Captured Groups in Regular Expressions

Recall that we use parentheses (...) to capture groups in a regular expression. The groups are numbered 1,2,3,... by the appearance of their opening parenthesis. We can then ask for any of the groups to be repeated in the pattern. This is different from using a non-capturing group and a quantifier as we did last lecture, since that was asking for a pattern to be repeated, rather than the exact substring. For example, suppose that we are looking for a pattern of the form `'a1a'` or `'b2b'` (a letter, followed by a digit, followed by the same letter again). We can do that as follows:

```
In:   re.search(r'([a-z])\d\1','h3g h4h').group()
Out:  'h4h'
```

And, the example from the lecture: looking for a repeated word. (Note the extra spaces around the pattern, this is to avoid finding consecutive words with similar ending/beginning.)

```
In:   s='this is is a sentence'
In:   re.search(r' (\w+) \1 ',s).group()
Out:  ' is is '
```

We can then use `re.sub()` to replace a pattern. In this case, remove the repeated word:

```
In:   re.sub(r' (\w+) \1 ',r' \1 ',s)
Out:  'this is a sentence'
```

It is not possible in a single regular expression to ask for a captured group to appear an arbitrary number of times, or to look for arbitrarily many groups, but we can often achieve such things by combining a regular expression with a loop. For example, suppose that we want to check that a word (in lower case letters) is a palindrome. We know that a word is a palindrome if the first and last letter are the same, and the part between them is a palindrome. So, we can work recursively.

```
def palindrome(word):
    if len(word)<=1:
        return True
    elif re.match(r'([a-z]).*\1',word):
        return palindrome(word[1:len(word)-1])
    else:
        return False
```

Exercises

- Write a function that finds sequences of letters repeated 3 times. For example, `'abaaaabbccc'` contains `'aaa'` twice, and `'ccc'` once.
- Write a regular expression that recognizes large integers with correct thousands separators, such as 10,000 and 3,746,982.
- Write a regular expression that recognizes valid times, such as 21:45, but not 25 : 98.
- Write a function that finds the longest word in an english text that appears at least three times.

- Write a function that uses regular expressions to test whether a list has repeated elements.
- Write a function that shortens sentences. “This is a sentence.” becomes“This ... sentence.”

Cleaning Data

We can use regular expressions to transform data from .txt or other types of files to python variables. Later on, we will work with modules (such as numpy and pandas) and variable types that are useful for data analysis, but for now we can practice with putting data into lists, using regular expressions. For example, suppose we have the following `marathon.txt` file, with data on marathon times.

```
Andrea    5:31
Ben       5:02
Carl      6:21
Didi      5:10
```

We read this into a string using the open() and read() functions. Make sure that your working directory (top right of the screen in Spyder) matches the directory of the file.

```
In:   times=open('marathon.txt', 'r').read(); times
Out:  'Andrea  5:31\nBen     5:02\nCarl    6:21\nDidi    5:10'
```

We can use the `re.split()` function to split our string first by the newline separators:

```
In:   Lrows=re.split(r'\n',times); Lrows
Out:  ['Andrea  5:31', 'Ben     5:02', 'Carl    6:21', 'Didi    5:10']
```

We then split each row by the space separators. In this case, it makes sense to split the time into two items: hours and minutes. Therefore, we also split by the “:” symbol.

```
In:   L=[re.split(r'\s+|:',i) for i in Lrows]; L
Out:  [['Andrea', '5', '31'], ['Ben', '5', '02'],
      ['Carl', '6', '21'], ['Didi', '5', '10']]
```

Finally, we would like thw hour and minute items to be integers, not strings:

```
In:   for i in L:
..:       for j in [1,2]:
..:         i[j]=int(i[j])
In:   L
Out:  [['Andrea',5,31], ['Ben',5,2], ['Carl',6,21], ['Didi',5,10]]
```

Exercises

- Find different types of data that are of interest to you, and think about how to clean it up and import it into python in a useful form.