HW 7: Calculating Molecular Weights

Your job is to create a program that receives molecular formulae from the user, and calculates the molecular weight of each one. This notebook should be called `molecular-weight-calculator.ipynb`.

```
Please enter a molecular formula, or hit enter to quit.
Formula: H2O
  The molecular weight is 18.0154 g/mole.
Formula: HOH
  The molecular weight is 18.0154 g/mole.
Formula: CO2
  The molecular weight is 43.9998 g/mole.
Formula: NoTvAlId
  I'm sorry.  That's not a valid formula.
Formula:
Goodbye!
```

Of course, if there is no digit after an atomic symbol, that means that its quantity is 1. You may assume that all formulae will consist of only proper atomic symbols and digits. You don't need to worry about parentheses at all, nor complicated hydrates. However, you must be careful that you still calculate the proper result when some symbol is present more than once (such as in `HOH` above). You should round all the final molecular weights to 4 fractional digits.

You will first need to download the file `element-weights.tsv`, and turn it into a dictionary. However, parsing the molecular formula for atomic symbols are their quantities will likely be challenging. Start by removing all spaces and other odd symbols from the formula. This will make what comes next easier:

1. Regular expressions will help. You can use the `finditer()` method in order to locate all of the atomic symbols in the formula. Make a list of all the matches you found.
2. A list of the symbol strings should be easy now. Make it so that all the symbols are in order. For example, `H2O` would yield the list `['H', 'O']`.)
3. Now, make another list that holds the quantities of each element. Use your list of matches to check the locations of each atomic symbol. If one of the symbols is at the very end of the formula, or another symbol is *immediately* after it, that means that no number was given. Its quantity must be 1. However if there is space after a symbol, that means that its quantity was given explicitly. That can be extracted from the string and converted to an `int`.
4. Now that you have two lists, you can return them. Do this by returning the two lists between a pair of parentheses. For example, if your lists are called `symbols` and `quantities`, you can write the line `return (symbols, quantities)`. This is called a `tuple`. It's similar to a list, but has some subtle differences. A tuple is indexed by

`int`, just like a list.

5. If you use this trick to return two things at the same time, be sure that you only call the function once!

Once you have both a list of symbols and their quantities, you can calculate the total weight easily. However, be kind to the user. If they enter a bad symbol, let them know nicely rather than crashing the program.