read_gml

`read_gml(path, label='label', destringizer=None)` [source]

Read graph in GML format from `path`.

Parameters:

path : ***filename or filehandle***

The filename or filehandle to read from.

label : ***string, optional***

If not None, the parsed nodes will be renamed according to node attributes indicated by `label`. Default value: ‘label’.

destringizer : ***callable, optional***

A `destringizer` that recovers values stored as strings in GML. If it cannot convert a string to a value, a `ValueError` is raised. Default value : None.

Returns:

G : ***NetworkX graph***

The parsed graph.

Raises:

NetworkXError

If the input cannot be parsed.

See also

`write_gml`, `parse_gml`
`literal_destringizer`

Notes

GML files are stored using a 7-bit ASCII encoding with any extended ASCII characters (iso8859-1) appearing as HTML character entities. Without specifying a `stringizer`/`destringizer`, the code is capable of writing `int`/`float`/`str`/`dict`/`list` data as required by the GML specification. For writing other data types, and for reading data other than `str` you need to explicitly supply a `stringizer`/`destringizer`.

For additional documentation on the GML file format, please see the GML url.

See the module docstring `networkx.readwrite.gml` for more details.

Examples

```
>>> G = nx.path_graph(4)
>>> nx.write_gml(G, "test.gml")
```

GML values are interpreted as strings by default:

```
>>> H = nx.read_gml("test.gml")
>>> H.nodes
NodeView(('0', '1', '2', '3'))
```

When a `destringizer` is provided, GML values are converted to the provided type. For example, integer nodes can be recovered as shown below:

```
>>> J = nx.read_gml("test.gml", destringizer=int)
>>> J.nodes
NodeView((0, 1, 2, 3))
```