Practical Python:

3: Libraries, Graphics and More: NumPy, MatPlotLib, NLTK

Dafydd Gibbon

First South African Workshop in Digital Humanities
North-Western University, Potchefstroom, SA
2015-04-04 to 2015-04-05

Installing packages and importing modules

- Before a module or module library can be imported into Python, the files containing the module or module library must be installed.

- Some libraries use modules in other libraries, and in general these other libraries are automatically installed when the library which needs them is installed.

- There are a number of Python installation tools:
  - packages which are included in Anaconda are installed for use in Miniconda with:
    `conda install <packagename>`
  - packages which are not included in the comprehensive Anaconda package are installed with:
    `pip install <packagename>`

Importing library modules and using objects in modules

`import numpy`

- This format requires the module name to be prefixed to functions:

```
numpy.median([3,2,3,4,5,4,3,2,3,4])
```

`import numpy as np`

- This format allows the prefixed module name to be abbreviated:

```
np.median([3,2,3,4,5,4,3,2,3,4])
```

`from numpy import median`

- Allows functions to be used without prefixing the module name – not normally a good idea, can lead to name clashes:

```
median([3,2,3,4,5,4,3,2,3,4])
```

`from numpy import *`

- Allows all submodules, functions and data in the module to be used without the module name, not normally a good idea:

```
median([3,2,3,4,5,4,3,2,3,4])
std([3,2,3,4,5,4,3,2,3,4])
```

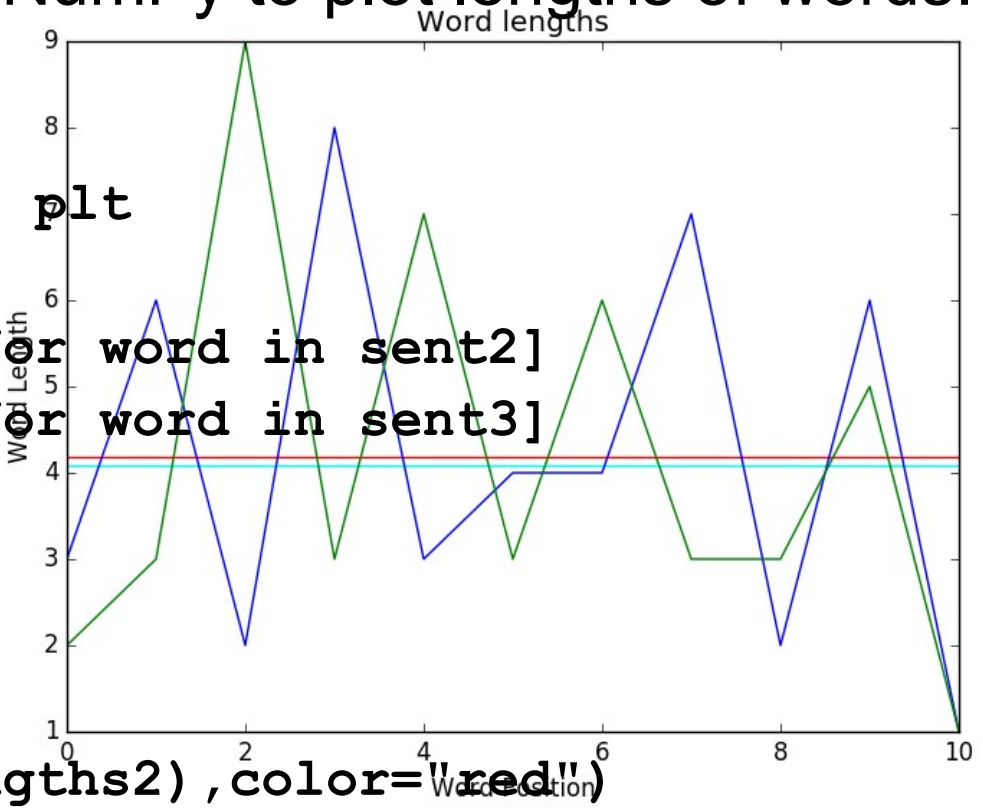
Importing submodules: plot graphics demonstration

- Submodules may also be imported, as in this demonstration of using MatPlotLib, NLTK and NumPy to plot lengths of words:

```
import numpy as np
import matplotlib.pyplot as plt
from nltk.book import *
wordlengths2 = [len(word) for word in sent2]
wordlengths3 = [len(word) for word in sent3]

plt.title("Word lengths")
plt.xlabel("Word Position")
plt.ylabel("Word Length")
plt.axhline(np.mean(wordlengths2),color="red")
plt.axhline(np.mean(wordlengths3),color="cyan")
plt.plot(wordlengths2)
plt.plot(wordlengths3)

plt.show()
plt.clf()
```



For further information check the MatPlotLib wiki: http://matplotlib.org/

Importing libraries

NumPy: mathematical library

- arrays with matrix operations, efficient algorithm implementations

```
import numpy
x = numpy.sqrt(25)

import numpy as np
x = np.sqrt(25)

from numpy import sqrt
x = sqrt(25)

a = [[1,2,3,4],[3,4,5,6]]
b = np.array([[1,2,3,4],[3,4,5,6]])
```

- `a * a` (throws an error)
- `b * b (yields the product of corresponding cells.`

Assignment:

Find and plot time series data on the web (e.g. some value per day, of month, year etc.).

Other libraries which I use frequently

Regular expression handling

```
import re
```

System calls

```
import sys, os, traceback, inspect
import time
```

Networking

```
import socket
```

Server-side Web apps

```
import cgi cgitb ; cgitb.enable()
```

Modules – your own libraries

- Code for any frequently used functionality can be re-used just like any other library, as a module.

- For example:

```
def errormessage(string):
    print 'Error message:', string
    return
```

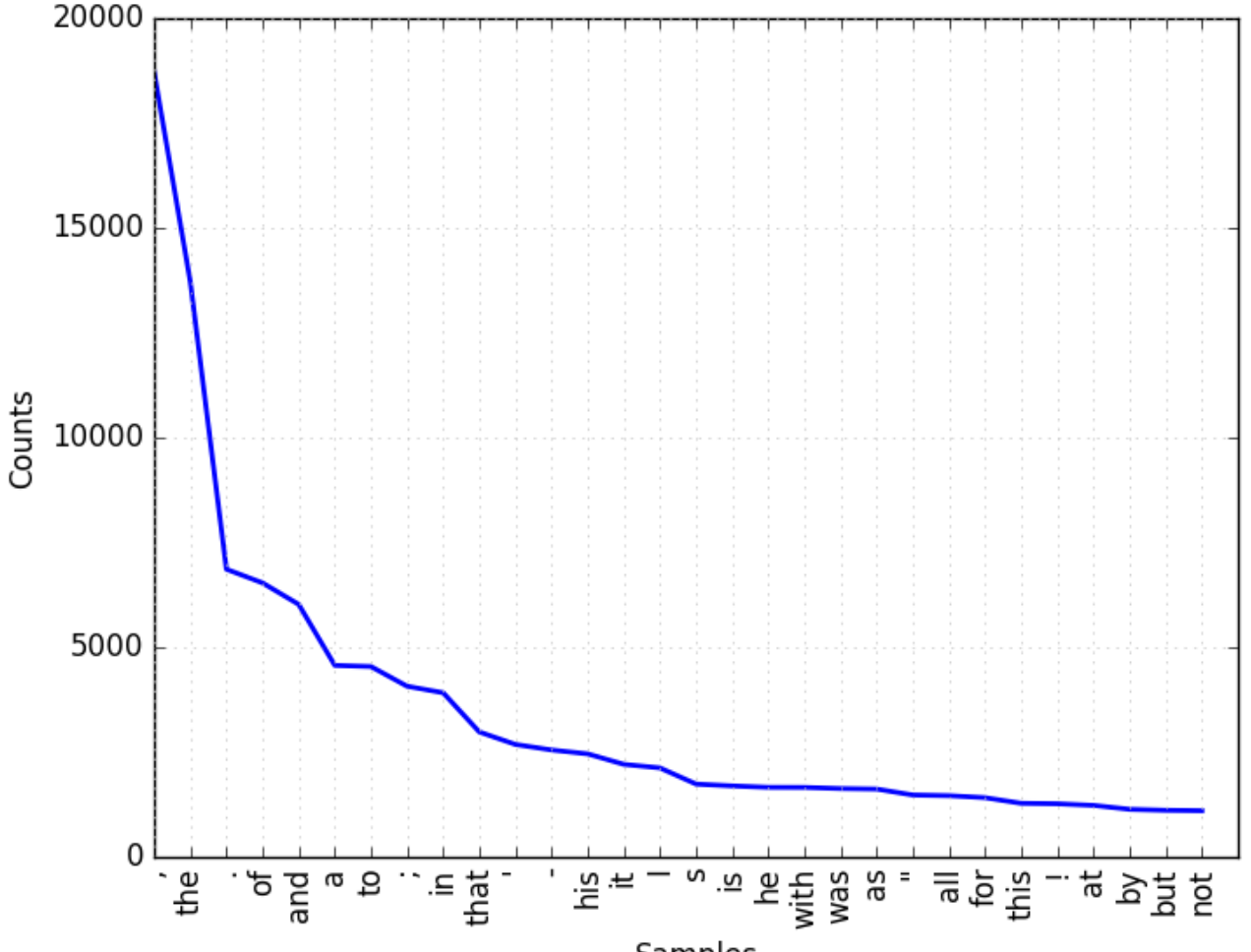
- Save as **`errhandler.py`**

- Then

```
import errhandler.py as eh
eh.errmessage('Sorry, that did not work.')
```

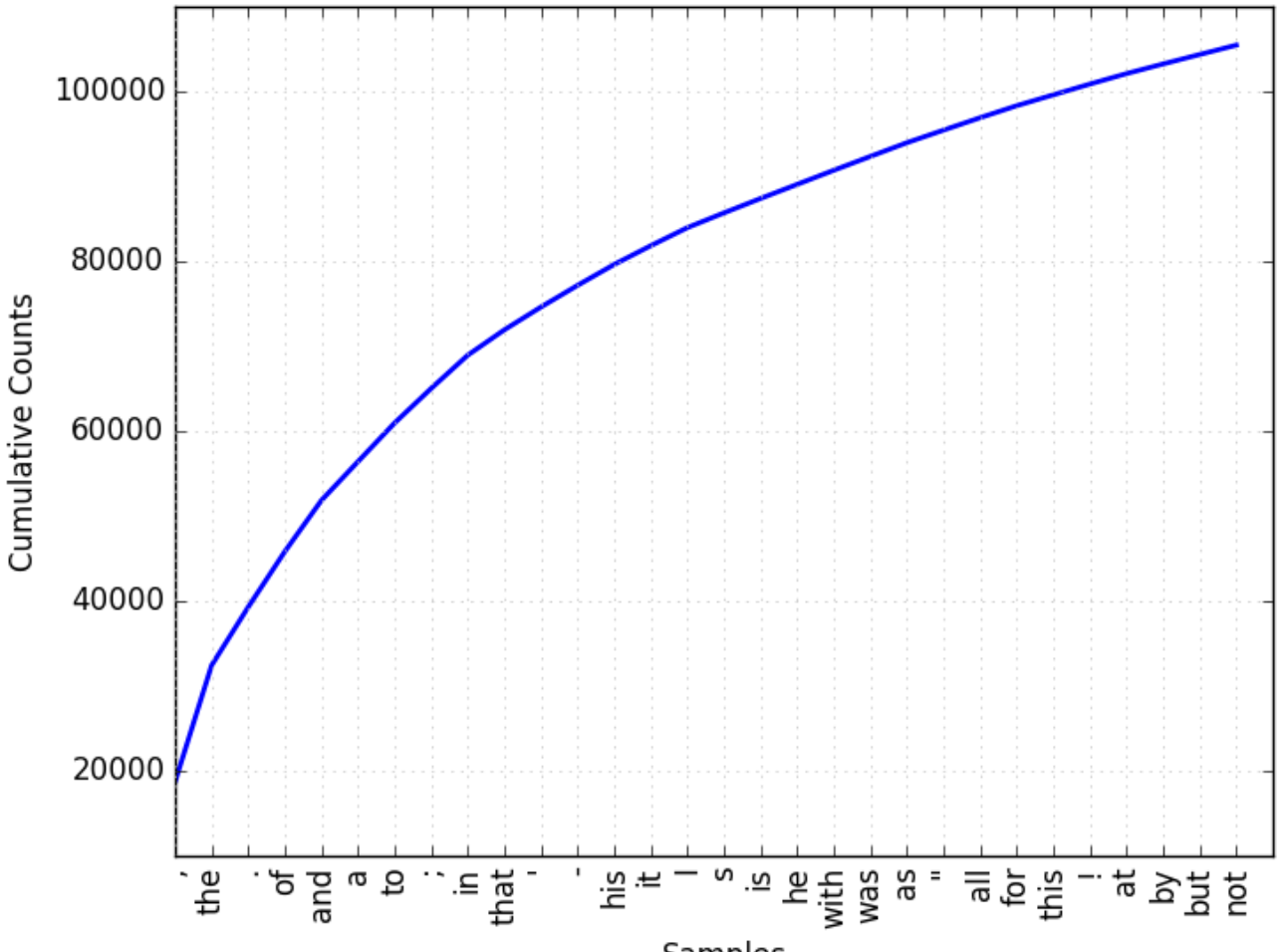
NLTK example: word frequency distributions

```
fdist1 = FreqDist(text1)
fdist1.plot(30)
```



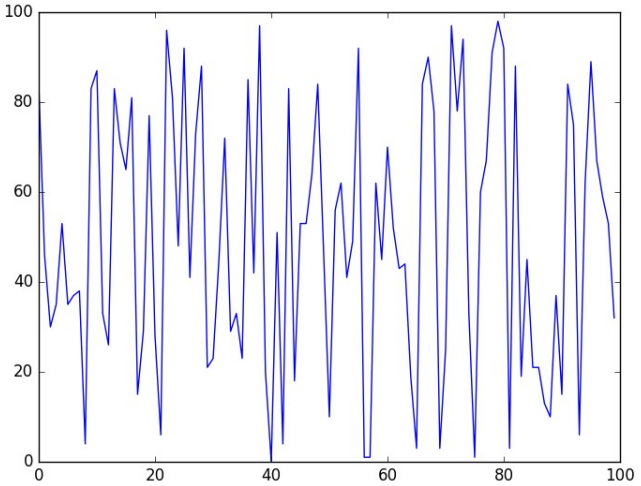
NLTK example: word frequency distributions

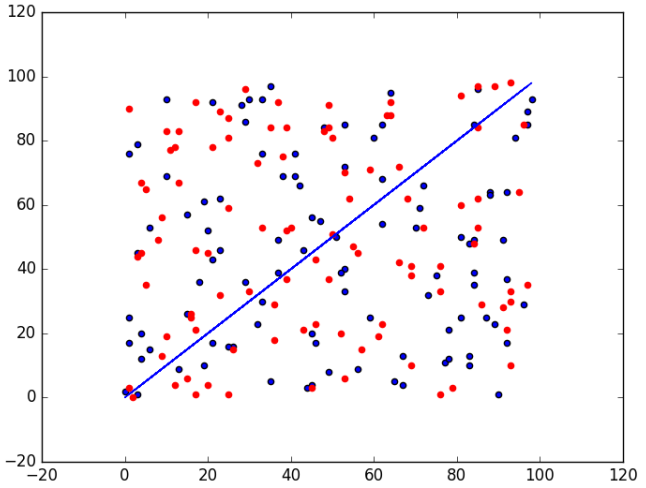
```
fdist1 = FreqDist(text1)
fdist1.plot(30,cumulative=True)
```

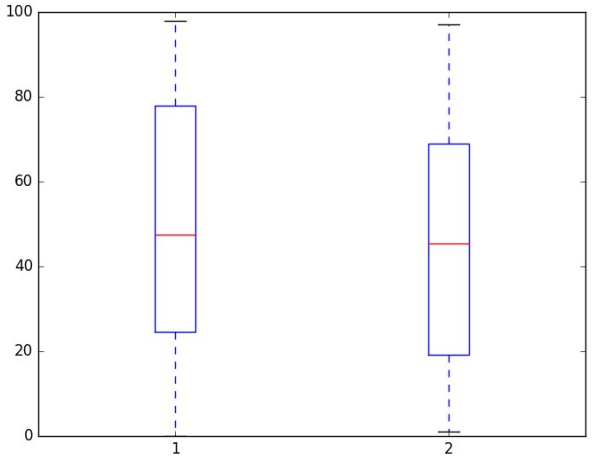


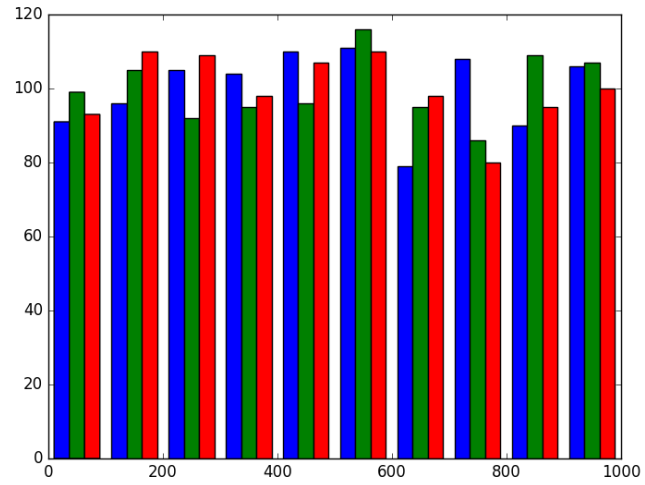
Specimen plots: sequence, scatter, box, histogram, tree

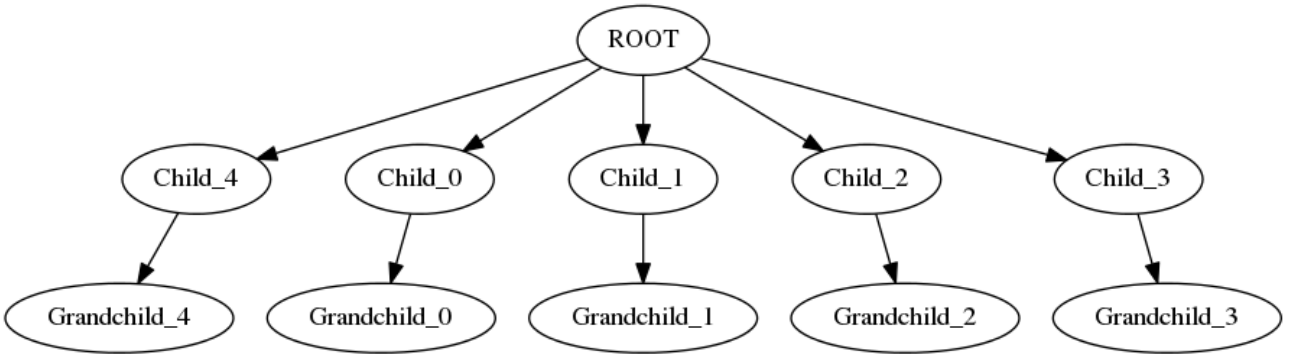
with MatPlotLib and friends











MatPlotLib: sequence plots

```
import matplotlib.pyplot as plt
import random as rnd

n = 100
rng = range(n)

a = [int(100*rnd.random()) for x in rng]
b = [int(100*rnd.random()) for x in rng]
c = [a, b]

for data in c:
    plt.plot(a)
    plt.plot(b,color='red')
plt.show()
plt.clf()
```

Assignment:

Find and plot time series data on the web (e.g. some value per day, month, year etc.).

MatPlotLib: sequence plots

```
import matplotlib.pyplot as plt
import random as rnd

n = 100
rng = range(n)

a = [int(100*rnd.random()) for x in rng]
b = [int(100*rnd.random()) for x in rng]
c = [int(100*rnd.random()) for x in rng]

for d in [a, b, c]:
    plt.plot(d)

plt.show()
plt.clf()
```

Assignment:

Find and plot time series data on the web (e.g. some value per day, month, year etc.).

MatPlotLib: scatter plots

```
import matplotlib.pyplot as plt
import random as rnd

n = 100
rng = range(n)

a = [int(100*rnd.random()) for x in rng]
b = [int(100*rnd.random()) for x in rng]

plt.scatter(a,b)

plt.show()
plt.clf()
```

Assignment:

Find and plot relational data on the web, e.g. age vs. height, age vs. wealth, ...

MatPlotLib: scatter plots

```
import matplotlib.pyplot as plt
import random as rnd

n = 100
rng = range(n)

a = [int(100*rnd.random()) for x in rng]
b = [int(100*rnd.random()) for x in rng]

plt.scatter(a,b)
plt.scatter(b,a,color='red')
plt.plot(a,a)
plt.show()
plt.clf()
```

Assignment:

Find and plot relational data as before, but separately for males and females.

MatPlotLib: boxplots

```
import matplotlib.pyplot as plt
import random as rnd

n = 100
rng = range(n)

a = [int(100*rnd.random()) for x in rng]
b = [int(100*rnd.random()) for x in rng]

plt.boxplot([a,b])
plt.show()
plt.clf()
```

Assignment:

Find and plot data sets to compare different categories (e.g. males vs. females)

MatPlotLib: histograms

```
import matplotlib.pyplot as plt
import random as rnd

n = 1000
rng = range(n)

a = [int(1000*rnd.random()) for x in rng]

plt.hist(a)
plt.show()
plt.clf()
```

Assignment:

Find and plot data sets to compare different categories (e.g. males vs. females)

Histograms

```
import matplotlib.pyplot as plt
import random as rnd

n = 1000
rng = range(n)

a = [int(1000*rnd.random()) for x in rng]
b = [int(1000*rnd.random()) for x in rng]
c = [int(1000*rnd.random()) for x in rng]

plt.hist([a,b,c])
plt.show()
plt.clf()
```

Assignment:

Find and plot data sets to compare different categories (e.g. males vs. females)

Creating a very basic graphics module (p.1)

- Create a file **`graphmod.py`** with the following definitions:

```
import matplotlib.pyplot as plt
import random as rnd

def sequence(listoflists):
    for d in listoflists:
       plt.plot(d)
    plt.show()
    plt.clf()

def box(listoflists):
    plt.boxplot(listoflists)
    plt.show()
    plt.clf()
```

Creating a very basic graphics module (p.2)

```
def histogram(listoflists):
    plt.hist(listoflists)
    plt.show()
    plt.clf()

def scatt(listoflistpairs):
    if len(listoflistpairs)>7:
        return
    clrs = [c for c in 'bgrcmyk']
    for d in listsofpairs:
        plt.scatter(d[0],d[1],color=clrs[0])
        clrs = clrs[1:]
    plt.show()
    plt.clf()
```

For each function, input is a list of datasets.
For **`scatter`**, each dataset (< 8) is a pair of equal length lists, e.g.

```
[ [ [1,2], [3,4] ],
  [ [7,9,8], [8,6,9] ],  …
]
```

Otherwise input is a list of lists of values.

Usage example:

```
import graphmod as gr
data = [ [[1,2],[3,4]], [[7,9,8],[8,6,9]] ]
gr.scatter(data)
```

Network plotting

- The following examples are concerned with the visualisation of tree and network graphs
- The examples are more advanced, in that they require additional software:
  - the module NetworkX, so that this module has to be installed into your Python installation, in order to be imported into your application
  - the independent graphics visualisation tool GraphViz, which has to be installed on your PC (independently of Python)
- Hint:
  - packages which are included in Anaconda are installed for use Miniconda with:

    `conda install <packagename>`
  - packages which are not included in the comprehensive Anaconda package are installed with one of the following:

    `pip install <packagename>`

    `easy_install <packagename>`

Tree plots (with NetworkX, pydotplus, GraphViz

Drawing trees with NetworkX (requires pydotplus):

```
#!/home/gibbon/miniconda2/bin/python

import networkx as nx

G = nx.DiGraph()
G.add_node("ROOT")
for i in xrange(5):
    G.add_node("Child_%i" % i)
    G.add_node("Grandchild_%i" % i)
    G.add_edge("ROOT", "Child_%i" % i)
    G.add_edge("Child_%i" % i, "Grandchild_%i" % i)
nx.drawing.nx_pydot.write_dot(G, 'test.dot')
```

Watch this space ...

Tree plots (with NetworkX, pydotplus, GraphViz - difficult

Drawing trees with NetworkX (plus dot and eog):

```
#!/home/gibbon/miniconda2/bin/python

import networkx as nx

G = nx.DiGraph()
G.add_node("ROOT")
for i in xrange(5):
    G.add_node("Child_%i" % i)
    G.add_node("Grandchild_%i" % i)
    G.add_edge("ROOT", "Child_%i" % i)
    G.add_edge("Child_%i" % i, "Grandchild_%i" % i)
nx.drawing.nx_pydot.write_dot(G, 'test.dot')
```

The additional modules must be installed and imported!

In order to see the tree graph you must install GraphViz and a viewer such as eog.

sudo apt-get install graphviz eog

Tree plots (with NetworkX, pydotplus, GraphViz - difficult

Drawing trees with NetworkX (plus dot and eog):

```
#!/home/gibbon/miniconda2/bin/python

import networkx as nx

G = nx.DiGraph()
G.add_node("ROOT")
for i in xrange(5):
    G.add_node("Child_%i" % i)
    G.add_node("Grandchild_%i" % i)
    G.add_edge("ROOT", "Child_%i" % i)
    G.add_edge("Child_%i" % i, "Grandchild_%i" % i)
nx.drawing.nx_pydot.write_dot(G, 'test.dot')

import os
os.system('dot -Tpng test.dot -o test.png && eog
test.png')
```

Assignment:

Design and plot a parse tree for a simple sentence.

Wordnet traversal - difficult

```
import networkx as nx
import matplotlib.pyplot as plt
from nltk.corpus import wordnet as wn

def traverse(graph,start,node):
    graph.depth[node.name] =
node.shortest_path_distance(start)
    for child in node.hyponyms():
        graph.add_edge(node.name,child.name)
        traverse(graph,start,child)
def hyponym_graph(start):
    G = nx.Graph()
    G.depth = {}
    traverse(G,start,start)
    return G
def graph_draw(graph):
    nx.draw_graphviz(graph,
        node_size = [16*graph.degree(n) for n in graph],
        node_color == [graph.depth[n] for n in graph]
        with_labels == False)
    plt.show()
```

```
dog = wn.synset('dog.n.01')
graph = hyponym.graph(dog)
graph_draw(graph)
```

End of Unit 3