Example: using the National Water Model (NWM) Data Component to download data

The NWM Data Component is implemented with a Python class called **BmiNwmHs**.

Import BmiNwmHs class and instantiate it. A conf guration f le (yaml f le) is required to provide the parameter settings for data download.

In [2]:
```
import matplotlib.pyplot as plt
import numpy as np
import cftime

from nwm import BmiNwmHs

# initiate a data component
data_comp = BmiNwmHs()
data_comp.initialize('config_file.yaml')
```

Use variable-related methods from BmiNwmHs class to check the variable information of the NWM dataset. This data component stores a f ow forecast variable.

In [3]:
```
# get variable info
var_name = data_comp.get_output_var_names()[0]
var_unit = data_comp.get_var_units(var_name)
print(' variable_name: {}\n var_unit: {}\n'.format(var_name, var_unit))
```

```
 variable_name: Flow Forecast
 var_unit: cfs
```

Use time-related methods to check the time information of the NWM dataset. The time values are stored in a format that follows the CF convention (http://cfconventions.org/Data/cf-conventions/cf-conventions-1.8/cf-conventions.pdf).

In [4]:
```
# get time info
start_time = data_comp.get_start_time()
end_time = data_comp.get_end_time()
time_step = data_comp.get_time_step()
time_unit = data_comp.get_time_units()
time_steps = int((end_time - start_time)/time_step) + 1
print(' start_time:{}\n end_time:{}\n time_step:{}\n'
      + ' time_unit:{}\n time_steps:{}\n'.format(start_time,
                                                 end_time,
                                                 time_step,
                                                 time_unit,
                                                 time_steps))
```

```
 start_time:{}
 end_time:{}
 time_step:{}
 time_unit:1503450000.0
 time_steps:1503511200.0
```

Loop through each time step to get the f ow and time values. stream_array stores f ow forecast values. cftime_array stores the numerical time values. time_array stores the corresponding Python datetime objects. get_value( ) method returns the f ow forecast value at each time step. update( ) method updates the current time step of the data component.

In [5]:
```
# initiate numpy arrays to store data
stream_value = np.empty(1)
stream_array = np.empty(time_steps)
cftime_array = np.empty(time_steps)

for i in range(0, time_steps):
    data_comp.get_value(var_name, stream_value)
    stream_array[i] = stream_value
    cftime_array[i] = data_comp.get_current_time()
    data_comp.update()

time_array = cftime.num2date(cftime_array,
                             time_unit,
                             only_use_cftime_datetimes=False,
                             only_use_python_datetimes=True)
```