IN1140: Introduksjon til språkteknologi

Forelesning #4

Egil Rønningstad

Universitetet i Oslo

16. september 2024



Tema for i dag

- Regulære uttrykk

Tema for i dag

- Regulære uttrykk



https://imgs.xkcd.com/comics/regular_expressions.png

Dagens leksjon

- Presentere regulære uttrykk
- Motivasjon for å mestre verktøyet: Blir brukt for å spesifisere tekststrenger i all slags type tekstprosessering og informasjonsutvinning
- Introduksjon til verktøyet
- Regex læres gjennom å bruke verktøyet. Dagens forelesning skal bare hjelpe deg i gang.
- Hele kapittel 2.1 er pensum, også substitusjon, capture-grupper og lookahead.
- Når uttrykkene blir litt kompliserte, kan det være forskjeller mellom Python-implemetasjonen og "ren" (PEARL?) regex. På eksamen blir du ikke spurt om Python-implementasjon.

Noen applikasjonseksempler

- Tokenisering
- Tekstprosessering (finne, erstatte)
  - Finne alle telefonnumre i en tekst, feks:
    - *Du kan også ringe Kundeservice på* **815 22 040**
    - *Du kan også ringe Kundeservice på* **815 22 041**
    - *Du kan også ringe Kundeservice på* **815 22 501**
  - Finne flere tilgrensende instanser av samme ord i en tekst
- Bestemme språket i en setning/tekst: spansk eller polsk?
  - *Sytuacja na Bliskim Wschodzie jest napieta, szczegolnie po wczorajszym ataku*

Noen applikasjonseksempler

- Validere felter i en database (datoer, e-postadresser, URL'er)
- Søk i et korpus etter lingvistiske mønstre
  - samle statistikk
- Tildele ordklasse til disse, selv om de ikke fins i ordboken. F.eks.:
  - *conurbation*, *verbify*, *disproportionality*, *Thatcherization*

Videre motivasjon

- I moderne språkteknologi ønsker vi å skape en kompakt tall-representasjon der (nesten) like ord representeres (nesten) likt.
- Regex gir oss en første tilnærming til dette.
- Regex er nyttig i seg selv, og den viser oss også behovet for andre metoder som vi lærer senere.

To setninger med ulike ord som representerer samme tingen:

- Jeg har en sykkel.
- Sykkelen min er grønn.

Regulære uttrykk

- Et regulært uttrykk er en beskrivelse av en mengde strenger
- Finnes en rekke UNIX-verktøy (grep, sed), editorer (emacs) og progammeringsspråk (perl, python, java) som har funksjonalitet for regulære uttrykk
- Som alle formalismer har ikke regulære uttrykk noe språklig (lingvistisk) innhold, men kan snarere brukes til å referere til lingvistiske enheter

Regulære uttrykk (RU)

To slags tegn ("characters"):

- **Bokstaver**
  - ethvert teksttegn er et RU og refererer til seg selv
- **Meta-tegn**
  - spesialtegn som lar deg kombinere RU på forskjellige måter
- Eksempel:
  - `/a/` refererer til `a`
  - `/a*/` refererer til ϵ (null) eller `a` eller `aa` eller `aaa` eller ...

Regulære uttrykk (RU)

To slags tegn ("characters"):

- **Bokstaver**
  - ethvert teksttegn er et RU og refererer til seg selv
- **Meta-tegn**
  - spesialtegn som lar deg kombinere RU på forskjellige måter
- Eksempel:
  - `/a/` refererer til `a`
  - `/a*/` refererer til ϵ (null) eller `a` eller `aa` eller `aaa` eller ...
- MERK: `//` (i feks `/a/`) er **ikke** en del av et RU. Brukes som notasjon for å gjøre det tydelig hva som er RU og hva som ikke er RU

Regulære uttrykk

Regulære uttrykk består av:

- Strenger bestående av tegn: `/b/`, `/IN1140/`, `/informatikk/`
- Disjunksjon:
  - vanlig disjunksjon: `/spise|ete/`, `/penge(r|ne)/`
  - tegnklasser:`/[Dd]en/`, `/m[ae]nn/`, `/bec[oa]me/`
  - intervaller ("ranges"): `[A-Z]`, `[a-z]`, `[0-9]`
- Negasjon:
  - Bruk av `^`
  - `[^b]`
  - `[^A-Z0-9]`

Regulære uttrykk

Regulære uttrykk består av:

- Strenger bestående av tegn: `/b/`, `/IN1140/`, `/informatikk/`
- Disjunksjon:
  - vanlig disjunksjon: `/spise|ete/`, `/penge(r|ne)/`
  - tegnklasser:`/[Dd]en/`, `/m[ae]nn/`, `/bec[oa]me/`
  - intervaller ("ranges"): `[A-Z]`, `[a-z]`, `[0-9]`
- Negasjon:
  - Bruk av `^`
  - `[^b]`
  - `[^A-Z0-9]`
  - `a\^b` ser etter sekvensen "`a^b`" i en streng
    - feks: "se etter `a^b` nå"
  - `e\^` ser etter sekvensen "`e^`" i en streng
    - feks: "se etter `e^` nå"

Regulære uttrykk

Regulære uttrykk består av (forts.):

- Tellere
  - opsjonalitet (0 eller 1): `?`
    - `/woodchucks?/` (fanger begge “woodchucks” og “woodchuck”)
    - `/colou?r/` (fanger begge “colour” og “color”)
  - hvilket som helst antall (0 eller flere): Kleene `*`
    - `/ba*!/` (fanger “b!”, “ba!”, “baa!”, “baaa!”, “baaaa!”, “baaaaaaaaaa!”)
    - `/[1-9][0-9]*/`
  - Minst en: `+`
    - `/ba+!/` (fanger “ba!”, “baa!”, “baaa!”, “baaaa!”, “baaaaaaaaaa!”)
    - `/[0-9]+ kroner/`

Regulære uttrykk

Regulære uttrykk består av (forts.):

- "wildcard" for et hvilket som helst tegn: .
  - `/beg.n/` (alt som finnes mellom *beg* og *n*)
    - "begin"
    - "beg'n"
    - "begun"
  - brukes ofte sammen med stjerne: "hva som helst":
  - `/beltedyr.*beltedyr/`
    - "beltedyr er en familie av gomlere som er i utgangspunktet en ren søramerikansk gruppe, og Sør-Amerika pluss de sørlige delene av Nord-Amerika er de eneste stedene hvor beltedyr finnes vilt i dag"
-

Regulære uttrykk

Regulære uttrykk kan fortelle oss:

`/beltedyr.*beltedyr/`

"beltedyr er en familie av gomlere som er i utgangspunktet en ren søramerikansk gruppe, og Sør-Amerika pluss de sørlige delene av Nord-Amerika er de eneste stedene hvor beltedyr finnes vilt i dag"

- *Matcher det regulære uttrykket teksten i sin helhet?*
  (Ja / Nei)
- *Inneholder teksten en streng som matches av det regulære uttrykket?*
  (Ja / Nei)
- *Hva er (den første) strengen som som matches av det regulære uttrykket?*
  beltedyr er en familie av gomlere som er i utgangspunktet en ren søramerikansk gruppe, og Sør-Amerika pluss de sørlige delene av Nord-Amerika er de eneste stedene hvor beltedyr

Regulære uttrykk

Regulære uttrykk består av (forts.):
Ankere: spesielle tegn som forankrer det regulære uttrykket til spesifikt sted i strengen/teksten

- `^` - begynnelsen av linjen
  - `/^Den/` matcher "Den" bare i begynnelsen av en linje

Regulære uttrykk

Regulære uttrykk består av (forts.):
Ankere: spesielle tegn som forankrer det regulære uttrykket til spesifikt sted i strengen/teksten

- `^` - begynnelsen av linjen
  - `/^Den/` matcher "Den" bare i begynnelsen av en linje
  - `^` har tre mulige bruk:
    - Matche begynnelsen av en linje
    - Indikerer negasjon innefor firkantede parenteser (`[]`) altså tegnklasser
    - Bare karakteren `^`

Regulære uttrykk

Regulære uttrykk består av (forts.):
Ankere: spesielle tegn som forankrer det regulære uttrykket til spesifikt sted i strengen/teksten

- `^` - begynnelsen av linjen
  - `/^Den/` matcher "Den" bare i begynnelsen av en linje
  - `^` har tre mulige bruk:
    - Matche begynnelsen av en linje
    - Indikerer negasjon innefor firkantede parenteser (`[]`) altså tegnklasser
    - Bare karakteren `^`
- `$` - slutten av linjen
  - `/der\.$/` matcher "der." på slutten av en linje
- `/^Det hvite huset\.$/` ?

Regulære uttrykk

Regulære uttrykk består av (forts.):
Ankere: spesielle tegn som forankrer det regulære uttrykket til spesifikt sted i strengen/teksten

- `^` - begynnelsen av linjen
  - `/^Den/` matcher "Den" bare i begynnelsen av en linje
  - `^` har tre mulige bruk:
    - Matche begynnelsen av en linje
    - Indikerer negasjon innefor firkantede parenteser (`[]`) altså tegnklasser
    - Bare karakteren `^`
- `$` - slutten av linjen
  - `/der\.$/` matcher "der." på slutten av en linje
- `/^Det hvite huset\.$/` ?
  - Matcher kun en linje som inneholder frasen "Det hvite huset."

Regulære uttrykk

Regulære uttrykk består av (forts.):

- Ankere `\b` og `\B`
- `\b` matcher en "word boundary" (altså ordgrense)
  - `/\bthe\b/` matcher ordet "the" men ikke "other"
- `\B` matcher en "non-boundary"

Regulære uttrykk

Disjunksjon, gruppering, og presedens:

- Ser etter “katter” og “hunder” i en tekst:

Regulære uttrykk

Disjunksjon, gruppering, og presedens:

- Ser etter “katter” og “hunder” i en tekst:
  - Kan ikke bruke “[” og “]” for å søke for “katter **eller** hunder”

Regulære uttrykk

Disjunksjon, gruppering, og presedens:

- Ser etter “katter” og “hunder” i en tekst:
  - Kan ikke bruke “[” og “]” for å søke for “katter **eller** hunder”
  - Må bruke **disjunksjon**: `/katter|hunder/`
- Ser etter “guppy” og “guppies”:

Regulære uttrykk

Disjunksjon, gruppering, og presedens:

- Ser etter "katter" og "hunder" i en tekst:
  - Kan ikke bruke "[" og "]" for å søke for "katter **eller** hunder"
  - Må bruke **disjunksjon**: `/katter|hunder/`
- Ser etter "guppy" og "guppies":
  - Kan vi si `/guppy|ies/`?

Regulære uttrykk

Disjunksjon, gruppering, og presedens:

- Ser etter "katter" og "hunder" i en tekst:
  - Kan ikke bruke "[" og "]" for å søke for "katter **eller** hunder"
  - Må bruke **disjunksjon**: `/katter|hunder/`
- Ser etter "guppy" og "guppies":
  - Kan vi si `/guppy|ies/`?
    - Nei! Vil matche strengene "guppy" og "ies"

Regulære uttrykk

Disjunksjon, gruppering, og presedens:

- Ser etter "katter" og "hunder" i en tekst:
  - Kan ikke bruke "[" og "]" for å søke for "katter **eller** hunder"
  - Må bruke **disjunksjon**: `/katter|hunder/`
- Ser etter "guppy" og "guppies":
  - Kan vi si `/guppy|ies/`?
    - Nei! Vil matche strengene "guppy" og "ies"
    - Sekvensen `guppy` har **presedens** over disjunksjonen "`|`"

Regulære uttrykk

Disjunksjon, gruppering, og presedens:

- Ser etter "katter" og "hunder" i en tekst:
  - Kan ikke bruke "[" og "]" for å søke for "katter **eller** hunder"
  - Må bruke **disjunksjon**: `/katter|hunder/`
- Ser etter "guppy" og "guppies":
  - Kan vi si `/guppy|ies/`?
    - Nei! Vil matche strengene "guppy" og "ies"
    - Sekvensen `guppy` har **presedens** over disjunksjonen "`|`"
  - Må derfor bruke "(" og ")" , (**gruppering**)
    - `/gupp(y|ies)/`

Regulære uttrykk

Kleene * operator:

- Vi vil fange repetisjoner av en instans, feks: Kolonne 1 Kolonne 2 Kolonne 3.

Regulære uttrykk

Kleene * operator:

- Vi vil fange repetisjoner av en instans, feks: Kolonne 1 Kolonne 2 Kolonne 3.
- Hva vil `/Kolonne␣[0-9]+␣*/` fange?

Regulære uttrykk

Kleene * operator:

- Vi vil fange repetisjoner av en instans, feks: Kolonne 1 Kolonne 2 Kolonne 3.
- Hva vil `/Kolonne␣[0-9]+␣*/` fange?
  - "Kolonne 1" etterfulgt av fritt antall mellomrom (feks. "Kolonne 1␣␣␣␣...")

Regulære uttrykk

Kleene * operator:

- Vi vil fange repetisjoner av en instans, feks: Kolonne 1 Kolonne 2 Kolonne 3.
- Hva vil `/Kolonne␣[0-9]+␣*/` fange?
  - "Kolonne 1" etterfulgt av fritt antall mellomrom (feks. "Kolonne 1␣␣␣␣...")
- Må bruke "`()`":
  - `/(Kolonne␣[0-9]+␣*)*/`
- Her har "`()`" presedens over "`*`"

Regulære uttrykk

Kleene * operator:

- Vi har `/[a-z]*/`
- Setningen “once upon a time”
- Hva vil `/[a-z]*/` fange?

Regulære uttrykk

Kleene * operator:

- Vi har `/[a-z]*/`
- Setningen “once upon a time”
- Hva vil `/[a-z]*/` fange?
  - ingenting? “o”? “on”? “onc”? “once”? “once␣”? “once␣u”? ...

Regulære uttrykk

Kleene * operator:

- Vi har `/[a-z]*/`
- Setningen “once upon a time”
- Hva vil `/[a-z]*/` fange?
  - ingenting? “o”? “on”? “onc”? “once”? “once␣”? “once␣u”? ...
  - “once”
- RU matcher alltid lengste strengen, de er kallt **greedy**

Regulære uttrykk

Presedens av RU operatorer:

1	Parentes	`()`
2	Tellere (counters)	`+ - ?  {}`
3	Sekvenser og ankere	`the ^my end$`
4	Disjunksjon	`|`

Regulære uttrykk

- Nyttige forkortelser:

RU	=	Eksempel
`\d`	`[0-9]`	Lag␣av␣5
`\D`	`[^0-9]`	Blå␣himmel␣
`\w`	`[A-Za-z0-9]`	Dag
`\W`	`[^A-Za-z0-9]`	!!!!!
`\s`	`[ \r\t\n]`	
`\S`	`[^\s]`	På␣Fløyen

Regulære uttrykk

- Nyttige forkortelser:

RU	Matcher
`*`	null eller flere
`+`	en eller flere
`?`	null eller en
`.`	wildcard, matcher hva som helst
`{n}`	n forekomster
`{n,m}`	fra n til m forekomster
`{n,}`	minst n forekomster

Regulære uttrykk

- Det som trenger "backslash":

RU	Eksempel
`\*`	I*N*1*1*4*0
`\.`	Dr. Nesehorn
`\?`	Dr. Nesehorn?
`\n`	ny linje
`\t`	tab

- Men også `/\[/`, og `/\\/` ...


UNIVERSITAS OSLOENSIS
MDCCCXI

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

- Riktig dato?

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

- Riktig dato?
- Hva er formatet?

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

- Riktig dato?
- Hva er formatet?
  - siffer: 0-9
  - mellomrom: \s
  - ord (sekvens av bokstaver): [A-Za-z]
  - mellomrom: \s
  - tall: 1900-2099

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

- Riktig dato?
- Hva er formatet?
  - siffer: 0-9
  - mellomrom: \s
  - ord (sekvens av bokstaver): [A-Za-z]
  - mellomrom: \s
  - tall: 1900-2099

$$regex \rightarrow [0-9] + \backslash s[A-Za-z] + \backslash s((19|20)[1-9]+)$$

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

- Riktig dato?
- Hva er formatet?
  - siffer: 0-9
  - mellomrom: \s
  - ord (sekvens av bokstaver): [A-Za-z]
  - mellomrom: \s
  - tall: 1900-2099

$$regex \rightarrow [0-9] + \backslash s[A-Za-z] + \backslash s((19|20)[1-9]+)$$

Fanger den kun riktige strenger?

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

$$regex \rightarrow [[0-9] + \backslash s[A-Za-z] + \backslash s((19|20)[1-9]+)$$

Fanger følgende:

- 6 januar 1992
- 3342 november 1988
- 9 fredag 1999
- 1 mars 2018

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

$$regex \rightarrow [[0-9] + \backslash s[A-Za-z] + \backslash s((19|20)[1-9]+)$$

Fanger følgende:

- 6 januar 1992
- 3342 november 1988
- 9 fredag 1999
- 1 mars 2018

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

- Hva er formatet?

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

- Hva er formatet?
  - siffer: 0-9
  - mellomrom: \s
  - ord (sekvens av bokstaver): (januar|februar|mars|april|mai|juni|juli|august|september|oktober|november|desember)
  - mellomrom: \s
  - tall: ((19|20)[0-9]+)

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

- Hva er formatet?
  - siffer: 0-9
  - mellomrom: \s
  - ord (sekvens av bokstaver): (januar|februar|mars|april|mai|juni|juli|august|september|oktober|november|desember)
  - mellomrom: \s
  - tall: ((19|20)[0-9]+)

$regex \rightarrow [[0-9]+\backslash s$
$(januar|februar|mars|april|mai|juni|juli|august|september|$
$oktober|november|desember)\backslash s\ ((19|20)[0-9]+)$

Fanger den kun riktige strenger?

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

$$regex \rightarrow [[0-9]+\backslash s (januar|februar|mars|april|mai|juni|juli|august|september|oktober|november|desember)\backslash s ((19|20)[0-9]+)$$

Den fanger:

- 6 januar 1992
- 25 juni 2005
- 3342 november 1988
- 1 mars 2018
- 30 juli 190221

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

$$regex \rightarrow [[0-9]+\backslash s (januar|februar|mars|april|mai|juni|juli|august|september|oktober|november|desember)\backslash s ((19|20)[0-9]+)$$

Den fanger:

- 6 januar 1992
- 25 juni 2005
- 3342 november 1988
- 1 mars 2018
- 30 juli 190221

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

- Hva er formatet?

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

- Hva er formatet?
  - siffer: 1-9, 1 etterfulgt av 0-9, 2 etterfulgt av 0-9, 3 etterfulgt av 0 eller 1
  - mellomrom: \s
  - ord (sekvens av bokstaver): (januar|februar|mars|april|mai|juni|juli|august|september|oktober|november|desember)
  - mellomrom: \s
  - tall: (19|20) etterfulgt av maks to siffer 0-9

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

- Hva er formatet?
  - siffer: 1-9, 1 etterfulgt av 0-9, 2 etterfulgt av 0-9, 3 etterfulgt av 0 eller 1
  - mellomrom: \s
  - ord (sekvens av bokstaver): (januar|februar|mars|april|mai|juni|juli|august|september|oktober|november|desember)
  - mellomrom: \s
  - tall: (19|20) etterfulgt av maks to siffer 0-9

$regex \rightarrow [([1-9]|[12][0-9]|3[01])\backslash s$
$(januar|februar|mars|april|mai|juni|juli|august|september|$
$oktober|november|desember)\backslash s\ ((19|20)\backslash d\{2\})$

Fanger den kun riktige strenger?

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

Eksempel 1

liste = ['6 januar 1992', '25 juni 2005', '3342 november 1988', '9 fredag 1999', '1 mars 2018', '30 juli 190221']

$$regex \rightarrow [([1-9]|[12][0-9]|3[01])\backslash s (januar|februar|mars|april|mai|juni|juli|august|september|oktober|november|desember)\backslash s ((19|20)\backslash d\{2\})$$

Den fanger:

- 6 januar 1992
- 25 juni 2005
- 1 mars 2018

Regulære uttrykk

- **Et eksempel: Finne alle forekomster av *og* i en tekst**
  - `/og/`
    - Problem: *Og det ble kveld, og det ble morgen . . .*

Regulære uttrykk

- **Et eksempel: Finne alle forekomster av *og* i en tekst**
  - `/og/`
    - Problem: *Og det ble kveld, og det ble morgen . . .*
  - `/[Oo]g/`
    - Problem: *Togets rutetider*

Regulære uttrykk

- **Et eksempel: Finne alle forekomster av *og* i en tekst**
  - `/og/`
    - Problem: *Og det ble kveld, og det ble morgen . . .*
  - `/[Oo]g/`
    - Problem: *Togets rutetider*
  - `/\s[Oo]g\s/`

Regulære uttrykk

- **Et eksempel: Finne alle forekomster av *og* i en tekst**
  - `/og/`
    - Problem: *Og det ble kveld, og det ble morgen . . .*
  - `/[Oo]g/`
    - Problem: *Togets rutetider*
  - `/\s[Oo]g\s/` eller `/\b[Oo]g\b/`
- Minimere antall falske positiver (uriktig matchet, feks *Toget*) og falske negativer (uriktig utelukket, feks *Og*)

Regulære uttrykk - sub

- Substitusjon (perl format):
  - `s/color/colour/`
  - `s/versjon\s2\.4/versjon\s2\.5/`

Regulære uttrykk - sub

- Substitusjon (perl format):
  - `s/color/colour/`
  - `s/versjon\s2\.4/versjon\s2\.5/`
- Substitusjon (python format):
  - re.sub(“color”, “colour”, text)
  - re.sub(“versjon 2.4”, “versjon 2.5”, text)

Regulære uttrykk

- ELIZA (Weizenbaum 1966):

User:	*Men are all alike*
ELIZA:	IN WHAT WAY?
User:	*They're always bugging us about something*
ELIZA:	CAN YOU THINK OF AN EXAMPLE?
User:	*Well, my boyfriend made me come here*
ELIZA:	YOUR BOYFRIEND MADE YOU COME HERE.
User:	*He says I am depressed much of the time*
ELIZA:	I'M SORRY TO HEAR THAT YOU ARE DEPRESSED.

Regulære uttrykk

- ELIZA (Weizenbaum 1966):

User:	*Men are all alike*
ELIZA:	IN WHAT WAY?
User:	*They're always bugging us about something*
ELIZA:	CAN YOU THINK OF AN EXAMPLE?
User:	*Well, my boyfriend made me come here*
ELIZA:	YOUR BOYFRIEND MADE YOU COME HERE.
User:	*He says I am depressed much of the time*
ELIZA:	I'M SORRY TO HEAR THAT YOU ARE DEPRESSED.

```
s/.* I am (depressed|sad) .*
/ I am sorry to hear that you are \1/
s/.* (all|always) .*
/ Can you think of an example /
```

Prøv hjemme med og uten Python

- Beskriv strengene som aksepteres av følgende regulære uttrykk
  - `/ab+a/`
  - `/(ab)*/`
  - `/([^aeiou][aeiou])/`
  - `/\sdis[a-z]+\s/`