Exercise: Power Plant Atlas

Summary

This exercise focuses on building and using complex data objects using data on power plants in Oswego County, New York. Each plant has some plant-specific attributes and then a set of zero or more generators. Each generator, in turn, has a number of generator-specific characteristics. The result of exercise will be a JSON file describing the plants in the county.

Input Data

Two input files are provided: one that gives the characteristics of the plants as a whole and one that gives information about individual generators. Here are the details:

- **plants_oswego.txt**: Has one record (line) for each plant in the county. Each record has six fields separated by vertical bars (|): (1) the name of the utility that owns the plant, (2) a code number identifying the plant, (3) the plant's name, (4) its street address, (5) its city, and (6) the voltage, in kilovolts (kV), of the electrical transmission line connecting the plant to the grid.
- **generators_oswego.txt**: Has one record for each generator in the county. Each record has six fields separated by vertical bars: (1) the code number of the plant that has the generator, (2) the generator's name, (3) its technology, (4) its power generating capacity, in megawatts (MW), (5) the year it began operating, and (6) its status. In case you're curious, the main status codes are "OP" if the generator is operating or "SB" if it is on standby.

Deliverables

Your finished repository should have the following new files: **atlas.py**, **atlas.json**, and **ninemile.md**. They're described below.

Instructions

The conceptual approach will be to read the plant file, create a dictionary for each plant to hold its characteristics, and then store these dictionaries in an outer dictionary called `atlas`. Then, the generator file will be used to create a dictionary for each generator's characteristics, and the generators will be added to the corresponding plant entry in `atlas`.

Please create a script called `atlas.py` and have it do the following:

1. Import the `json` module.
2. Create an empty dictionary called `atlas`.
3. Open *plants_oswego.txt* and set up a for-loop to read it line by line. Within the loop do the following:
    1. Use `strip()` to remove leading and trailing blank space.
    2. Use `split("|")` to break the line into a list of attributes.
    3. Create a new dictionary called `new_plant` to hold the information. Use the following keys and set the value of each one to the appropriate part of the list created in the previous step: `utility` (the name of the utility), `name` (the name of the plant), `address`, `city`, and `kv`. Don't include the plant code in the dictionary: instead, create a variable called `code` and set it to the plant's code.
    4. Add one additional key, `generators`, to `new_plant` and set it to an empty list.
    5. Add the plant to the `atlas` dictionary. Use `code` as the key and `new_plant` as the value.
4. Close the file.
5. Open *generators_oswego.txt* and set up a for-loop to read it line by line. Within that loop do the following:

    1. Use `strip()` and `split("|")` to break the line into a list of generator attributes.
    2. Create a new dictionary called `new_gen` and set the following keys to the appropriate parts of the line: `name`, `tech`, `mw`, `year`, and `status`. When you set `mw` use the `float()` call to convert the string to a number. When setting `year` use `int()` to convert that string to a number. Don't include the plant code in `new_gen`; instead, use it to create a variable called `code`.
    3. Use `code` to look up the plant containing the generator in the `atlas` dictionary.
    4. Use the `append()` call to add `new_gen` to the plant's `generators` list.
6. Close the file.
7. Set up a for-loop to iterate over `atlas` using a running variable called `code` (since the loop will iterate over the keys of the dictionary, which are plant codes). Within the loop do the following:
    1. Set a variable called `plant` to the value of `atlas` for `code`.
    2. Set a variable called `num_gen` to the length of the plant's generator list.
    3. Set a variable called `total_mw` to 0.
    4. Use a for-loop to iterate over the generator list for `plant`. Use `gen` as the running variable and for each generator in the list add its `gen["mw"]` to `total_mw`.
    5. Go back out one level of indenting (so the line lines up with the `for` and `total_mw = 0` lines above and you're in the loop over `atlas` but not the inner loop over generators)
    6. Create three new keys for `plant`: one called `mw` set to `total_mw`, one called `num_gen` set to `num_gen`, and one called `mean_mw` set to `total_mw/num_gen`.
8. Open a file called `atlas.json` for writing.
9. Create a JSON string from the atlas using `json.dumps()` with an indenting level of 4. Call that variable `atlas_json`.
10. Use `write()` to write `atlas_json` to the output file.
11. Close the file.
12. Finally, to practice reading JSON notation, open `atlas.json` with Atom or another editor and have a look for the Nine Mile Point plant. Then create a short Markdown file called `ninemile.md` that starts with the line `# About Nine Mile Point`, has a blank line, and then has a couple sentences of text describing the plant in words. It doesn't have to be elaborate or polished: just a short summary of the plant for someone who doesn't have the underlying data.

Submitting

Once you're happy with everything and have committed all of the changes to your local repository, please push the changes to GitHub. At that point, you're done: you have submitted your answer.

Tips

- Atom understands JSON and will color-code it appropriately. It can also fold up sections of the JSON to help you see the big picture, which can be very useful when trying to make sense of a JSON file someone else has built. The folding is done by clicking on small arrows that will appear next to the beginnings of lists or dictionaries if you hover over the line numbers on the left. It can fold Python code and other languages as well.