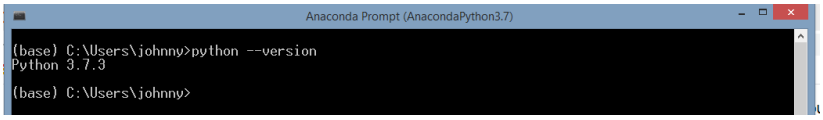
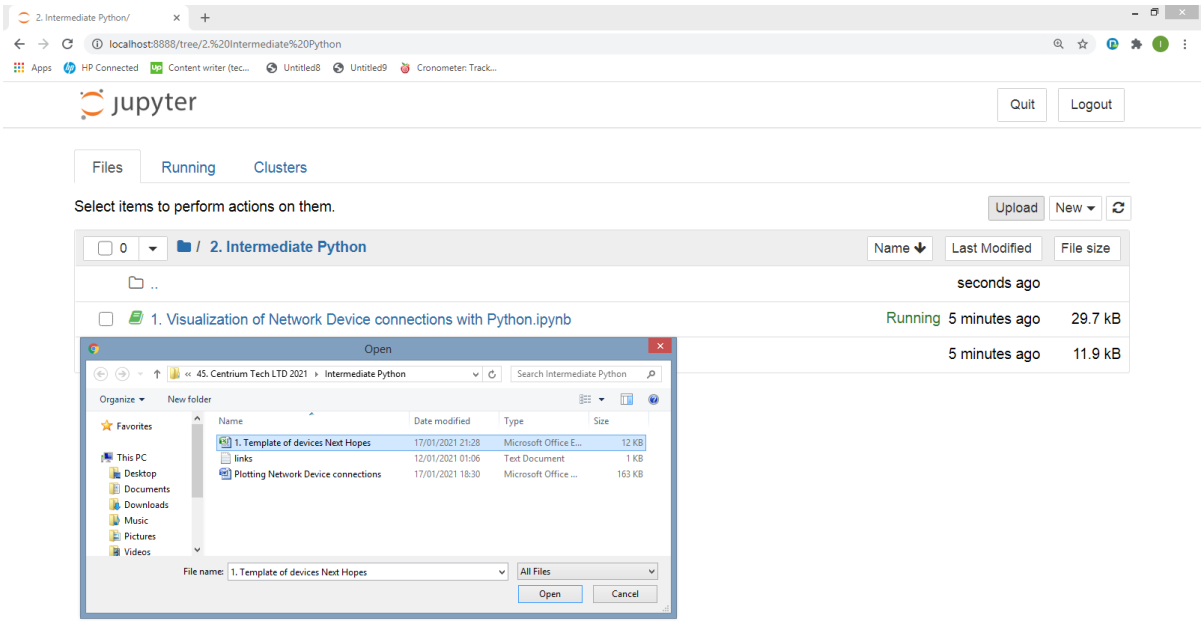
Lab Environment Setup:

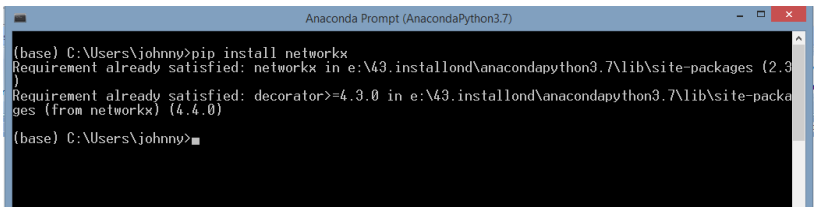
1. We need Jupiter notebook installed on our PC. We used Python 3.7.3 version.



2. We need to be able to create Excel document and upload it in the folder where our .ipynb file is. In our case:



3. We need to install on Anaconda Prompt (part of Jupyther Notebook tools) NetworkX using following command: *pip install networkx*



For this Lab following links with web materials were used as starting points :

https://en.wikipedia.org/wiki/Graph_theory

https://www.python-course.eu/graphs_python.php

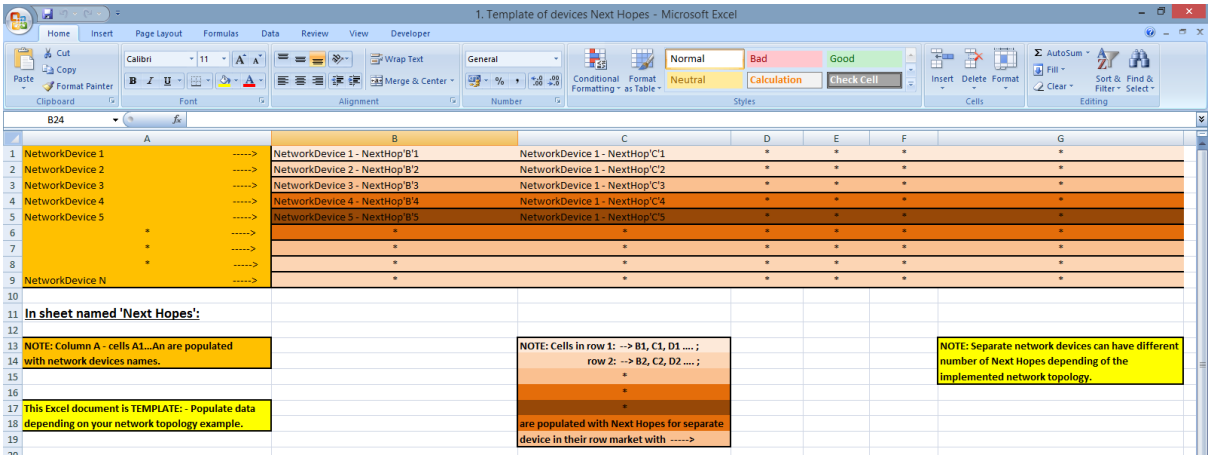
https://networkx.org/documentation/stable/index.html

I. Creation of Excel template

We need to create Excel template file where we will enter our data related to the network we want to plot.

In our case Excel template file is named: *1. Template of devices Next Hopes.xlsx*.

Explanation how to use this template is given in *README* Sheet.



In column A we enter network devices.
In any sepatate row we enter Next Hopes (devices that connect with this present device described with its name in 1st cell of that sspecific row.
Any device can have different number of Next hopes, or network devices that connects to.

I.1 Example of entrance data in this template

In Excel Sheet named *Next Hopes* we have entered 8 network devices (R1 to R8) in column A.
Any of them has different Next Hopes. For example:
- Next Hopes for R1 router are given in 1st row and is: R2.
- Next Hopes for R2 router are given in 2nd row and are: R1, R5 and R6.

...

II. Python Code

```
_____________________________

    1ST PART of the code:
_____________________________
```

NOTE: We are reading entered data in excel file, and get those data in a format suitable for data manipulation. So, we want to use graphs to plot and because of this we are formatting data in a dictionary known graphs format.

```
import xlrd  #xlrd is a Python library for reading data and formatting information from Excel file

import pandas as pd #pandas is a Python package for data analysis/manipulation tool


def Len_of_rows():

   df = pd.read_excel('1. Template of devices Next Hopes.xlsx', sheetname="Next Hopes")

   return len(df) #Example of function - returns number of, occupied with data, rows in excel files


loc = ('1. Template of devices Next Hopes.xlsx') #Location of the excel file

wb = xlrd.open_workbook(loc) #Reading the excel file
sheet = wb.sheet_by_index(0)  #Extracting the worksheet

sheet.cell_value(0, 0) #initializing cell from the excel file through the cell position

Len_of_cols=sheet.ncols #Gives number of occupied with data, columns in excel files .
                # /sheet.nrows/ - Gives number of occupied with data, rows in excel files

List_NH = [] #This list is for Next Hopes data in any unique (separate) row
List_Device = [] #This list is for separate Device data in unique row

for j in range(1):
   InnerList_NH = [] #This list is for Next Hopes data in all rows
   InnerList_Devices = [] #This list is for Devices data in starting cell of rows

   for i in range(0, Len_of_rows()+1):
      x=sheet.row_values(i, start_colx=1, end_colx=Len_of_cols)
      #We create separate list for any row of the excel file, except first cell in rows

     #print(x)

      List_NH = ' '.join(x).split() #We clear empty cell data in a row and add them to List_NH
      #print(List_NH)


      y=sheet.row_values(i, start_colx=0, end_colx=1)
      #We create separate list for any row of the excel file, just for first cell in rows

      #print(y)

      List_Device = ' '.join(y).split() #We clear empty cell data in a row and add them to List_NH
      #print(List_Device)
```

```
        InnerList_NH.append(List_NH)  #Accumulates all List_NH elements
        InnerList_Devices.append(List_Device)  #Accumulates all List_Device elements

    List_NH.append(InnerList_NH)  #Accumulates previous elements of 'i'-th element inside 'for' cycle
    List_Device.append(InnerList_Devices)  #Accumulates previous elements of 'i'-th element inside 'for' cycle
    #print(InnerList_NH)
    #print(InnerList_Devices)

    List_NH.pop()   #Removes last junk element in the List_NH (it was something like [...] element )
    List_Device.pop()   #Removes last junk element in the List_NH (it was something like [...] element )

#print(InnerList_NH)
print(InnerList_Devices)

InputDataDict1={} #We define empty dictionary that will have 'Device:Next Hopes' pairs

for k in range(0,len(InnerList_Devices)):
    InputDataDict1[InnerList_Devices[k][0]]=InnerList_NH[k]
    #We populate empty dictionary InputDataDict1 with elements of lists:
    #InnerList_Devices:InnerList_NH

print(InputDataDict1)
type(InputDataDict1)
```

If we excecute this part of the code we will got dictionary in a form that we wanted:

```
[['R1'], ['R2'], ['R3'], ['R4'], ['R5'], ['R6'], ['R7'], ['R8']]
{'R1': ['R2'], 'R2': ['R1', 'R5', 'R6'], 'R3': ['R5', 'R6', 'R7'], 'R4': ['R1', 'R7'], 'R5': ['R2',
'R3'], 'R6': ['R2', 'R3'], 'R7': ['R3', 'R4'], 'R8': ['R2', 'R1', 'R7']}

dict
```

This form of data in the dictionary with *key:[value1, value2, ...]* format is used to be able to work with Graphs.

Also, there are #print lines if we unmark them as code we can check how data translation is done from one to other acceptable data format.

```
_____________________________

    2ND PART of the code:
_____________________________
```

NOTE: In this part of the code we are accepting data in a dictionary format (through InputDataDict1 variable) and we create connections between the network devices.

NOTE: Here is used Graph Theory (More on: https://en.wikipedia.org/wiki/Graph_theory). Edge in our case actually is line used to connect two devices.

```
graph = InputDataDict1

def generate_edges(graph):
   edges = []
   for node in graph:
      for neighbour in graph[node]:
         edges.append((node, neighbour))

   return edges #Function that takes input data in dictionary format and makes List of Network Edges

NetworkEdges= generate_edges(graph)
print(NetworkEdges)
type(NetworkEdges)  #Data are stored in List data format and prepared for plotting.
```

If we excecute this part of the code we will got connections between network devices in a List data format:

```
[('R1', 'R2'), ('R2', 'R1'), ('R2', 'R5'), ('R2', 'R6'), ('R3', 'R5'), ('R3', 'R6'), ('R3', 'R7'),
('R4', 'R1'), ('R4', 'R7'), ('R5', 'R2'), ('R5', 'R3'), ('R6', 'R2'), ('R6', 'R3'), ('R7', 'R3'), ('R
7', 'R4'), ('R8', 'R2'), ('R8', 'R1'), ('R8', 'R7')]

list
```

3RD PART of the code:

NOTE: We are accepting all Network Edges data and visualise the network connectivity between devices.

```
import networkx as nx #NetworkX provides data structures and methods for storing graphs.
import matplotlib.pyplot as plt

router_matrix = nx.MultiDiGraph() #We use directed type of graph
#(networkx.org/documentation/stable/reference/classes/index.html#module-networkx.classes.graphviews)

router_matrix.add_edges_from(NetworkEdges)  # Add all Network Edges as List

print(router_matrix.edges())

pos = nx.spring_layout(router_matrix)

nx.draw_networkx_nodes(router_matrix, pos)
nx.draw_networkx_labels(router_matrix, pos)
nx.draw_networkx_edges(router_matrix, pos, edge_color='r', arrows = True)

plt.show()
```

If we excecute this part of the code we will got connections between network devices and also we will got Visualization of our network:

```
[('R1', 'R2'), ('R2', 'R1'), ('R2', 'R5'), ('R2', 'R6'), ('R5', 'R2'), ('R5', 'R3'), ('R6', 'R2'),
('R6', 'R3'), ('R3', 'R5'), ('R3', 'R6'), ('R3', 'R7'), ('R7', 'R3'), ('R7', 'R4'), ('R4', 'R1'), ('R
4', 'R7'), ('R8', 'R2'), ('R8', 'R1'), ('R8', 'R7')]
```