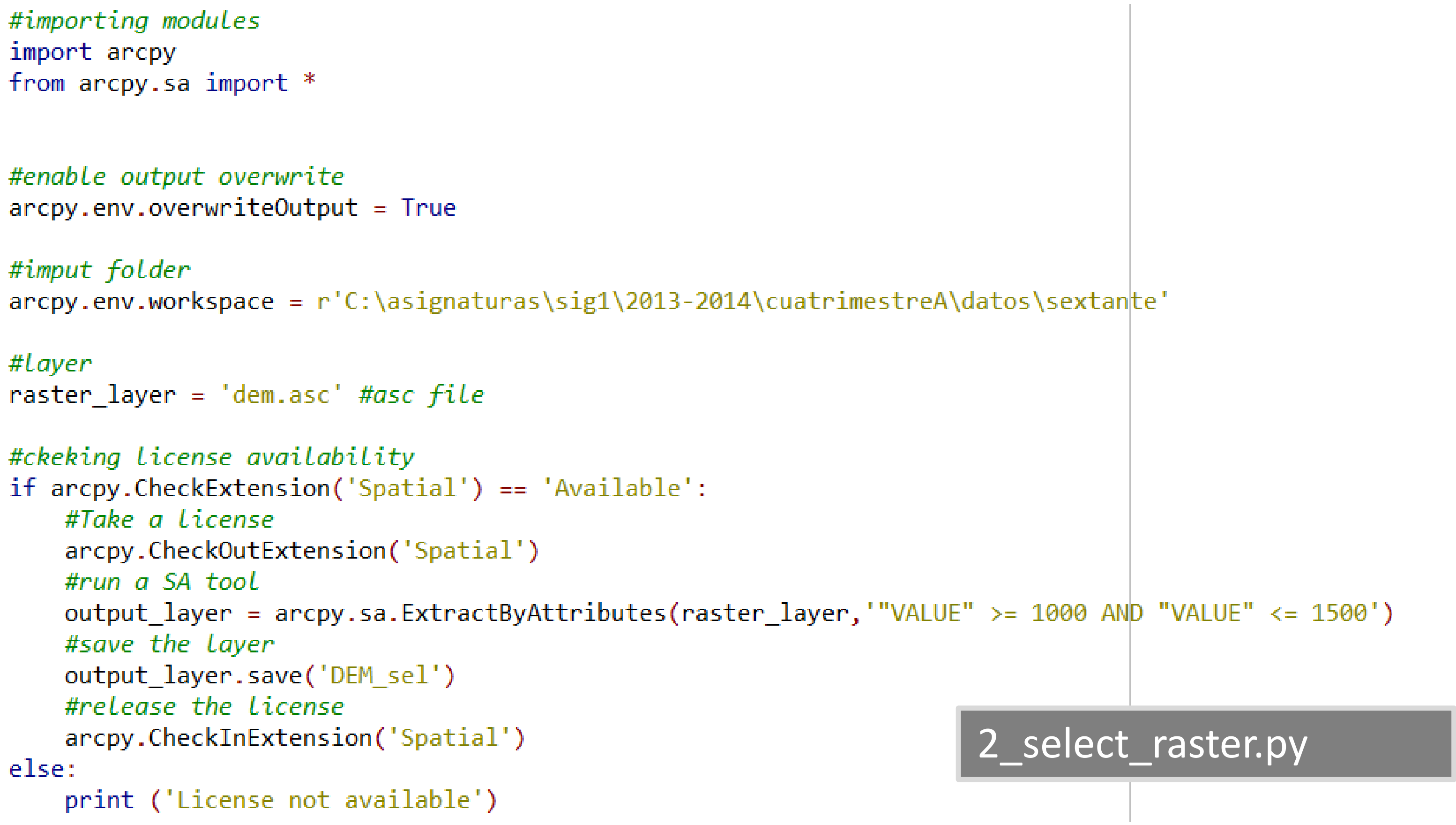


```
#importing modules
import arcpy

#enable output overwrite
arcpy.env.overwriteOutput = True

#Dinput folder
arcpy.env.workspace = r'I:\tutorial_gvsig\carto\datos\castilla-leon'

#output folder
output = 'I:\\asignaturas\\sig-I\\2012-2013\\cuatrimestreB\\teoria\\MT7\\salida'

#layers
layer_towns = 'NUCLEOS.shp'
layer_roads = 'CARRETERA.shp'

#Analysis:*****************
#Buffer
arcpy.Buffer_analysis(layer_towns,output+'\\buf_towns.shp',1000)
#Clip
arcpy.Clip_analysis(layer_roads,output+'\\buf_towns.shp',output+'\\clip_roads.shp')
```

1_extract_roads.py

# Spatial Analyst tools

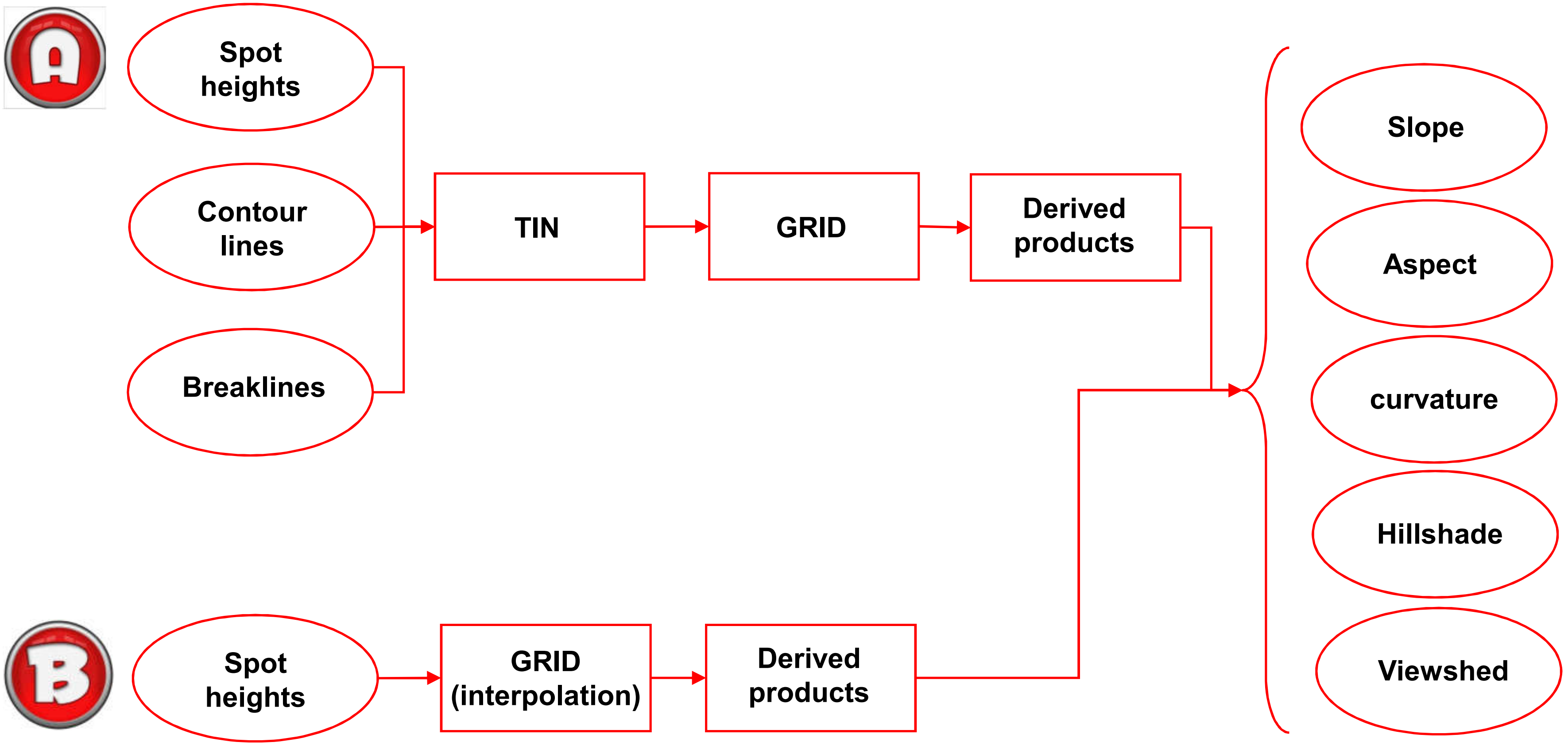
# We have to remember that if we want to use SA tools, we must follow the next rules about the license: checking the license availability, take a license, use the tool and finally release the license.



```
#importing modules
import arcpy
from arcpy.sa import *


#enable output overwrite
arcpy.env.overwriteOutput = True

#imput folder
arcpy.env.workspace = r'C:\asignaturas\sig1\2013-2014\cuatrimestreA\datos\sextante'

#layer
raster_layer = 'dem.asc' #asc file

#ckeking license availability
if arcpy.CheckExtension('Spatial') == 'Available':
    #Take a license
    arcpy.CheckOutExtension('Spatial')
    #run a SA tool
    output_layer = arcpy.sa.ExtractByAttributes(raster_layer,'"VALUE" >= 1000 AND "VALUE" <= 1500')
    #save the layer
    output_layer.save('DEM_sel')
    #release the license
    arcpy.CheckInExtension('Spatial')
else:
    print ('License not available')
```

2_select_raster.py

# DEM management. Workflow



# DEM management. Workflow

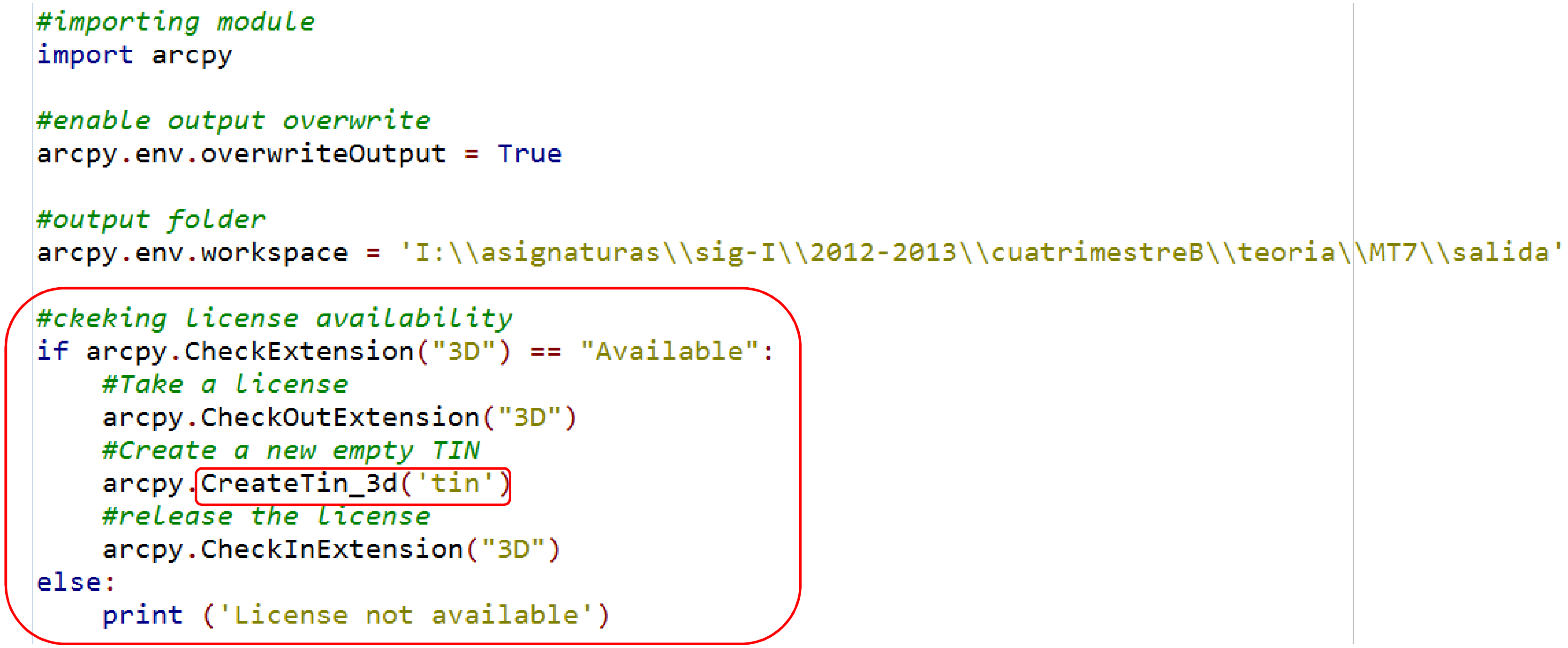
# First step: create a TIN (triangulated irregular network). The tool "CreateTin_3d" allows to add source layers. If not, we will obtain an empty TIN. If we want to add some source layers after, we will have to use the tool "EditTin_3d".

```
arcpy.CreateTin_3d(path)
```

#In this case, as these tools belong to the "3D analyst" extension, we also have to observe the known rules: checking the license availability, take a license, use the tool and finally release the license. The keyword for this extension is "3D".

**http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//002z0000000z000000**

# DEM management. Create a TIN



```python
#importing module
import arcpy

#enable output overwrite
arcpy.env.overwriteOutput = True

#output folder
arcpy.env.workspace = 'I:\\asignaturas\\sig-I\\2012-2013\\cuatrimestreB\\teoria\\MT7\\salida'

#ckeking license availability
if arcpy.CheckExtension("3D") == "Available":
    #Take a license
    arcpy.CheckOutExtension("3D")
    #Create a new empty TIN
    arcpy.CreateTin_3d('tin')
    #release the license
    arcpy.CheckInExtension("3D")
else:
    print ('License not available')
```

3_create_tin.py

# DEM management. Edit a TIN

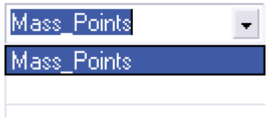
# Once the TIN have been built, we can edit it to add new feature classes (source layers)..

arcpy.EditTin_3d (in_tin, in_features, {constrained_delaunay}).

1 - Input TIN

2
- Height field (if there isn't, use "<none>")
- tag value fro triangles (if there isn't, use "<none>")
- Surface feature type:

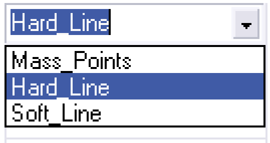
**Points**



Mass_Points

Mass_Points

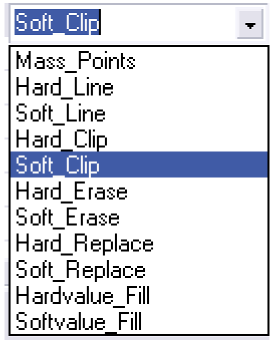
**Lines**



Hard_Line

Mass_Points
Hard_Line
Soft_Line

**Polygons**



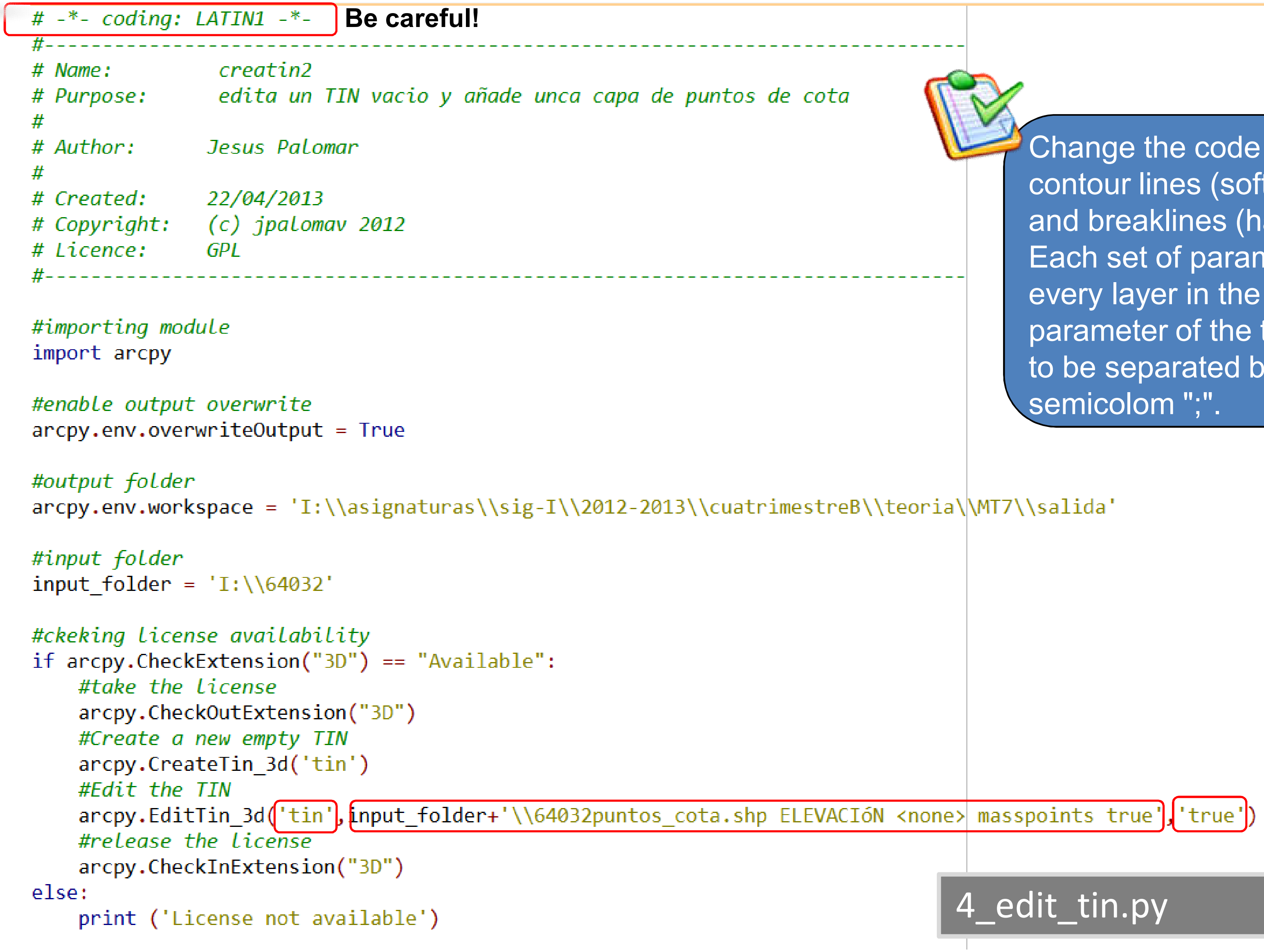
Soft_Clip

Mass_Points
Hard_Line
Soft_Line
Hard_Clip
Soft_Clip
Hard_Erase
Soft_Erase
Hard_Replace
Soft_Replace
Hardvalue_Fill
Softvalue_Fill

- Use geometry as Z value (boolean). In the event we have a height field, the value will be “False” .

3 - Constrained Delaunay (boolean): delaunay or Constrained Delaunay.

# DEM management. Edit a TIN



```
# -*- coding: LATIN1 -*-
#-------------------------------------------------------------------------------
# Name:        creatin2
# Purpose:     edita un TIN vacio y añade unca capa de puntos de cota
#
# Author:      Jesus Palomar
#
# Created:     22/04/2013
# Copyright:   (c) jpalomav 2012
# Licence:     GPL
#-------------------------------------------------------------------------------

#importing module
import arcpy

#enable output overwrite
arcpy.env.overwriteOutput = True

#output folder
arcpy.env.workspace = 'I:\\asignaturas\\sig-I\\2012-2013\\cuatrimestreB\\teoria\\MT7\\salida'

#input folder
input_folder = 'I:\\64032'

#ckeking license availability
if arcpy.CheckExtension("3D") == "Available":
    #take the license
    arcpy.CheckOutExtension("3D")
    #Create a new empty TIN
    arcpy.CreateTin_3d('tin')
    #Edit the TIN
    arcpy.EditTin_3d('tin',input_folder+'\\64032puntos_cota.shp ELEVACIóN <none> masspoints true','true')
    #release the license
    arcpy.CheckInExtension("3D")
else:
    print ('License not available')
```

**Be careful!**

Change the code to add contour lines (soft lines) and breaklines (hard lines). Each set of parameters for every layer in the second parameter of the tool have to be separated by a semicolom ";".

4_edit_tin.py

# DEM management. TIN to GRID

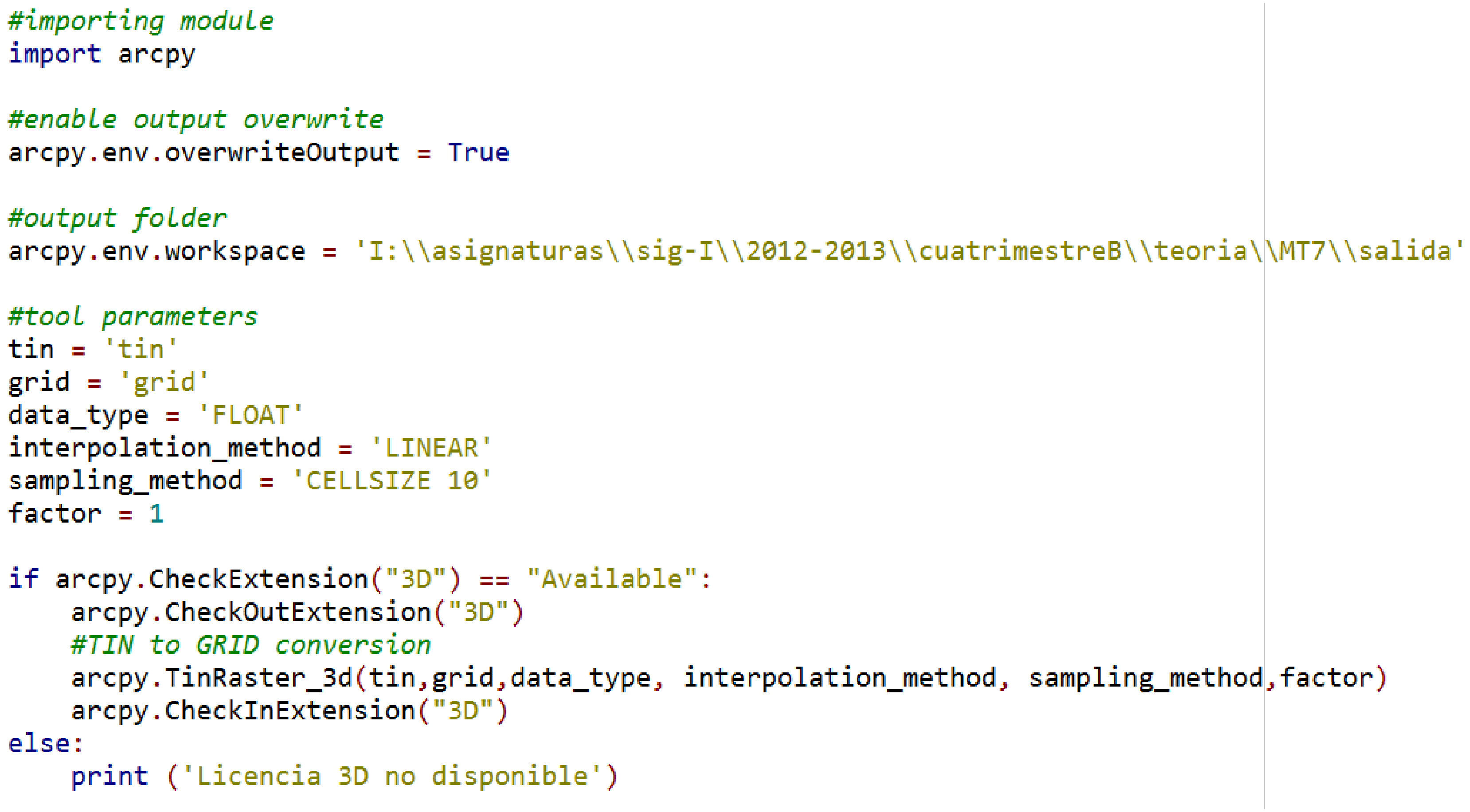
# Once we have made the edition, we are going to convert the TIN (vector) to a GRID (raster). The tool "TinRaster_3D" has the next parameters:

```
arcpy. TinRaster_3d (in_tin, out_raster, {data_type}, {method}, {sample_distance}, {z_factor})
```

- input TIN
- output raster (without extension, the output layer will be a GRID)
- data type (float or integer)
- interpolation method (linear or natural neigbors)
- sampling method (observations or cell size)
- z factor (terrain exaggeration)

**http://resources.arcgis.com/en/help/main/10.1/index.html#//005v00000027000000**

# DEM management. TIN to GRID



```
#importing module
import arcpy

#enable output overwrite
arcpy.env.overwriteOutput = True

#output folder
arcpy.env.workspace = 'I:\\asignaturas\\sig-I\\2012-2013\\cuatrimestreB\\teoria\\MT7\\salida'

#tool parameters
tin = 'tin'
grid = 'grid'
data_type = 'FLOAT'
interpolation_method = 'LINEAR'
sampling_method = 'CELLSIZE 10'
factor = 1

if arcpy.CheckExtension("3D") == "Available":
    arcpy.CheckOutExtension("3D")
    #TIN to GRID conversion
    arcpy.TinRaster_3d(tin,grid,data_type, interpolation_method, sampling_method,factor)
    arcpy.CheckInExtension("3D")
else:
    print ('Licencia 3D no disponible')
```

5_tin_to_grid.py

# DEM management. Slope

# To end the analysis we will perform an slope map. The tool will be Slope_3d.

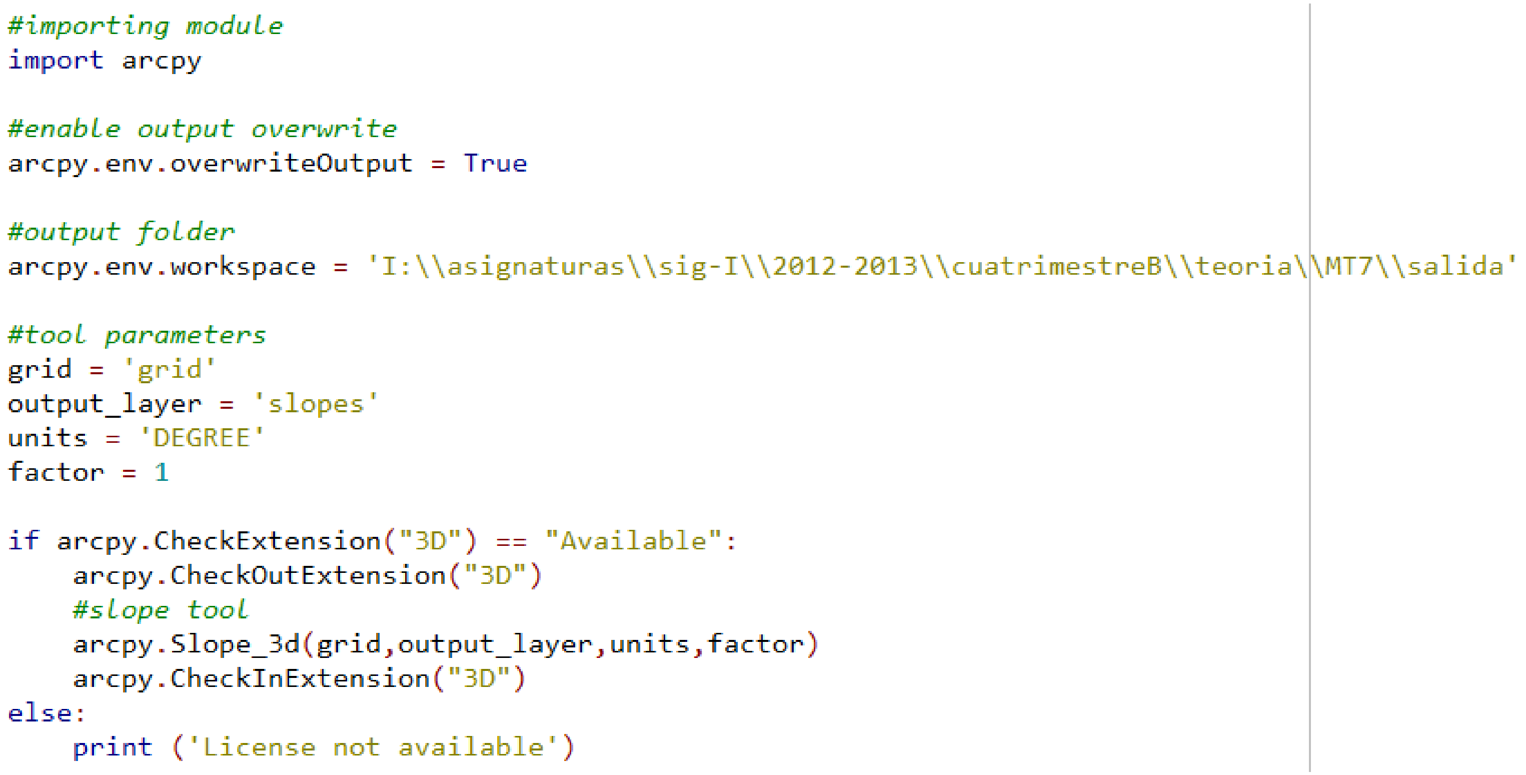
```
Arcpy.Slope_3d (in_raster, out_raster, {output_measurement}, {z_factor})
```

.

# Parameters:

- input raster
- output raster
- slope units (DEGREE or PERCENT_RISE)
- z factor (terrain exaggeration)

# DEM management. Slope



```
#importing module
import arcpy

#enable output overwrite
arcpy.env.overwriteOutput = True

#output folder
arcpy.env.workspace = 'I:\\asignaturas\\sig-I\\2012-2013\\cuatrimestreB\\teoria\\MT7\\salida'

#tool parameters
grid = 'grid'
output_layer = 'slopes'
units = 'DEGREE'
factor = 1

if arcpy.CheckExtension("3D") == "Available":
    arcpy.CheckOutExtension("3D")
    #slope tool
    arcpy.Slope_3d(grid,output_layer,units,factor)
    arcpy.CheckInExtension("3D")
else:
    print ('License not available')
```

6_slope_map.py

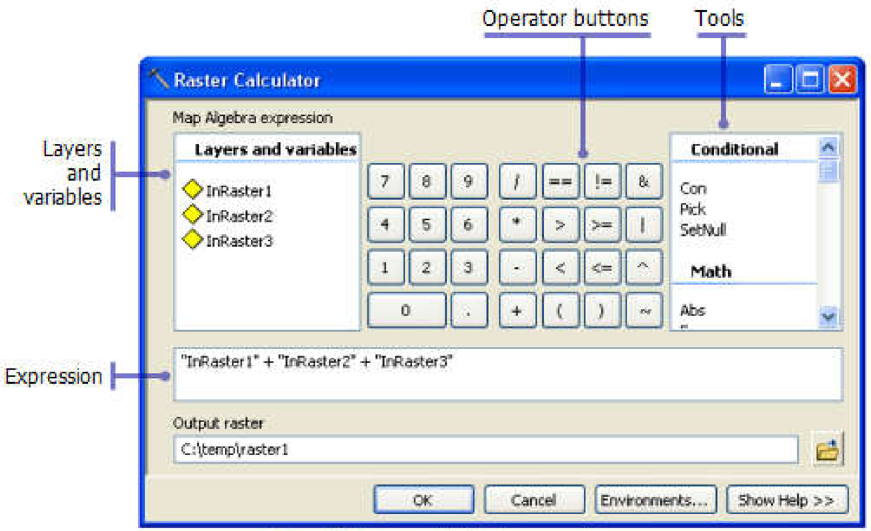
# DEM management. Slope

Write a new script to do all processes together: TIN creation, TIN edition, TIN to GRID conversion and 4 derived products: slope, aspect, curvature and hillshade.

# Raster processing. Map algebra

# Sometimes it is necessary to work with  mathematical expressions where raster layers are operands.

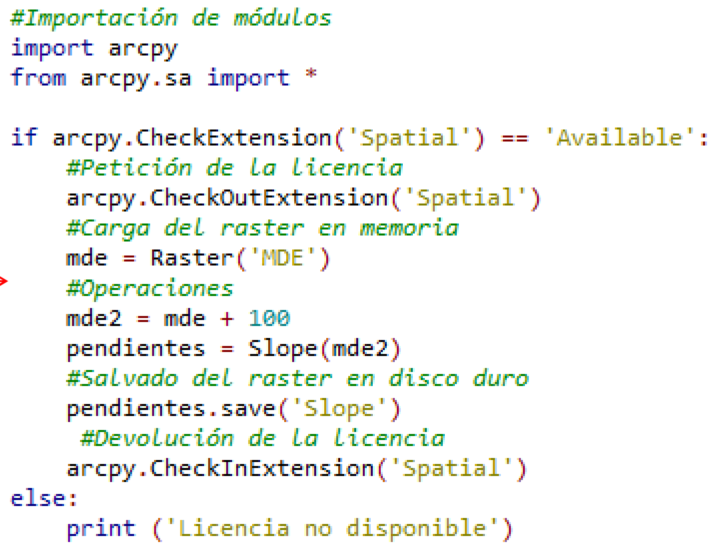
#ArcToolBox provides a tool called "Map algebra“ (or raster calculator). In python is not possible to use this tool directly, so we need to work with the SA module in order to do similar things (arcpy.sa)



"InRaster1" + "InRaster2" + "InRaster3"

Output raster: C:\temp\raster1

*Raster Calculator tool dialog box example*



```
#Importación de módulos
import arcpy
from arcpy.sa import *

if arcpy.CheckExtension('Spatial') == 'Available':
    #Petición de la licencia
    arcpy.CheckOutExtension('Spatial')
    #Carga del raster en memoria
    mde = Raster('MDE')
    #Operaciones
    mde2 = mde + 100
    pendientes = Slope(mde2)
    #Salvado del raster en disco duro
    pendientes.save('Slope')
     #Devolución de la licencia
    arcpy.CheckInExtension('Spatial')
else:
    print ('Licencia no disponible')
```

# Raster processing. Map algebra

# There are two ways to work from python:

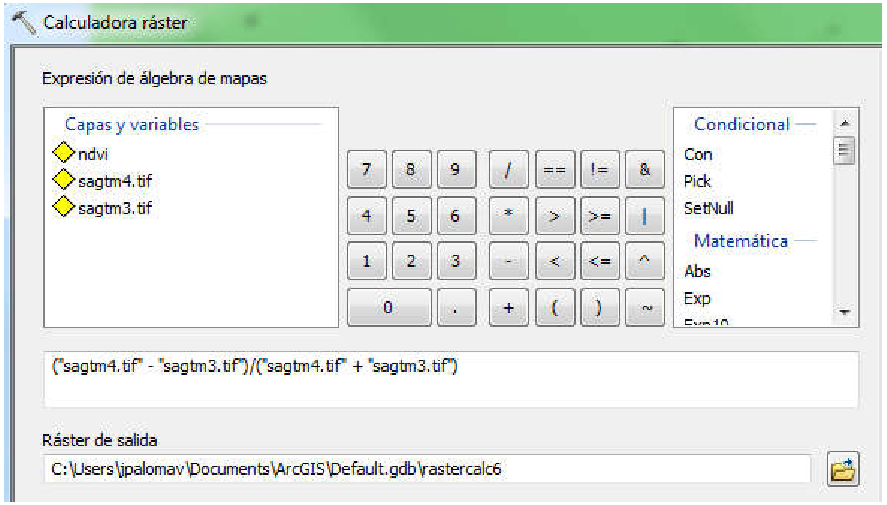
- use SA module functions (Spatial analyst).
- work pixel by pixel using arcpy (Raster class) and numpy..

#Using SA module.

#Example 1: calculate NDVI (Normalized Difference Vegetation Index)

#https://en.wikipedia.org/wiki/Normalized_Difference_Vegetation_Index

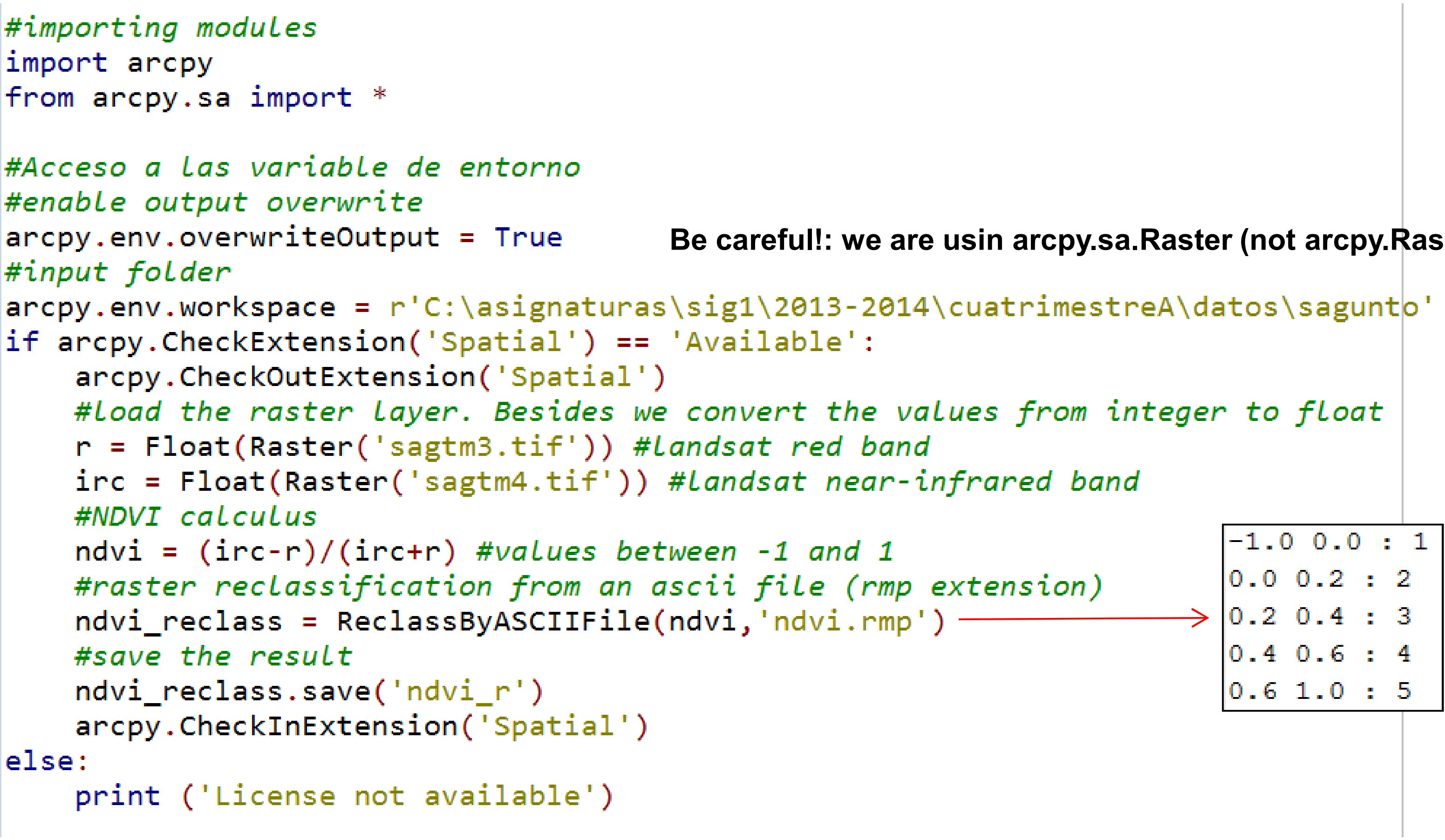
From ArcToolBox tool



("sagtm4.tif" - "sagtm3.tif")/("sagtm4.tif" + "sagtm3.tif")

Ráster de salida
C:\Users\jpalomav\Documents\ArcGIS\Default.gdb\rastercalc6

# Raster processing. Map algebra



```
#importing modules
import arcpy
from arcpy.sa import *

#Acceso a las variable de entorno
#enable output overwrite
arcpy.env.overwriteOutput = True
#input folder
arcpy.env.workspace = r'C:\asignaturas\sig1\2013-2014\cuatrimestreA\datos\sagunto'
if arcpy.CheckExtension('Spatial') == 'Available':
    arcpy.CheckOutExtension('Spatial')
    #load the raster layer. Besides we convert the values from integer to float
    r = Float(Raster('sagtm3.tif')) #landsat red band
    irc = Float(Raster('sagtm4.tif')) #landsat near-infrared band
    #NDVI calculus
    ndvi = (irc-r)/(irc+r) #values between -1 and 1
    #raster reclassification from an ascii file (rmp extension)
    ndvi_reclass = ReclassByASCIIFile(ndvi,'ndvi.rmp')
    #save the result
    ndvi_reclass.save('ndvi_r')
    arcpy.CheckInExtension('Spatial')
else:
    print ('License not available')
```

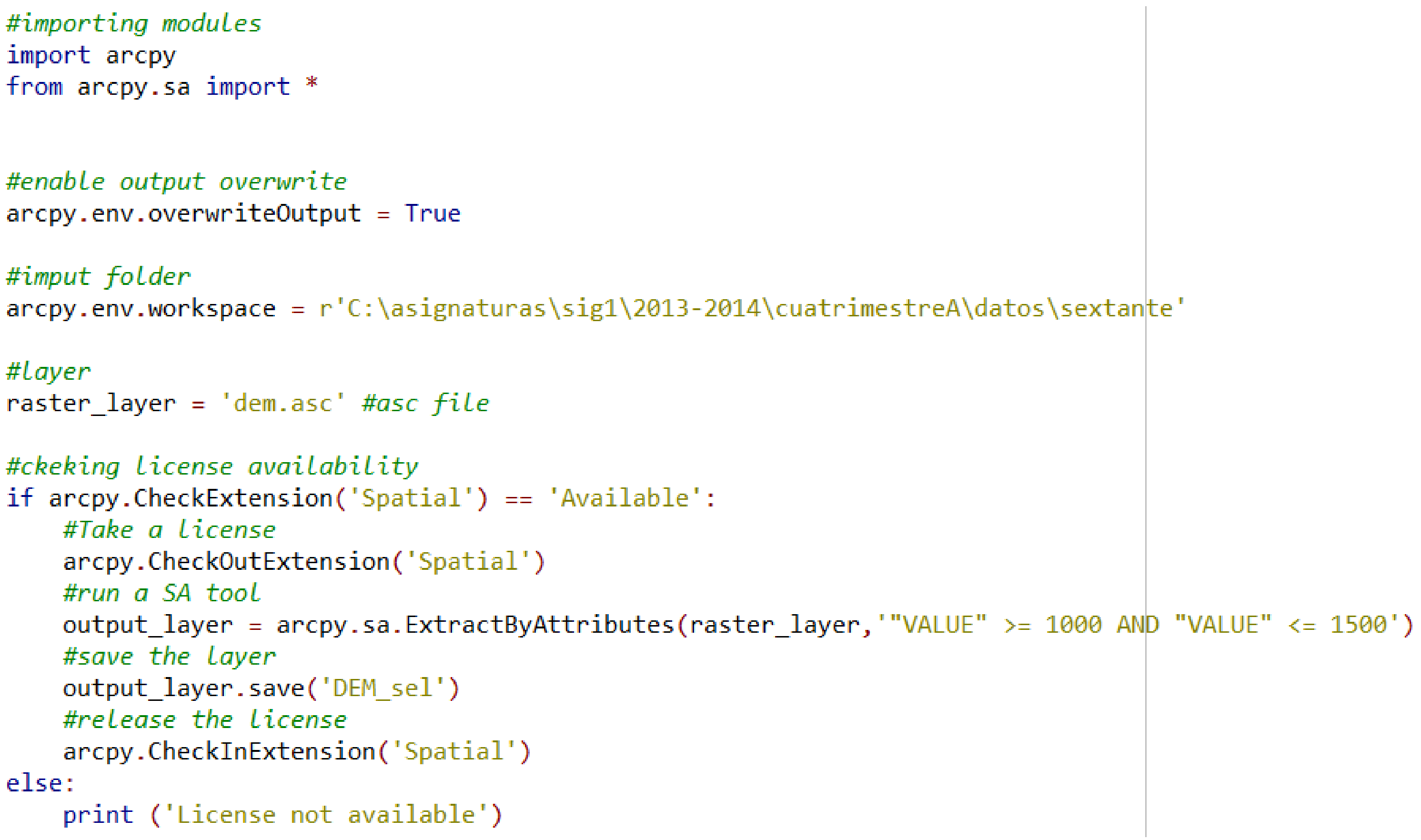
**Be careful!: we are usin arcpy.sa.Raster (not arcpy.Raster)**

```
-1.0 0.0 : 1
0.0 0.2 : 2
0.2 0.4 : 3
0.4 0.6 : 4
0.6 1.0 : 5
```

7_ndvi.py

# Raster processing. Map algebra

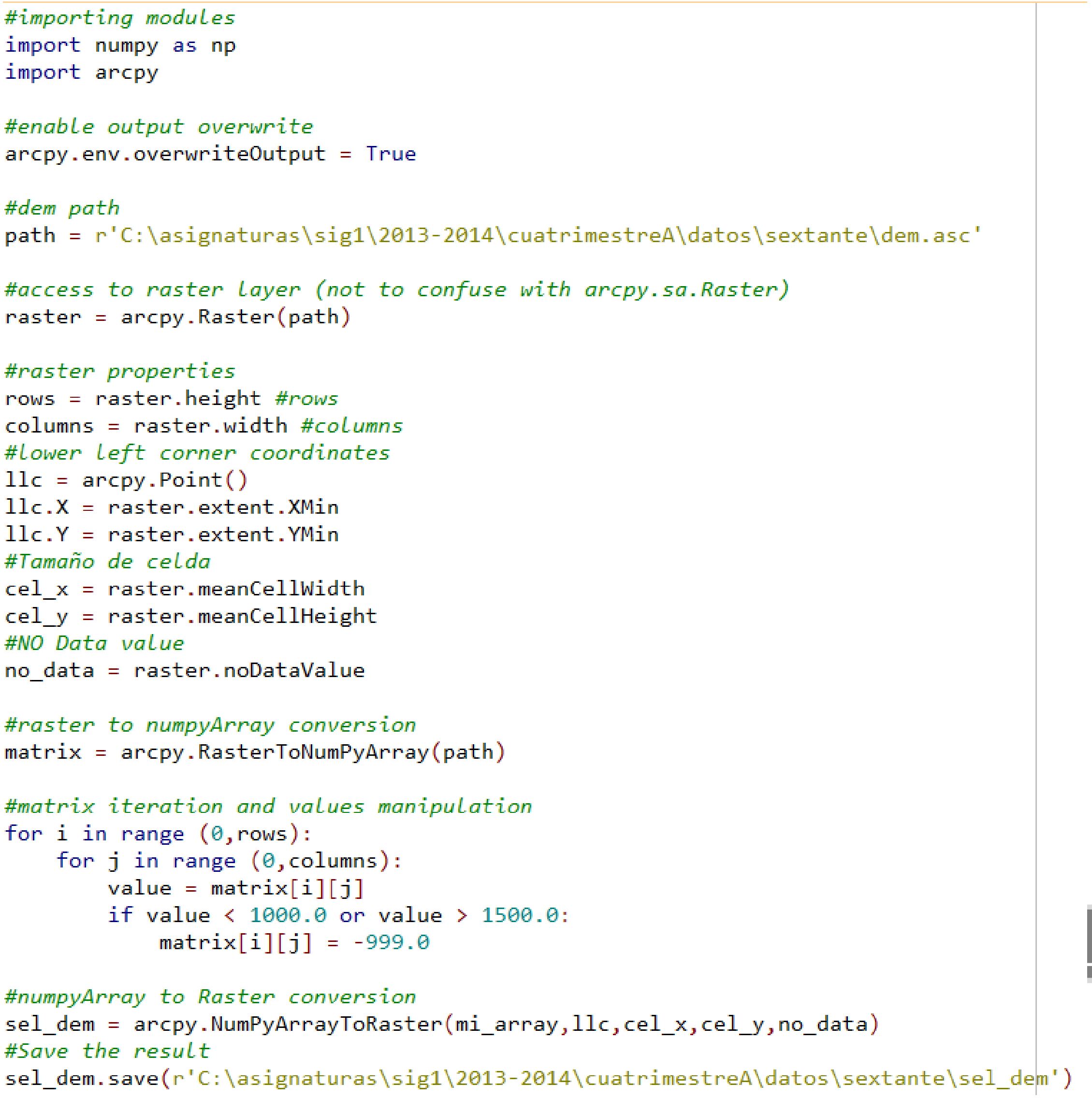
#Example 2: raster selection (using SA)



```
#importing modules
import arcpy
from arcpy.sa import *


#enable output overwrite
arcpy.env.overwriteOutput = True

#imput folder
arcpy.env.workspace = r'C:\asignaturas\sig1\2013-2014\cuatrimestreA\datos\sextante'

#layer
raster_layer = 'dem.asc' #asc file

#ckeking license availability
if arcpy.CheckExtension('Spatial') == 'Available':
    #Take a license
    arcpy.CheckOutExtension('Spatial')
    #run a SA tool
    output_layer = arcpy.sa.ExtractByAttributes(raster_layer,'"VALUE" >= 1000 AND "VALUE" <= 1500')
    #save the layer
    output_layer.save('DEM_sel')
    #release the license
    arcpy.CheckInExtension('Spatial')
else:
    print ('License not available')
```

# Operaciones raster. Algebra de mapas

#Example 2: raster selection
(pixel by pixel)



```
#importing modules
import numpy as np
import arcpy

#enable output overwrite
arcpy.env.overwriteOutput = True

#dem path
path = r'C:\asignaturas\sig1\2013-2014\cuatrimestreA\datos\sextante\dem.asc'

#access to raster layer (not to confuse with arcpy.sa.Raster)
raster = arcpy.Raster(path)

#raster properties
rows = raster.height #rows
columns = raster.width #columns
#lower left corner coordinates
llc = arcpy.Point()
llc.X = raster.extent.XMin
llc.Y = raster.extent.YMin
#Tamaño de celda
cel_x = raster.meanCellWidth
cel_y = raster.meanCellHeight
#NO Data value
no_data = raster.noDataValue

#raster to numpyArray conversion
matrix = arcpy.RasterToNumPyArray(path)

#matrix iteration and values manipulation
for i in range (0,rows):
    for j in range (0,columns):
        value = matrix[i][j]
        if value < 1000.0 or value > 1500.0:
            matrix[i][j] = -999.0

#numpyArray to Raster conversion
sel_dem = arcpy.NumPyArrayToRaster(mi_array,llc,cel_x,cel_y,no_data)
#Save the result
sel_dem.save(r'C:\asignaturas\sig1\2013-2014\cuatrimestreA\datos\sextante\sel_dem')
```

_raster_pix.py