Getting started with regular expressions(RegExp)

- `awk` It is a command line utitlity that is used to extract or filter some text that matches a certain pattern. It is one of the most used utility for regular expressions.
  Example:

```bash
$ cat example.txt | awk '/physical/ {print $0}'
The physical properties of iron at very high pressures and temperatu
```

It matches the word physical in the text file and outputes it to the stdout.

Pattern matching techniques

- `^` (Caret): **Match the string that starts with the word.**

Example:

```bash
$ cat example.txt | awk '/^Some/ {print $0}'
Some controversial experimental evidence exists for a stable β phase
```

- `$` (Dollar symbol): **Match the string that ends with the provided word.**
  Example:

```bash
$ cat example.txt | awk '/phase$/ {print $0}'
Some controversial experimental evidence exists for a stable β phase
```

- `[G-K]` (Range): **Match the string that lies between the provided range (Case sensitive)**
  Example:

```bash
$ cat example.txt | awk '/^[A-z]he/ {print $0}'
The first three forms are observed at ordinary pressures.
The physical properties of iron at very high pressures and temperatu
The higher-temperature γ-phase also changes into ε-iron
```

For matching
lower-case letters: `[a-z]`
upper-case letters: `[A-Z]`
upper as well as lower-case letters: `[A-z]`

Ranges can also be used on numbers,
for example,

```Bash
$ cat example.txt | awk '/[0-9]/ {print $0}'
The physical properties of iron at very high pressures and temperatu
```

- `!` (Exclamation/Negation): **Match everything except the one that matches the expression**
  Example:

```Bash
$ cat example.txt | awk '!/IRON/ {print $0}'
As molten iron cools past its freezing point of 1538 °C.
ARON The physical properties of iron at very high pressures and temp
...
```

- `*` (Asterisk): **Check if a character appears once or more than once, or not at all**
  Example:

```Bash
$ cat example.txt | awk '/colou*r/ {print $0}'
color
colour
```

- `|` (Pipe): **Check if the expression contains one or the other character**
  Example:

```Bash
$ cat example.txt | awk '/gre|ay/ {print $0}'
gray
grey
```

- `{}` (Count): **Provide a certain count for a character**
  Example:

```bash
$ cat example.txt | awk '/she{2}sh/ {print $0}'
(meaning e should only appears twice in the word of the expression)
```

- {2}: The character should appear only two times
- {2,4}: The character can appear any number of times between 2 and 5
- {2,}: The character should appear for atleast 2 times
- {,2}: The character should appear for maximum of 2 times
- {,}: The character can appear any number of times

- `.` (Period): **Match any character**
  Example:

```bash
Prints the entire file
$ cat example.txt | awk '/./ {print $0}'

Match any character any number of times
$ cat example.txt | awk '/.*/ {print $0}'
```

Finding and replacing using regular expressions.

sed: `sed` (Streamlined EDitor) is a command line tool that enables us to find and replace strings in a text file in place or output it to the terminal.

```bash
$ cat example.txt | sed -iE 's/the/THE/g'
```

- The `-i` tells sed to edit the file in place. Without this, `sed` simply outputs to the stdout.
- The `-E` stands for Extended Expression
- The `s` stands for substitute
- The `g` stands for globally
- The text between the first and second forward slashes contains the text to replace.
- The text inside the second and third forward slashes contains the replacement
- We can use all the expression that we've learned in `awk` with some exceptions and edge cases.