Regular Expressions

Regular expressions are a language to describe patterns in strings. They are extremely useful when working with data sets. To start, we need the module `re`, and two functions from it: `search` and `findall`. The function `re.search` takes as input a pattern and a string, and returns an object (of a class that is specific to the `re` module) that contains the first substring matching the pattern, together with its location in the string. We can call the former using the group() attribute, and the latter using the start() and end() attributes. For example:

```
In:   import re
In:   re.search(r'abc','abcd').group()
Out:  'abc'
In:   re.search(r'abc','abcd').start()
out:  0
In:   re.search(r'abc','abcd').end()
Out:  3
```

The group() attribute allows us to group the pattern, using parentheses, such that we can call different parts of it. (Imagine, for example, that we are loooking for email addresses and wish to sometimes only call the username or the domain.) These groups may be nested; they are ordered by their leftmost parenthesis. The 0-group always returns the whole pattern. For example:

```
In:    re.search(r'((a)b)c','abcd').group(1)
Out:  'ab'
In:    re.search(r'((a)b)c','abcd').group(2)
out:  'a'
```

The findall() functions simply returns a list containing all substrings that match the pattern, without information about their location. To ask for more general patterns, we use placeholders, such as `.` for any character, `\d` for any digit, `[a-z]` for lower case letters, `[A-Z]` for upper case letters. We also use multipliers, such as `?` for 0 or 1, `+` for 1 or more, `*` for any number. If we want to look for these special characters, instead of using them as placeholders, we can use `[]`. We can also use `|` inside square brackets to mean "or". So if we need an `a` or a `b`, we ask for `a|b`. We can also use lookforwards and lookbackwards. These let us filter out patterns that are preceded or followed by another pattern, without asking for that pattern. For example, suppose that we have a list of real numbers that all have decimals, and we would like to filter out the integer parts, so we use a lookforward, indicated by `(?=)` to a period:

```
In:   s='1.86 5.30 8.54 13.75'
In:   re.findall(r'\d+(?=[.])',s)
Out:  ['1', '5', '8', '13']
```

The cheat sheet linked to from my page is very useful to keep in the background. For example, suppose that we have a piece of text, and we are looking for words that are capitalized, but that are not at the beginning of a sentence. These are words that are preceded by a lower case letter and then a space:

```
In:    s='October arrived, spreading a damp chill over the grounds and
       into the castle. Madam Pomfrey, the nurse, was kept busy by a
       sudden spate of colds among the staff and students. Her Pepperup
       potion worked instantly, though it left the drinker smoking at the
       ears for several hours afterward. Ginny Weasley, who had been
       looking pale, was bullied into taking some by Percy. The steam
       pouring from under her vivid hair gave the impression that her
       whole head was on fire. '
In:    re.findall(r'(?<=[a-z] )[A-Z][a-z]+','s')
out:  ['Pomfrey', 'Pepperup', 'Weasley', 'Percy']
```

Exercises

- Write a regular expression that extracts all full sentences that are questions from a piece of english text.
- Write a regular expression that finds the maximum of all numbers (either integers or floats) that are in a string. The pattern '26' should be counted as twenty six, rather than a 2 and a 6.
- Write a regular expression that finds all short sentences in a piece of english text (sentences with no more than 10 words).
- Write a regular expression that lists the first letters of every sentence in a piece of english text.
- Write a regular expression that checks whether a string contains both upper and lower case letters.
- Write a regular expression that checks whether a string contains upper, lower case letters and at least one digit.
- Use regular expressions to add commas between two numbers in a list of numbers separated only by spaces. So, we want to turn `'1 3 8 2'` into `'1, 3, 8, 2'`. Then, also add square brackets, so that the list is written in a way that can be used in python.
  We want `'[1, 3, 8, 2]'`. Put this all together to create a function that takes as input a string of space separated numbers, and outputs a python list of those numbers (not strings).