22. september 2022

IN1140 Gruppe 3

Denne økta

Gjennomgang av regulære uttrykk

Regulære uttrykk i Python

Ukesoppgaver

Gjennomgang av oblig 1b

Arbeid og hjelp



Regulære uttrykk

Beskrivelse av en mengde strenger

Har ikke språklig innhold, men refererer til lingvistiske enheter

Kan anvendes i en rekke programmeringsspråk og digitale verktøy

`[Hh]ei`	`IN\d{4}`	`\d{2}:\d{2}`
Hei	IN1140	14:15
hei	IN2110	01:30
	IN3050	72:90

Spesialbokstaver

Det regulære uttrykket `[A-Za-z]+` gjenkjenner ord som ...

hei, Ola, Kari, skole, matpakke

... men ikke ...

br**ø**dskive, Gaustadall**é**en, nytt**å**r, b**æ**r, gr**ü**nder, se**ñ**or

Løsning: `[A-ZÆØÅa-zæøåéüñ]+`

... eller `\w+`

(fanger også opp ord med tall)

Prøveoppgave

Skriv et regulært uttrykk som gjenkjenner ordene ...

kaffekopp, tekopp, lattekopp,
kaffekoppen, tekoppen, lattekoppen

Løsning: `(kaffe|te|latte)kopp(en)?`

`(kaff|t|latt)ekopp(en)?`

Regulære uttrykk bør være så spesifikke som mulig, slik at vi ikke gjenkjenner strenger som vi ikke er interesserte i.

Prøveoppgave

Løsning fra forrige side: `(kaffe|te|latte)kopp(en)?`

Hvorfor ikke `(kaffe|te|latte)koppen?` ?

`?` påvirker kun `n` i det andre uttrykket. Dermed kan vi ende opp med følgende uønskede treff:

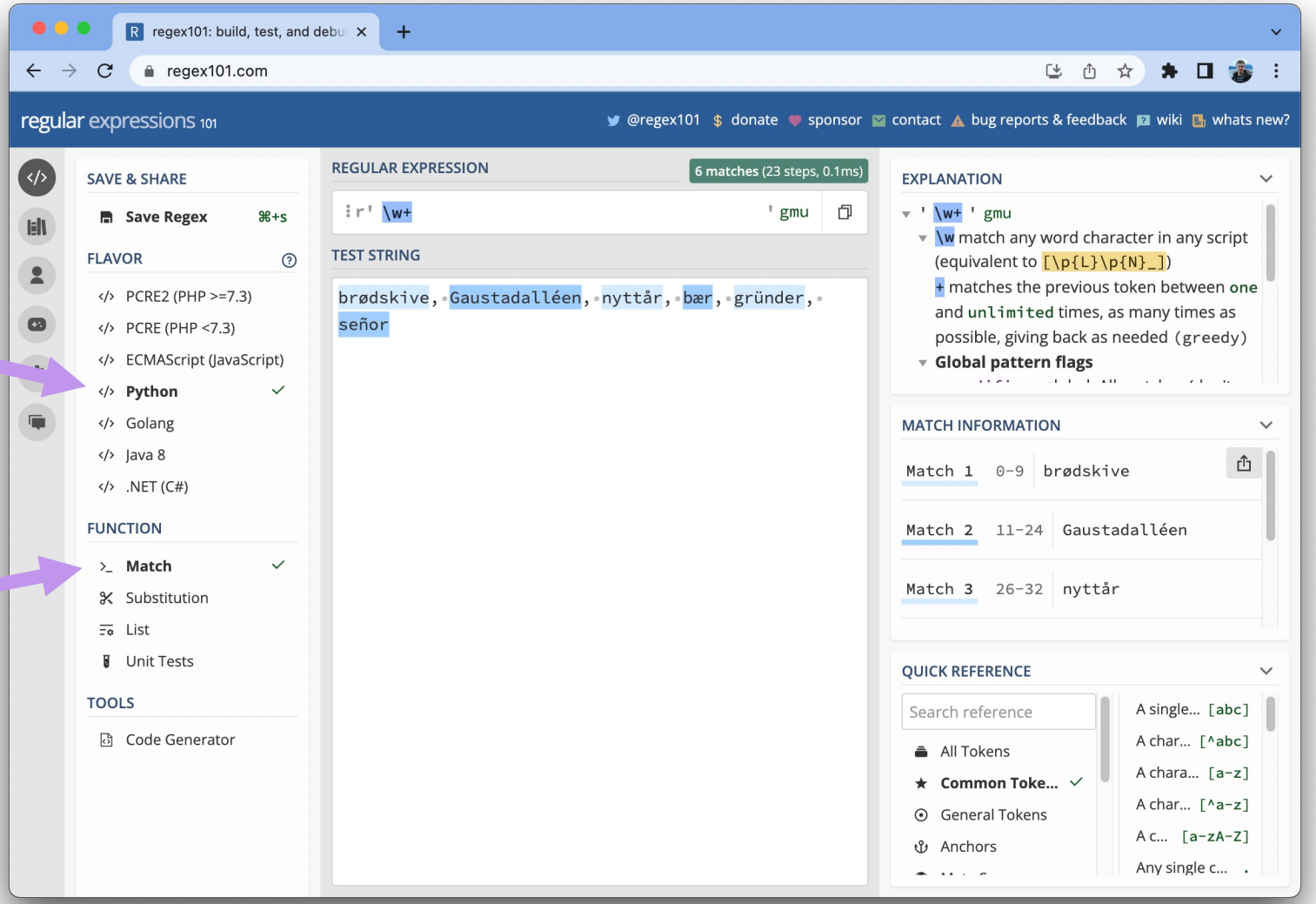
kaffekoppe, tekoppe, osv.

I det første uttrykket påvirker `?` hele den foregående gruppa `(en)` .
Dermed blir hele bøyningssuffikset *-en* valgfritt.

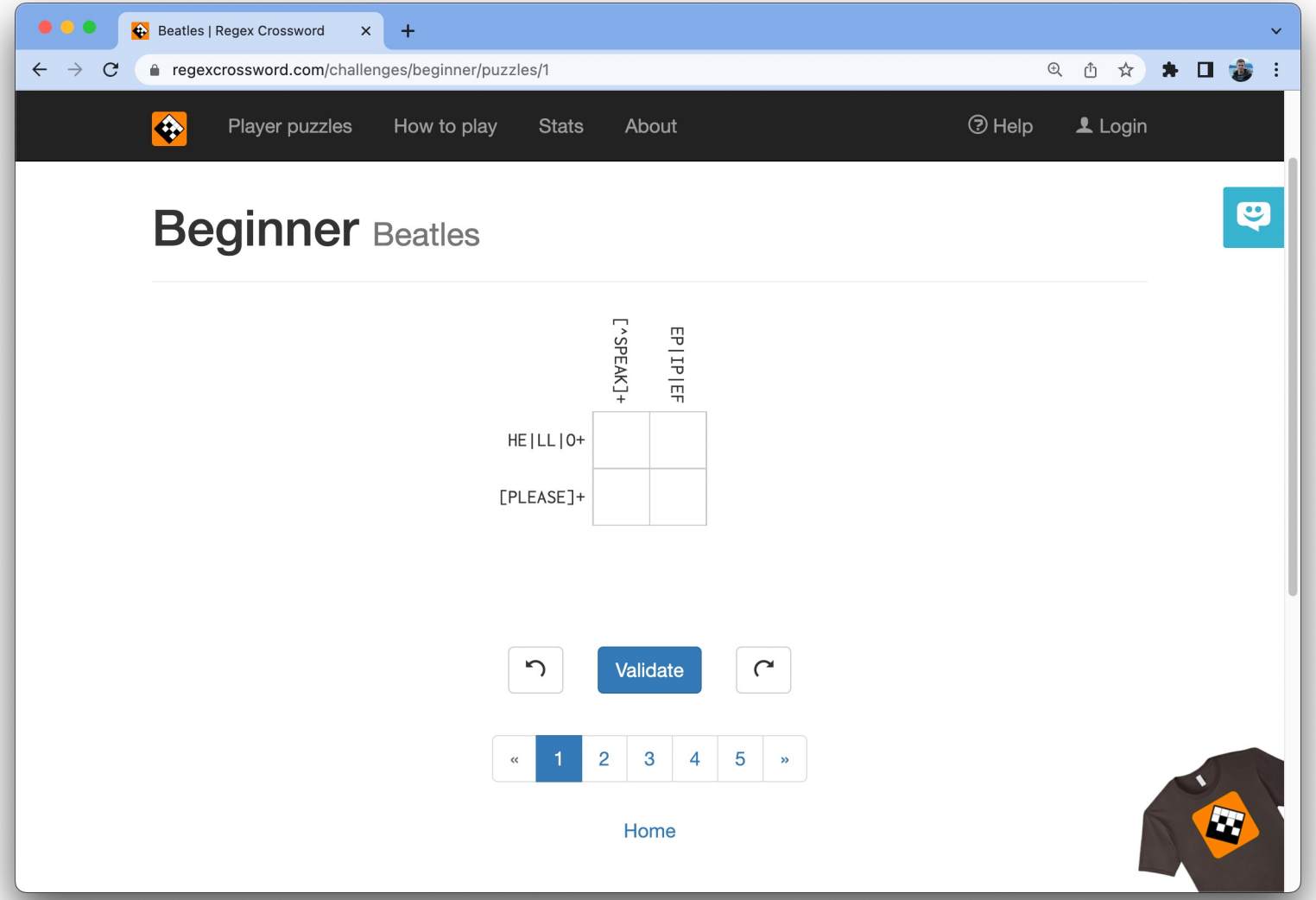
Det samme gjelder andre kvantifiserere, som `+` , `*` og `{n}` .

Hjelpeverktøy

IKKE PENSUM
IKKE TILGJENGELIG PÅ EKSAMEN



Kryssord med regulære uttrykk



... i Python

Funksjonalitet for regulære uttrykk finnes i modulen `re` .

Denne må importeres før bruk: `import re`

Vi skriver uttrykket vårt som en streng, ofte som en rå streng (r-streng):

```
uttrykk = r'(kaffe|te|latte)kopp(en)?'
```

Rå strenger er nyttige når vi ønsker at Python ikke skal prøve å tolke spesielle tegnsekvenser, f.eks. for linjeskift og tabulator, i strengene.

Metoder i `re`

`re.search(uttrykk, tekst)`
Går gjennom *`tekst`* og finner første stedet hvor *`uttrykk`* gir et treff.

`re.fullmatch(uttrykk, tekst)`
Sjekker om hele *`tekst`* gir et treff for *`uttrykk`*.

`re.match(uttrykk, tekst)`
Sjekker om *`uttrykk`* gir et treff i starten av *`tekst`*.

`re.findall(uttrykk, tekst)`
Returnerer en liste med alle ikke-overlappende treff for *`uttrykk`* i *`tekst`*.

`re.search()`, `re.match()` og `re.fullmatch()` returnerer et såkalt match-objekt hvis de finner et treff, ellers `None`.

Hentet fra `https://docs.python.org/3/library/re.html`
21. september 2022

Tokenisering

Regulære uttrykk kan brukes til tokenisering.

I innlevering 1a tokeniserte vi med `split()` kun ved mellomrom:

```
['Bæra', 'er', 'antioksidantbomber', 'og', 'inneheld', 'i', 'tillegg',
'E-vitamin,', 'C-vitamin', 'og', 'pektin.']
```

I innlevering 1b skal vi bruke et regulært uttrykk for å ende opp med tegnsetting som egne token:

```
['Bæra', 'er', 'antioksidantbomber', 'og', 'inneheld', 'i', 'tillegg',
'E-vitamin', ',', 'C-vitamin', 'og', 'pektin', '.']
```