

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 newline (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 provide 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 uk
↳ optional

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

$\backslash A(_)$: Begin with given char(s)

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

$\backslash B(_)$: shouldn't begin/end "

$\backslash d$: $[0-9]$

$\backslash D$: $[\^0-9]$

$\backslash s$: whitespaces

$\backslash S$: Non whitespaces

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

$\backslash W$: Any non-alpha numeric char

$\backslash Z$: string ends with given regex

Note: Inside \backslash s

$[+]$: No special meaning

$[+, *, \cdot, |, (), \$, \{, \}]$

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!)

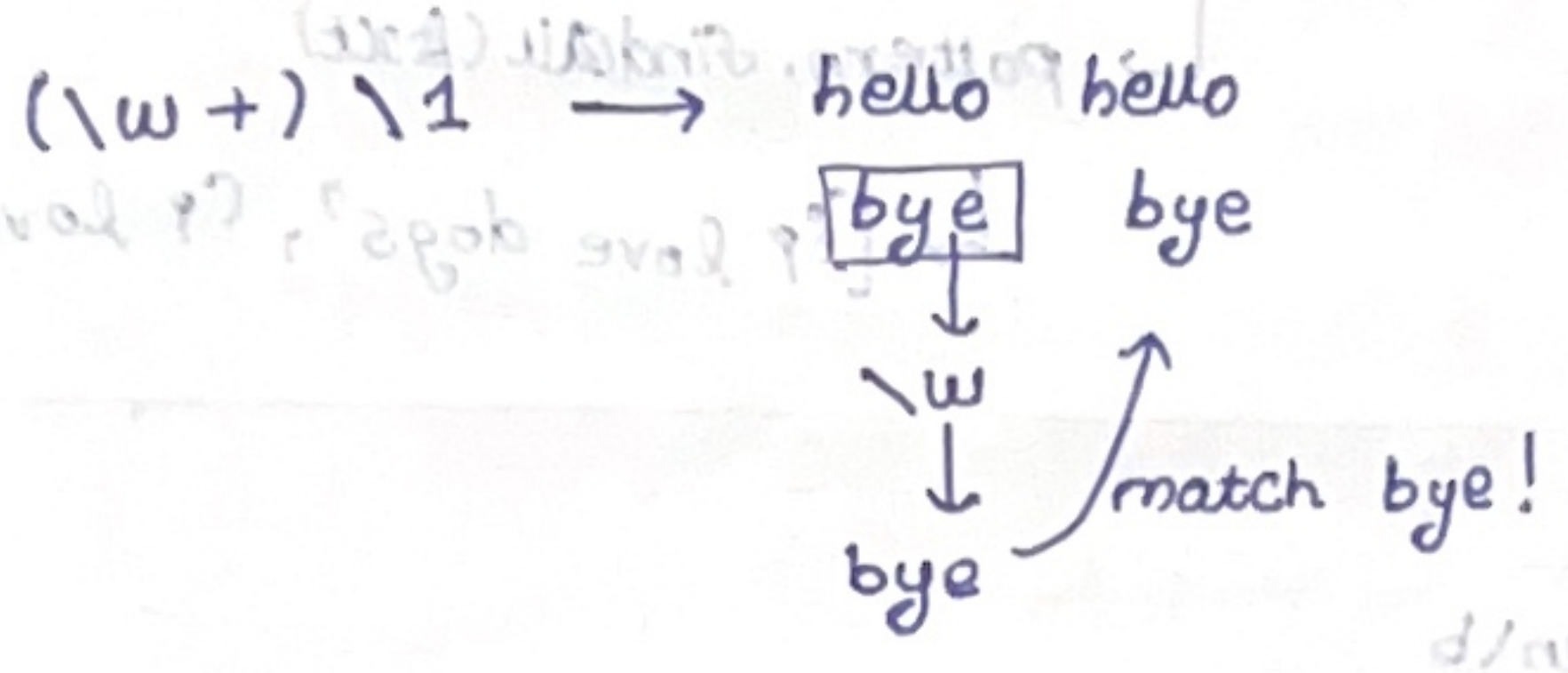
Stick (Stick Searching from 'lastIndex' property)

Flags:

- i: Ignore Case
- m: multiline
- S: make . match newline also
- U: make $\{\backslash w \backslash W \backslash b \backslash B\}$ follow unicode rules
- L: " Locale (lang, time, ...): Not reComm!
- A: " ASCII only
- x: Allow Comments in regex
- DEBUG: Get info about regex compilation pattern!
- g: global

Back Referencing:

* Match string using o/p of regex group 1



Named groups:

$(?P<first Name> \backslash w +) (?P<last> \backslash w +)$

`match.group('first Name')`

Swap using regex:

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

`Pattern.sub("^(?P<last> \w+)(?P<first> \w+)", txt)`

Same First & last name:

"John John"

(or) $\left[\begin{array}{l} \text{Pattern} = \text{re.compile}("(\backslash w +) \backslash 1") \\ \text{Pattern} = \text{re.compile}("(?P<first> \w+) (?P=first)") \end{array} \right.$

`Pattern.match(txt)`

`match.groupdict() → {'first': 'John', 'last': 'John'}`

Non Capturing groups:

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

'I love (cats|dogs)' → pattern.findall(txt)

['cats', 'dogs'] → returns group matches

case:

I want entire string instead of groups: Non matching groups

(?: pattern) → 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' → ['green']

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

Powerful zero-width assertions: Look around
Look ahead
Look behind

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

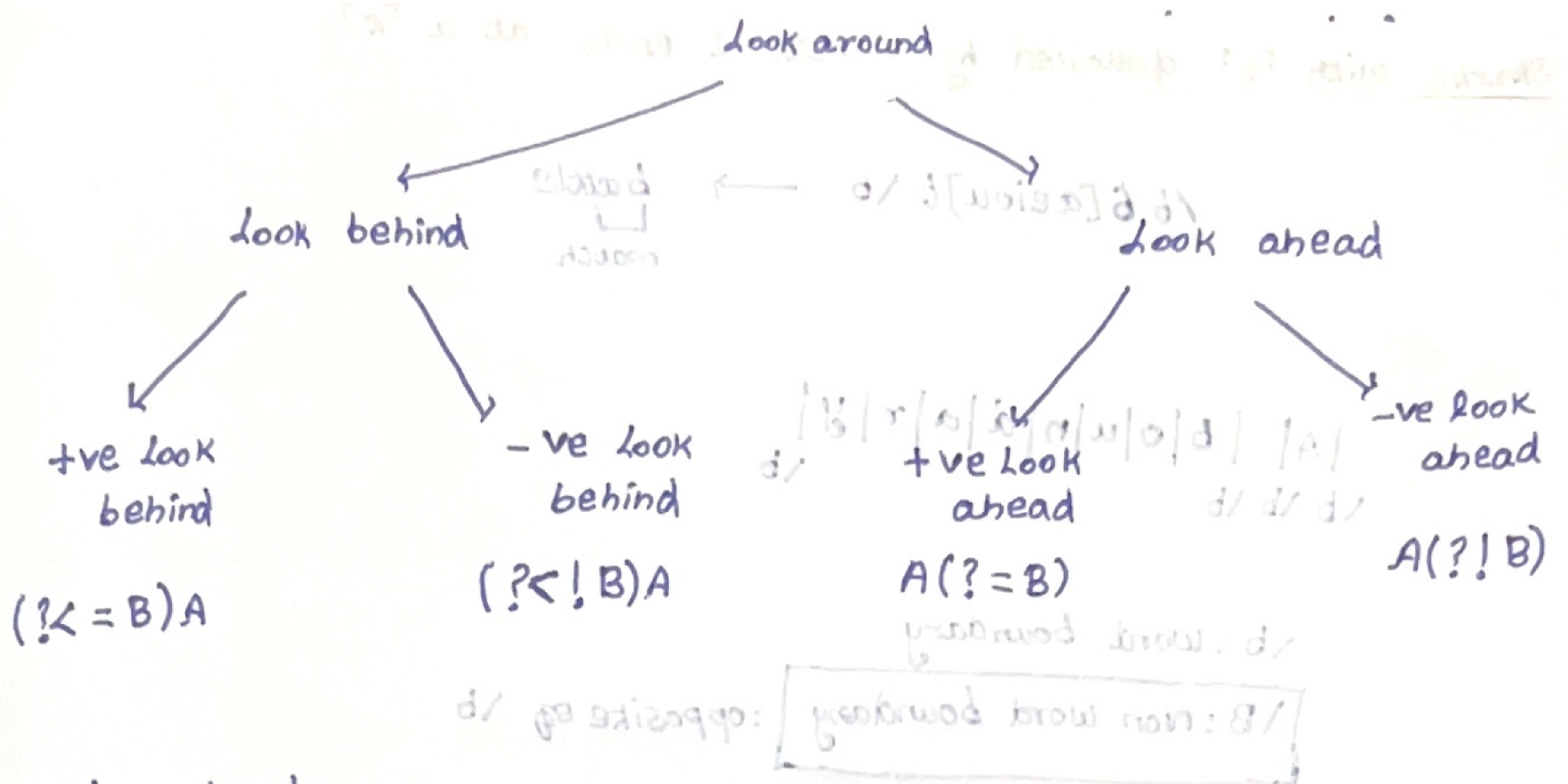
Look around (2 parts)

Actual expression
(final result)

Non-consuming expression

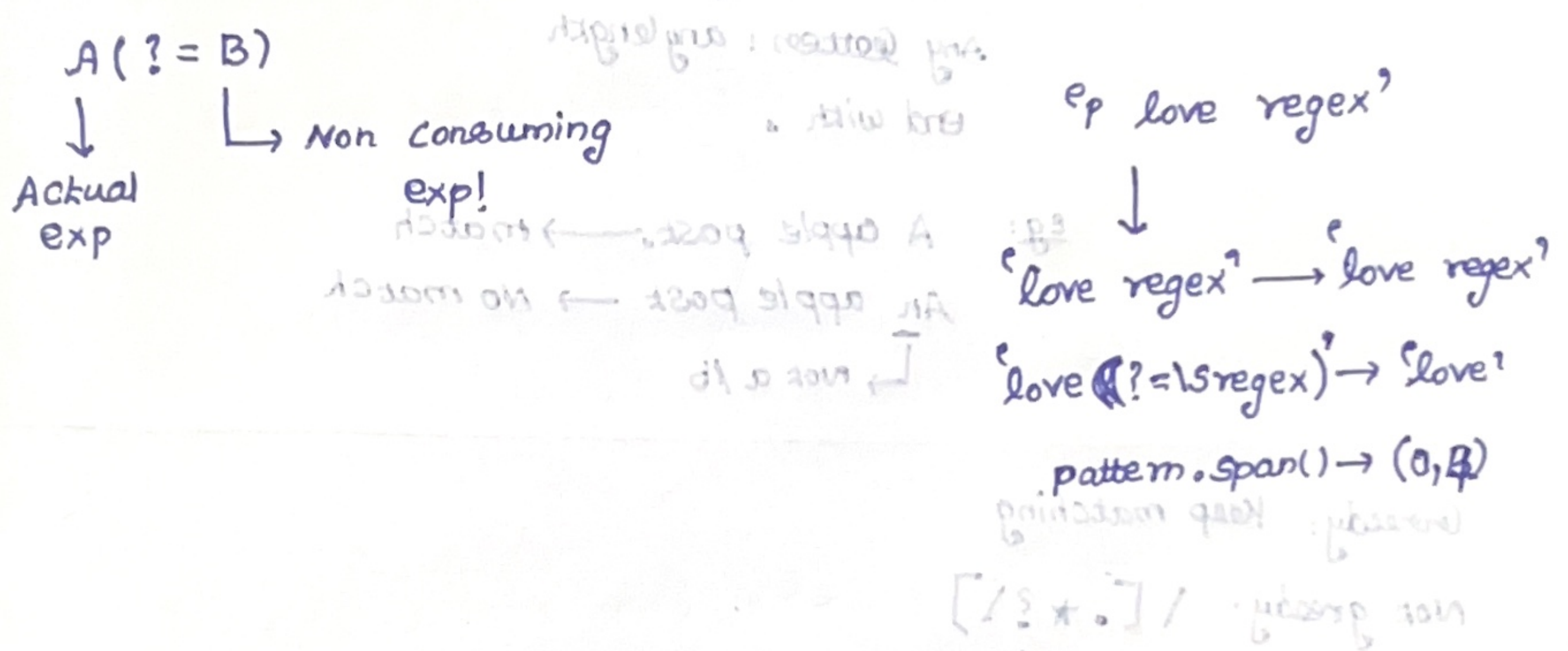
(Just to check position like can succeed: start/end)

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



+ve Look ahead:

* Check for pattern ahead of a regex expression



eg: "My favourite colors are red, green, and blue?"

All words followed by comma (,) or fullstop.

$\backslash w + (?=[, .])$

$\backslash w + (?=[, .])$

Negative look ahead:

* match only not followed by a regex

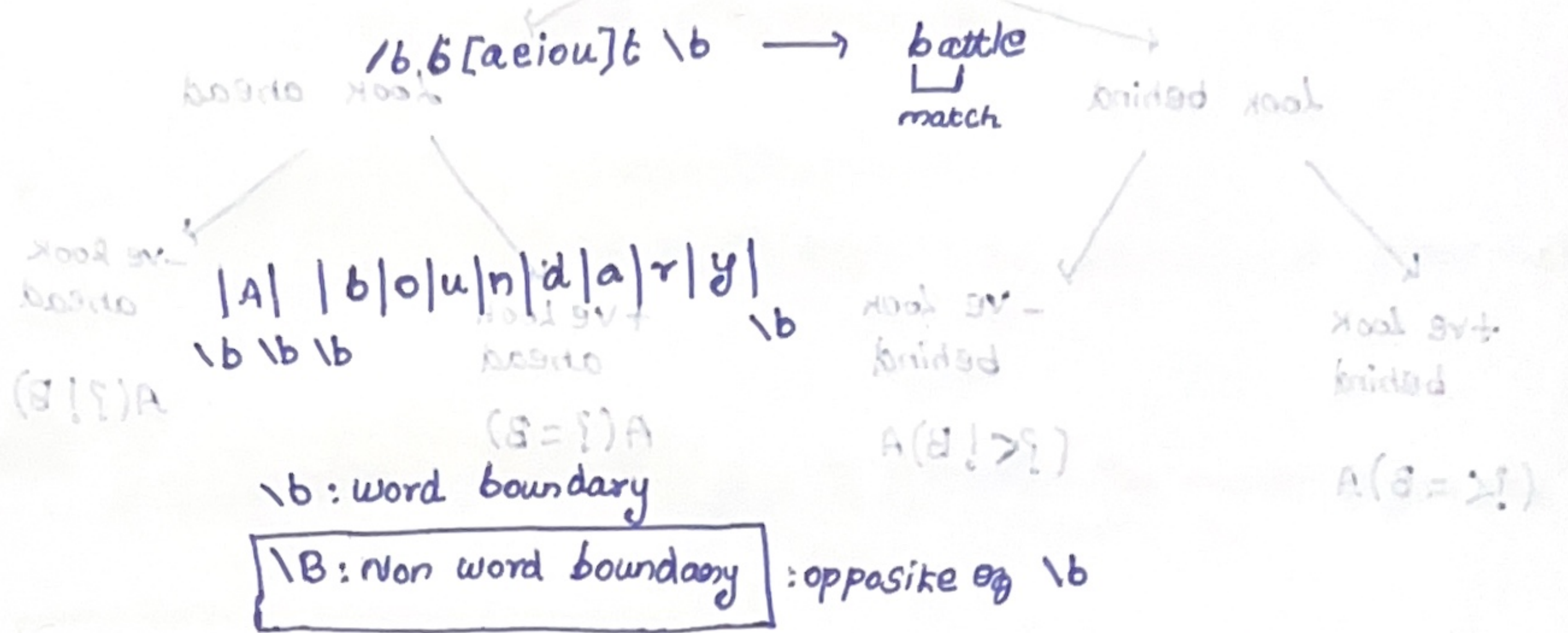
eg: 'love' not followed by 'regex'

eg: 'love physics, I love regex'

'love (?!regex)' → Span: (2, 6)

Same with look behind

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



`^A\b.*\.$` → Starts with A, word boundary, Any pattern: 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]?` → `Google (url) test`
`[?bm] (url) (.*?)` → `bm (url) .*`

`\[.*\]` → `'[Google](url)[test]'`

`\[.*?\]` → `'[Google]', '[test]'`

Group within groups:

`(cat(\d\d)+)+` → `CatCat30Cat5084`
`Cat54Cat30`

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

'The Code Is 45cat78/4cat' \rightarrow 45cat78/4cat

" 45cat_78/4catcat' \rightarrow match nothing

? : and ? = (confusion)

* whenever ? = (equal) we are talking about assermen.