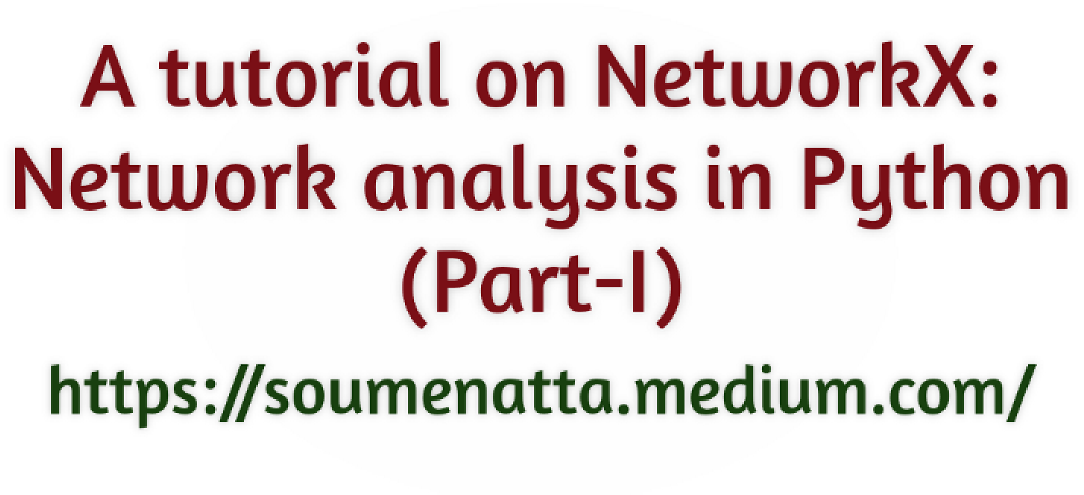
A Tutorial on NetworkX: Network Analysis in Python (Part-I)

medium.com/swlh/a-tutorial-on-networkx-network-analysis-in-python-part-i-43c1d35830b6

March 24, 2022



In this tutorial, we will learn about the NetworkX package of Python. NetworkX stands for network analysis in Python. It is mainly used for creating, manipulating, and study complex graphs. This is the Part-I of the tutorial on NetworkX. The remaining tutorial will be posted in different parts.

Prerequisites: Basic knowledge about graph theory and Python programming.

> “NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.” — https://networkx.org/

Installation

If the NetworkX package is not installed in your system, you have to install it at first. You can use the following command to install it.

The above command will install the NetworkX package in your system. Now, you are ready to use it. Sometimes, the above command may issue an error message. In that case, you are advised to use pip3 command instead of pip. It depends on how your system is configured.

Importing

You can use the 'networkx' module by importing it using the following command:

Now, the 'networkx' module is available with the alias 'nx'. You can use any alias names, though 'nx' is the most commonly used alias for 'networkx' module in Python.

Python Pandas DataFrame Tutorial for Beginners

This is a tutorial on Python Pandas DataFrame for absolute beginners. Pandas is a fast, powerful, flexible, and easy to…

medium.com

Create an empty graph

To create an empty graph, we use the following command:

The above command will create an empty graph. An empty graph is a graph whose vertex set and the edge set are both empty.

To access the vertex set and the edge set of the graph G, we can use the following command:

```
 # returns a list # returns a list
```

Both G.nodes() and G.edges return Python lists. Obviously, the above two commands will return two empty lists because we have not added any nodes or edges to graph G.

Adding nodes

Suppose, we want to add a vertex (also called a node) in G. In this tutorial, vertex and node will be used synonymously. We can add a node in G as follows:

The above command will add a single node A in the graph G. If we want to add multiple nodes at once, then we can use the following command:

The above command will add four vertices (or, nodes) in graph G. Now, graph G has five vertices A, B, C, D, and E. These are just isolated vertices because we have not added any edges to the graph G.

Adding edges

We can add an edge connecting two nodes A and B as follows:

The above command will create an edge (A, B) in graph G. Multiple edges can be added at once using the following command:

The above command will create four more edges in G. Now, G has a total of five edges.

Descriptive Statistics using Pandas: An Introductory Tutorial

In this tutorial, we will learn how to compute descriptive statistics using Python's Pandas library. We use a…

medium.com

Accessing vertex and edge sets

The vertex set and the edge set of G can be accessed using G.nodes() and G.edges(), respectively. These two commands will return Python lists.

The output of the above command is shown below:

```
Vertex set: [‘A’, ‘B’, ‘C’, ‘D’, ‘E’]
```

Similarly, we can access the edge set of G, as follows:

The output of the above print statement is mentioned below:

```
Edge set: [(‘A’, ‘B’), (‘A’, ‘C’), (‘B’, ‘D’), (‘B’, ‘E’), (‘C’, ‘E’)]
```

Exploratory Data Analysis using Python Pandas: A Tutorial

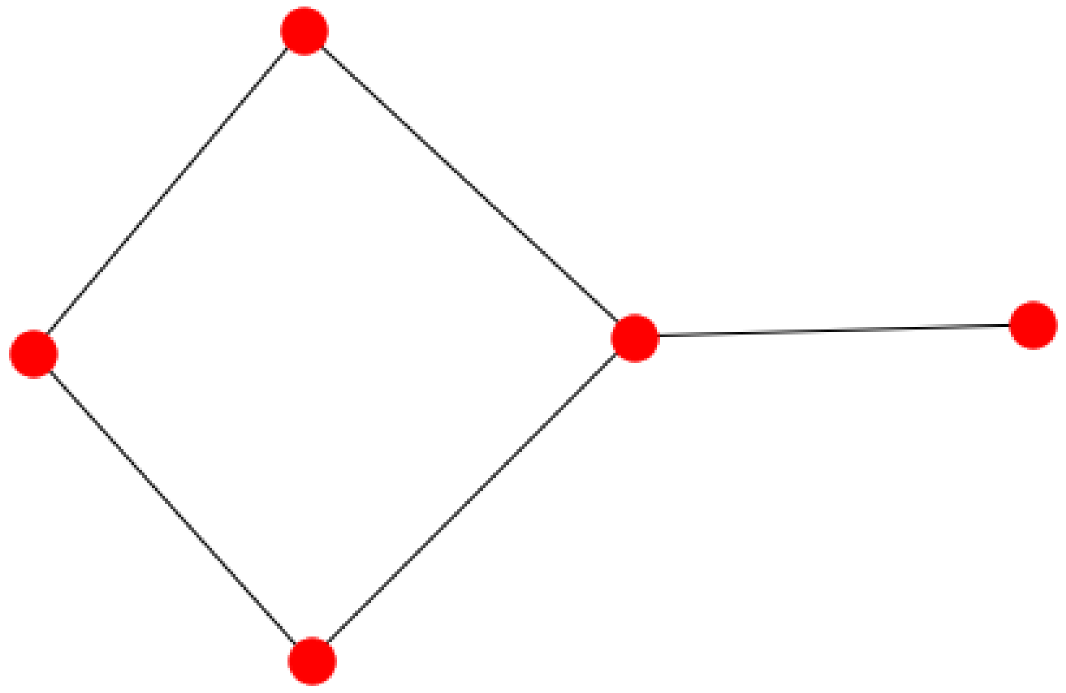
In this tutorial, we will learn about exploratory data analysis using Python Pandas. In exploratory data analysis, we…

soumenatta.medium.com

Drawing graph

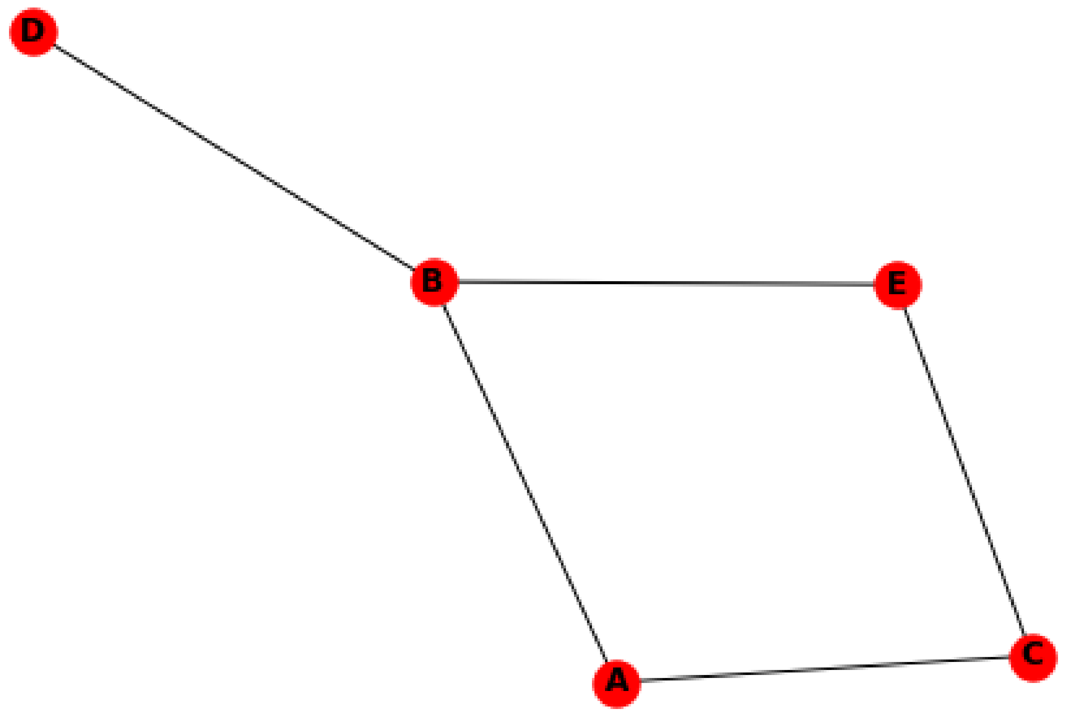
We can easily draw a graph using ‘networkx’ module. We use the ‘matplotlib’ library to draw it. So, we need to import it at first.

Now, the graph (G) created above can be drawn using the following command:



We can use the savefig() function to save the generated figure in any desired file format. In the following command, it is saved in PNG format. We can also save it as EPS, JPEG, etc.

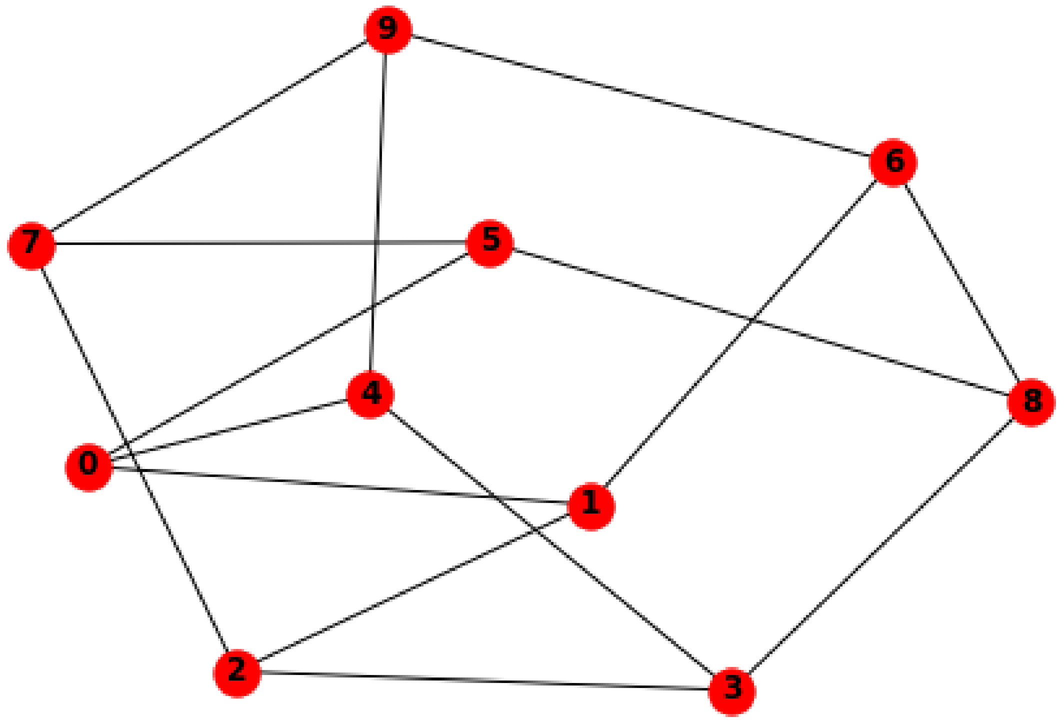
We can also use the following attributes in nx.draw() function, to draw G with vertex labels.



Note that we may get the different layouts of the same graph G, in different runs of the same code. Eventually, they represent the same graph G. In Figure 2, vertex labels are mentioned.

Using 'nextworkx' module, we can create some well-known graphs, for example, Peterson's graph. The command is mentioned below:

Here, GP is Peterson’s graph. Now, we draw graph GP as discussed above.



Publishing Online Books using Jupyter Book and GitHub Pages

In this beginner-friendly tutorial, we will learn how to publish online books using Jupyter Book and GitHub Pages. In…

soumenatta.medium.com

Adjacency view

We can get the adjacency view of a graph using ‘networkx’ module. This is the same as the adjacency list of a graph. In the following command, we print the adjacency view of G.

The above print statement will generate the adjacency view of graph G.

```
{‘A’: {‘B’: {}, ‘C’: {}}, ‘B’: {‘A’: {}, ‘D’: {}, ‘E’: {}}, ‘C’: {‘A’: {}, ‘E’: {}}, ‘D’: {‘B’:
{}}, ‘E’: {‘B’: {}, ‘C’: {}}}
```

Therefore, vertex A is adjacent to the vertices B, C, and so on (refer to Figure 2).

Degree of a vertex

The degree of a vertex is defined by the number of edges incident to it. The following command determines the degree of vertex A in the graph G.

The output of the above statement is 2. This can also be verified with the adjacency view of G.

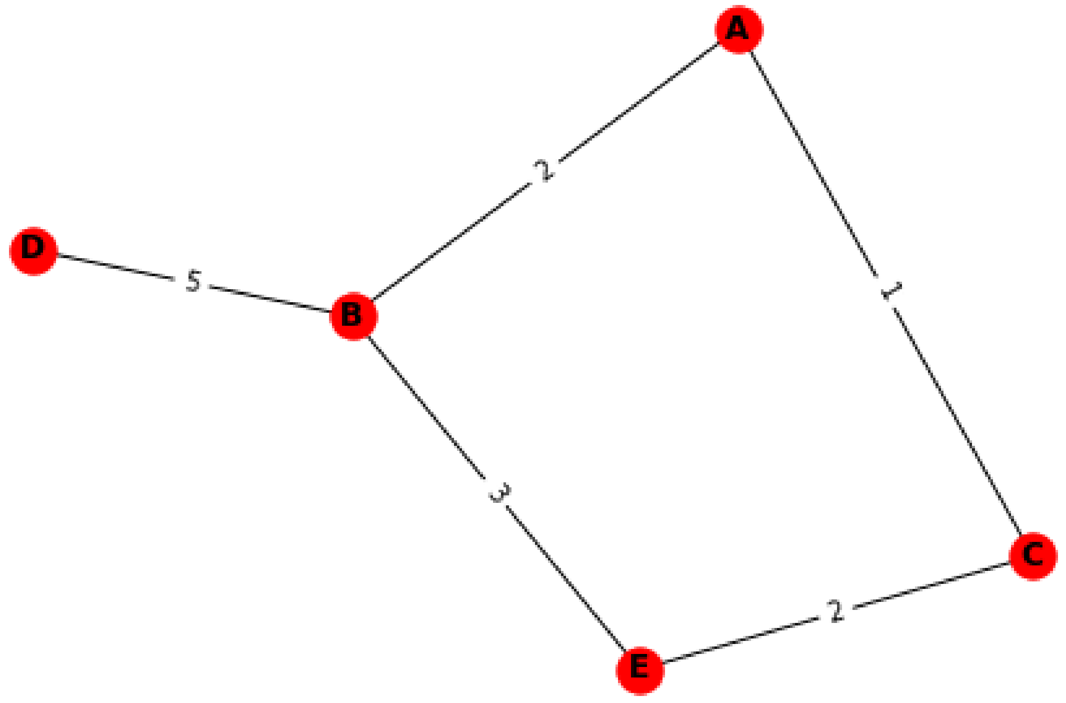
Creating weighted graph

Now, we will learn how to create a weighted graph using ‘networkx’ module in Python. Here, a weighted graph represents a graph with weighted edges. In general, we consider the edge weights as non-negative numbers.

In the following example, E is a Python list, which contains five elements. Each of these elements is a Python tuple having three elements. The first two elements denote the two endpoints of an edge and the third element represents the weight of that edge. Finally, we need to add these weighted edges to G.

We have already seen above how to draw an unweighted graph. Now, we will learn how to draw a weighted graph using ‘networkx’ module in Python. The complete code is mentioned below:

The above code segment will draw the graph as shown in Figure 4.



This is the end of Part-I of this tutorial. In the coming parts of this tutorial, more features of 'networkx' module in Python will be discussed.