### Regular Expressions Anchors
- `^` Start of string, or start of line in multi-line pattern
- `\A` Start of string
- `$` End of string, or end of line in multi-line pattern
- `\Z` End of string
- `\b` Word boundary
- `\B` Not word boundary
- `<` Start of word
- `>` End of word

### Regular Expressions Character Classes
- `\c` Control character
- `\s` White space
- `\S` Not white space
- `\d` Digit
- `\D` Not digit
- `\w` Word
- `\W` Not word
- `\x` Hexadecimal digit
- `\O` Octal digit

### Regular Expressions POSIX
- `[upper:]` Upper case letters
- `[lower:]` Lower case letters
- `[alpha:]` All letters
- `[alnum:]` Digits and letters
- `[digit:]` Digits
- `[xdigit:]` Hexadecimal digits
- `[punct:]` Punctuation
- `[blank:]` Space and tab
- `[space:]` Blank characters
- `[ctrl:]` Control characters
- `[graph:]` Printed characters
- `[print:]` Printed characters and spaces
- `[word:]` Digits, letters and underscore

### Regular Expressions Quantifiers
- `*` 0 or more
- `+` 1 or more
- `?` 0 or 1
- `(3)` Exactly 3
- `(3,)` 3 or more
- `(3,5)` 3, 4 or 5
- `?` Add a ? to a quantifier to make it ungreedy.

### Regular Expressions Escape Sequences
- `\` Escape following character
- `\\` End literal sequence

*“Escaping” is a way of treating characters which have a special meaning in regular expressions literally, rather than as special characters.*

### Regular Expressions Pattern Modifiers
- `g` Global match
- `i` Case-insensitive
- `m` Multiple lines
- `s` Treat string as single line
- `x` Allow comments and white space in pattern
- `e` Evaluate replacement
- `U` Ungreedy pattern

### Regular Expressions String Replacement
- `$n` nth non-passive group
- `$2` “xyz” in `(abc)(xyz)="/2"
- `$1` “xyz” in `(?:abc)(xyz)="/1"
- `$` Before matched string
- `$+` Last matched string
- `$&` Entire matched string

Some regex implementations use \ instead of $.