Text Wrangling and Regular Expressions

Pandas `str` methods

Function	Description
`s.str.len()`	Returns a Series containing length of each string
`s.str[a:b]`	Returns a Series where each element is a slice of the corresponding string indexed from `a` (inclusive, optional) to `b` (non-inclusive, optional)
`s.str.lower()/s.str.upper()`	Returns a Series of lowercase/uppercase versions of each string
`s.str.replace(pat, repl)`	Returns a Series that replaces occurences of substrings matching the regex `pat` with string `repl`
`s.str.contains(pat)`	Returns a boolean Series indicating if a substring matching the regex `pat` is contained in each string
`s.str.extract(pat)`	Returns a Series of the first subsequence of each string that matches the regex `pat`. If `pat` contains capturing group(s), outputs a DataFrame with one column for each group.
`s.str.split(pat)`	Splits the strings in `s` at the delimiter `pat`. Returns a Series of lists, where each list contains strings of the characters before and after the split.

Regex patterns

Operator	Description	Operator	Description
`.`	Matches any character except `\n`	**`*`**	Matches preceding character/group zero or more times
`\`	Escapes metacharacters	**`+`**	Matches preceding character/group one or more times
`|`	Matches expression on either side of expression; has lowest priority of any operator	**`^`**	Matches the beginning of the string
`\d`, **`\w`**, **`\s`**	Predefined character group of digits (0-9), alphanumerics (a-z, A-Z, 0-9, and underscore), or whitespace, respectively	**`$`**	Matches the end of the string
`\D`, **`\W`**, **`\S`**	Inverse sets of **`\d`**, **`\w`**, **`\s`**, respectively	**`( )`**	Capturing group or sub-expression
`{m}`	Matches preceding character/group exactly `m` times	**`[ ]`**	Character class used to match any of the specified characters or range (e.g. `[abcde]` is equivalent to `[a-e]`)
`{m, n}`	Matches preceding character/group at least `m` times and at most `n` times. If either `m` or `n` are omitted, set lower/upper bounds to 0 and ∞, respectively	**`[^ ]`**	Invert character class; e.g. `[^a-c]` matches all characters except `a`, `b`, `c`

Python `re` methods

Function	Description
`re.match(pattern, string)`	Returns all matching characters if zero or more characters at beginning of `string` matches `pattern`, else None
`re.search(pattern, string)`	Returns all matching characters if zero or more characters anywhere in `string` matches `pattern`, else None
`re.findall(pattern, string)`	Returns a list of all non-overlapping matches of `pattern` in `string` (if none, returns empty list). If `pattern` includes capturing groups, only return captured characters.
`re.sub(pattern, repl, string)`	Returns `string` after replacing all occurrences of `pattern` with `repl`