Data 100 Regular Expressions

(Fall 2023)
Here's a complete list of metacharacters:

`. ^ $ * + ? { } [ ] \ | ( )`

Some reminders on what each can do (this is not exhaustive):

`"^"` matches the position at the beginning of string (unless used for negation `"[^]"`)

`"$"` matches the position at the end of string character.

`"?"` match preceding literal or sub-expression 0 or 1 times.

`"+"` match preceding literal or sub-expression *one* or more times.

`"*"` match preceding literal or sub-expression *zero* or more times

`"."` match any character except new line.

`"[ ]"` match any one of the characters inside, accepts a range, e.g., `"[a-c]"`.

`"( )"` used to create a sub-expression

`"\d"` match any *digit* character. `"\D"` is the complement.

`"\w"` match any *word* character (letters, digits, underscore). `"\W"` is the complement.

`"\s"` match any *whitespace* character including tabs and newlines. `\S` is the complement.

`"*?"` Non-greedy version of *. Not fully discussed in class.

`"\b"` match boundary between words. Not discussed in class.

`"+?"` Non-greedy version of +. Not discussed in class.

`"{m,n}"` The preceding element or subexpression must occur between m and n times, inclusive.

Some useful `re` package functions:

`re.split(pattern, string)` split the `string` at substrings that match the `pattern`. Returns a list.

`re.sub(pattern, replace, string)` apply the `pattern` to `string` replacing matching substrings with `replace`. Returns a string.

`re.findall(pattern, string)` Returns a list of all matches for the given `pattern` in the `string`.