Practical 5 - Dictionaries

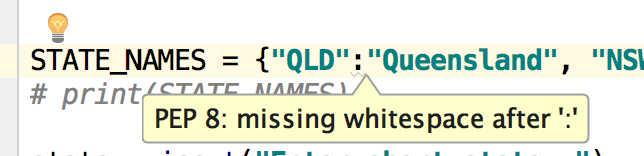
**Did you finish last week’s work? Including the practice section?** If not, make sure to complete it during the week. If you do not understand anything, bring those questions to your tutor the following week.

# Walkthrough

Copy the code from: <https://github.com/CP1404/Practicals2016/blob/master/Prac05/stateNames.py>

This is a program that uses a ‘constant’ (name is ALL\_CAPS) dictionary to store the Australian state abbreviations and names - e.g. QLD is Queensland. It asks the user for their 'short' state and prints the full state name by looking it up in the dictionary.

**Things to do:**

1. Run the program to see how it works.
2. Right now the code formatting of this dictionary is incorrect and inconsistent. Dictionary literals should be formatted with no space before and one space after the colon, like {"A": 1, "B": 2}  
   Thankfully, PyCharm knows this and can fix it for you. If you move your mouse over the grey line near the first colon, PyCharm pops up the problem... Click on it to see the action icon (light bulb) to the left... Click on that to see the options... Choose reformat file... Problem solved.   
   You can also choose **Code > Reformat Code** or use the shortcut any time. It does the whole file or whatever's selected. No more excuses for dodgy formatting!  
   This is a great way to learn what the "PEP 8" standards are for Python code formatting style.  
   
3. Currently the program requires you to enter the states in capitals. Fix this so lowercase inputs also work to show the state names. (There are two places to add a string method.)
4. Do this next part on paper first (then in PyCharm: Write a loop that prints all of the states and names neatly (line them up with str.format), like:

NSW is New South Wales

QLD is Queensland

TAS is Tasmania

# Intermediate Exercises

Based on the state name example program above, create a program that allows you to look up hexadecimal colour codes like those at <http://www.color-hex.com/color-names.html>

Use a constant dictionary of about 10 names and write a program that allows a user to enter a name and get the code, e.g. entering **AliceBlue** should show **#f0f8ff**.

# Do-from-scratch Exercise

Write a program to count the occurrences of words in a string. The program should ask the user for a string, then print the counts of how many of each word are in the file.  
The output should look like this:

Text: this is a collection of words of nice words this is a fun thing it is

a : 2

collection : 1

fun : 1

is : 3

it : 1

nice : 1

of : 2

thing : 1

this : 2

words : 2

**Hints:** use a dictionary where the keys are the words and the values are the counts; when you find a word, check if it’s in the dictionary...

Notice that the sample output is sorted. As a refinement step, **after** you have the program working, make your program do this sorting.

As a further refinement, align the numbers so they are in one column. You will need to find the longest word in the list first, and then you need to know how to use the str.format method to take a variable width. The former is up to you, but the latter can be done with another {} placeholder, like:

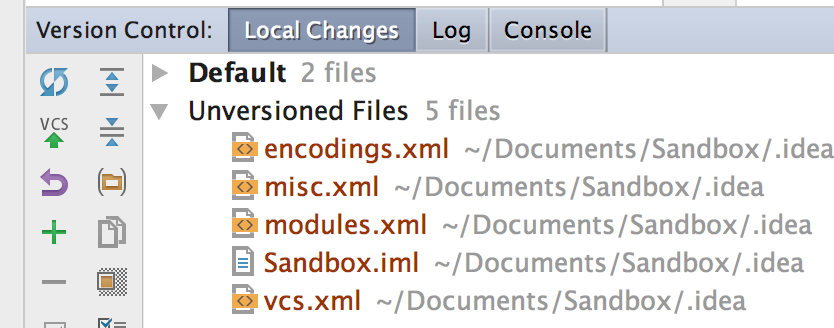
print(**"{:{}} : {}"**.format(a, b, c))

This formats the first placeholder value, a, with a width of b, then prints a literal : then the value of c. Nice.

**.gitignore**

Before we're done, let's learn one more Git thing. Ignoring files.

If you have files in your project that you don't want stored in your repo (like PyCharm metadata files), you can just choose not to add them (as we've done until now) but they do show up as "unversioned files", which is kind of a warning... We'd prefer this to show us files we probably should add.



The solution is to add a file called **.gitignore** to your repository. Note the exact spelling, including the dot at the start. On Unix-like systems, then dot makes a file/folder hidden.

This is just a plain text file that stores the names of any files or folders you want Git not to track and not to warn you about.

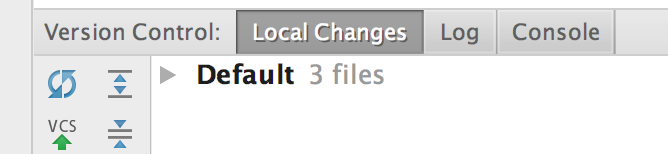
Do this now – create the file, and let PyCharm add it to Git.

Then enter one line:

.idea

Which is the folder that PyCharm stores its project metadata in.

Now look at that Version Control tool window... problem solved!



Commit.

# Practice & Extension Work

1. Convert parallel lists into a dictionary…

Recall that it’s possible to represent information in the form of parallel lists where the indices determine how the information is related across lists. For example:

names = ["Jack", "Jill", "Harry"]

dobs = [(12, 4, 1999), (1, 1, 2000), (27, 3, 1982)]  
This means Jack was born on 12/4/1999, Jill was born on 1/1/2000, and Harry was born on 27/3/1982.   
Write a program using a dictionary instead of the above parallel lists that allows the user to enter the date-of-birth details for 5 people, and have it display their individual ages.  
**Hint:** you can **split()** a string like “12/4/1999”, as we did in the lecture last week.

1. Write a function that takes two parallel lists as input parameters and returns a dictionary where keys are from the first list and the values are from the second. Use the above example as a test case.
2. In practical 1 you should have created an electricity bill estimator using constant values for the tariff amounts like:

TARIFF\_11 = 0.244618

TARIFF\_31 = 0.136928

|  |
| --- |
| Electricity Bill Estimator 2.0  Which tariff? 11 or 31: 11  Enter daily use in kWh: 13.4  Enter number of billing days: 90  Estimated bill: $295.01 |

Now create a version of the above electricity program that uses a **dictionary** to store the tariffs and the corresponding cost.   
In the prompt, list all of the tariffs (all of the dictionary keys) and make sure a valid one is selected.  
Use the appropriate cost from the dictionary to calculate the bill total.

You will need to change how you present the "Which tariff" prompt, since these values come from the dictionary.

To show the benefit of this, add three more tariffs (make them up).   
You should find that this is a very simple step for you, and your program can handle it without any extra coding.