Regular Expressions with regex Crate

Learn to Code with Rust / Section Review

The **regex** Crate

- The **regex** library crate is the community standard for regular expressions in Rust.
- A **regular expression** (RegEx) is a search pattern for text.
- RegEx uses symbols to declare a sequence of characters to look for within a string.
- The **regex** crate uses the terminology "haystack" to describe the string that the RegEx searches within.

Character Matches

- Use **Regex::new** to create a new pattern and call the **unwrap** method on the **Result**.
- Create the regex with a raw string to avoid conflicts with Rust's escape characters (like **\n** or **\t**).
- Write an **r** before the double quotes in the string.
- A regex can match sequences of exact characters (alphabetic, numeric, etc).

The **find** and **find_iter** Methods

- The **find** method on the **Regex** struct searches for the first match of the pattern within the haystack.
- The **find** method returns an **Option** to account for the possibility of no matches.
- The **Match** struct in the **Some** variant stores the start byte, the end byte, and the exact, concrete match.
- The **find_iter** method returns an iterable with *all* matches throughout the haystack. There is no need to worry about **Options** here.

Character Classes

- **\d** matches a digit.
- **\D** matches a non-digit.
- **\w** matches a word character (letters, digits, underscores).
- **\W** matches non-word characters (punctuation, spaces, other symbols).
- **\s** matches a whitespace.
- **\S** matches non-whitespace characters.

Word Boundaries

- The **\b** symbol matches a word boundary.
- A word boundary represents the gap between a word character and a non-word character.
- Combine the **\b** with another search character to look for matches at the beginning and end of words.

The Dot Metacharacter

- A **metacharacter** is a character that has special significance in a RegEx.
- The **dot** metacharacter matches any character. Technically, the **\n** newline is not included.
- Combine the dot with other symbols to create hardcoded + dynamic search patterns.
- Use **\.** to search for a literal dot/period in the pattern.

Square Brackets

- Place multiple characters inside square brackets to match any of them (**[kze]**).
- Use a dash to create a character range (**[a-m]**).
- Lowercase and uppercase characters are treated differently. Provide multiple character ranges (**[A-Ma-z]**) to capture all variations.

Number of Matches

- Use **{n}** to declare an exact number of matches.
  - **\d{4}** matches exactly 4 digits in a row
- Use **{n,}** to declare "at least n" matches.
  - **\d{4,}** matches at least 4 digits in a row
- Use **{n,m}** to declare a range of possible numeric matches (lower bound, upper bound).
  - **\d{4, 8}** matches between 4 and 8 digits in a row
- Use **+** to indicate "one or more".
  - **\d+** is equivalent to **\d{1,}**

Or Logic

- Use a vertical pipe inside parentheses to declare either/or logic.
- **(\d|\s)** will match either a single digit or a single whitespace.
- We can combine symbols inside the parentheses.
- **(\d|\s{2})** will match either a single digit or 2 whitespaces in a row.

Anchors

- Anchor symbols mark the beginning or end of a string.
- The **$** symbol designates the end of the string.
  - **\d+$** looks for 1 or more digits at the end of a string.
- The **^** symbol designates the start of the string.
  - **^\w** looks for 1 word character at the start of the string.

Capture Groups

- A **capture group** isolates and optionally names a chunk/segment of the regular expression.
- Use **(?<my_name>)** before a symbol to assign it a capture group name.
- Use the **captures** method on the **Regex** struct and pass in the haystack.
- Capture groups are accessible by index position or by the custom names.
- Index 0 will represent the full RegEx match. Subsequent index positions hold the capture groups.

The **replace_all** Method

- The **replace_all** method on the **Regex** struct swaps all pattern matches with an alternate piece of text.
- Use **$name** to access a dynamic capture group value by its name.