# Regular Expressions

Class 5

# Overview

1. Announcements
2. Review + Exercises
3. Q&A
4. Basic assignment

# Announcements

- Shell assignments due Oct 30
- Regex assignments due Nov 6
- Shell survey closing tonight!
- If you ever get stuck, please ask a question!
  - Piazza and Discord
- Office hours change
  - Tuesday 1-3 PM **in person at the BBB 4th floor spiral staircase**

# Misc special characters

- `.` matches *any* single character
  - `...` matches three consecutive characters
- `|` for an OR between regexes
  - `hello|world` matches a string that is "hello" or "world"
- `\` for special expressions/escapes
  - `\b` matches the empty string at the edge of a "word"
  - There's more: check the GNU `grep` manual for the rest
- `(`, `)` enclose a whole expression as a *subexpression*

# Brackets

- `[`, `]` enclose a set to match for **one character**
  - `[abc]` matches 'a', 'b', or 'c'
- `-`: range
  - `[A-Za-z0-9]`: capital and lowercase numbers and digits
- `^`: not in set
  - `[^ab]`: everything not 'a' or 'b'
- Named classes
  - e.g. `[:alnum:]` (alphanumeric characters)
  - See the Grep documentation for more
  - `[:alnum:]` is the actual name for the class, goes inside of `[`, `]`
  - `[[:alnum:]]` to actually use
  - to further illustrate: `[[:alpha:]0-5]`

# Quantifiers

- Specify how many of a preceding regex to match
- `?`: ≤1 time
- `*`: ≥0 times
- `+`: ≥1 times
- `{n}`: *n* times
- `{n,}`: ≥*n* times
- `{,m}`: ≤*m* times
- `{n,m}`: *x* times where $n \leq x \leq m$

# Exercise 1

- If you want to test these with `grep`, try using `grep -E`
  - Default `grep` uses BRE, which requires you to `\` escape a lot of things (more on this at the end)
- Write regexes that matches against:
  1. "hello" or "world"
  2. 3 of any character, "cat", then at least 5 of any character

# Exercise 2

Write regexes that match against:

1. 3 English vowels (a, e, i , o, u) in a row
2. 5 non-number characters in a row
3. "Odd" and a single digit odd number
4. "Even" and an even number
    - For simplicity's sake, leading 0s are allowed

# Anchors

- Perform *positional* matching
- `^`: match empty string at the beginning of a line
- `$`: match empty string at the end of a line

# Exercise 3

Write regexes that match against:

1. File names that end in ".txt"
2. File names that start with "file" with an odd number after and ends in ".txt"
3. A phone number with an optional country code
    - In the +X (XXX) XXX-XXXX format

# Backreferences

- Match previous parenthesized `( )` subexpression
- `\n`: match *n* th parenthesized subexpression

# BRE vs ERE

- In BRE `?`, `+`, `{`, `|`, `(`, and `)` must be escaped with `\`
  - (and `}` if you're specifying an interval)

# Exercise 4

- `sed` is a command that performs text transformations
- The `s` command can perform search-and-replace
- `sed 's/hello/world/g'` replaces instances of "hello" with "world"
  - The `g` at the end allows for multiple replacements on a line
  - The first character specifies the delimiter: doesn't have to be `/`
  - `s/pattern/replacement/flags`
  - `-E` as an argument puts `sed` into ERE mode
- Write a `sed` command that replaces instances of four digit numbers with "(XXXX)"
  - To be a number, it can't have letters next to it
  - `\<` and `\>` are positional matches for word boundaries

# Q&A

# Basic assignment