Regex

* match/check Series of characters (email validation: .com, @, URL)

Char Set

[A-Za-z] : A-Z, a-z

[^a-z] : except a-z

[0-9] : 0-9

Repeating:

[a-z]+ : Atleast one (1/more)

[a-z]{10} : 10 times exactly

[a-z]{5,8} : B/w 5 to 8

[a-z]{5,} : 5 to ∞

Metacharacters:

\d : digits [0-9]

\w : [a-z][A-Z][0-9] -

\s : Any whitespaces except new line (Spaces, tabs)

\t : Tabs only

\ : Escape character

\d{3}\s\w{5} : 3 digits, 1 space, 5 letters

Special characters:

+ : 1/more

\ : Escape character

[] : char set

[^] : Negate Symbols inside char set

? : zero | one

. : Any char (except newline)

* : 0/more

() : group regex

Start with:

^[a-z]{5}

Start & end with: ^[a-z]{5}$

OR:

p|kyre : $\boxed{p}$kyre : matches 'p' only

(p|k)yre: matches p|k then yre.

eg:

(name) @ (domain). (extension)(.again)

boss @ ninja. com$\boxed{.uk}$
↳ optional

^([a-zA-Z0-9\.-]+)@([a-zA-Z\d -]+)\.([a-zA-Z]{2,8})
(\.[a-zA-Z]{2,8})? $

\A(___) : Begin with given char(s)

\b(_) : begin/end with given character(s)

\B(_) : Shouldn't begin/end "

\d : [0-9]

\D : [^0-9]

\s : whitespaces

\S : Non whitespaces

\w : [a-zA-Z0-9 _]

\W : Any non-alpha numeric char

\Z : String ends with given regex

Note: Inside sets

[+] : No special meaning
[+, *, ., |, (), $, {}]

Python functions:

re.findAll() : Return non-overlapping matches : order maintained

re.compile() : Compile to pattern (Search, Substitute)

re.split() : Upon finding split

re.sub() : Replace Content

re.subn() : Tuple with Count of replacement, new String (replaced)

re.escape() : Return String : remove all non-alphanumerics backslashed (match String have regex metachar)

re.search() : Return None / re (more than extracting : used for Testing)!

y' Sticky (Start Searching from 'lastIndex' property)

Flags:

i : Ignore Case

M : multiline

S : make . match newline also

U : make {\w \W \b \B} follow unicode rules

L : " Locale (lang, time, ..) : Not recomm!

A : " ASCII only

X : Allow Comment in regex

DEBUG : Get info about regex Compilation pattern!

g : global

Back Referencing:

* Match string using o/p of regex group 1

(\w+) \1 $\longrightarrow$ hello hello

$$\begin{tikzcd}
\boxed{bye} \arrow[d] & bye \\
\backslash w \arrow[d] & \\
bye \arrow[ruu, "match\ bye!"'] &
\end{tikzcd}$$

Named groups:

(?P<firstName>\w+) (?P<last>\w+)

$\boxed{\text{match.group('firstName')}}$

Swap using regex:

Pattern.sub("\2 \1", txt)

Pattern.sub(r"\g<last> \g<first>", txt)

Same First & last name:

'John John'

(or) [pattern = re.compile ("(\w+) \1")
Pattern = re.compile ("(?P<first>\w+) (?P=first)")

$\boxed{\text{Pattern.match(txt)}}$

match.groupdict() $\longrightarrow$ {'first' : 'John', 'last' : 'John'}

Non Capturing groups:

★ Eg: alternation (Something can't be altered)

'I love (cats|dogs)' $\longrightarrow$ pattern.findAll(txt)

['cats', 'dogs'] $\rightarrow$ returns group matches

case:

I want entire string instead of groups: Not matching groups

$\boxed{\text{(?: pattern)} \longrightarrow \text{Non matching group!}}$

re.compile('I love (?:cats/dogs)')

↳ pattern.findAll(txt)

↳ ['I love dogs', 'I love cats']

---

word boundaries (\b):

\bgreen\b

'red green blue' $\longrightarrow$ ['green']

Zero-width assertions: characters indicating positions (^, $) rather than content (like assertion)

Powerful zero-width assertion: Look around

Look ahead

Look behind

Look around: Contain regex before & after a word/regex!

Look around (2 parts)

↙ Actual expression (final result)

↘ Non-consuming expression (Just to check position like can succeed: start/end)

↓ doesn't match: move to next char & do searching matching again.

$$\begin{tikzcd}
 & & \text{look around} \arrow[dll] \arrow[drr] & & \\
\text{look behind} \arrow[d] \arrow[dr] & & & & \text{look ahead} \arrow[dl] \arrow[d] \\
\text{+ve look behind} & \text{-ve look behind} & & \text{+ve look ahead} & \text{-ve look ahead} \\
(?<=B)A & (?<!B)A & & A(?=B) & A(?!B)
\end{tikzcd}$$

+ve look ahead:

* check for pattern ahead of a regex expression

A(?=B)
↓ Actual exp
└→ Non consuming exp!

eg 'love regex'
↓
'love regex' → 'love regex'
'love(?=\sregex)' → 'love'
pattern.span() → (0,4)

eg:
'my favourite colors are red, green, and blue'

All words followed by comma (or) fullstop:

\w+(?=[,.])
\w+(?=,|\.)

Negative look ahead:

* match only not followed by a regex
eg: 'love' not followed by 'regex'

eg. 'I love physics, I love regex'
'love(?!\sregex)' → Span: (2,6)

$\boxed{\text{'Same with look behind'}}$

Starts with 'b' followed by vowels & ends at a 't'

/b.b[aeiou]t \b ⟶ battle
match

|A| |b|o|u|n|d|a|r|y| \b
\b \b \b

\b : word boundary

$\boxed{\text{\B : Non word boundary}}$: opposite of \b

^A\b.*\.$ ⟶ Start with A
word boundary
Any letter : any length
end with .

eg: A apple post. ⟶ match
An apple post ⟶ No match
↳ Not a /b

Greedy: Keep matching
Not greedy: \[.*?\]
↳ Not greedy

greedy: match until ']' no longer available.

[Google] (____url____) [test]
[ibm] (____url____)

\[.*\] ⟶ '[Google](url)[test]'
\[.*?\] ⟶ '[Google]', '[test]'

Group within groups:

(cat(\d\d)+)+ ⟶ Catcat30cat5084
Cat54Cat30

Note: Subgroups : tedious task (make sure of detail such as parenthesis, a minute change : different result.)

'The Code is 45cat7814Cat' $\longrightarrow$ 45cat7814 cat

e " 45cat_7814catcat' $\longrightarrow$ match nothing

?: and ?= (confusion)

★ whenever ?= (equal) we are talking about assertion.