

RegEx Exercises

Create a new Python file called `re_functions.py` (or something similar). When these exercises are complete there should only be functions in the file. Any function calls or `print()` should be removed. Make sure to have good, informative [doc strings](#) and, of course, [appropriate commenting](#).

1. Write a function that takes a string as an argument to test the first character of a string is uppercase or not.

```
string_1a = "This should return True"
string_1b = "this should return False"
```

2. Write a function that takes a string as an argument and returns a list containing all words in the string that are capitalized. Use [sample_sentences.txt](#).

3. Write a function that takes `a_string` as an argument and returns all words that contain 2 vowels in a row (ex - "goes").

```
string2 = "Sam goes to school. Sam comes home and studies. Sam is a good boy."
```

4. Write a function to count the number of vowels in a given string. Use `string2` from previous problem.

5. Write a function to find all words that contain a numeric character contained in it.

```
string4 = "a2c if3 clean 001mn10 string asw21"
```

6. Write a function `fix_punctuation`, which replaces multiple occurrences of "!" (exclamation mark), "?" (question mark), and "." (period) with single instances of each. It should take a string as an argument and return its copy with all repetitions replaced.

Sample call: `fix_punctuation('Wow!!! But why??? Who knows...')`

Sample result: `'Wow! But why? Who knows.'`

7. Write four functions, `gerunds`, `abbrevs`, `cv words`, and `time stamps`, which match the following types of strings. Your functions should take a string as an argument and return True

or False depending on whether a match has been found. Split your regular expressions into parts and explain what each one does.

- Gerund forms (ending with “-ing”) which are at least seven characters long.
- Abbreviations such as U.S.A., U.S.S.R.
- Words with a consonant-vowel (CV) structure, that is in which every consonant is followed by exactly one vowel
- Time stamps in the (24-hour) format hour:minutes:seconds (00:00:00). Think about valid digits in each position.

8. Write a function that takes a sequence of strings of the format seen below and returns a list of strings extracting the two years appearing in each string element. So, as an example, the first element becomes “from 1993 to 2003”.

```
tuple_7 = ("AT0ACH10000700100dymax.1-1-1993.31-12-2003",
           "AT0ILL10000700500dymax.1-1-1990.31-12-2011",
           "AT0PIL10000700500dymax.1-1-1992.31-12-2011",
           "AT0SON10000700100dymax.1-1-1990.31-12-2011",
           "AT0STO10000700100dymax.1-1-1992.31-12-2006",
           "AT0VOR10000700500dymax.1-1-1991.31-12-2011",
           "AT110020000700100dymax.1-1-1993.31-12-2008",
           "AT2HE190000700100dymax.1-1-1993.31-12-2000",
           "AT2KA110000700500dymax.1-1-1991.31-12-2010",
           "AT2KA410000700500dymax.1-1-1991.31-12-2011")
```

9. Write a function that takes a string as an argument and returns a list of words with all non-alphabetic characters removed. Use [sample_sentences.txt](#) to test.

10. Write a function that takes a string and number of letters (make this argument default to length 1) as arguments and returns a list of words, in the string, of the specified length. Use this function and [sample_sentences.txt](#) to plot the 4-letter words and their frequency.