```
# -*- coding: utf-8 -*-
"""
@author: campioni

Cap 6

"""
import matplotlib.pyplot as plt
import sys
import networkx as nx
import numpy as np
sys.path.insert(0,' D/PYTHON/Programmi/code')
from networkx.algorithms import approximation as app
from  thinkstats2 import Pmf
import thinkplot
def degrees(G):
    return[G.degree(u) for u in G]
def read_graph(filename):
    G=nx.Graph()
    array=np.loadtxt(filename, dtype=int)
    G.add_edges_from(array)
    return G
ba=nx.barabasi_albert_graph(4039, 22) #il secondo numero rappr. il numero di
archi
pmf_ba=Pmf(degrees(ba))
print('-------------------------')
print()
print('Valor medio di B-A = ',round(pmf_ba.Mean(),4))
print()
print('La sua deviazione standard= ',round(pmf_ba.Std(),4))

print('-------------------------')
print()
print()
print('I grafici sono i seguenti')
print()

plt.style.use('ggplot')     #sottofondo grigio e griglia bianca
plt.tick_fig, ax = plt.subplots(facecolor='teal', alpha=0.1)
plt.tick_params(labelcolor='y')
plt.xlabel('Gradi', color='y')
plt.ylabel('PMF', color='y')
thinkplot.Pdf(pmf_ba,color='r', label='Barabasi-Albert - legge di potenza')
thinkplot.Show()
print()

fig, ax = plt.subplots(facecolor='teal', alpha=0.1)
plt.style.use('ggplot')     #sottofondo grigio e griglia bianca
plt.tick_params(labelcolor='y')
print('BA in scala logaritmica')
print()
ax.set_xlabel('log gradi', color='y')
ax.set_ylabel('log PMF', color='y')

thinkplot.Pdf(pmf_ba, label='Barabasi-Albert  - legge di potenza',color='g')
thinkplot.Show(xscale='log',yscale='log')
print()
```