# Exercise: Dictionary Basics

## Summary

Practice with basic operations involving dictionaries in Python.

## Input Data

One input file is provided, **nato-alphabet.txt**, that provides the standard words used for spelling things verbally using the NATO phonetic alphabet.

## Deliverables

There are two deliverables for this assignment: **basics.py** and **nato.py**.

## Instructions

### A. Script basics.py

The first part of this assignment is to build a script called `basics.py` that does the steps below.

1. Include the line `import json` (shown below) near the top of the file to import the JSON module. JSON is short for JavaScript Object Notation and the `json` module allows complex data objects to be read in or written out in very clear notation. We'll use JSON a lot during the semester.

   ```
   import json
   ```

2. Create four dictionaries called `ca`, `tx`, `fl`, and `ny`. They will be used to store information about the four US states with the largest populations in 2016. Each dictionary should have three keys: `po` for the state's two-letter postal code, `name` for its name, and `pop` for its population. Use the information below to set the corresponding values for each dictionary:

   ```
   name         po  pop
   California   CA  38654206
   Florida      FL  19934451
   New York     NY  19697457
   Texas        TX  26956435
   ```

   To be clear, the data should be coded directly into the script: it is not necessary to store it in a file and then read the file.

3. Print the `ny` dictionary.

4. Now create a list called `state_list` that has one element for each of the state dictionaries. That is, it should consist of the four dictionaries created above: `ca`, `tx`, `fl`, `ny`.

5. Print `state_list`.

6. Now print `state_list` with better formatting using the `dumps()` call from the `json` module as shown below. The name "dumps" is short for "dump (write out) to a string" and the indent parameter says to indent each level of the object being printed by 4 spaces. It will be clearer from the output what it does.

   ```
   print( json.dumps(state_list,indent=4) )
   ```

   If all goes, well, it should be very clear that `state_list` is a list with four dictionaries as elements.

7. Add a variable called `uspop` that is equal to 317899153, which was the total population of the states in 2016 (it excludes Washington, DC, Puerto Rico, and other US territories).

8. Print a message `Percent of the US population:` to create a heading for what comes next.
9. Loop through `state_list` using `state` as the running variable (the variable name following `for` in a for-loop). Within the loop, do the following:
    1. Create variables `name`, `po`, and `pop` that are equal to that state's name, postal code, and population.
    2. Create a variable called `po_str` that is equal to a string formed like this: `"("+po+")"`.
    3. Create a variable called `pct` that is equal to 100 times the ratio of the state's population to `uspop`. Use the `round()` call to round the results to two decimal places.
    4. Print out `name`, `po_str` and `pct` (one line per state). Each line should look something like this:

       ```
       California (CA) 12.16
       ```

**B. Script nato.py**

For the second part of the assignment, build a script called `nato.py` that does the following:

1. Create a variable called `natofile` equal to the string "nato-alphabet.txt".
2. Create an empty dictionary called `to_nato`.
3. Open `natofile`.
4. Loop through the lines of the file doing the following to each line:
    1. Use the `strip()` call to remove leading and trailing blank space;
    2. Use the `lower()` call to convert it all to lower case;
    3. Use `split()` to break the result into a list of words called `words`.
    4. Finally, use element 0 of `words` (the letter) as a dictionary key and set the value of `to_nato` for that key to `words` element 1 (the NATO word).
5. Create a variable called `syr_str` that is equal to "syracuse" and create an empty list called `syr_list`.
6. Use a for-loop to go through the letters in `syr_str`, look each one up in `to_nato`, and then append that word to the end of `syr_list`.
7. Then join `syr_list` with spaces to create `syr_nato` and print out a message with `syr_str`,"is:", `syr_nato`.
8. Create a variable `myname_str` equal to your first name *in lower case*, and then repeat the last couple of steps to translate it into its phonetic alphabet equivalent as `myname_nato` and print out a message like the one for Syracuse. Please note: if your first name has spaces or punctuation you'll need to omit them because the "nato-alphabet.txt" file we're using only has letters: anything else will lead to an error. A more complex script could work around that limitation.

## 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

- This just scratches the surface of what can be done with dictionaries. They are the most powerful and flexible of Python's core data structures.

- If you know how to define and use functions, you may want to set one up to translate a given string to its NATO equivalent and then print out the result. You could then call the function on `syr_str` and then `my_str` without having any duplicate code.
- The `demo.py` file shows how to build a dictionary of dictionaries, which is more versatile than a list. However, that step is not required for this exercise.