# Importing network data from other formats, using nx\_pgnet Python module

[Importing network data from other formats, using nx\_pgnet Python module 1](#_Toc346009187)

[Supported Formats 2](#_Toc346009188)

[Spatial Networks 3](#_Toc346009189)

[Example imports – create a database connection 3](#_Toc346009190)

[GEXF 3](#_Toc346009191)

[Pajek 3](#_Toc346009192)

[YAML 4](#_Toc346009193)

[GraphML 4](#_Toc346009194)

[GML 4](#_Toc346009195)

[Gephi-compatible node/edge lists 4](#_Toc346009196)

[Aspatial Networks 5](#_Toc346009197)

[Example imports – create a database connection 5](#_Toc346009198)

[GEXF 5](#_Toc346009199)

[Pajek 5](#_Toc346009200)

[YAML 5](#_Toc346009201)

[GraphML 5](#_Toc346009202)

[GML 5](#_Toc346009203)

[Gephi-compatible node/edge lists 5](#_Toc346009204)

## Supported Formats

NetworkX supports the ability to import graphs / networks from a number of different formats. The definitive list of supported formats can be found at the NetworkX website: <http://networkx.lanl.gov/reference/readwrite.html>.

A subset of those formats are supported within the Python nx\_pgnet module, allowing a user to create a graph / network by reading from one of the supported formats, that can then be written to the database schema for storage. This operation is performed by the import\_graph class. This could be useful if a user wishes to share a network that they have created with another user. Of the formats listed at the above URL, the nx\_pgnet module supports importing from:

* Graph Exchange XML (GEXF) - <http://gexf.net/format/>
* YAML - <http://www.yaml.org/>
* GraphML - <http://graphml.graphdrawing.org/>
* GML - <http://www.infosun.fim.uni-passau.de/Graphlet/GML/gml-tr.html>
* Gephi-compatible node/edge lists
* Pajek - <http://vlado.fmf.uni-lj.si/pub/networks/pajek/>

**NOTE: There are a number of elements that must be adhered to in order for the nx\_pgnet module to be able to read the input network from one of the supported formats and then subsequently, at some point, be able to store that within the network schema. These elements will be described within each specific format import section below.**

In general, the main difference between importing a spatial network that can then be written to the database schema, compared with an aspatial network, is that the spatial network MUST define the coordinates of the node or edge as the identifier of that node or edge.

# Spatial Networks

## Example imports – create a database connection

Import nx\_pgnet as nx\_pgnet

Import osgeo.ogr as ogr

db\_connection = ogr.Open("PG: host=’<insert\_server\_host>’ dbname=’<insert\_database\_name>’ user=’<insert\_user\_name>’ password=’<insert\_password>’”)

conn = ogr.Open("PG: host='127.0.0.1' dbname='database' user='postgres', password='password'")

### GEXF

nx\_pgnet.import\_graph().import\_from\_gexf(‘<insert\_path\_to\_gexf\_file>’, ‘<insert\_graph\_name>’, <insert\_type\_to\_convert\_node\_str\_attr\_to>, <insert\_relabel\_boolean>)

nx\_pgnet.import\_graph().import\_from\_gexf(‘C://TEMP//transportnetwork.gexf’,’TransportNetwork’,str,False)

**NOTE: In order to be able to read the gexf file and create a network that can then be stored in the schema, the gexf file must have the node id and label values set to a tuple of the coordinates of the node e.g.**

<node id="(137623.20999999999, 28977.740000000002)" label="(137623.20999999999, 28977.740000000002)">

**NOTE: Furthermore, the edge source and target values must be set to the tuple of the coordinates of the nodes between which the edge exists e.g.**

<edge id="34" source="(381796.03000000003, 384130.82000000001)" target="(622026.53000000003, 313738.46000000002)">

If this information is not configured in this manner the network will not be able to be created correctly.

### Pajek

nx\_pgnet.import\_graph().import\_from\_pajek(‘<insert\_path\_to\_pajek\_file>’,’<insert\_graph\_name>’, ‘<insert\_encoding>’)

nx\_pgnet.import\_graph().import\_from\_pajek(‘C://TEMP//transportnetwork.net’,’TransportNetwork’,’utf-8’)

**NOTE: In order to be able to read the pajek file and create a network that can then be stored in the schema, the pajek file must have the node labels, and x/y parameters, set to be equal to a tuple of the coordinates of that node e.g.**

\*network NetworkX

\*vertices 48

1 "(137623.20999999999, 28977.740000000002)" 137623.21 28977.74 ellipse y 28977.74 x 137623.21 name "Land's End / St. Just Airport"

**NOTE: Currently no edge attributes are supported at this time.**

### YAML

nx\_pgnet.import\_graph().import\_from\_yaml(‘<insert\_path\_to\_yaml\_file>’,’<insert\_graph\_name>’)

nx\_pgnet.import\_graph().import\_from\_yaml(‘C://TEMP//transportnetwork.yaml’,’TransportNetwork’)

### GraphML

nx\_pgnet.import\_graph().import\_from\_graphml(‘<insert\_path\_to\_graphml\_file>’,’<insert\_graph\_name>’,<insert\_type\_to\_convert\_node\_str\_attr\_to>)

nx\_pgnet.import\_graph().import\_from\_graphml(‘C://TEMP//transportnetwork.graphml’,’TransportNetwork’,str)

**NOTE: node id values must be set to equal a tuple of coordinates denoting the geometry of the node e.g.**

<node id="(137623.20999999999, 28977.740000000002)">

**NOTE: edge id and label values must set to equal a tuple of coordinates denoting the geometry of the source and target node of an edge e.g.**

<edge source="(128198.08, 537323.33999999997)" target="(314388.65999999997, 673846.98999999999)">

### GML

nx\_pgnet.import\_graph().import\_from\_gml(‘<insert\_path\_to\_gml\_file>’,’<insert\_graph\_name>’,<insert\_relabel\_boolean>,’<insert\_encoding>’)

nx\_pgnet.import\_graph().import\_from\_gml(‘C://TEMP//transportnetwork.gml’,’TransportNetwork’, False, ‘utf-8’)

### Gephi-compatible node/edge lists

nx\_pgnet.import\_graph().import\_from\_gephi\_node\_edge\_lists(‘<insert\_path\_to\_gephi\_node\_file>’,’<insert\_path\_to\_gephi\_edge\_file>’,’<insert\_network\_name>’,’<insert\_attribute\_name\_for\_node\_geom\_wkt\_string>’,’<insert\_attribute\_name\_for\_edge\_geom\_wkt\_string>’,<insert\_directed\_boolean>)

nx\_pgnet.import\_graph().import\_from\_gephi\_node\_edge\_lists(‘C://TEMP//nodes.csv’,’C://TEMP//edges.csv’,’TransportNetwork’,’geometry\_text’,’geometry\_text’,’geom’,’geom’,False)

# Aspatial Networks

## Example imports – create a database connection

Import nx\_pgnet as nx\_pgnet

Import osgeo.ogr as ogr

db\_connection = ogr.Open("PG: host=’<insert\_server\_host>’ dbname=’<insert\_database\_name>’ user=’<insert\_user\_name>’ password=’<insert\_password>’”)

conn = ogr.Open("PG: host='127.0.0.1' dbname='database' user='postgres', password='password'")

### GEXF

nx\_pgnet.import\_graph().import\_from\_gexf(‘<insert\_path\_to\_gexf\_file>’, ‘<insert\_graph\_name>’, <insert\_type\_to\_convert\_node\_str\_attr\_to>, <insert\_relabel\_boolean>)

nx\_pgnet.import\_graph().import\_from\_gexf(‘C://TEMP//genericnetwork.gexf’,’GenericNetwork’,str,False)

### Pajek

nx\_pgnet.import\_graph().import\_from\_pajek(‘<insert\_path\_to\_pajek\_file>’, ‘<insert\_graph\_name>’, ‘<insert\_boolean\_denoting\_aspatial\_network>’, ‘<insert\_encoding\_value>’)

nx\_pgnet.import\_graph().import\_from\_pajek('C://Temp//genericnetwork.net', 'A\_Generic\_Network', spatial=True, encoding='UTF-8')

### YAML

nx\_pgnet.import\_graph().import\_from\_yaml(‘<insert\_path\_to\_yaml\_file>’, ‘<insert\_graph\_name>’)

nx\_pgnet.import\_graph().import\_from\_yaml('C://Temp//a\_network.yaml', 'A\_Network')

### GraphML

Nx\_pgnet.import\_graph().import\_from\_graphml(‘<insert\_path\_to\_graphml\_file>’, ‘<insert\_graph\_name>’, ‘<insert\_boolean\_denoting\_aspatial\_network>’, ‘<insert\_type\_to\_convert\_node\_str\_attr\_to>’)

nx\_pgnet.import\_graph().import\_from\_graphml('C://Temp//an\_aspatial\_network.graphml', 'An\_Aspatial\_Network', spatial=False, nodetype=str)

### GML

nx\_pgnet.import\_graph().import\_from\_gml(‘<insert\_path\_to\_gml\_file>’, ‘<insert\_graph\_name>’, ‘<insert\_encoding\_value>’)

nx\_pgnet.import\_graph().import\_from\_gml('C://Temp//a\_spatial\_network.gml', 'A\_Network', encoding='UTF-8')

### Gephi-compatible node/edge lists

Nx\_pgnet.import\_graph().import\_from\_gephi\_node\_edge\_lists(‘<insert\_path\_to\_node\_file>’, ‘<insert\_path\_to\_edge\_file>’, ‘<insert\_graph\_name>’, ‘<insert\_boolean\_denoting\_aspatial\_network>’, ‘<insert\_node\_column\_name\_geometry\_as\_text>’, ‘<insert\_edge\_column\_name\_geometry\_as\_text>’, ‘<insert\_node\_column\_name\_geometry\_raw>’, ‘<insert\_edge\_column\_name\_geometry\_raw>’)

nx\_pgnet.import\_graph().import\_from\_gephi\_node\_edge\_lists('C://Temp//nodes.csv', 'C://Temp//edges.csv', 'An\_Aspatial\_Network', spatial=False)

**NOTE: A value for node\_column\_name\_geometry\_as\_text, edge\_column\_name\_geometry\_as\_text, node\_column\_name\_geometry\_raw, edge\_column\_name\_geometry\_raw are not required for an aspatial network**