1 Multiscale modelling

Program structure is following.

$\langle$***$\rangle\equiv$
  $\langle$*Import*$\rangle$
  $\langle$*main.py*$\rangle$

Dependencies and import We need lxml.etree for XML parsing. To print error and warning messages, I wrote a small module DebugModule by myself. This module is in file debug.nw. We'd be needing these modules in almost every python file. Lets keep them in one chunck Import.

Imports Lets just import the mimimal required. We'll try import cElement tree first and then some other implementation of etree. Then we'll add a command line parser to get filepaths of XML. And why not, let parse these models too.

2 Parser

Let's include the parser here.

3 XML parser

This document deals with parser of XML models.

⟨*parser.py*⟩≡
```
import debug
import os

def getModels(models) :

    """

    Parses given xml models. We can pass either one or two models; one described
    in neuroML and the other in sbml.

    Notes: Document is properly. See

      http://www.biomedcentral.com/1752-0509/7/88/abstract

    sent by Aditya Girla. It a online composition tool for SBML. In its
    references, some other tools are mentioned.

    Args :

    Raises :

    Return
    return a list of elementTree of given models.

    """
    elemDict = dict()
    if models.nml :
        # Get the schema
        with open(models.nml, "r") as nmlFile :
            elemDict['nml'] = models.nml

    if models.sbml :
        elemDict['sbml'] = models.sbml
    return elemDict
```

Parse xml Let's write a function to parse xml and return the root element.

⟨*parser.py*⟩+≡

```
def parseModel(modelPath):
    if not os.path.exists(modelPath) :
      pass
```

Print debugging messages
Different type of messages are printed in different colors.

⟨*debug.py*⟩≡
```
HEADER = '\033[95m'
OKBLUE = '\033[94m'
OKGREEN = '\033[92m'
WARNING = '\033[93m'
ERR = '\033[91m'
ENDC = '\033[0m'
RED = ERR
WARN = WARNING
INFO = OKGREEN
TODO = OKBLUE
DEBUG = HEADER

prefix = dict(
    ERR = ERR
    , WARN = WARN
    , FATAL = ERR
    , INFO = INFO
    , TODO = TODO
    , NOTE = HEADER
    , DEBUG = DEBUG
    )

def colored(msg, label) :
    """
    Return a colored string. Formatting is optional.
    """
    global prefix
    if label in prefix :
        color = prefix[label]
    else :
        color = ""
    return "[{0}] {1} {2}".format(label, color+msg, ENDC)

def printDebug(label, msg):
    print(colored(msg, label))
```

⟨*main.py*⟩≡
```
⟨Import⟩
⟨functions in main⟩
⟨argument parser⟩
⟨parse models⟩
```

We prefer cElementTree for its speed. If it is not available then go for something else.

⟨*Import*⟩≡
```
import logging
import debug
logger = logging.getLogger('multiscale')
try:
    import cElementTree as etree
    debug.printDebug("DEBUG", "running with lxml.etree")
except ImportError:
    try:
        # Python 2.5
        import xml.etree.cElementTree as etree
        debug.printDebug("DEBUG", "running with cElementTree")
    except ImportError:
        try:
            # Python 2.5
            import xml.etree.cElementTree as etree
            debug.printDebug("DEBUG", "running with ElementTree")
        except ImportError:
            try:
              # normal cElementTree install
              import cElementTree as etree
              debug.printDebug("DEBUG", "running with cElementTree")
            except ImportError:
                try:
                    # normal ElementTree install
                    import elementtree.ElementTree as etree
                    debug.printDebug("DEBUG", "running with ElementTree")
                except ImportError:
                    try :
                      import lxml.etree as etree
                    except ImportError :
                        debug.prefix("FATAL", "Failed to import ElementTree")
                        os._exit(1)
```

Agrument parser Paths of models files are to be passed from the command line. More than one xml file can be passed. We use `argparse` library to build a command-line interface.

⟨*argument parser*⟩≡
```
import argparse

# This section build the command line parser
argParser = argparse.ArgumentParser(description= 'Mutiscale modelling of neurons')
argParser.add_argument('--nml', metavar='nmlpath'
    , help = 'File having neuron described in neuroML'
    )
argParser.add_argument('--sbml', metavar='sbmlpath'
    , help = 'File having neuron described in SBML'
    , required = False
    )
args = argParser.parse_args()
```

Parse xml models We pass two kind of models from command line to this application, `sbml` and `neuroML`. Parse them and we'll think of next step. We need at least one model to start with (neuroML?).

But before we parse, we need a helper function to check if given paths are correct.

⟨*functions in main*⟩≡
```
def ifPathsAreValid(paths) :
  ''' Verify if path exists and are valid. '''
  if paths.nml :
    if os.path.isfile(paths.nml) : pass
    else :
      debug.printDebug("ERROR", "Filepath {0} is not valid".format(paths.nml))
      return False
  if paths.sbml :
    if os.path.isfile(paths.sbml) : pass
    else :
      debug.printDebug("ERROR", "Filepath {0} is not valid".format(paths.sbml))
      return False
  return True
```

Parse xml files.

⟨*parse models*⟩≡

```
# There must be at least one model present
import parser
if args.nml or args.sbml :
  if ifPathsAreValid(args) :
    logger.info("Started parsing XML models")
    debug.printDebug("INFO", "Started parsing XML models")
    etreeList = parser.parseModels(args)
  else :
    debug.printDebug("FATAL", "One or more model file does not exists.")
    sys.exit()
else :
  debug.printDebug("FATAL", "Please provide at least one model. None given.")
  sys.exit()

debug.printDebug("INFO", "Parsing of models is done")
```