Import Firt Module

In [6]:
```
#!pip install selenium
import time
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
import networkx as nx
```

Prepare Selenium

In [6]:
```
chrome_options = Options()
chrome_options.add_argument("--headless")
chrome_path = './chromedriver'
driver = webdriver.Chrome(executable_path=chrome_path, options=chrome_options)
```

Crawling And Create Huge Graph

In [7]:
```
Links=["https://www.deu.edu.tr/"]
G = nx.Graph()
count=0
G.add_node(Links[count])
try:
    while count< len(Links):
        try:
# ============================================================================
#             Acces Website
# ============================================================================
            driver.get(Links[count])
            elems = driver.find_elements_by_xpath("//a[@href]")

            #Write Website
            pageSource = driver.page_source
            fileToWrite = open("./All_Website_Link/{}.html".format(count), "w",encoding='utf-8')
            fileToWrite.write(pageSource)
            fileToWrite.close()

            #Find New Link that Inside The Links
            for elem in elems:
                try:
                    link=elem.get_attribute("href")
                    if link:# check null
                        if "deu.edu" in link  and not( #the html must include deu.edu
                           "mailto" in link or #eleminate social media
                           "whatsapp" in link or
                           "facebook" in link or
                           "addtoany" in link or
                           "twitter" in link or
                           link[-1]=='#' or #eleminate nj
                           link=="javascript:void(0);" or #eleminate js
                           link=="javascript:void(0)" or
                           '.jpg' in link or #eleminate multimedia
                           'css' in link or
                           'json' in link or
                           '.jpeg' in link or
                           '.png' in link or
                           '.rar' in link or
                           '.zip' in link or
                           link.count('/')>5 or #eleminate deep html
                           '/feed/' in link or #eleminate not essential page
                           'lang=en' in link or #eleminate en page
                           'google.' in link or #eleminate big site
                           '.pdf' in link or #eleminate file format
                           '.doc' in link or
                            '#offcanvas' in link or
                            '?OGR_NO=' in link or
                           '.odt' in link or
                           'wp-content' in link or
                            '?SEARCH=' in link or
                           '/en/' in link or #eleminate en page
                           'acikerisim.deu.edu.tr' in link or #high resource page
                           'katalog.deu.edu.tr/search?' in link or
                            ('debis.deu.edu.tr/ders-katalog/' in link and
                             '/2020-2021/tr' not in link) or
                           'online.deu.edu.tr' in link or #eleminate login page
                           'onlinetip.deu.edu.tr' in link):#chech everything
                            if link in Links:
                                 G.add_edge(Links[count],link)
                            else:
                                if 100000>(len(Links)):
                                    Links.append(link)
                                    G.add_node(link)
                                    G.add_edge(Links[count],link)
                                else:
                                    continue
                except Exception:
                    continue
except Exception as e:
    print("!!!{}. Index and {} Link else went wrong with {}".format(count,Links[count],e))
    continue
finally:
    print("Completed :%{}\nRight Now On: {}. Index\nQue: {}".format((count/len(Links))*100,count,len(Links)))
            count+=1
finally:
    driver.quit()
```

...

Read Links

In [8]:
```
#Write All Crawded Site
with open("All_Sub_Links.txt", 'w') as f:
    for link in Links:
        f.write("%s\n" % link)
f.close()
print(len(Links))
```

```
20000
```

Import Second Module

In [4]:
```
#!pip install TurkishStemmer
#!pip install wordcloud
#!pip install textblob
#!pip install nltk

from TurkishStemmer import TurkishStemmer
from wordcloud import WordCloud
import pandas as pd
import nltk
from textblob import TextBlob,Word
import matplotlib.pyplot as plt
import os
import numpy as np
from PIL import Image
import nltk
#nltk.download('punkt')
```

Read Datas

In [2]:
```
Links=[]
f = open("All_Sub_Links.txt", "r")
Links=f.read().splitlines()
f.close()
len(Links)
```

Out[2]:
```
20000
```

In [5]:
```
RawDataFrame=[]
Name=[]
for i in os.listdir("All_Website_Link/"):#
    file = open("All_Website_Link/{}".format(i), "r" ,encoding='utf-8')#
    Name.append("{}".format(Links[int(i[:-5])]))
    RawDataFrame.append(file.read())
    file.close()

#dataframe=pd.DataFrame(RawDataFrame,columns=["Body"])
#dataframe['WebPage']=Name
#df=dataframe['Body']
```

Pre-Process

In [16]:
```
df=df.apply(lambda x: " ".join(x.lower() for x in x.split()))
df=df.replace('<script([\S\s]*?)>([\S\s]*?)<\/script>',' ',regex=True)#script tag
df=df.replace('<style([\S\s]*?)>([\S\s]*?)<\/style>',' ',regex=True)#style tag
df=df.replace('<[a-zA-Z\/][^>]*>',' ',regex=True)# html tag


df=df.replace('[^\w\s:)(]',' ',regex=True)
df=df.replace('\d+',' ',regex=True)

f = open("StopWord.txt", "r",encoding='utf-8')
sw=f.read().splitlines()
f.close()
stemmer = TurkishStemmer()

df=df.apply(lambda x: " ".join(stemmer.stem(x) for x in x.split() if x not in sw))
df=df.apply(lambda x: TextBlob(x).words)
dataframe["Body"]=df
```

Calculate Frequency

In [17]:
```
Frequency={}
count=0
for i in dataframe["Body"]:
    print("{}. Index was Calculated".format(count))
    count+=1
    for j in set(i):
        if j in Frequency.keys():
            Frequency[j][0]+=i.count(j)
            Frequency[j][1]+=1
        else:
            Frequency[j]=[i.count(j),1]

Frequency=sorted(Frequency.items(), key=lambda x: x[1][0],reverse=True)
with open("Frequency.csv", 'w' ,encoding='utf-8') as f:
    f.write("{},{},{}".format("Word","Frequency","Frequency For Page"))
    for i in Frequency:
        f.write("{},{},{}\n".format(i[0],i[1][0],i[1][1]))
f.close()
```

...

Plt Show

In [ ]:
```

wordcloud = WordCloud(max_font_size = 50,
                      max_words = 100,
                      background_color = "white").generate(" ".join(x for x in np.transpose(Frequency[:100])[0]))
plt.figure()
plt.imshow(wordcloud, interpolation = "bilinear")
plt.axis("off")
plt.show()

wordcloud.to_file("HighFreq_Words.png");
deu_mask = np.array(Image.open("deu.png"))
masked_cloud = WordCloud(background_color = "white",
                      max_words = 1000,
                      mask = deu_mask,
                      contour_width = 3,
                      contour_color = "firebrick")

masked_cloud.generate(" ".join(x for x in np.transpose(Frequency[:100])[0]))

masked_cloud.to_file("deu_mask.png")

plt.figure(figsize = [10,10])
plt.imshow(masked_cloud, interpolation = "bilinear")
plt.axis("off")
plt.show()
```

Centrality Analysy

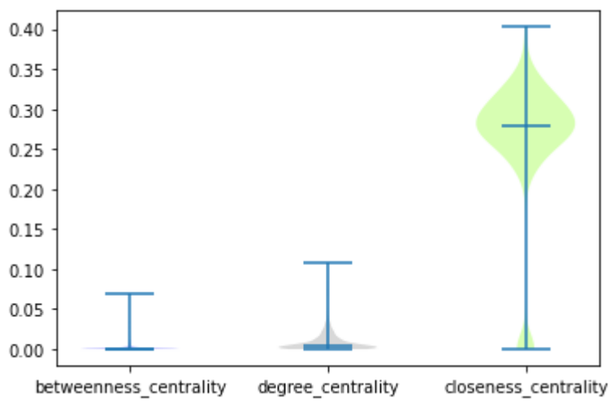
In [55]:
```
#!pip install networkx==2.5.1
#!pip install decorator==5.0.5
betweenness_centrality=nx.betweenness_centrality(G)
betweenness_centrality=sorted(betweenness_centrality.items(), key=lambda x: x[1],reverse=True)

closeness_centrality=nx.closeness_centrality(G)
closeness_centrality=sorted(closeness_centrality.items(), key=lambda x: x[1],reverse=True)

degree_centrality=nx.degree_centrality(G)
degree_centrality=sorted(degree_centrality.items(), key=lambda x: x[1],reverse=True)
#20000^2*log20000+200000*1000000 is high comp so it got small
```

Print Centrality Analysy

In [48]:
```
with open("betweenness_centrality.csv", 'w' ,encoding='utf-8') as f:
    for i in betweenness_centrality:
        f.write("{},{}\n".format(i[0],i[1]))
f.close()

with open("closeness_centrality.csv", 'w' ,encoding='utf-8') as f:
    for i in closeness_centrality:
        f.write("{},{}\n".format(i[0],i[1]))
f.close()

with open("degree_centrality.csv", 'w' ,encoding='utf-8') as f:
    for i in degree_centrality:
        f.write("{},{}\n".format(i[0],i[1]))
f.close()
with open("centrality.csv", 'w' ,encoding='utf-8') as f:
    f.write("{},{},{},{},{},{}\n".format("WebPage","degree_centrality","WebPage","betweenness_centrality","WebPage","closeness_centrality"))
    i=0
    while i<len(degree_centrality):
        f.write("{},{},{},{},{},{}\n".format(degree_centrality[i][0],degree_centrality[i][1],betweenness_centrality[i][0],betweenness_centrality[i][1],closeness_centrality[
        i+=1
f.close()
```

Plt Show

In [59]:
```
colors = ['blue', 'grey', 'lawngreen']
betweenness_centrality_p=np.transpose(betweenness_centrality)[1].astype(float)
degree_centrality_p=np.transpose(degree_centrality)[1].astype(float)
closeness_centrality_p=np.transpose(closeness_centrality)[1].astype(float)

vp = plt.violinplot(np.transpose([betweenness_centrality_p,degree_centrality_p,closeness_centrality_p]),
                    showmedians=True)

plt.xticks([1, 2, 3], ['betweenness_centrality', 'degree_centrality', 'closeness_centrality'])

for i in range(len(vp['bodies'])):
    vp['bodies'][i].set(facecolor=colors[i])

plt.show()
plt.savefig('centrality.png')
```




```
<Figure size 432x288 with 0 Axes>
```

Get Small and Good For Centrality Analysy

In [ ]:
```
limited=[i for i in Links[:3000] if not 'hastane' in i and not 'debis' in i and not 'haber' in i and not 'kutuphane' in i and not 'bid' in i and not 'kisi' in i and not 'en
```

In [46]:
```
from lxml import etree
from bs4 import BeautifulSoup
import networkx as nx
i=0
G = nx.Graph()
while(i<len(limited)):
    if not limited[i] in G.nodes():
        G.add_node(limited[i])
    i+=1
i=0
while(i<len(limited)):
    html = RawDataFrame[Links.index(limited[i])]

    soup = BeautifulSoup(html, 'html.parser',from_encoding="iso-8859-1")
    for link in soup.find_all(['a', 'link'], href=True):
        if link['href'] in limited:
            G.add_edge(link['href'],Links[Links.index(limited[i])])
    i+=1
    print([len(G.nodes),i])
```

...

Drawing Centratily Analysis

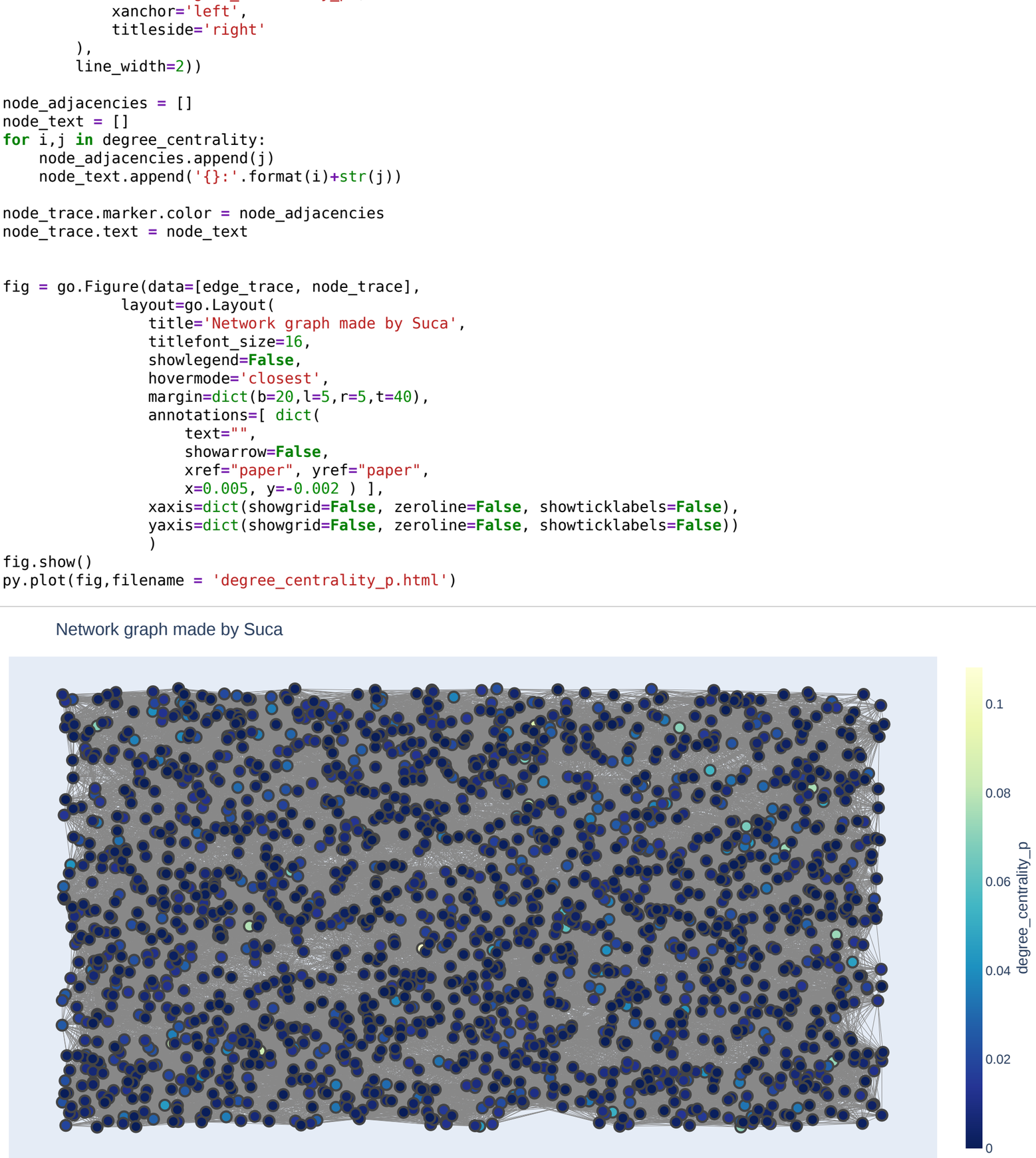
In [66]:
```
!pip install plotly
import plotly.graph_objects as go
import plotly.offline as py
import networkx as nx
```

```
Collecting plotly
  Downloading plotly-5.1.0-py2.py3-none-any.whl (20.6 MB)
     |████████████████████████████████| 20.6 MB 12.7 MB/s eta 0:00:01
Requirement already satisfied: six in /home/ggezbabe123/anaconda3/lib/python3.8/site-packages (from plotly) (1.15.0)
Collecting tenacity>=6.2.0
  Downloading tenacity-7.0.0-py2.py3-none-any.whl (23 kB)
Installing collected packages: tenacity, plotly
Successfully installed plotly-5.1.0 tenacity-7.0.0
```

In [ ]:

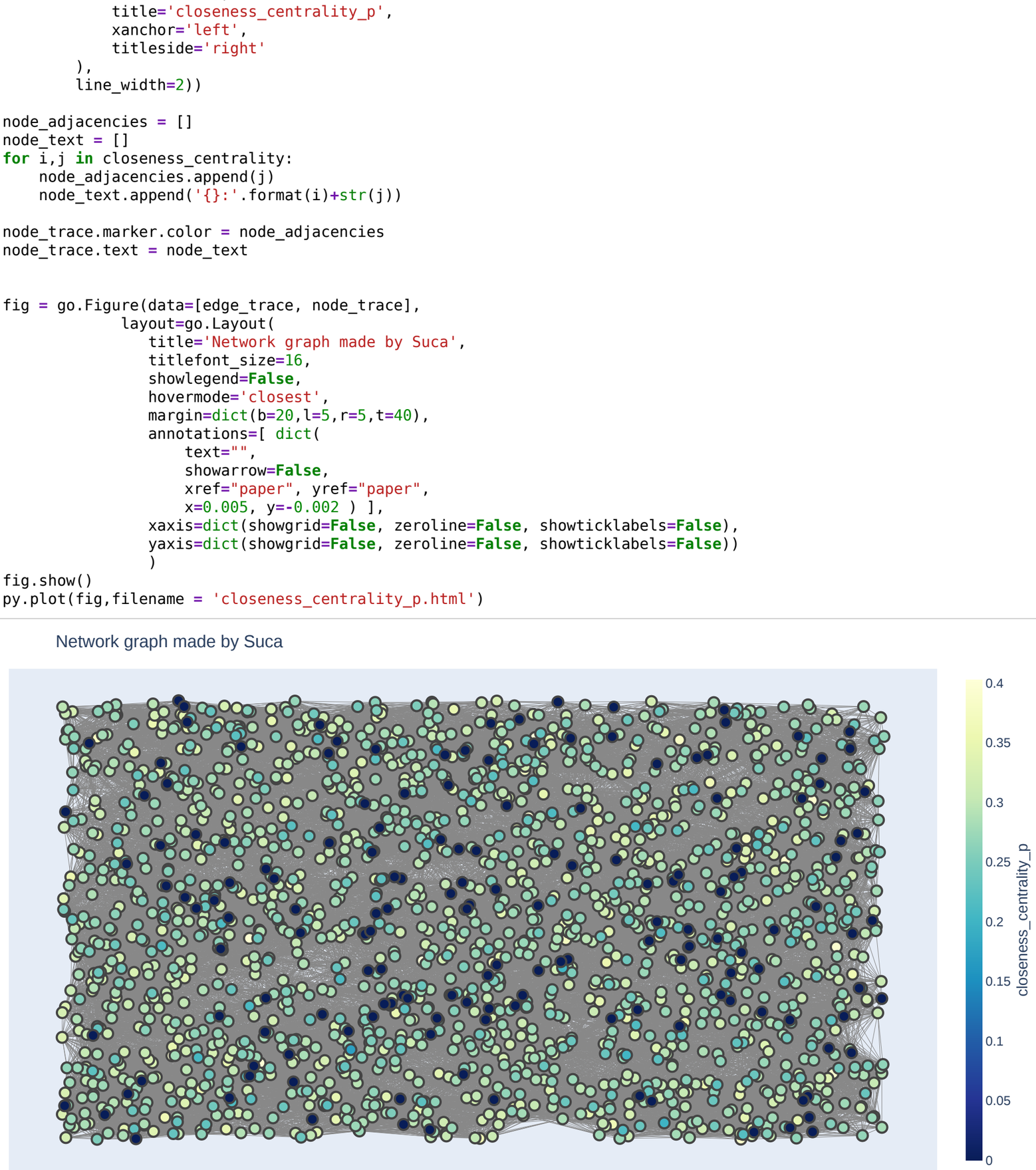
In [ ]:

In [70]:
```
edge_x = []
edge_y = []
for edge in G.edges():
    x0, y0 = G.nodes[edge[0]]['pos']
    x1, y1 = G.nodes[edge[1]]['pos']
    edge_x.append(x0)
    edge_x.append(x1)
    edge_x.append(None)
    edge_y.append(y0)
    edge_y.append(y1)
    edge_y.append(None)

edge_trace = go.Scatter(
    x=edge_x, y=edge_y,
    line=dict(width=0.5, color='#888'),
    hoverinfo='none',
    mode='lines')

node_x = []
node_y = []
for node in G.nodes():
    x, y = G.nodes[node]['pos']
    node_x.append(x)
    node_y.append(y)

node_trace = go.Scatter(
    x=node_x, y=node_y,
    mode='markers',
    hoverinfo='text',
    marker=dict(
        showscale=True,
        colorscale='YlGnBu',
        reversescale=True,
        color=[],
        size=10,
        colorbar=dict(
            thickness=15,
            title='degree_centrality_p',
            xanchor='left',
            titleside='right'
        ),
        line_width=2))

node_adjacencies = []
node_text = []
for i,j in degree_centrality:
    node_adjacencies.append(j)
    node_text.append('{}:'.format(i)+str(j))

node_trace.marker.color = node_adjacencies
node_trace.text = node_text


fig = go.Figure(data=[edge_trace, node_trace],
             layout=go.Layout(
                title='Network graph made by Suca',
                titlefont_size=16,
                showlegend=False,
                hovermode='closest',
                margin=dict(b=20,l=5,r=5,t=40),
                annotations=[ dict(
                    text="",
                    showarrow=False,
                    xref="paper", yref="paper",
                    x=0.005, y=-0.002 ) ],
                xaxis=dict(showgrid=False, zeroline=False, showticklabels=False),
                yaxis=dict(showgrid=False, zeroline=False, showticklabels=False))
                )
fig.show()
py.plot(fig,filename = 'degree_centrality_p.html')
```



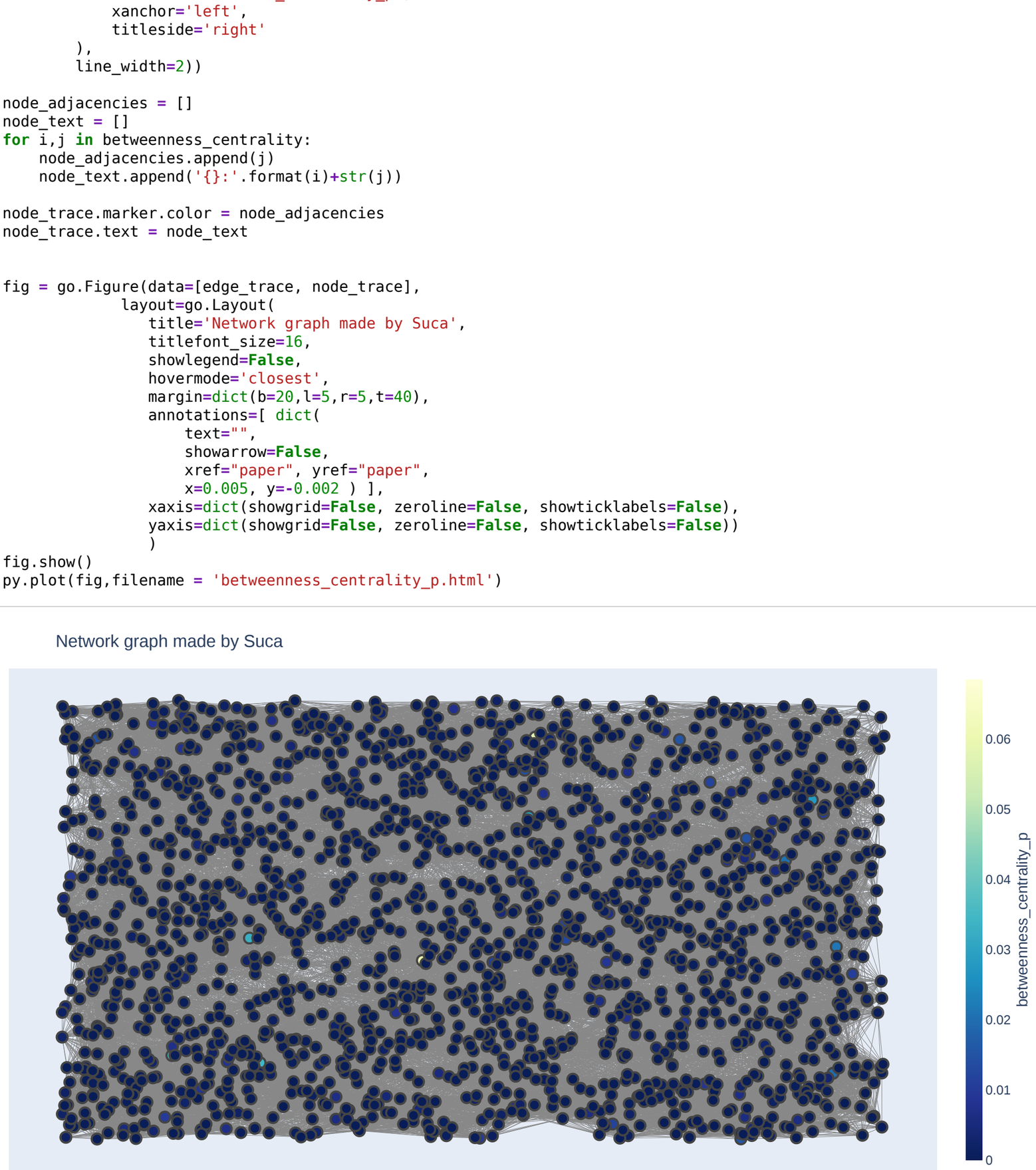
Out[70]: 'degree_centrality_p.html'

In [69]:
```
edge_x = []
edge_y = []
for edge in G.edges():
    x0, y0 = G.nodes[edge[0]]['pos']
    x1, y1 = G.nodes[edge[1]]['pos']
    edge_x.append(x0)
    edge_x.append(x1)
    edge_x.append(None)
    edge_y.append(y0)
    edge_y.append(y1)
    edge_y.append(None)

edge_trace = go.Scatter(
    x=edge_x, y=edge_y,
    line=dict(width=0.5, color='#888'),
    hoverinfo='none',
    mode='lines')

node_x = []
node_y = []
for node in G.nodes():
    x, y = G.nodes[node]['pos']
    node_x.append(x)
    node_y.append(y)

node_trace = go.Scatter(
    x=node_x, y=node_y,
    mode='markers',
    hoverinfo='text',
    marker=dict(
        showscale=True,
        colorscale='YlGnBu',
        reversescale=True,
        color=[],
        size=10,
        colorbar=dict(
            thickness=15,
            title='closeness_centrality_p',
            xanchor='left',
            titleside='right'
        ),
        line_width=2))

node_adjacencies = []
node_text = []
for i,j in closeness_centrality:
    node_adjacencies.append(j)
    node_text.append('{}:'.format(i)+str(j))

node_trace.marker.color = node_adjacencies
node_trace.text = node_text


fig = go.Figure(data=[edge_trace, node_trace],
             layout=go.Layout(
                title='Network graph made by Suca',
                titlefont_size=16,
                showlegend=False,
                hovermode='closest',
                margin=dict(b=20,l=5,r=5,t=40),
                annotations=[ dict(
                    text="",
                    showarrow=False,
                    xref="paper", yref="paper",
                    x=0.005, y=-0.002 ) ],
                xaxis=dict(showgrid=False, zeroline=False, showticklabels=False),
                yaxis=dict(showgrid=False, zeroline=False, showticklabels=False))
                )
fig.show()
py.plot(fig,filename = 'closeness_centrality_p.html')
```



Out[69]: 'closeness_centrality_p.html'

In [68]:
```
edge_x = []
edge_y = []
for edge in G.edges():
    x0, y0 = G.nodes[edge[0]]['pos']
    x1, y1 = G.nodes[edge[1]]['pos']
    edge_x.append(x0)
    edge_x.append(x1)
    edge_x.append(None)
    edge_y.append(y0)
    edge_y.append(y1)
    edge_y.append(None)

edge_trace = go.Scatter(
    x=edge_x, y=edge_y,
    line=dict(width=0.5, color='#888'),
    hoverinfo='none',
    mode='lines')

node_x = []
node_y = []
for node in G.nodes():
    x, y = G.nodes[node]['pos']
    node_x.append(x)
    node_y.append(y)

node_trace = go.Scatter(
    x=node_x, y=node_y,
    mode='markers',
    hoverinfo='text',
    marker=dict(
        showscale=True,
        colorscale='YlGnBu',
        reversescale=True,
        color=[],
        size=10,
        colorbar=dict(
            thickness=15,
            title='betweenness_centrality_p',
            xanchor='left',
            titleside='right'
        ),
        line_width=2))

node_adjacencies = []
node_text = []
for i,j in betweenness_centrality:
    node_adjacencies.append(j)
    node_text.append('{}:'.format(i)+str(j))

node_trace.marker.color = node_adjacencies
node_trace.text = node_text


fig = go.Figure(data=[edge_trace, node_trace],
             layout=go.Layout(
                title='Network graph made by Suca',
                titlefont_size=16,
                showlegend=False,
                hovermode='closest',
                margin=dict(b=20,l=5,r=5,t=40),
                annotations=[ dict(
                    text="",
                    showarrow=False,
                    xref="paper", yref="paper",
                    x=0.005, y=-0.002 ) ],
                xaxis=dict(showgrid=False, zeroline=False, showticklabels=False),
                yaxis=dict(showgrid=False, zeroline=False, showticklabels=False))
                )
fig.show()
py.plot(fig,filename = 'betweenness_centrality_p.html')
```



Out[68]: 'betweenness_centrality_p.html'

In [ ]: