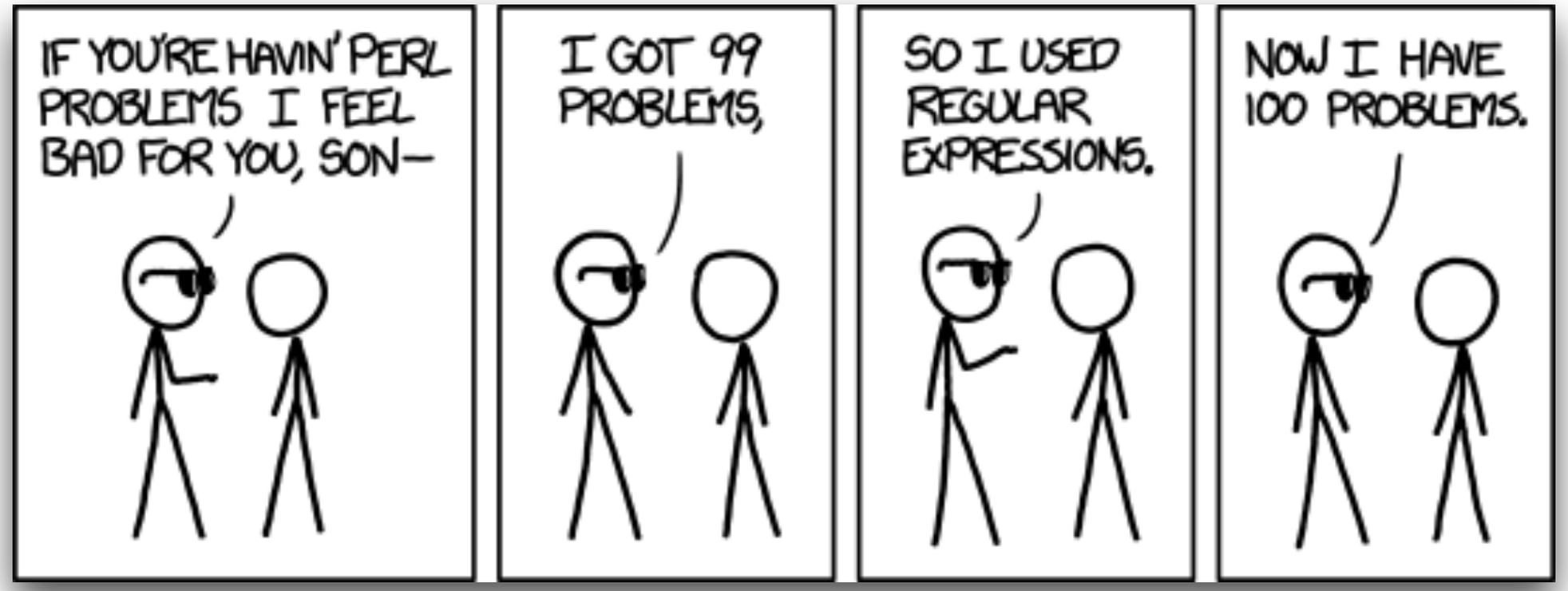
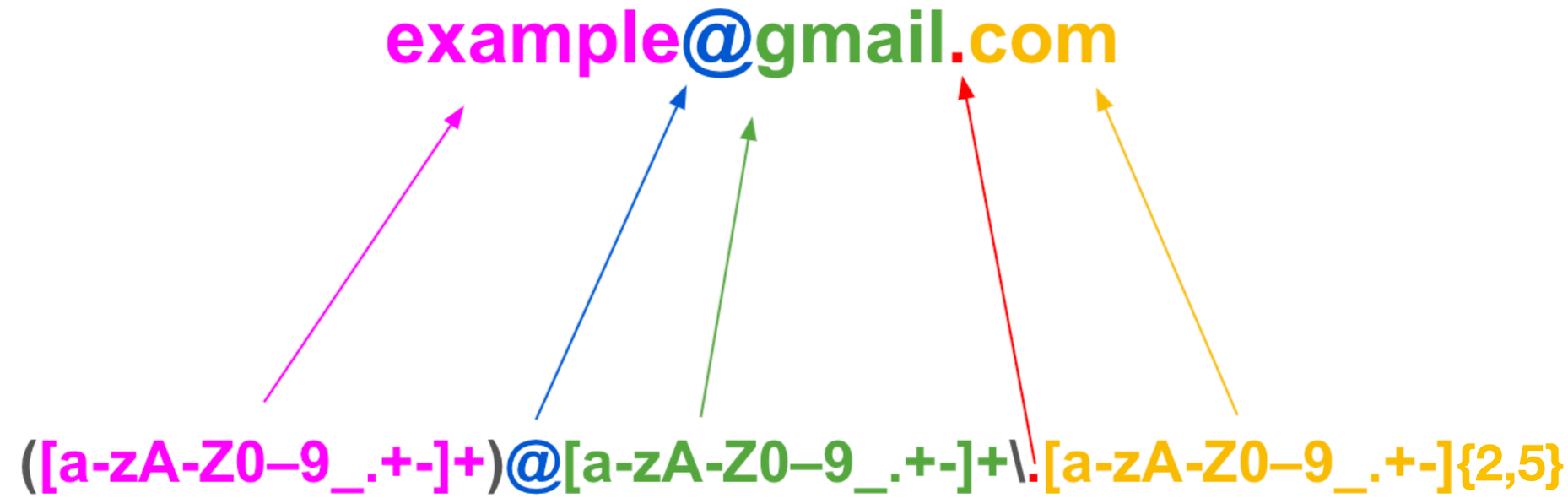
IN1140 – Uke 5

Regulære uttrykk
Ukesoppgaver
Oblighjelp



Regulære uttrykk



Regex

Kunsten å kaste nettet akkurat stort nok

Deler av et regulært uttrykk

- Start- og sluttmarkører: / og /
- Strenger av tegn: /Det var en gang/
    - Oppfører seg som forventa -> Matcher «Det var en gang» i tekst
- Grupper: (i parenteser)
    - Lar oss gruppere deler av uttrykket vårt:

Deler av et regulært uttrykk

- Kvantorer {antall ganger}:
  - ? – 0 eller 1 /bei?n/ –> ben eller bein
  - * – 0 eller flere /bei*n/ –> ben, bein, beiin, …
  - + – 1 eller flere /bei+n/ –> bein, beiin, beiiin, …
  - {n} – n antall ganger /Yes!{4}/ –> Yes!!!!
  - {n, m} – mellom n og m antall ganger /(ha){1, 3}/ –> ha, haha, hahaha
  - {n, } – minst n antall ganger /(ha){3, }/ –> hahaha, hahahaha, …

Deler av et regulært uttrykk

- Disjunksjoner (enten | eller):
  - Eller-operatoren: a | b — /ma|øt/ –> mat eller møt
  - Velg en: [abcde] — /oblig1[ab]/ – oblig1a, oblig1b
    - Velg en i intervallet: [A-E] — /[1-3] år/ – 1 år, 2 år, 3 år

^ og \

- ^ er enten:
    - startmarkør (av en streng)
    - [^abc] – ikke a, b eller c
- \ – gjør det vanlige spesielt og det spesielle vanlig
    - \. —> matcher tegnet .
        - Ellers er . et wildcard! Det matcher hva som helst!
    - \b —> word boundary
    - \w —> matcher "alle" alfanumeriske tegn
        - Kortform for [a-zA-Z0-9_] (i Python også æ, ø, å osv.)
        - Stor bokstav er det motsatte av liten! Så \W er alt \w ikke er, og motsatt

Deler av et regulært uttrykk

- Masse masse mer!

REGEX CHEAT SHEET

Anchors

^	Start of string, or start of line in multi-line pattern
\A	Start of string
$	End of string, or end of line in multi-line pattern
\Z	End of string
\b	Word boundary
\B	Not word boundary
\<	Start of word
\>	End of word

Character Classes

\c	Control character
\s	White space
\S	Not white space
\d	Digit
\D	Not digit
\w	Word
\W	Not word
\x	Hexadecimal digit
\O	Octal digit

POSIX

[:upper:]	Upper case letters
[:lower:]	Lower case letters
[:alpha:]	All letters
[:alnum:]	Digits and letters
[:digit:]	Digits
[:xdigit:]	Hexadecimal digits
[:punct:]	Punctuation
[:blank:]	Space and tab
[:space:]	Blank characters
[:cntrl:]	Control characters
[:graph:]	Printed characters
[:print:]	Printed characters and spaces
[:word:]	Digits, letters and underscore

Quantifiers

*	0 or more	{3}	Exactly 3
+	1 or more	{3,}	3 or more
?	0 or 1	{3,5}	3, 4 or 5

Add a ? to a quantifier to make it ungreedy.

Escape Sequences

\	Escape following character
\Q	Begin literal sequence
\E	End literal sequence

"Escaping" is a way of treating characters which have a special meaning in regular expressions literally, rather than as special characters.

Common Metacharacters

^	[	.	$
{	*	(	\
+	)	|	?
<	>		

The escape character is usually \

Special Characters

\n	New line
\r	Carriage return
\t	Tab
\v	Vertical tab
\f	Form feed
\xxx	Octal character xxx
\xhh	Hex character hh

Groups and Ranges

.	Any character except new line (\n)
(a|b)	a or b
(...)	Group
(?:...)	Passive (non-capturing) group
[abc]	Range (a or b or c)
[^abc]	Not (a or b or c)
[a-q]	Lower case letter from a to q
[A-Q]	Upper case letter from A to Q
[0-7]	Digit from 0 to 7
\x	Group/subpattern number "x"

Ranges are inclusive.

Pattern Modifiers

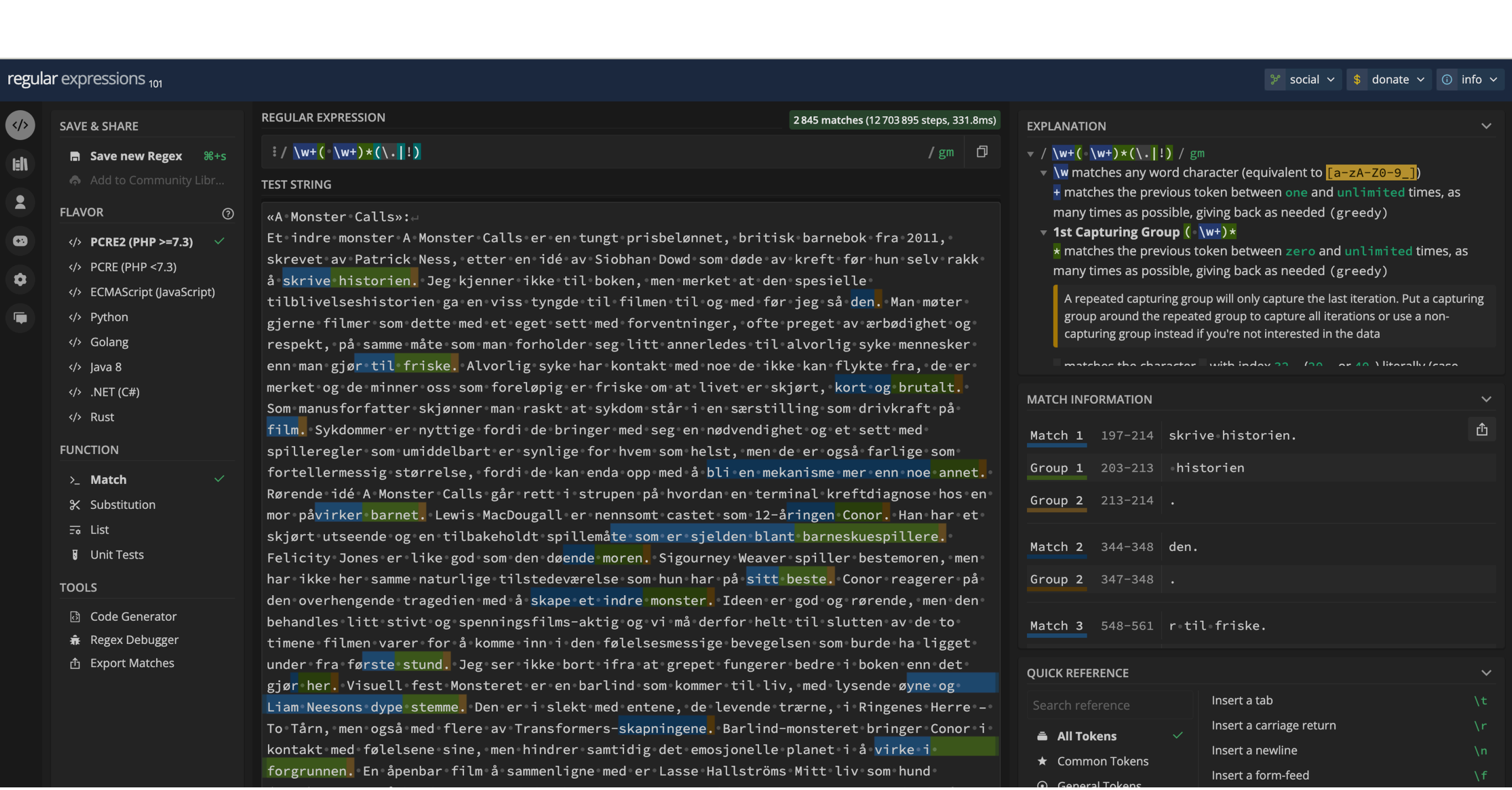
g	Global match
i *	Case-insensitive
m *	Multiple lines
s *	Treat string as single line
x *	Allow comments and whitespace in pattern
e *	Evaluate replacement
U *	Ungreedy pattern

* PCRE modifier

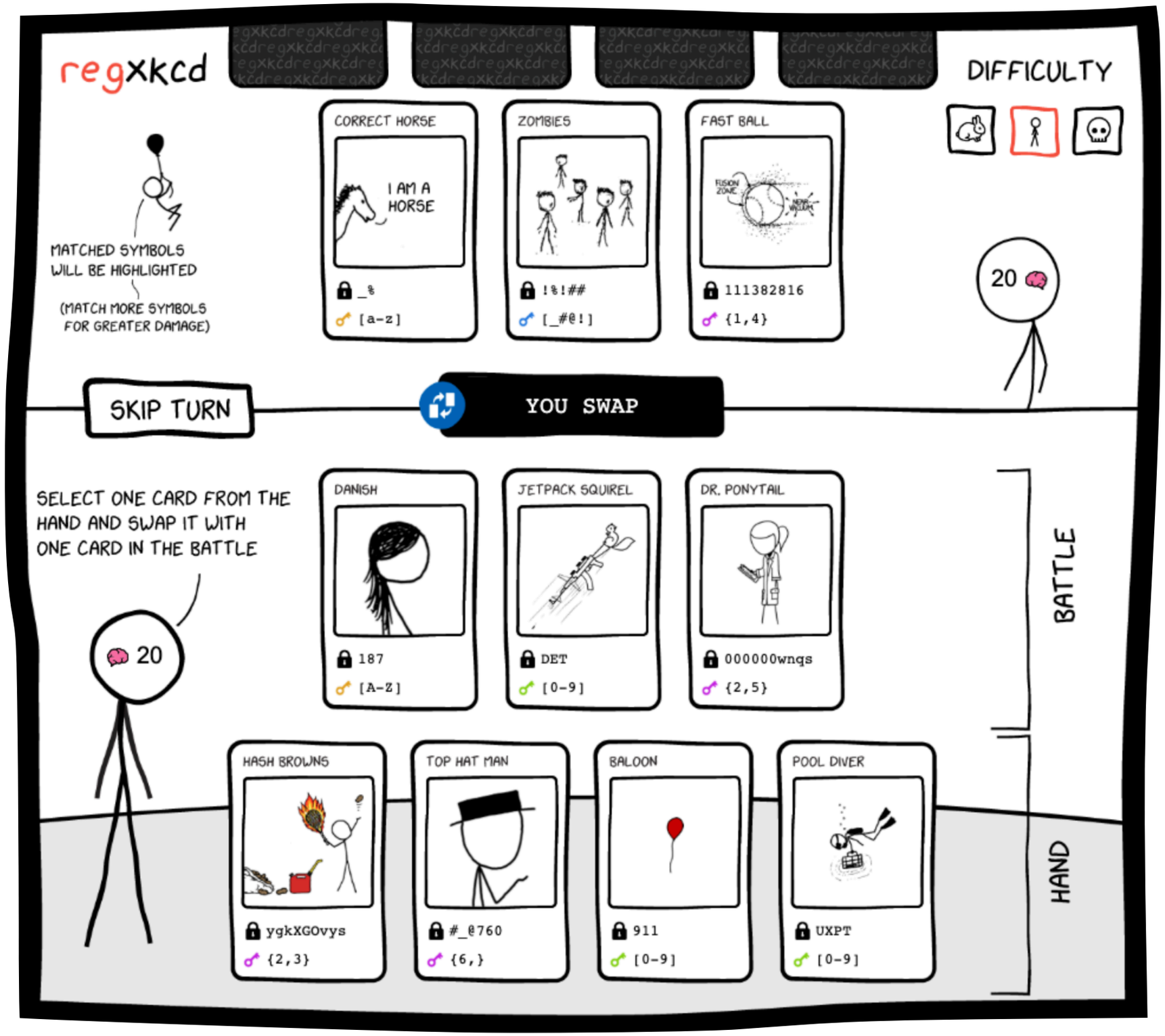
String Replacement

$n	nth non-passive group
$2	"xyz" in /^(abc(xyz))$/
$1	"xyz" in /^(?:abc)(xyz)$/
$`	Before matched string
$'	After matched string
$+	Last matched string
$&	Entire matched string

Some regex implementations use \ instead of $.



regex101.com



mekagem.itch.io/regxkcd

`hello`	
`gray|grey`	
`gr(a|e)y`	
`gr[ae]y`	
`b[aeiou]bble`	
`[b-chm-pP]at|ot`	
`colou?r`	
`rege(x(es)?|xps?)`	
`go*gle`	
`go+gle`	
`g(oog)+le`	
`z{3}`	
`z{3,6}`	
`z{3,}`	
`^dog`	
`dog$`	
`^dog$`	

Regulære uttrykk i Python

```python
import re
import regex
```

```python
regulært_uttrykk = r"[A-Z][a-z]+" # Ingen start og slutt
```

```python
liste_med_matcher = re.findall(regulært_uttrykk, tekst)
```

```python
splitta_tekst = re.split(regulært_uttrykk, tekst)
```

```python
streng_med_bytte = re.sub(regulært_uttrykk, "bytt til dette", tekst)
```

Regulære uttrykk i Python

```
import re

regulært_uttrykk = r"(\w+(\.\w+)*@(\w+)\.uio\.no)"
print(re.findall(regulært_uttrykk, tekst))
>>> [('haakonhy@ifi.uio.no', '', 'ifi')]
```

Vi kan unngå denne fordelinga med spesialtegnet ?:

```
regulært_uttrykk = r"(\w+(?:\.\w+)*@(?:\w+)\.uio\.no)"
print(re.findall(regulært_uttrykk, tekst))
>>> ['haakonhy@ifi.uio.no']
```