Regular expressions: Exercises

Exercises 1–4 are for in-class evaluation, the other exercises are to be handed in via the Student Portal. Exercises marked with an asterisk are required for a "pass with distinction". Additional files needed for the assignments can be found here.

Please make sure that your code *complies with the specification*, above all that all functions take the right number of arguments, and return objects of the right type.

For the in-class exercises, please prepare your solutions at home in advance, and then demonstrate them to and discuss them with the examiner, rather than write all code on the spot.

1. Which of the following regular expressions are syntactically incorrect? Why? What kind of strings do the valid ones match?

- `?.\b`
- `[?].\b`
- `[^\D]{4,1}`
- `^[^^]`
- `++`
- `+\+`
- `\++`
- `\+\+`

2. What kind of strings will be matched by the following regular expressions? In your answer, break them up into parts and explain what each one does.

- `^[+-]?\d+(\.\d+)?`
- `\b[aeiou][a-z]{,4}\b`
- `[.?!]\s+([A-Z][A-Za-z]*)`

3. Write four functions, `gerunds`, `abbrevs`, `cv_words`, and `time_stamps`, which match the following types of strings. Your functions should take a string 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.

4. 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 invocation

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

Sample result

```
'Wow! But why? Who knows.'
```

4. Grep is a small Unix program which goes over a file and returns a list of lines matching the provided regular expression. Write a function `grep` replicating this behaviour. It should accept two arguments: path to a file and a regular expression. It should print out all lines which match the regular expression. Test your function on file matches.txt.

Sample invocation

```
grep('matches.txt', r'h[eao]{2}d')
```

Sample result

```
['She said "heed" very clearly!',
 'She said "head" very clearly!',
 'She said "hood" very clearly!']
```

5*. Extend your `grep` function to take another optional argument which defaults to `True` but when set to `False` makes grep return a list of lines which *do not* match the regular expression.

Sample invocation

```
grep('matches.txt', r'h[eao]{2}d')
```

Sample result

```
['She said "heed" very clearly!',
 'She said "head" very clearly!',
 'She said "hood" very clearly!']
```

Sample invocation

```
grep('matches.txt', r'h[eao]{2}d', True)
```

Sample result

```
['She said "heed" very clearly!',
 'She said "head" very clearly!',
 'She said "hood" very clearly!']
```

Sample invocation

```
grep('matches.txt', r'h[eao]{2}d', False)
```

Sample result

```
['She said "hid" very clearly!',
 'She said "had" very clearly!',
 'She said "hod" very clearly!',
 'She said "hawed" very clearly!',
 'She said "who'd" very clearly!']
```

6*. Write a function `extract_url` which goes over an HTML file and returns a list of addresses of all websites linked from this file. Specifically, your function should find all tags of the form:

```
<a href="https://en.wikipedia.org/wiki" title="Wikipedia">
```

and extract the value of the `href` property. Test your function on file href.html Notice that there might be more than one link per line!

Sample invocation

```
extract_url('href.html')
```

Sample result

```
['https://en.wikipedia.org/wiki',
 'https://en.wikipedia.org/wiki/Theoretical_computer_science',
 'https://en.wikipedia.org/wiki/Formal_language_theory', ...]
```