IN1140 – Uke 4

Oblig 1b
Regulære uttrykk
Ukesoppgaver
Oblighjelp

Oblig 1b er ute!

Obligatorisk innlevering 1b - Regulære uttrykk &
tokenisering med regulære uttrykk i Python

Høst 2023

Husk å lese gjennom tilbakemeldingen din på forrige innlevering i Devilry før du begynner på denne.

Innleveringsfrist: fredag 29. september kl. 23.59

Innlevering av oppgaven skjer i Devilry. Det er viktig at du setter deg inn i retningslinjene for obligatoriske oppgaver. Besvar oppgavene i én fil som angir brukernavnet ditt slik:

`oblig1b_brukernavn.py`

Du skal også levere inn filen `in01.txt` som du fikk utlevert med forrige innlevering. Den skal brukes i denne innleveringen også. Ikke pakk filene inn i en ZIP-mappe. Besvarelsen må kunne kjøres med Python i terminalen, slik som tidligere.

Poeng

En perfekt løsning av denne innleveringen er verdt 100 poeng. Koden må kunne kjøres og må inneholde kommentarer som forklarer hva koden gjør. Løsningen på oppgave 1, 2, 3 og 4.1 skal skrives som kommentarer i Python-filen din. Du trenger kun å programmere i oppgave 4.2 og 4.3.

1 Regulære uttrykk (eksamen høsten 2018) (10 poeng)

Hvilket av alternativene kan ikke gjenkjennes med det følgende regulære uttrykket:

```
[1-9][0-9]*\s((cent(s)?)|(dollar(s)?\s+([1-9][0-9]*\scent(s)?)))
```

Velg ett alternativ:

1. `1 dollar 35 cents`
2. `35 dollars`
3. `99 cents`
4. `99 dollars 1 cent`

2 Regulært uttrykk for verb (eksamen høsten 2017) (20 poeng)

Skriv ett regulært uttrykk som gjenkjenner formene imperativ, infinitiv, presens og preteritum av følgende norske verb: *spise*, *kjøpe*, *tenke*, *rope*. Du skal skrive ett enkelt uttrykk som kun gjenkjenner verbene med formene spesifisert. Bruk tabellen som referanse.

Infinitiv	Imperativ	Presens	Preteritum
spise	spis	spiser	spiste
kjøpe	kjøp	kjøper	kjøpte
tenke	tenk	tenker	tenkte
rope	rop	roper	ropte

3 Regulært utrykk for filnavn (20 poeng)

Skriv ett regulært uttrykk som gjenkjenner filnavn som ender på:

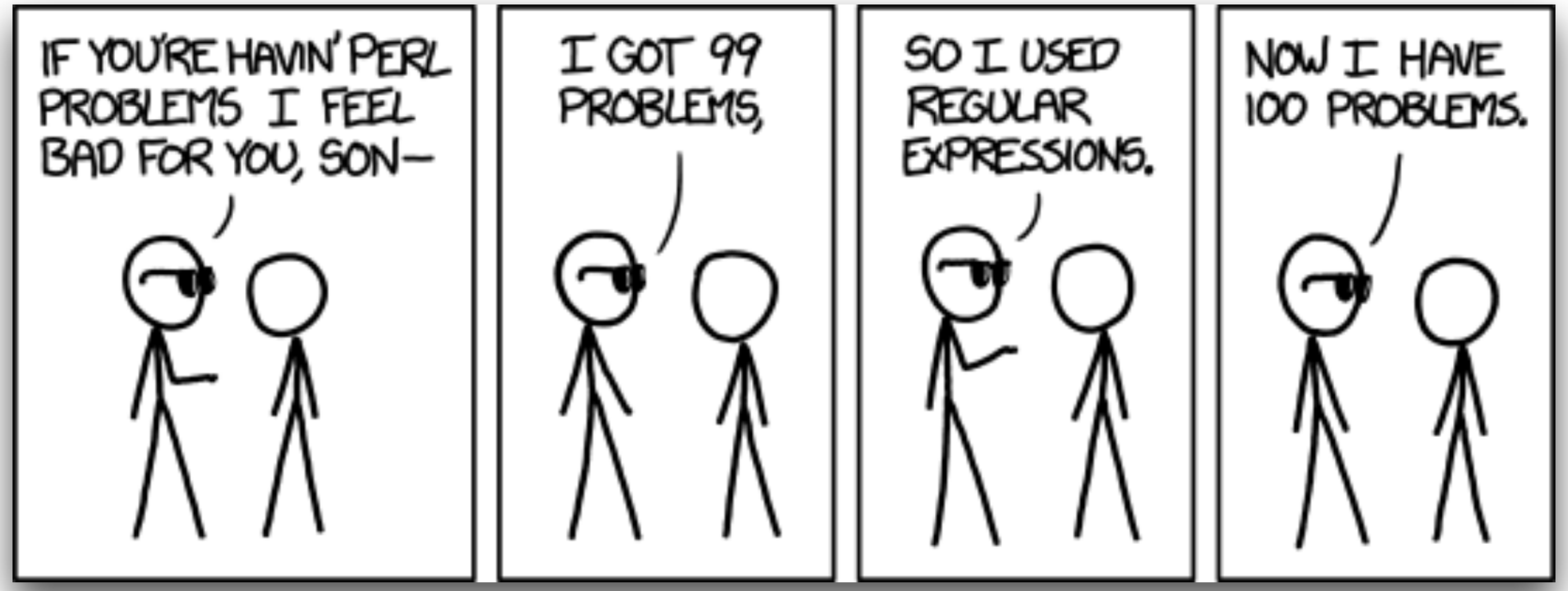
- .gif
- .jpg
- .png
- .jpeg
- .pdf
- .rtf

Husk at selve navnet også må gjenkjennes (altså f.eks. `utgave_3.pdf` og `Møre_og_Romsdal.jpeg`), ikke bare endelsen alene. Du kan anta at navnet kun kan bestå av bokstaver, tall og enkel tegnsetting som understrek og bindestrek, og at det må bestå av minst ett tegn.

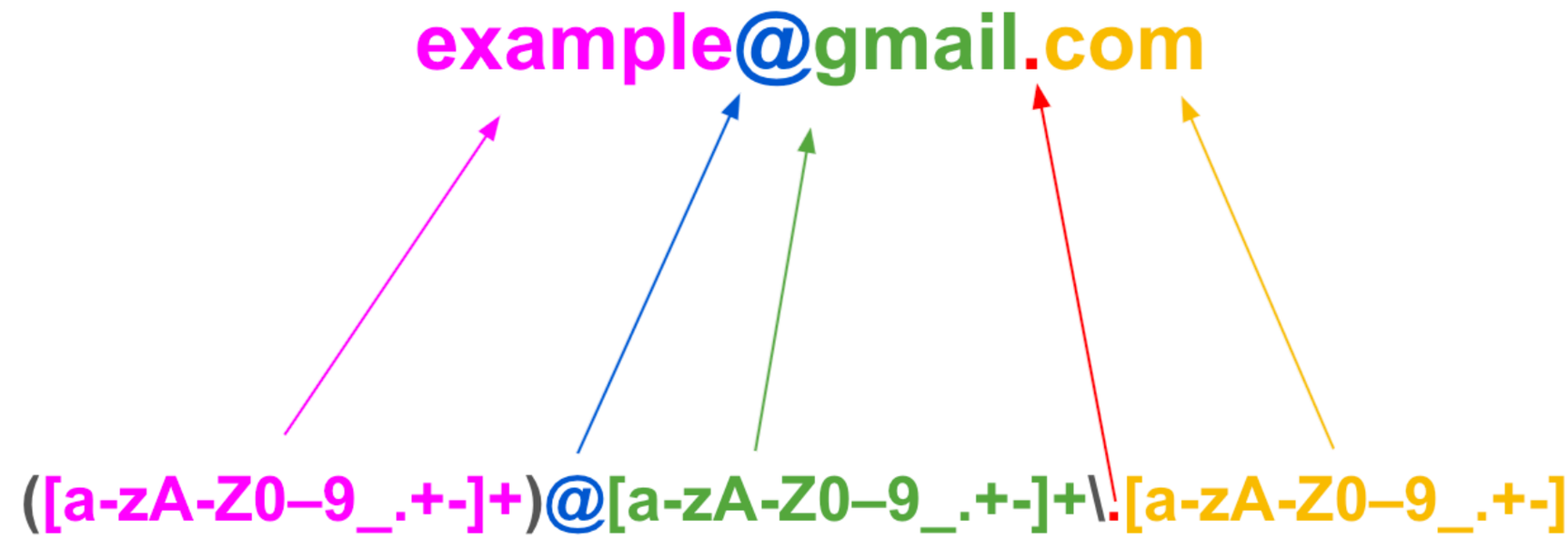
4 Tokenisering med regulære uttrykk (50 poeng)

I denne oppgaven skal vi jobbe med å forbedre tokeniseringen fra **innlevering 1a**, basert på feilanalysen fra oppgave 5 i innlevering 1a. Du skal her forsøke å rette noen av de feilene du fant under feilanalysen din.

1. Vi starter med å formulere et regulært uttrykk som definerer et gyldig ord. Her bør du bruke disjunksjon for å beskrive forskjellige typer ord. For eksempel, bør det regulære uttrykket ditt skille mellom bokstavsekvenser og tegnsetting, da et gyldig ord ikke inkluderer tegnsetting, og ulike typer tegnsetting skal skilles ut som egne token. Eksempelvis bør *"Løp!"* tokeniseres som `Løp` og `!` hver for seg. Kommenter kort hvordan uttrykket ditt fungerer.
2. Benytt deg av `re.findall()` for å hente ut ordene som gjenkjennes av uttrykket ditt, fra teksten i filen `in01.txt`. Denne metoden tar inn et regulært uttrykk og en streng, og returnerer en liste med alle treff. F.eks. vil `re.findall('[0-9]','in1140')` returnere listen `['1', '1', '4', '0']`. Bruk det regulære uttrykket du lagde i **oppgave 4.1**. Du kan definere det i koden din slik: `ru = r'DITT UTTRYKK HER'`
3. Skriv ut tokeniseringen til en fil slik som i **innlevering 1a**, og inspiser resultatet av den forbedrede tokeniseringen. Gjenstår det noen feil? Gi i så fall noen eksempler.



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
- Disjunksjoner (enten | eller):
  - Eller-operatoren: a | b
  - Velg en: [abcde]
    - Velg en i intervallet: [A-E]

Deler av et regulært uttrykk

- Kvantorer {antall ganger}:
  - ? – 0 eller 1
  - * – 0 eller flere
  - + – 1 eller flere
  - {n} – n antall ganger
  - {n, m} – mellom n og m antall ganger
  - {n, } – minst n antall ganger

Deler av et regulært uttrykk

- Grupper (alt inni parentesen)
  - (mat) —> «mat»
  - (ma|at) —> «ma» eller «at»
  - ([mat]) —> «m», «a» eller «t»

^ og \

- ^ – startmarkør (av en streng)
  - [^abc] – ikke a, b eller c
- \ – gjør det vanlige spesielt og det spesielle vanlig
  - \. —> matcher tegnet .
  - \b —> word boundary
  - \w —> matcher "alle" alfanumeriske tegn
    - Kortform for [a-zA-Z0-9_] (i Python også æ, ø, å osv.)

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

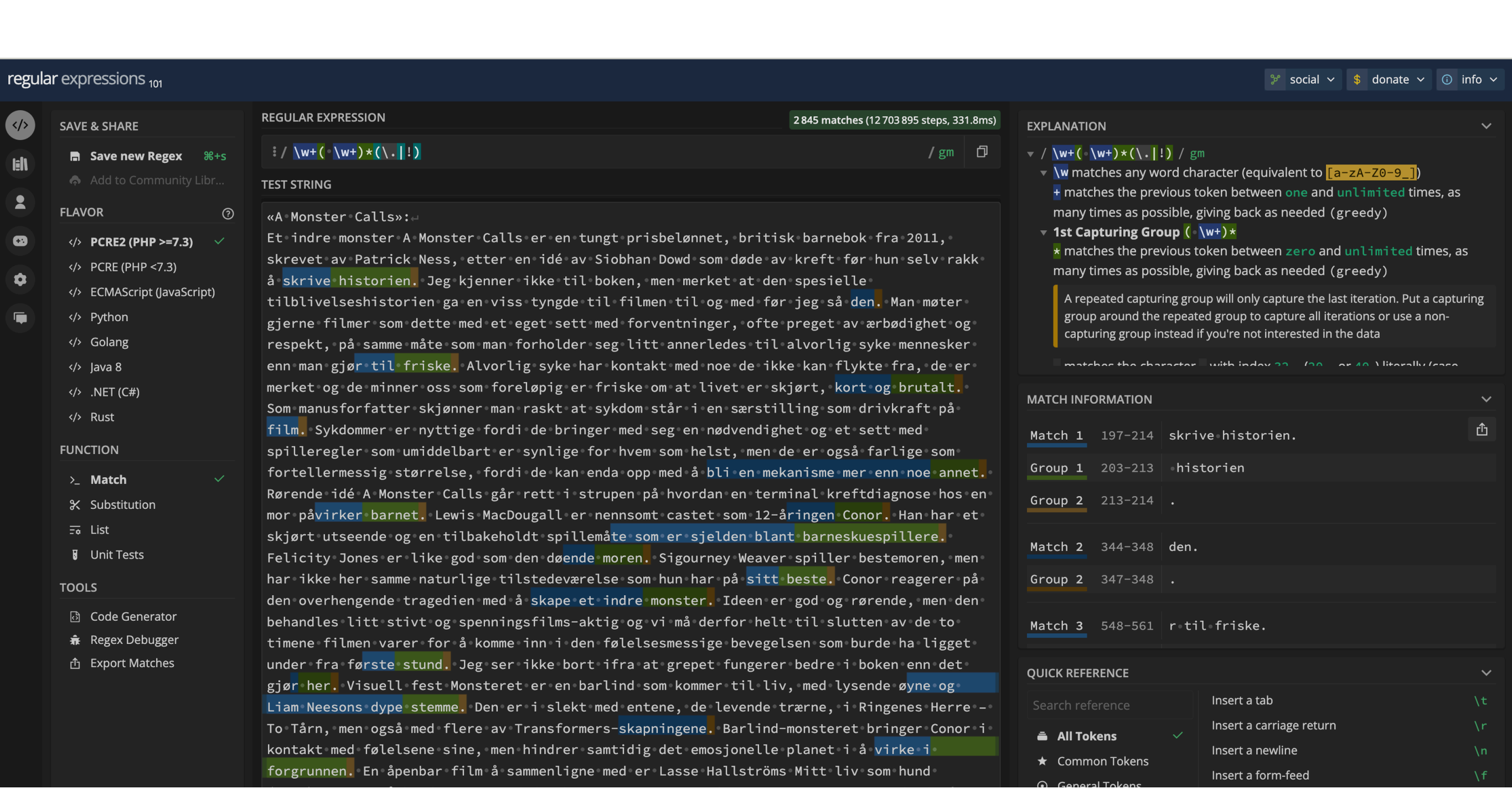
g	Global match
i *	Case-insensitive
m *	Multiple lines
s *	Treat string as single line
x *	Allow comments and whitespace in pattern
e *	Evaluate replacement
U *	Ungreedy pattern

* PCRE modifier

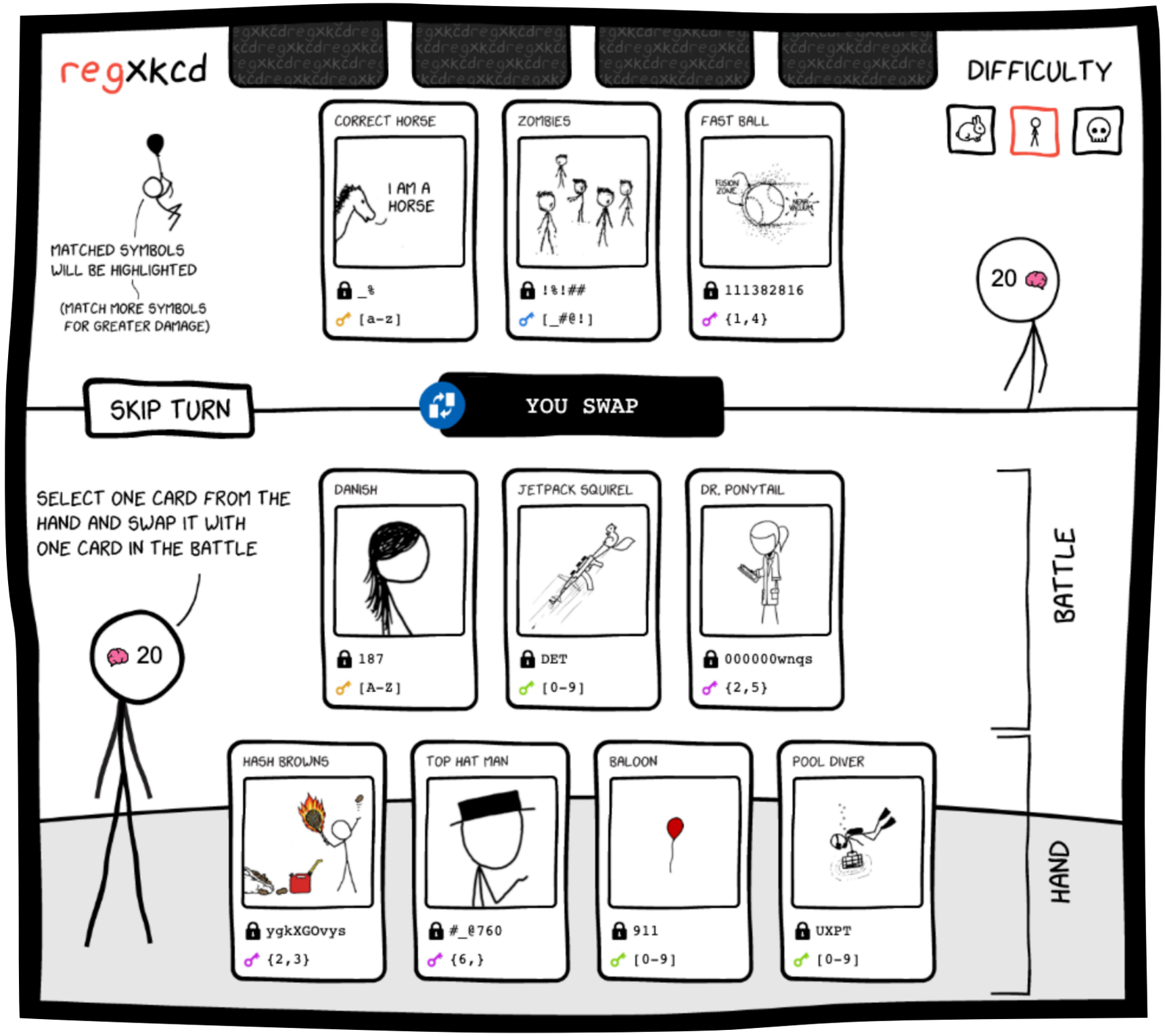
String Replacement

$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`	contains {hello}
`gray|grey`	contains {gray, grey}
`gr(a|e)y`	contains {gray, grey}
`gr[ae]y`	contains {gray, grey}
`b[aeiou]bble`	contains {babble, bebble, bibble, bobble, bubble}
`[b-chm-pP]at|ot`	contains {bat, cat, hat, mat, nat, oat, pat, Pat, ot}
`colou?r`	contains {color, colour}
`rege(x(es)?|xps?)`	contains {regex, regexes, regexp, regexps}
`go*gle`	contains {ggle, gogle, google, gooogle, goooogle, ...}
`go+gle`	contains {gogle, google, gooogle, goooogle, ...}
`g(oog)+le`	contains {google, googoogle, googoogoogle, googoogoogoogle, ...}
`z{3}`	contains {zzz}
`z{3,6}`	contains {zzz, zzzz, zzzzz, zzzzzz}
`z{3,}`	contains {zzz, zzzz, zzzzz, ...}

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']
```

Neste uke

- Sannsynlighetsregning fra [A-ZÆØÅ] (det dere trenger i hvert fall)

- Oblighjelp

LI:ST – Generalforsamling

