Chapter 3: From Data to Networks

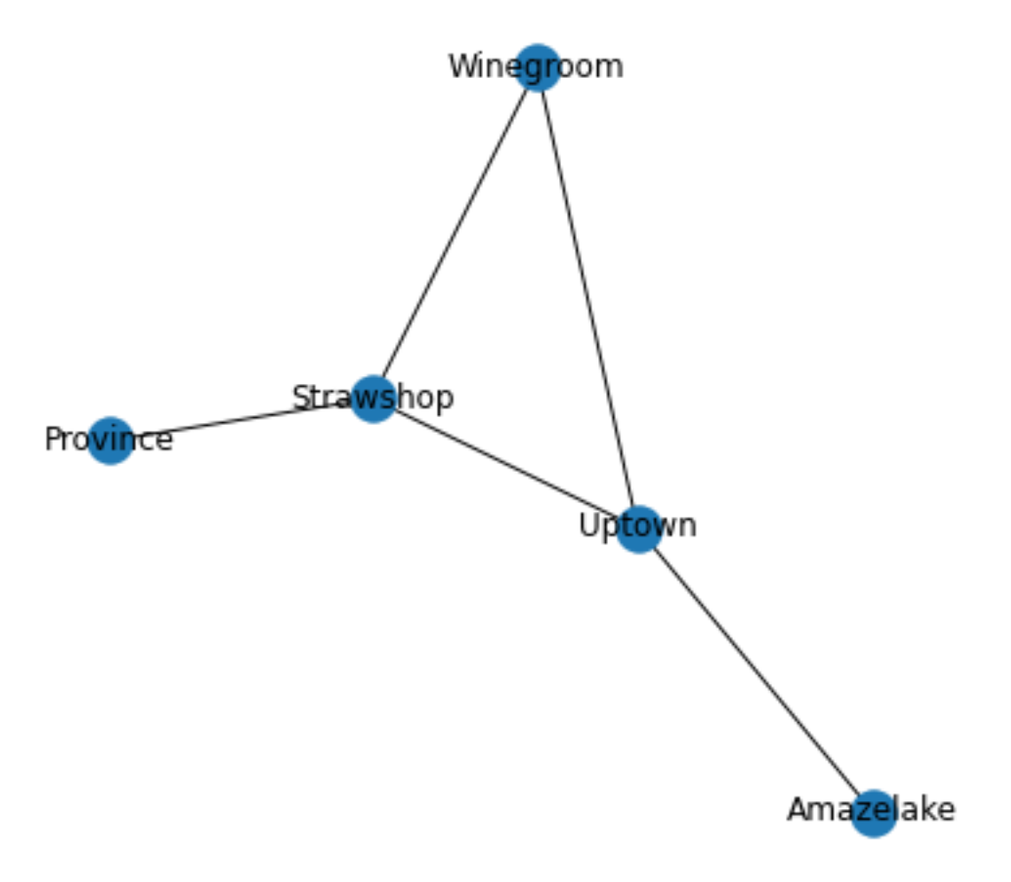
In [ ]:
```python
# Configure plotting in Jupyter
from matplotlib import pyplot as plt
%matplotlib inline
plt.rcParams.update({
    'figure.figsize': (7.5, 7.5),
    'axes.spines.right': False,
    'axes.spines.left': False,
    'axes.spines.top': False,
    'axes.spines.bottom': False})
# Seed random number generator
from numpy import random as nprand
import random
seed = hash("Network Science in Python") % 2**32
nprand.seed(seed)
random.seed(seed)
```

In [ ]:
```python
import networkx as nx
```

Reading and Writing Network Files

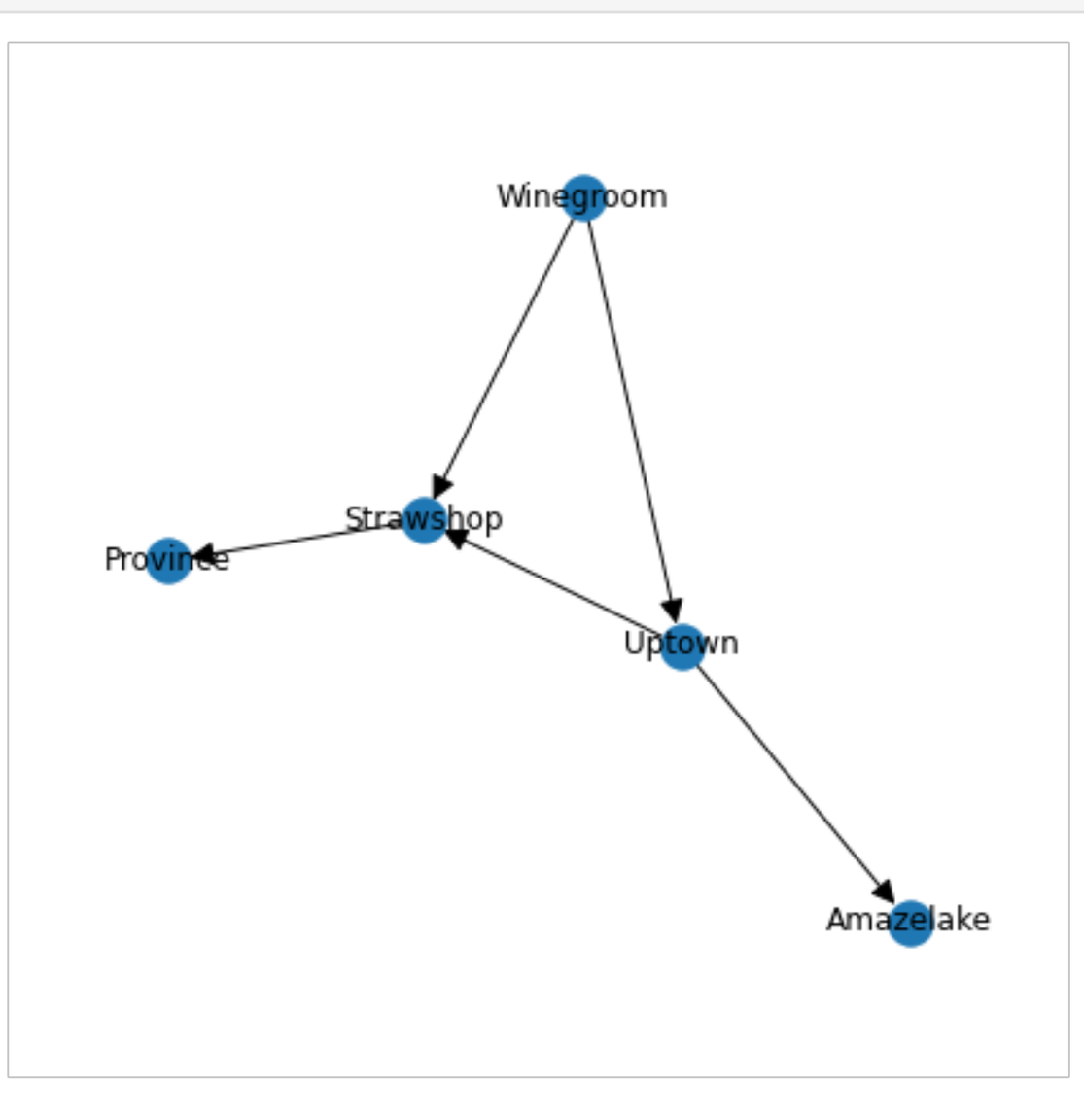
Edge list

In [ ]:
```python
# Create data directory path
from pathlib import Path
data_dir = Path('.') / 'data'
# Read edge list
G = nx.read_edgelist(str(data_dir / 'example.edgelist'))
# Draw network
pos = nx.spring_layout(G)
nx.draw_networkx(G, pos)
plt.gca().margins(0.15, 0.15)
```



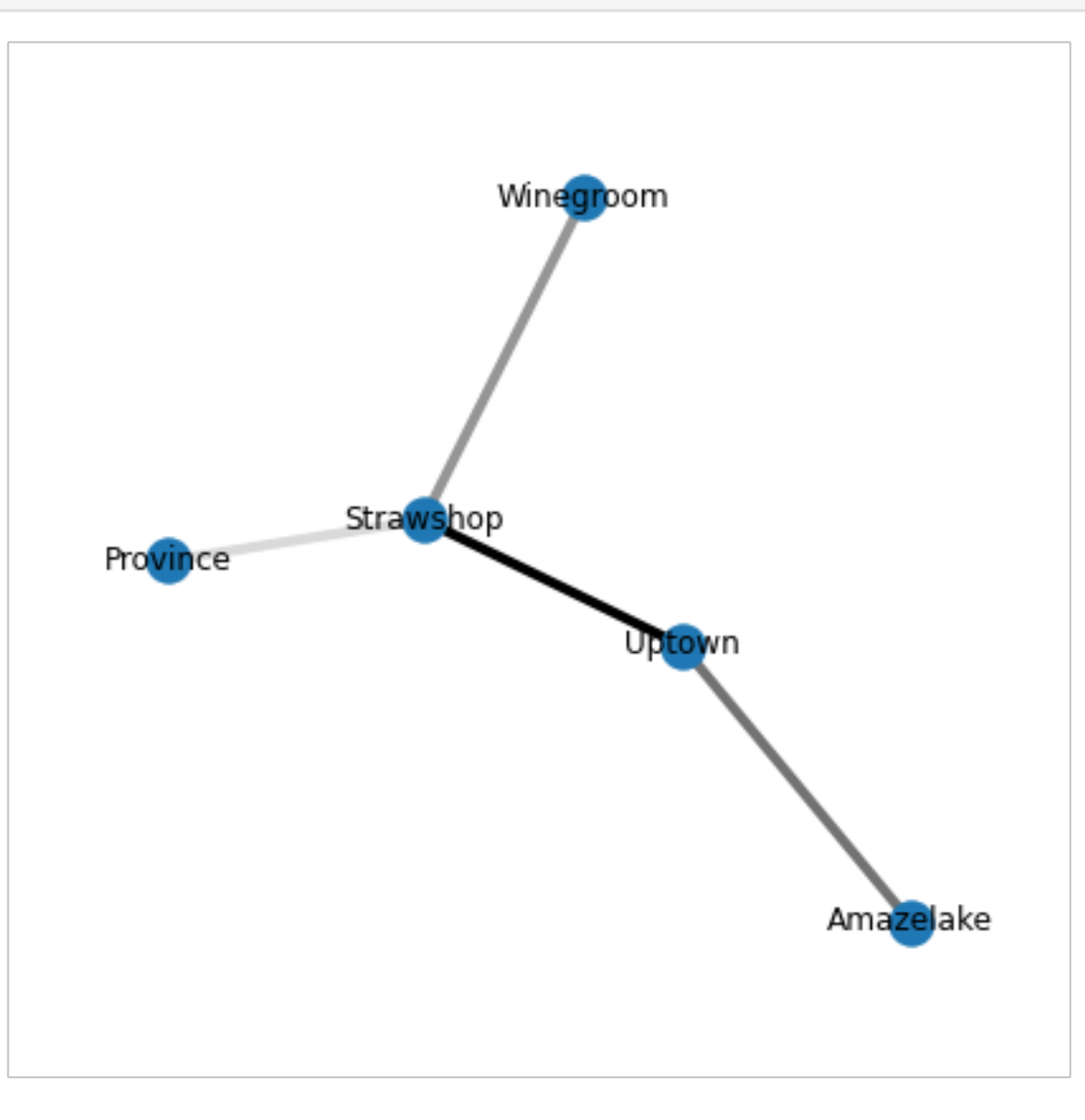
Directed edge list

In [ ]:
```python
# Read edge list
G = nx.read_edgelist(
    str(data_dir / 'example.edgelist'),
    create_using=nx.DiGraph)
# Draw network
nx.draw_networkx(G, pos, arrowsize=20)
plt.gca().margins(0.15, 0.15)
```



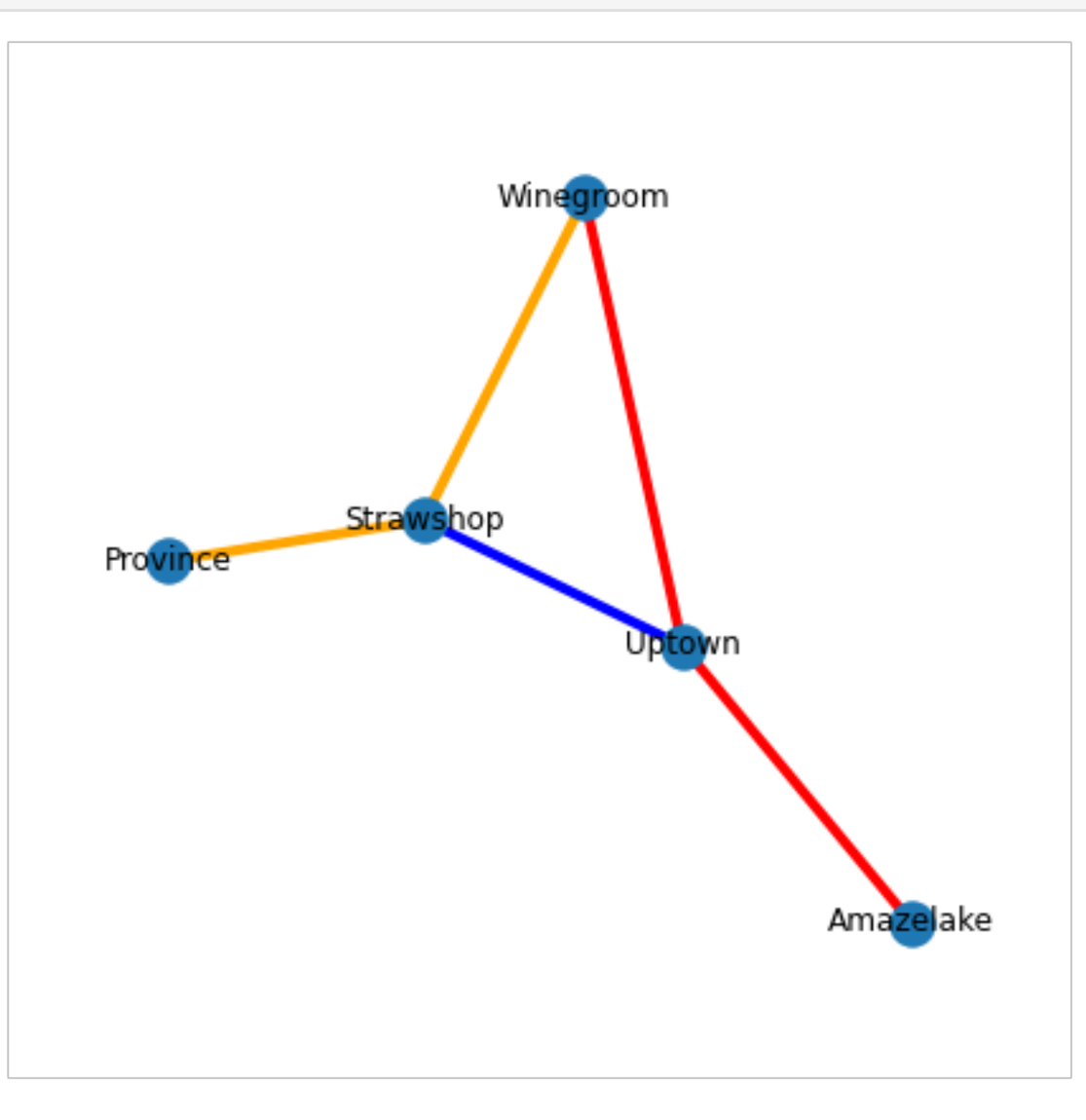
Weighted edge list

In [ ]:
```python
# Read edge list
G = nx.read_weighted_edgelist(
    str(data_dir / 'weighted.edgelist'))
# Extract weights
weights = [d['weight'] for s, t, d in G.edges(data=True)]
# Draw network
ax = plt.subplot(111)
nx.draw_networkx(
    G, pos, width=4, edge_color=weights, edge_cmap=plt.cm.Greys)
ax.margins(0.15, 0.15)
```



Edge list with edge attributes

In [ ]:
```python
# Read edge list
G = nx.read_edgelist(
    str(data_dir / 'attributes.edgelist'),
    data=[('weight', float), ('color', str)])
# Extract colors
colors = [d['color'] for s, t, d in G.edges(data=True)]
# Draw network
ax = plt.subplot(111)
nx.draw_networkx(
    G, pos, width=4, edge_color=colors)
ax.margins(0.15, 0.15)
```



Adjacency list

In [ ]:
```python
# Read adjacency list
G_adj = nx.read_adjlist(str(data_dir / 'example.adjlist'))
```

GEXF

In [ ]:
```python
for v in G.nodes:
    G.nodes[v]['abbreviation'] = v[0]
import sys
nx.write_gexf(G, sys.stdout)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/Users/NathanBick/Documents/Graduate School/MATH517 - Social Network Analysis/Network-Science-with-Python-and-NetworkX-Quick-Start-Guide-master/Chapter03/Chapter_03.ipynb Cell 15' in <cell line: 4>()
      <a href='vscode-notebook-cell:/Users/NathanBick/Documents/Graduate%20School/MATH517%20-%20Social%20Network%20Analysis/Network-Science-with-Python-and-NetworkX-Quick-Start-Guide-master/Chapter03/Chapter_03.ipynb#ch0000014?line=1'>2</a>     G.nodes[v]['abbreviation'] = v[0]
      <a href='vscode-notebook-cell:/Users/NathanBick/Documents/Graduate%20School/MATH517%20-%20Social%20Network%20Analysis/Network-Science-with-Python-and-NetworkX-Quick-Start-Guide-master/Chapter03/Chapter_03.ipynb#ch0000014?line=2'>3</a> import sys
----> <a href='vscode-notebook-cell:/Users/NathanBick/Documents/Graduate%20School/MATH517%20-%20Social%20Network%20Analysis/Network-Science-with-Python-and-NetworkX-Quick-Start-Guide-master/Chapter03/Chapter_03.ipynb#ch0000014?line=3'>4</a> nx.write_gexf(G, sys.stdout)

File <class 'networkx.utils.decorators.argmap'> compilation 27:5, in argmap_write_gexf_23(G, path, encoding, prettyprint, version)
      3 import gzip
      4 import inspect
----> 5 import itertools
      6 import re
      7 from collections import defaultdict

File ~/miniconda3/envs/network/lib/python3.10/site-packages/networkx/readwrite/gexf.py:87, in write_gexf(G, path, encoding, prettyprint, version)
     85 writer = GEXFWriter(encoding=encoding, prettyprint=prettyprint, version=version)
     86 writer.add_graph(G)
---> 87 writer.write(path)

File ~/miniconda3/envs/network/lib/python3.10/site-packages/networkx/readwrite/gexf.py:662, in GEXFWriter.write(self, fh)
    660     self.indent(self.xml)
    661 document = ElementTree(self.xml)
--> 662 document.write(fh, encoding=self.encoding, xml_declaration=True)

File ~/miniconda3/envs/network/lib/python3.10/xml/etree/ElementTree.py:732, in ElementTree.write(self, file_or_filename, encoding, xml_declaration, default_namespace, method, short_empty_elements)
    730         encoding = "us-ascii"
    731 enc_lower = encoding.lower()
--> 732 with _get_writer(file_or_filename, enc_lower) as write:
    733     if method == "xml" and (xml_declaration or
    734             (xml_declaration is None and
    735              enc_lower not in ("utf-8", "us-ascii", "unicode"))):
    736         declared_encoding = encoding

File ~/miniconda3/envs/network/lib/python3.10/contextlib.py:142, in _GeneratorContextManager.__exit__(self, typ, value, traceback)
    140 if typ is None:
    141     try:
--> 142         next(self.gen)
    143     except StopIteration:
    144         return False

File ~/miniconda3/envs/network/lib/python3.10/xml/etree/ElementTree.py:780, in _get_writer(file_or_filename, encoding)
    777     yield write
    778 else:
    779     # wrap a binary writer with TextIOWrapper
--> 780     with contextlib.ExitStack() as stack:
    781         if isinstance(file_or_filename, io.BufferedIOBase):
    782             file = file_or_filename

File ~/miniconda3/envs/network/lib/python3.10/contextlib.py:576, in ExitStack.__exit__(self, *exc_details)
    572 try:
    573     # bare "raise exc_details[1]" replaces our carefully
    574     # set-up context
    575     fixed_ctx = exc_details[1].__context__
--> 576     raise exc_details[1]
    577 except BaseException:
    578     exc_details[1].__context__ = fixed_ctx

File ~/miniconda3/envs/network/lib/python3.10/contextlib.py:561, in ExitStack.__exit__(self, *exc_details)
    559 assert is_sync
    560 try:
--> 561     if cb(*exc_details):
    562         suppressed_exc = True
    563         pending_raise = False

File ~/miniconda3/envs/network/lib/python3.10/contextlib.py:449, in _BaseExitStack._create_cb_wrapper.<locals>._exit_wrapper(exc_type, exc, tb)
    448 def _exit_wrapper(exc_type, exc, tb):
--> 449     callback(*args, **kwds)

File ~/miniconda3/envs/network/lib/python3.10/site-packages/ipykernel/iostream.py:513, in OutStream.write(self, string)
    503 """Write to current stream after encoding if necessary
    504 
    505 Returns
   (...)
    509 
    510 """
    512 if not isinstance(string, str):
--> 513     raise TypeError(
    514         f"write() argument must be str, not {type(string)}"
    515     )
    517 if self.echo is not None:
    518     try:

TypeError: write() argument must be str, not <class 'bytes'>
```

JSON

In [ ]:
```python
nx.node_link_data(G)
```

Out[ ]:
```
{'directed': False,
 'multigraph': False,
 'graph': {},
 'nodes': [{'abbreviation': 'W', 'id': 'Winegroom'},
  {'abbreviation': 'U', 'id': 'Uptown'},
  {'abbreviation': 'S', 'id': 'Strawshop'},
  {'abbreviation': 'A', 'id': 'Amazelake'},
  {'abbreviation': 'P', 'id': 'Province'}],
 'links': [{'weight': 1.0,
   'color': 'red',
   'source': 'Winegroom',
   'target': 'Uptown'},
  {'weight': 5.0,
   'color': 'orange',
   'source': 'Winegroom',
   'target': 'Strawshop'},
  {'weight': 9.0, 'color': 'blue', 'source': 'Uptown', 'target': 'Strawshop'},
  {'weight': 6.0, 'color': 'red', 'source': 'Uptown', 'target': 'Amazelake'},
  {'weight': 3.0,
   'color': 'orange',
   'source': 'Strawshop',
   'target': 'Province'}]}
```

Creating a Network with Code

In [ ]:
```python
# Ignore articles, pronouns, etc.
stop_words = set([
    'the', 'of', 'and', 'i', 'to', 'my', 'in', 'was', 'that', 'thy',
    'a', 'had', 'my', 'with', 'but', 'he', 'she', 'you', 'your',
    'me', 'not', 'as', 'will', 'from', 'on', 'be', 'it', 'which',
    'for', 'his', 'him', 'chapter', 'at', 'who', 'by', 'have',
    'would', 'is', 'been', 'when', 'they', 'there', 'we', 'are',
    'our', 'if', 'her', 'were', 'than', 'this', 'what', 'so',
    'yet', 'more', 'their', 'them', 'or', 'could', 'an', 'can',
    'said', 'may', 'do', 'these', 'shall', 'how', 'shall', 'asked',
    'before', 'those', 'whom', 'am', 'even', 'its', 'did', 'then',
    'abbey', 'tintern', 'wordsworth', 'letter', 'thee', 'thou', 'oh',
    'into', 'any', 'myself', 'nor', 'himself', 'one', 'all', 'no', 'yes'
    'now', 'upon', 'only', 'might', 'every', 'own', 'such', 'towards',
    'again', 'most', 'ever', 'where', 'after', 'up', 'soon', 'many',
    'also', 'like', 'over', 'us', 'thus', 'has', 'about']
    + [str(x) for x in range(24)])
```

In [ ]:
```python
# This example uses regular expressions from the re package
import re
# Construct a network from a text
def co_occurrence_network(text):
    # Create a new network
    G = nx.Graph()
    # Split the text into sentences and iterate through them
    sentences = text.split('.')
    for s in sentences:
        # Remove punctuation and convert to lowercase
        clean = re.sub('[^\w\n ]+', '', s).lower()
        clean = re.sub('_+', '', clean).strip()
        # Create list of words separated by whitespace
        words = re.split('\s+', clean)
        # Create an edge for each pair of words
        for v in words:
            # Update word count, add node if necessary
            try:
                G.nodes[v]['count'] += 1
            except KeyError:
                G.add_node(v)
                G.nodes[v]['count'] = 1
            # Update edge count for each pair of words in this sentence
            for w in words:
                # Skip stop words
                if v == w or v in stop_words or w in stop_words:
                    continue
                # Skip blank space
                if len(v) == 0 or len(w) == 0:
                    continue
                # Add one to the edge's count
                try:
                    G.edges[v, w]['count'] += 1
                except KeyError:
                    # Edge doesn't exist, create it
                    G.add_edge(v, w, count=1)
    return G
```

In [ ]:
```python
# Read the text
with open(data_dir / 'shelley1818' / 'frankenstein.txt') as f:
    text = f.read()
# Create a network from the text
G = co_occurrence_network(text)
```

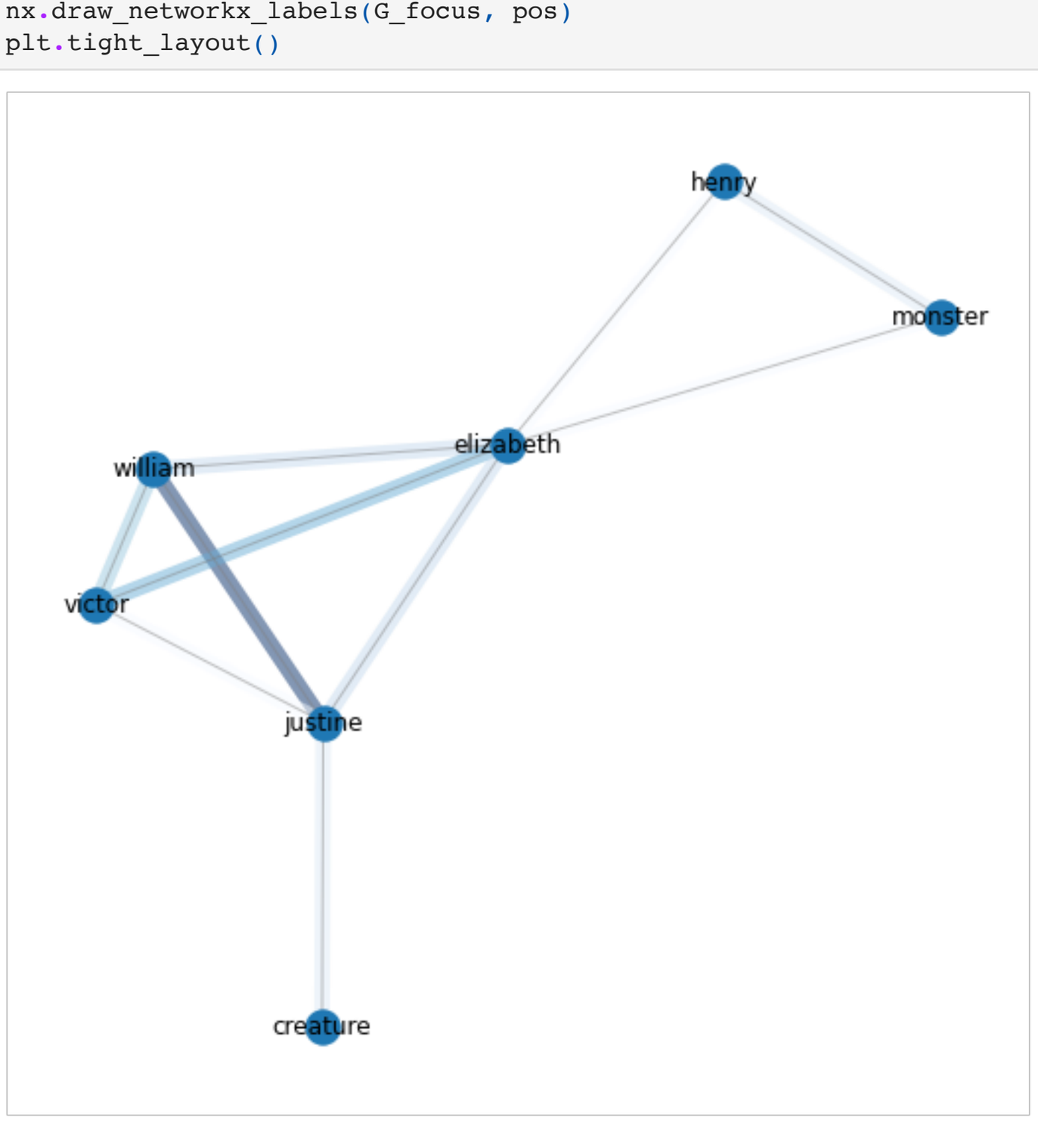
In [ ]:
```python
pairs = sorted(
    G.edges(data=True),
    key=lambda e: e[2]['count'],
    reverse=True)
pairs[0:10]
```

Out[ ]:
```
[('man', 'old', {'count': 68}),
 ('country', 'native', {'count': 38}),
 ('first', 'now', {'count': 32}),
 ('death', 'life', {'count': 32}),
 ('human', 'being', {'count': 32}),
 ('natural', 'philosophy', {'count': 32}),
 ('eyes', 'tears', {'count': 30}),
 ('first', 'eyes', {'count': 28}),
 ('some', 'time', {'count': 28}),
 ('night', 'during', {'count': 28})]
```

In [ ]:
```python
pos=nx.spring_layout(G)
nx.draw_networkx_nodes(G, pos, alpha=0)
nx.draw_networkx_edges(
    G, pos, edge_color="#333333", alpha=0.05)
# Zoom in for a better view
plt.xlim([-0.1,0.1]); plt.ylim([-0.1, 0.1])
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/Users/NathanBick/Documents/Graduate School/MATH517 - Social Network Analysis/Network-Science-with-Python-and-NetworkX-Quick-Start-Guide-master/Chapter03/Chapter_03.ipynb Cell 23' in <cell line: 1>()
----> <a href='vscode-notebook-cell:/Users/NathanBick/Documents/Graduate%20School/MATH517%20-%20Social%20Network%20Analysis/Network-Science-with-Python-and-NetworkX-Quick-Start-Guide-master/Chapter03/Chapter_03.ipynb#ch0000022?line=0'>1</a> pos=nx.spring_layout(G)
      <a href='vscode-notebook-cell:/Users/NathanBick/Documents/Graduate%20School/MATH517%20-%20Social%20Network%20Analysis/Network-Science-with-Python-and-NetworkX-Quick-Start-Guide-master/Chapter03/Chapter_03.ipynb#ch0000022?line=1'>2</a> nx.draw_networkx_nodes(G, pos, alpha=0)
      <a href='vscode-notebook-cell:/Users/NathanBick/Documents/Graduate%20School/MATH517%20-%20Social%20Network%20Analysis/Network-Science-with-Python-and-NetworkX-Quick-Start-Guide-master/Chapter03/Chapter_03.ipynb#ch0000022?line=2'>3</a> nx.draw_networkx_edges(
      <a href='vscode-notebook-cell:/Users/NathanBick/Documents/Graduate%20School/MATH517%20-%20Social%20Network%20Analysis/Network-Science-with-Python-and-NetworkX-Quick-Start-Guide-master/Chapter03/Chapter_03.ipynb#ch0000022?line=3'>4</a>     G, pos, edge_color="#333333", alpha=0.05)

File <class 'networkx.utils.decorators.argmap'> compilation 13:4, in argmap_spring_layout_10(G, k, pos, fixed, iterations, threshold, weight, scale, center, dim, seed)
      2 import collections
      3 import gzip
----> 4 import inspect
      5 import itertools
      6 import re

File ~/miniconda3/envs/network/lib/python3.10/site-packages/networkx/drawing/layout.py:476, in spring_layout(G, k, pos, fixed, iterations, threshold, weight, scale, center, dim, seed)
    474 if len(G) < 500:  # sparse solver for large graphs
    475     raise ValueError
--> 476 A = nx.to_scipy_sparse_array(G, weight=weight, dtype="f")
    477 if k is None and fixed is not None:
    478     # We must adjust k by domain size for layouts not near 1x1
    479     nnodes, _ = A.shape

File ~/miniconda3/envs/network/lib/python3.10/site-packages/networkx/convert_matrix.py:923, in to_scipy_sparse_array(G, nodelist, dtype, weight, format)
    921         r += diag_index
    922         c += diag_index
--> 923     A = sp.sparse.coo_array((d, (r, c)), shape=(nlen, nlen), dtype=dtype)
    924 try:
    925     return A.asformat(format)

AttributeError: module 'scipy.sparse' has no attribute 'coo_array'
```

In [ ]:
```python
# Count co-occurrences for characters only
characters = [
    'creature', 'monster', 'victor', 'elizabeth',
    'william', 'henry', 'justine']
G_focus = G.subgraph(characters)
# Create list of edge counts
counts = [G_focus.edges[e]['count'] for e in G_focus.edges]

# Create spring layout
pos = nx.spring_layout(G_focus)

# Create figure and draw nodes
plt.figure()
nx.draw_networkx_nodes(G_focus, pos)
# Draw edges
nx.draw_networkx_edges(
    G_focus, pos, width=8,
    edge_color=counts, edge_cmap=plt.cm.Blues, alpha=0.5)
nx.draw_networkx_edges(G_focus, pos, edge_color="#7f7f7f",alpha=0.5)
# Draw labels
nx.draw_networkx_labels(G_focus, pos)
plt.tight_layout()
```



In [ ]: