CS 61BL Lab 22

Ryan Purpura

Announcements

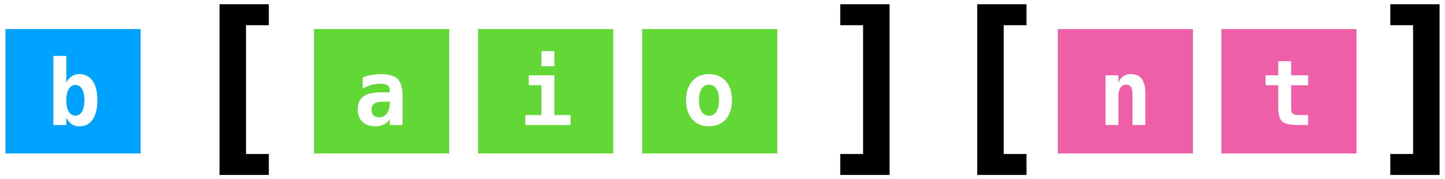
- Last lab! Thank you for a great semester!

Regular Expressions

- Regex lets you specify a pattern for ***matching*** whole strings or ***finding*** substrings.
- E.g. "gr[ae]y" matches both "grey" and "gray"
- How do we tame this black magic?

Brackets [...]

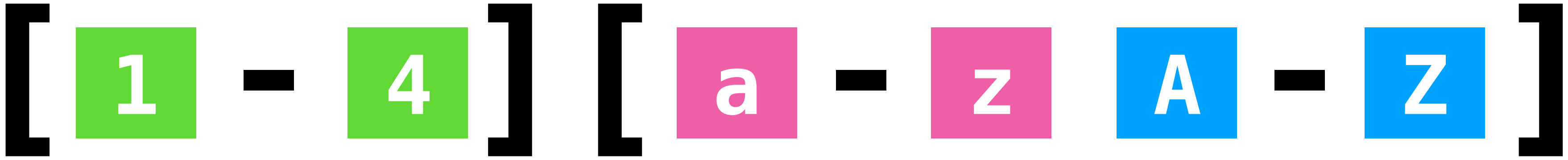
- Brackets allow you to match different characters in the same position.
- "b[aio][nt]" matches "ban", "bat", "bin", "bit", "bon", "bot"



b [a i o] [n t]

Ranges in Brackets [A-Z]

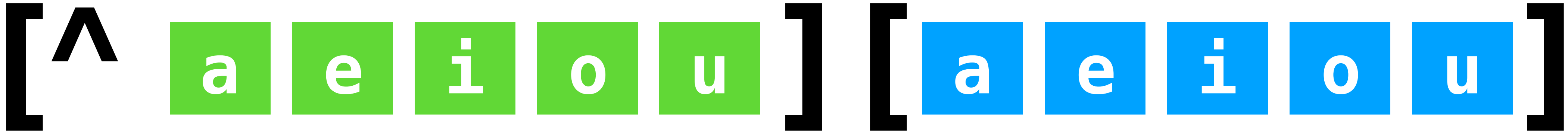
- Use ranges to quickly specify a range of characters. You can multiple ranges in a single bracket expression
- "[1-4][a-zA-Z]" matches "1a", "3F", "2g", etc.



[1 - 4] [a - z A - Z]

Carets with Brackets [^...]

- Adding a caret (^) after the opening bracket inverts the bracket expression.
- `"[^aeiou][aeiou]"` matches `"be"`, `"so"`, etc. but not `"aa"`, `"io"`, etc.



[^ a e i o u][a e i o u]

Character Classes

- **`\d`** is a digit [0-9]
- **`\w`** is a word character [A-Za-z0-9]
- **`\s`** matches whitespace (space, newline, tab)
- **`\D`**, **`\W`**, **`\S`** matches the inverse of the above.
- **`.`** (period) matches any non-newline character.
- E.g. `"\wat"` and `".at"` matches `"cat"`, `"hat"`, but not `"at"`

\w a t

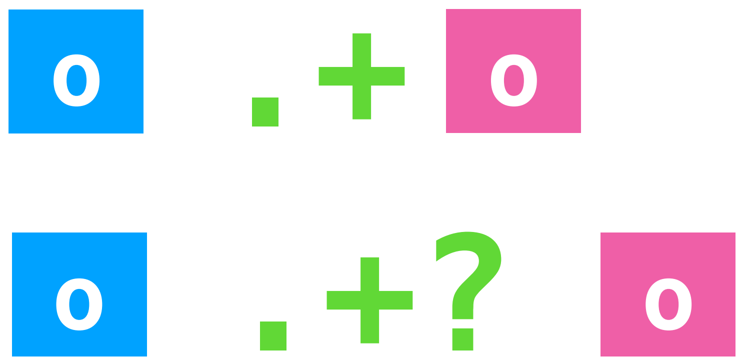
Quantifiers

- "+" matches one or more of preceding element
- "*" matches zero or more of preceding element
- "{x}" matches between x of the preceding element
- "{x,y}" matches between x and y of the preceding element
- "?" is shorthand for "{0, 1}"
- "\d+.*" matches "1", "34", "20cat", "42cs61bl"

```
\d+ .*
```

Greedy vs. Reluctant

- `+, ?,` `and` `*` are greedy (captures as many elements as possible)
- When *finding* within the string "owo owwwo", the pattern `"o.+o"` would *find*
    - `"owo  owwwo"`
- To make them reluctant (capture as few elements as possible), add "?" after
    - Also known as "lazy"
- The pattern `"o.+?o"` would find
    - "owo" and "owwwo" separately

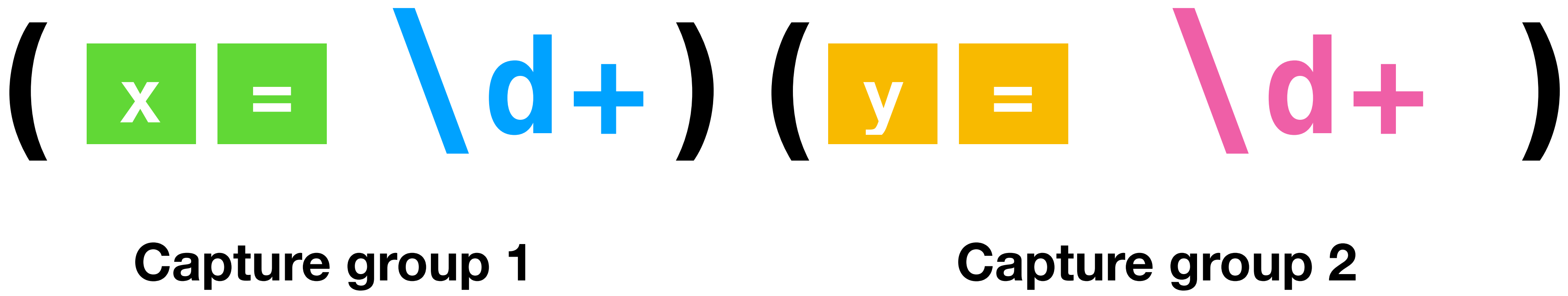


o .+ o

o .+? o

Capturing and Boundaries

- Parentheses indicate a capturing group: allowing you to extract text from a string.
- **`"(x=\d+) (y=\d+)"`** would match string **`"x=31 y=210"`**.
  - Capture group 1 is **`"x=31"`** and capture group 2 is **`"y=210"`**.
- **`"\b"`** matches the end of a word (a word boundary) and is zero-width
  - The pattern **`"cat\b"`** would be found in **`"a cat"`** but not **`"a catch"`**.



(x = \d+) (y = \d+)

Capture group 1

Capture group 2