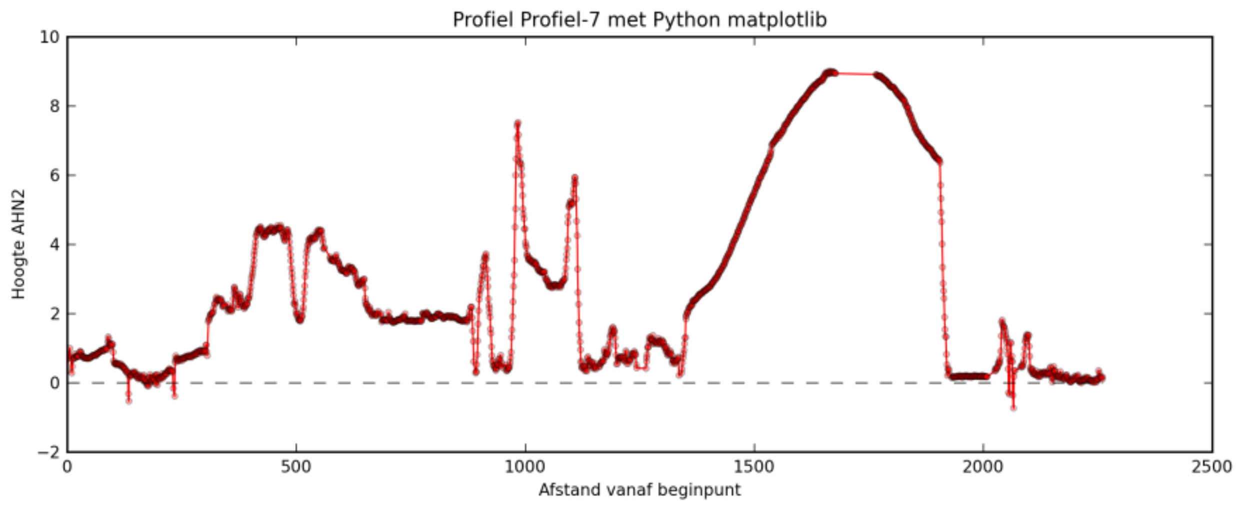
Using Python and matplotlib to create profile graphs
Some time ago I came across a blog post on ArcGIS 10.1 and matplotlib by Another GIS blog. From this blog I learned that matplotlib is part of ArcGIS 10.1 and you can create advanced graphs with it (see the matplotlib gallery).

The blog describes how to plot a profile based on a table holding x and y values (of d and z for that matter). I wanted to go a little further and extract the altitude directly from a raster based on points on a polyline. I didn't want to create intermediate featureclasses and make use of Spatial Analyst, so the Extract Values to Points tool was not an option.

Since ArcPy was introduced at 10.0 you can make use of Numpy arrays. Numpy arrays can be created from rasters in ArcPy. A Numpy array allows you to process raster data in a fast way. There are however some memory issues. Trying to convert a large raster (20-40GB) of a Water board into a Numpy will not work due to memory problems. It is necessary to extract only the relevant part of a raster.

See below an example of a profile graph created with matplotlib (*sorry for the Dutch titles*):



See code below, used to create a graph like this:

```
import arcpy
import os
import matplotlib.pyplot as plt
import numpy

#  settings for datasources
linesPath = "C:/Path/To/Folder/or/FileGeodatabase.gdb/myLines"
rasterPath = "C:/Path/To/Folder/or/FileGeodatabase.gdb/myDEM"

# settings for output profiles PNG's
profileFolder = "C:/Path/To/Output/Folder/png"
profilePrefix = "ArcPy_"
profilePostfix = "_new.png"
fldName = "Name"
```

```
# standard NoData mapping
NoDataValue = -9999

# describe raster
inDesc = arcpy.Describe(rasterPath)
rasMeanCellHeight = inDesc.MeanCellHeight
rasMeanCellWidth = inDesc.MeanCellWidth

# search cursor
fldShape = "SHAPE@"
flds = [fldShape, fldName]
with arcpy.da.SearchCursor(linesPath, flds) as curs:
    for row in curs:
       feat = row[0]
       profileName = row[1]

       # read extent of feature and create point for lower left corner
       extent = feat.extent
       print "Processing: {0}".format(profileName)
       pntLL = arcpy.Point(extent.XMin, extent.YMin)

       # determine number of rows and cols to extract from raster for this feature
       width = extent.width
       height = extent.height
       cols = int(width / rasMeanCellWidth)
       rows = int(height / rasMeanCellHeight)

       # create Numpy array for extent of feature
       arrNP = arcpy.RasterToNumPyArray(rasterPath, pntLL, cols, rows, NoDataValue)

       # create empty arrays for distance (d) and altitude (z) and NAP line (zv)
       d = []
       z = []
       zv = []

       # loop through polyline and extract a point each meter
       for dist in range(0, int(feat.getLength("PLANAR"))):
              XYpointGeom = feat.positionAlongLine(dist, False)
              XYpoint = XYpointGeom.firstPoint

              # translate  XYpoint to  row, col (needs additional checking)
              c = int((XYpoint.X - pntLL.X) / rasMeanCellWidth)
              r2 = int((XYpoint.Y - pntLL.Y) / rasMeanCellHeight)
              r = rows - r2
              if c >= cols:
                     c = cols-1
              if r >= rows:
                     r = rows-1

              # extract value from raster and handle NoData
              zVal = arrNP[r,c]
              if not zVal == NoDataValue:
                     d.append(dist)
                     z.append(arrNP[r,c])
                     zv.append(0)
```

```
        # define output profile filename and figure size settings
        elevPNG = profileFolder + os.sep + profilePrefix + profileName +
                     profilePostfix
        fig = plt.figure(figsize=(10, 3.5)) # inches

        # plot profile, define styles
        plt.plot(d,z,'r',linewidth=0.75)
        plt.plot(d,z,'ro',alpha=0.3, markersize=3)
        plt.plot(d,zv,'k--',linewidth=0.5)
        plt.xlabel('Distance from start')
        plt.ylabel('Elevation')
        plt.title('Profile {0} using Python matplotlib'.format(profileName))

        # change font size
        plt.rcParams.update({'font.size': 8})

        # save graph as PNG
        fig.savefig(elevPNG, dpi=300)

# delete some objects
del arrNP, XYpointGeom, XYpoint
print "PNGs stored in: {0}".format(profileFolder)
```

How it works

The code below basically does the follwing thing:

- access raster and get the MeanCellHeight and MeanCellWidth (Cell size)
- create a cursor with the polyline features using the new and enhanced data access module arcpy.da
- for each feature read the extent of the feature
    - obtain lower left corner of extent
    - calculate the number of rows and columns required for extent
    - extract the extent from raster into the numpy array
        - extract points every 1 meter interval on polyline
        - determine raster value at point and populate some arrays with distance from FromPoint and raster value
    - plot the resulting arrays using matplotlib
    - export the graph to PNG

Resources

- Another GIS blog on ArcGIS 10.1 and matplotlib
- Matplotlib User's Guide online version
- Accessing data using cursors in Python
- Python wiki on For Loops
- Matplotlib color API
- Matplotlib customizing
- Matplotlib gallery
- Matplotlib How to and FAQ
- Numpy
- Using Numpy arrays in ArcGIS