Introduction to Data Science and Programming

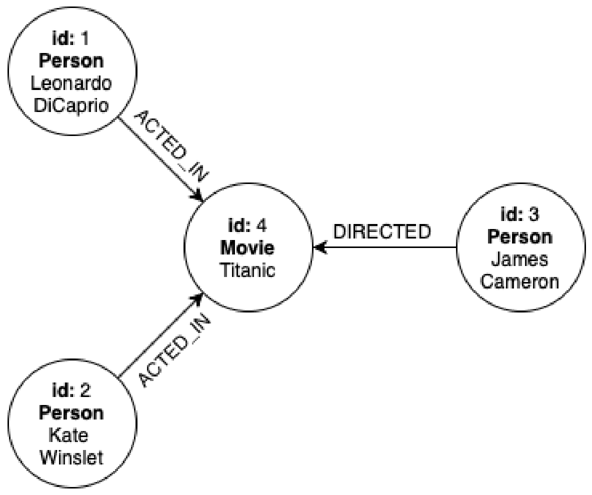
Assignment 3

Due October 3, 2019, 23:59

Description

This week we are going to work with dictionaries and process graph concepts such as nodes and edges. We've provided you with two files: `edge.csv` and `nodes.csv`. Each line in `nodes.csv` describes a node which either represents a person or a movie. The first value column is a unique identifier for the node, the second is its label (name of person or movie), and the third is its type (person or movie). Each line in `edges.csv` contains an edge between a person a movie. The first value is an identifier for the source node (a person node), the second is an identifier for the second node (a movie node), and the third is what type of relationship they have (`DIRECTED/ACTED_IN`). Each file and column is described in the table below. Please inspect the contents of both files to see how the data looks like.

filename	**column 1**	**column 2**	**column 3**
edges.csv	source node id	target node id	edge type
nodes.csv	node id	node label	node type



Submission

- Turn in a Python file (`movies.py`) containing your solutions.
- Remember to write the Python file as a library.
- Remember that your principal function calls must reside inside the `main` function.
- Make sure that we are able to run the Python file by running the following command: `python movies.py`.
- Also, make sure that we can import your data library without automatically running the `main` function.

Assignment

1. Write a function, called `read_csv(filename)`, to read in the two provided files: `edges.csv`, `nodes.csv`. The return value should be a list of tuples. Feel free to use loops or recursion, however note, since the function must accept the name of the file, a recursive solution may need to have a helper function to read the lines before splitting. *For this, consider reusing part 1 and 2 from last week's assignment and combine them.*
   Dummy example output:
   `[ ("1", "Leonardo DiCaprio", "Person"), ("2", "Kate Winslet", "Person"), ("4", "Titanic", "Movie") ]`

2. Write a function called `generate_name_map(nodes)` that takes a list of nodes and returns a dictionary that maps each 'node id' to the corresponding 'node label'. This allows us to lookup the label of uninformative unique identifiers. You may assume that no two nodes have the same id.
   Dummy example:
   `generate_name_map( [ ("1", "Leonardo DiCaprio", "Person"), ("2", "Kate Winslet", "Person"), ("4", "Titanic", "Movie") ] )`
   should return
   `{ "1":"Leonardo DiCaprio", "2":"Kate Winslet", "4":"Titanic" }`

3. Write a function called `name_edges(edges, name_map)` that takes the edges and the dictionary from `generate_name_map`. It should return the edges with names/labels, instead of unique identifiers. See the code snippets in the next section for an example.
   Dummy example:
   `name_edges( [ ("1", "4", "ACTS_IN") ],`
   `{ "1":"Leonardo DiCaprio", "2":"Kate Winslet", "4":"Titanic" } )`
   should return
   `[ ("Leonardo DiCaprio", "Titanic, "ACTS_IN") ]`

4. Write a function called `generate_movie_dictionary(named_edges)` that builds a dictionary that we can use to lookup which actors appeared in a movie. A requirement is that we want to filter out all non-actors, as the dictionary should only contain persons that appeared in the actual movie. Use the last value in the tuple to filter this out.
   Dummy output example *(This is a dummy example and does not work on your network. See the Examples section for an example given your network)*:
   `movie_dict = generate_movie_dictionary(named_edges)`
   `named_edges["Titanic"]`
   should return
   `{"Leonardo DiCaprio", "Kate Winslet"}`

5. Write a function called `get_actor_friends(movie_dictionary)` that takes the dictionary returned by exercise 4 and returns a dictionary where the key is an actor's name and the value is all the actors that they have worked with. For this exercise you'll have to iterate over the dictionary. *Hint:* python documentation.

Examples

Below is an example of how the calls to the different functions and their output may look like. It is also very close to what a `main` function could look like. The outputs below may or may not be correct, therefore, do not evaluate your code by replicating these exact results and instead convince yourself that your results are correct.

```
# 1
nodes = read_csv("nodes.csv")
edges = read_csv("edges.csv")
edges[0]
> ('1062', '66986', 'DIRECTED')
nodes[0]
> ('345', 'Avatar', 'Movie')

# 2
namemap = generated_name_map(nodes)
namemap["345"]
> Avatar

# 3
named_edges = name_edges(edges, name_map):
named_edges[0]
> ('Clint␣Eastwood', 'The␣Gauntlet', 'DIRECTED')

# 4
movie_dict = generate_movie_dictionary(named_edges)
movie_dict["The␣Matrix"]
> {'Anthony␣Ray␣Parker',
 'Belinda␣McClory',
  ...,
}

# 5
actor_friends = get_actor_friends(movie_dict)
actor_friends["Keanu␣Reeves"]
> {'Adrian␣Rayment',
 'Aitana␣Sanchez-Gijon',
 'Al␣Pacino',
 'Alan␣Arkin',
 'Alan␣Parnaby'
 ...
}
```