Agenda

- Regular Expressions
  - Simple pattern matching
  - `.` character
  - `\` character
- Metacharacters
  - Matching digits (and non digits) - \d , \D
  - Matching word (and non-word) characters - \w , \W
  - Matching whitespace characters - \s , \S
- Anchors
  - Word boundary (and non-boundary) anchors - \b , \B
  - Beginning and end anchors - ^ , $
- Character Set
  - Range notation - [a-z] ,[A-Z] , [1-6]
  - Negation - [^a-z]
- Quantifiers
  - *, +, ?, {3}, {3, 4}
- Groups
  - `()` and `|`
- Functions in `re` library

Regex

Regular Expression

- Shorthand notations for:
  - Searching
  - Parsing
  - manipulating

Complex Patterns in String

Q: Why regex ?

- Help in NLP
- Extracting Information from Unstructured Text Data

Business Use-Cases

- Extracting customer info from feedbacks
- Validating Texts such as Email
- Masking PII data such as Phone-numbers, address etc.

Ex: Email Validation

* Note

1. Regex is case sensitive
2. Regex is Order sensitive

Meta Characters

- Special characters that don't match themselves

1. \d : Matches all digits
2. \D : Matches Everything Except digits
3. \w : Matches Alphanumeric and _ (Underscore)
4. \W : Matches Non-Alphanumeric
5. \s : Matches White space characters (space, tab, newline)
6. \S : Matches non White space characters

* Note: Uppercase counterparts of meta characters 'Negate' the Lower case ones

Other Special Characters

```
.       - Any Character Except New Line
\d      - Digit (0-9)
\D      - Not a Digit (0-9)
\w      - Word Character (a-z, A-Z, 0-9, _)
\W      - Not a Word Character
\s      - Whitespace (space, tab, newline)
\S      - Not Whitespace (space, tab, newline)

\b      - Word Boundary
\B      - Not a Word Boundary
^       - Beginning of a String
$       - End of a String

[]      - Matches Characters in brackets
[^ ]    - Matches Characters NOT in brackets
|       - Either Or
( )     - Group
```

meta characters

Anchors

Documentation: https://docs.python.org/3/howto/regex.html

Anchors

* Anchors don't match with any characters
* They match with positions instead

1. \b : Word boundary
   - new-line or space is considered as boundary
2. ^ : Matches only the begining of a string
3. $: Matches only the end of string

Character Set

- Matches for any of the characters defined inside set

Ex [ab]

[.-]

* numeric character set
  - [1 2 3 4 5 6]
  - [1-6]
  - [1-9]

* Alphabet charset

[a-z] [A-Z] [a-zA-Z]

* Negation in character-set

[^a-z] ⇒ match everything but lowercase char

Quantifiers and Groups

Quantifiers : matching more than one char at a time

① ? : 0 or One match

② + : 1 or more

③ * : 0 or more

④ {} → {3} → Exactly 3 times

→ {a,b} → Range of nums (min, max)
{3,4} a b

Groups

(r|s|rs)

re - functions

① match : Checks for a match for input at the beginning of string

② search : Finds only the first Occ. of a given Expression

③ findall : Find all the Occ. of a given Expression

④ finditer : Finds all occ. one by one as an iterable

* For String manipulation

① sub : Used for substitution operation

Ex: re.sub('ab', 'ba', 'ababc')
output: babac

② split : Used for splitting the input string 'into list

re Flags

① re.IGNORECASE or re.I

- Ignore case sensitivity

② re.VERBOSE or re.X

- It allows for spacing indentation and comments while writing Expression
- You can use this when you have highly non-understandable complex pattern.