# Assignment 1: Reading a NEWICK tree

1. use the Python fragment `firsttree.py` and complete the function `myread(self, newick, p)` in the `Tree` class. `newick` is the string that contains the Newick tree, for example test your program using this tree: `((frog:0.1,fish:0.2):0.5,(cat:0.1,dog:0.1):0.8);`. But your program should be able to read any tree (I will use a different Newick string to test your code). Using one of the two algorithms specified in the handout to solve this assignment (recursive or non-recursive tree reading. If you are uncertain about Python, talk to Kyle and see him during the open-lab time.

2. Ideally, you would read the Newick string from a file, but if you are a Python beginner, then define the Newick string near the beginning of your main code section.

Make sure that your code is well documented and follows good programming practices. Mail the source code, a user description (what is input, how to run, what is output), and the example data sets to Peter (beerli@fsu.edu) no later than **Tuesday September 12**. Best practices are to send a compressed folder (zip or .tar.gz) containing all files. The name of the folder should be your first name and the assignment number (for example peter1).

```
# node and tree class
# class Node defines:
#     -Node (see __init__)
#     -tip          : defines tip node and sets label
#     -interior     : defines interior node and sets
    connections left and right
#     -myprint      : prints name and branchlength associated
    with a node
#     -debugprint   : prints the content of a node
# class Tree defines:
#     -Tree (see __init__) : defines root
#     -myprint      : prints the tree in NEWICK format
#your assignment     -myread       : reads a NEWICK string and
    creates a tree
#     -printTiplabels: prints tip labels
#
# PB Oct 2011 (2017 reviewed and revised)
from __future__ import print_function
import sys
import random
import math


class Node:
    """
    Node class: this is a container for content that is saved
    at nodes in a tree
    """
    def __init__(self):
        """
        basic node init
        """
        self.name = ""
        self.left = -1
        self.right = -1
        self.ancestor = -1
        self.blength = -1
        self.sequence = []

    def tip(self,name, blength=-1):
        """
        sets the name of a tip
        """
        self.name = name
        self.left = -1
```

```
        self.right = -1
        self.ancestor = -1
        self.blength = blength

    def interior(self,left,right,blength=-1):
        """
        connects an interior node with the two descendents
        """
        self.name = ""
        self.left = left
        self.right = right
        self.ancestor = -1
        self.blength = blength

    def myprint(self):
        """
        Prints the content of a node: name if any and
            branchlengths if any
        """
        if(self.name!=""):
            print(self.name,end=' ')
        if(self.blength != -1):
            print(":{}".format(self.blength),end=' ')

    def debugprint(self):
        """
        Prints the content of a node: name if any and
            branchlengths if any
        """
        print("Name:", self.name)
        print("Descendents:", self.left, self.right)
        print("Branch-length: {}".format(self.blength))

class Tree(Node):
    """
    Node class: this is a container for content that is saved
    at nodes in a tree
    """
    i = 0

    def __init__(self, rootnode):
        self.root = rootnode
        #self.root.name = "root"

    def myprint(self):
```

```
        self.mynodeprint(self.root)
        print(";")


    def mynodeprint(self,p):
        """
        prints nodes in a tree recursively in Newick format
        """
        if(p.left != -1):
            print("(",end=' ')
            self.mynodeprint(p.left)
            print(",",end=' ')
        if(p.right != -1):
            self.mynodeprint(p.right)
            print(")",end=' ')
        p.myprint()
        print("",end=' ')


    # assignment 1 create this function
    def myread(self,newick, p):
        """
        reads a tree in newick format
        """
        pass



if __name__ == '__main__':
    # we create a small tree as a test for the printing
        functions
    # with 3 species, 1 interior node and 1 root node
    a = Node()
    a.tip('A',1)
    b = Node()
    b.tip('B',1)
    c = Node()
    c.tip('C',3)
    d = Node()
    d.interior(a,b,2)
    e = Node()
    e.interior(d,c,0)
    mytree = Tree(e)
    mytree.myprint()
    a.debugprint()
```

```
    d.debugprint()
```