WPS primer (using Python)

Extracting data from the WPS server is a three step process

1. Get information about available processes (GetCapabilities)
2. Get information about a specific process (DescribeProcess)
3. Execute the process (Execute)

Prerequisites

Handling WPS processes from Python require at least the following package to be installed:

- OWSLib package (link to geopython github)
- Matplotlib (for plotting)
- pandas (for dataframe/table processes)
- cStringIO (for in memory file handling)

GetCapabilities

The first step is to ask the server which processes are being offered. To get this information we do a GetCapabilities request. The request returns an xml response that contains a list of processes that are provided by the server.

OWSLib GetCapabilities

```
# import owslib wps part dealing with WebProcessingServices
from owslib.wps import WebProcessingService

# define the URL of the WPS
url = 'http://wps.openearth.nl/wps'

# define the WPS
wps = WebProcessingService(url, verbose=False, skip_caps=True)
wps.getcapabilities()
wps.identification.title

>>> 'PyWPS Server'

wps.identification.abstract

>>> wps.identification.abstract

# find out which processes there area
for process in wps.processes:
    print process.identifier, process.title

# printing the process characteristics result in the following list of available processes
- constituents Lookup constituents based on their short name
- tidal_predict Tidal prediction tool
- IDT_simple Interactive Dredge Planning Tool
- hymos_doublemass Double Mass Plot
- hymos_timeseries_example Example timeseries
- gwdlr_get_river_profiles This process returns shape files with riverprofiles and locations from the gwdlr
database
- addtwonumbers None
- oil_spill None
- t_tide None
- tide_analysis None
- tide_predict None
- WPSdataprofeet None
```

DescribeProcess

After inspecting the GetCapabilities document and choosing the process you like, you need to find out how to execute the process. For execution you need at least all all requested input parameters. This is done by means of the 'DescribeProcess' request. This request returns an xml containing all in- and outputs.

OWSLib describeprocess

```
# in analogy of the WPS Primer using matlab tidal_predict will be described
# DescribeProcess gives insight in input and output parameters

process = wps.describeprocess('tidal_predict')
process.abstract
for input in process.dataInputs:
    print input.title, input.identifier, input.dataType, input.defaultValue

for output in process.processOutputs:
    print output.title, output.identifier, output.dataType, output.defaultValue
```

Execute

Once you've found and specified all input parameters you are ready to execute the process by means of the 'Execute' request. The service returns an xml containing all process outputs. These outputs can be read by Matlab and can also be plotted or used for other operations.

OWSLib Exectute process

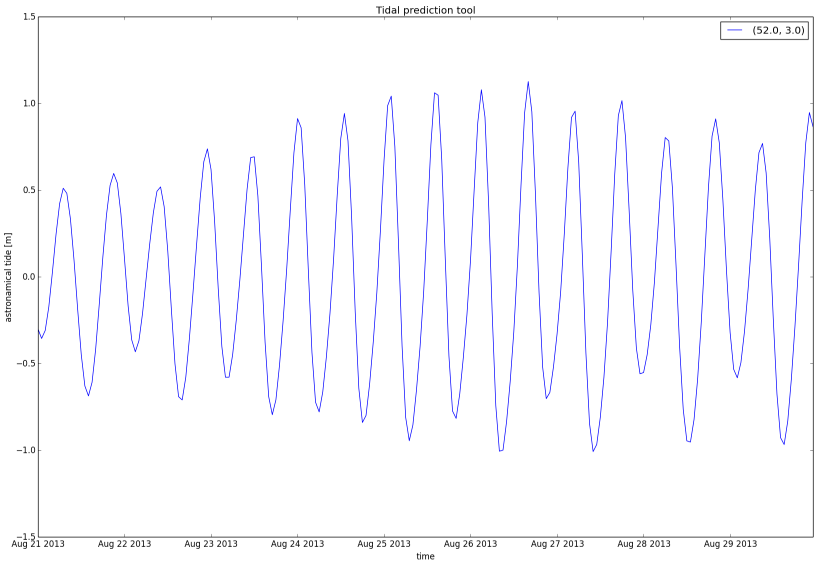
```
# define inputs, which is an array with several objects of several dataTypes (as listed below)
inputs = [('location','POINT(3 52)'),
                  ('startdate','2013-08-21 00:00'),
                  ('enddate','2013-08-29 23:00'),
                  ('frequency','HOURLY')]

execution = wps.execute(process.identifier,inputs)
for output in execution.processOutputs:
    print output.identifier
    print output.data

# this will yield data for tide for 1 location for the given datetime range
```

Plotting the data

Output.data can be plotted. Following code can be used to derive this plot.



Plot WPS ouput

```
import pandas
import matplotlib.pyplot as plt
import cStringIO

# data is in table format and can be transferred to a in memory file object
f = cStringIO.StringIO(output.data[0])

# use pandas dataframe to work with the data
df = pandas.DataFrame.from_csv(f)
grouped = df.groupby(by=['lat','lon'])

# make figure and axes
fig, ax = plt.subplots(figsize=(20,13))
# add a line for each group
for (index, group) in grouped:
    ax.plot(group.index, group.h, label=str(index))
# add the legend
ax.legend(loc='best')
ax.set_xlabel('time')
ax.set_ylabel('astronamical tide [m]')
ax.set_title("{}".format(process.title))

# save the figure
fig.savefig(r'd:\temp\wps.png')
```

Besides points the code above also works for lines. Check this example on OpenEarth Notebooks github for ready to use IPython Notebook code.