Tutorial MATH 1MP3 – February 9, 2017

The following questions are to be done in groups of two or three.

- Open PyCharm and create a new project (File -> New Project, make sure the drop down box in "interpreter" points to your Anaconda installation!)
- In this project create a new python file (File -> New, then click "Python File" in the menu that appears)
- Write the code for the following questions in that python file. Just have each question answer follow the previous one in the code. (I recommend doing each question one at a time, though)

For the following questions, write a python program (in PyCharm or otherwise) that prints the answer.

For all the following questions your answers should be python functions that work on any input, not just the example inputs given!

Don't forget docstrings!!

1) In Python, adding lists together works in the way we'd like it too. For instance,
   ```
   [1, 2, 3] + [4, 5, 6] # [1, 2, 3, 4, 5, 6]
   ```
   However this doesn't work for dictionaries!
   ```
   {"a":1, "b":2} + {"c": 3, "d":4} # Gives TypeError
   ```
   a) Write a Python function called `add_dictionaries(dict_a, dict_b)` that takes in two dictionaries and returns them added together.
   For instance:
   ```
   add_dictionaries({"a":1, "b":2}, {"c": 3, "d":4})
   # Should return {"a":1, "b":2, "c": 3, "d":4}
   ```
   b) Try running.
   ```
   add_dictionaries({"a":1, "b":2}, {"a": 3, "d":4})
   ```
   What happens? How should we deal with this?
   Modify your function so that it can "merge" two dictionaries with similar keys. So that
   ```
   add_dictionaries({"a":1, "b":2}, {"a": 3, "d":4})
   # Should return {"a":4, "b":2, "d":4}
   ```

2)
   a) Write a python function called `no_spaces(dict)` that takes in a dictionary whose keys are all strings, and returns a new dictionary whose keys have no spaces and instead are replaced by underscores. I.e.
   ```
   no_spaces({"first key": 2, "second key": 3, "third": 7, "
   ":9})
   # Should return
   {"first_key":2,"second_key":3,"third":7,"_":9}
   ```
   b) Similar to Question 1b) try running
   ```
   no_spaces({"first key": 2, "first_key": 3})
   ```
   Will likely give an error, depending on how you wrote your code. Modify it so that it works as in 1b) i.e.
   ```
   no_spaces({"first key": 2, "first_key": 3})
   # Should return {"first_key":5}
   ```

3)
   a) Write a python function called `string_to_dict(s)` that takes in a string s and returns a dictionary whose keys are the letters of s and whose values are the number of times that letter shows up.

i.e.:

```
string_to_dict("Hello World")
# {"H":1, "e":1, "l":3, "W":1, "r":1, "d":1, " ":1, "o":2}
```

b) Download the file: http://www.gutenberg.org/files/2600/2600-0.txt and save it in your PyCharmProjects directory so that you can open it!

c) Write some python code (not a function, just code!) that opens the file and reads it all into a string (use `file.read()` to read it all at once, don't bother looping through by each line!)

d) Using your function from part a), tell me how many spaces are in *War and Peace* (the .txt file you just saved!). Dictionaries and Python are very, very good at performing text analysis like this very quickly!

e) Write a loop that loops through all the lower-case letters of the alphabet. (Consider using `for letter in "abcdefghijklmnopqrstuvwxyz"`) and `print`s out the letter and its number of occurences in *War and Peace*!

```
# Output should be something like
a 196155
b 31052 #etc..
```

4)

a) Write a python function called `dictionary_max(dict)` that returns the maximum value of all the values in a dictionary

```
dictionary_max({"a":4, "b":2, "d":3}) # Should return 4
```

b) Modify the function so that it returns the key that corresponds to the maximum value in the dictionary

```
dictionary_max({"a":4, "b":2, "d":3}) # Should return "a"
```

c) Now give the function a second optional argument called `n` (initially `n=1`) (i.e. now you should be declared like

```
def dictionary_max(dict, n=1):
 #some_stuff
```

So that dictionary_max now returns the top "n" keys in a dictionary, in order

```
dictionary_max({"a":4, "b":2, "d":3}, 2)
# Should return ["a","d"]
dictionary_max({"a":4, "b":2, "d":3, "e":4}, 3)
# Should return ["a","e", "d"] or ["e","a","d"]
# Here the order of "a" and "e" depends upon your computer,
# because dictionaries don't necessarily respect order!
```

d) If you've done number 3d), try running running `dictionary_max` to get the top 5 letters used in *War and Peace.*