Språk och Datorer (729G49)

Språkteknologi 2:
Lingvistiskt uppmärkt text 1:
Ord och ordklasser

7/4/2025

Innehåll

- Ord och normalisering
- Lingvistiskt uppmärkt text
- Ordklasser
- Generatorfunktioner

Ord och Normalisering

Graford och Ordtyper (1)

A rose is a rose is a rose

(Gertrude Stein)

- Tre ordformer/**ordtyper** (*a*, *rose*, *is*).
- Åtta tokens/**graford** (*A*, *rose*, *is*, *a*, *rose*, *is*, *a*, *rose*).

Graford och Ordtyper (2)

Korpus	Antal graford	Antal ordtyper
Shakespeare	ca 884,000	ca. 31,000
Riksmöte 2012/2013	4,645,560	96,114
Google Ngrams	1,176,470,663	13,588,391

FineWeb

The finest collection of data the web has to offer

> *“15 trillion tokens of the finest data the 🌐 web has to offer”*

What is it?

The 🍷 FineWeb dataset consists of more than **15T tokens** of cleaned and deduplicated english web data from CommonCrawl. The data processing pipeline is optimized for LLM performance and ran on the 🏭 `datatrove` library, our large scale data processing library.

FineWeb

The finest collection of data the web has to offer

> *“15 trillion tokens of the finest data the 🌐 web has to offer”*

What is it?

The 🍷 FineWeb dataset consists of more than **15T tokens** of cleaned and deduplicated english web data from CommonCrawl. The data processing pipeline is optimized for LLM performance and ran on the 🏭 `datatrove` library, our large scale data processing library.

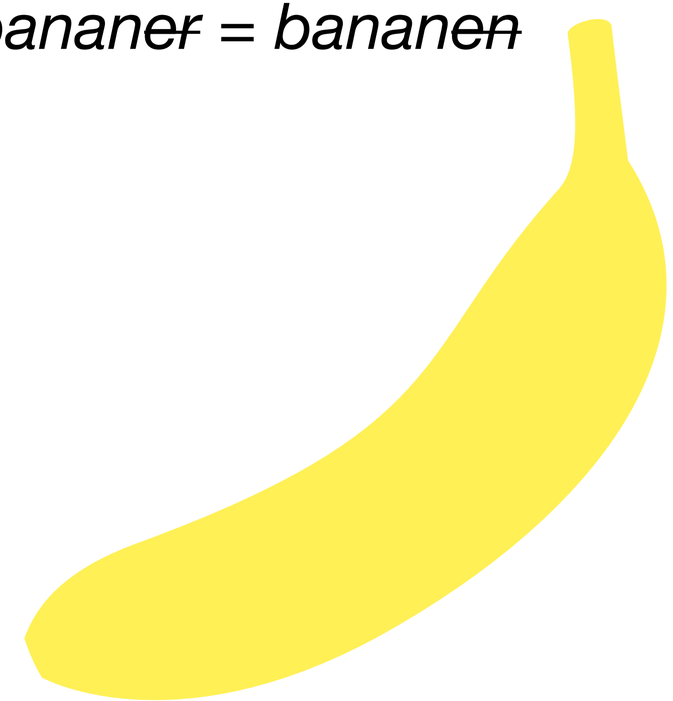
Normalisering

- Tokenisering ger oss möjlighet att hitta graford och ordtyper.
    - Det gjorde vi i labb 1!

- Men ibland vill vi även sammanfatta ordtyper som är sig mycket lika:
    - *Jag ska äta en **banan** nu.*
    - *Jag gillar **bananer**.*
    - ***Bananer** är goda.*

Exempel på Normalisering

- Ändra alla bokstäver till gemener: *Banan* = *banan*
- Harmonisering av stavningsvarianter: *colour* = *color*; *metre* = *meter*
- Avstamning (borttagandet av suffix): *banan* = *banane~~r~~* = *banane~~n~~*



Lexem och Lemma

- Begreppet **lexem** betecknar en mängd ordformer som representerar samma grundläggande betydelse.
    - Ordformer: *tanke*, *tanken*, *tankar*, *tankarna*, *tankarnas*,…
    - Lexem: *TANKE*
- Begreppet **lemma** betecknar den form av ett lexem som brukar användas för att representera lexemet i t. ex. en ordbok.
    - För substantiv: nominativ singularis (*tanke*),
    - För verb: infinitiv *(att tänka)*

Lingvistiskt uppmärkt text

1	Genom	genom	PP	3	AA
2	skattereformen	skattereform	NN	1	PA
3	införs	införa	VB	0	ROOT
4	individuell	individuell	JJ	5	AT
5	beskattning	beskattning	NN	3	SS
6	(	(	PAD	5	IR
7	särbeskattning	särbeskattning	NN	5	AN
8	)	)	PAD	5	JR
9	av	av	PP	5	ET
10	arbetsinkomster	arbetsinkomst	NN	9	PA
11	.	.	MAD	3	IP

Lingvistiska Annotationer

1	Genom	genom	PP	3	AA
2	skattereformen	skattereform	NN	1	PA
3	införs	införa	VB	0	ROOT
4	individuell	individuell	JJ	5	AT
5	beskattning	beskattning	NN	3	SS
6	(	(	PAD	5	IR
7	särbeskattning	särbeskattning	NN	5	AN
8	)	)	PAD	5	JR
9	av	av	PP	5	ET
10	arbetsinkomster	arbetsinkomst	NN	9	PA
11	.	.	MAD	3	IP

- Graford *skattereformen*
- Lemma *skattereform*
- Ordklass NN (substantiv)
- Huvudord Ord nummer 3 i meningen (*införs*)
- Grammatisk funktion SS (subjekt)
- Morfologiska egenskaper utrum, singular, definit, nominativ
- Betydelse skattereform..nn.1

Lingvistiska Annotationer

1	Genom	genom	PP	3	AA
2	skattereformen	skattereform	NN	1	PA
3	införs	införa	VB	0	ROOT
4	individuell	individuell	JJ	5	AT
5	beskattning	beskattning	NN	3	SS
6	(	(	PAD	5	IR
7	särbeskattning	särbeskattning	NN	5	AN
8	)	)	PAD	5	JR
9	av	av	PP	5	ET
10	arbetsinkomster	arbetsinkomst	NN	9	PA
11	.	.	MAD	3	IP

- **Graford** ***skattereformen***
- ~~Lemma~~ ~~*skattereform*~~
- **Ordklass** **NN (substantiv)**

Labb 2

- Huvudord Ord nummer 3 i meningen (*införs*)
- Grammatisk funktion SS (subjekt)
- Morfologiska egenskaper utrum, singular, definit, nominativ
- ~~Betydelse~~ ~~skattereform..nn.1~~

Lingvistiska Annotationer

1	Genom	genom	PP	3	AA
2	skattereformen	skattereform	NN	1	PA
3	införs	införa	VB	0	ROOT
4	individuell	individuell	JJ	5	AT
5	beskattning	beskattning	NN	3	SS
6	(	(	PAD	5	IR
7	särbeskattning	särbeskattning	NN	5	AN
8	)	)	PAD	5	JR
9	av	av	PP	5	ET
10	arbetsinkomster	arbetsinkomst	NN	9	PA
11	.	.	MAD	3	IP

- **Graford** ***skattereformen***
- ~~Lemma~~ ~~*skattereform*~~
- **Ordklass** **NN (substantiv)**

Labb 2

- Huvudord Ord nummer 3 i meningen (*införs*)
- Grammatisk funktion SS (subjekt)

Labb 3

- Morfologiska egenskaper utrum, singular, definit, nominativ
- ~~Betydelse~~ ~~skattereform..nn.1~~

Olika Filformat

Representation av Korpusdata

- Det finns många möjligheter att representera korpusdata:
    - Tabulerade data
    - Extensible Markup Language (XML)
    - JavaScript Object Notation (JSON)

- I labben kommer vi att titta på tabulerade data i CoNLL-formatet.

Tabulerade Data: CoNLL-Format (1)

1	Genom	genom	PP	3	AA
2	skattereformen	skattereform	NN	1	PA
3	införs	införa	VB	0	ROOT
4	individuell	individuell	JJ	5	AT
5	beskattning	beskattning	NN	3	SS
6	(	(	PAD	5	IR
7	särbeskattning	särbeskattning	NN	5	AN
8	)	)	PAD	5	JR
9	av	av	PP	5	ET
10	arbetsinkomster	arbetsinkomst	NN	9	PA
11	.	.	MAD	3	IP

Tabulerade Data: CoNLL-Format (2)

- Filen struktureras i **rader** och **kolumner**.
- Rader skiljs åt med ett **nyrad-tecken**.
- Kolumner skiljs åt med ett **separatortecken** (ex. tab).

1	Genom	genom	PP	3	AA
2	skattereformen	skattereform	NN	1	PA
3	införs	införa	VB	0	ROOT
4	individuell	individuell	JJ	5	AT
5	beskattning	beskattning	NN	3	SS
6	(	(	PAD	5	IR
7	särbeskattning	särbeskattning	NN	5	AN
8	)	)	PAD	5	JR
9	av	av	PP	5	ET
10	arbetsinkomster	arbetsinkomst	NN	9	PA
11	.	.	MAD	3	IP

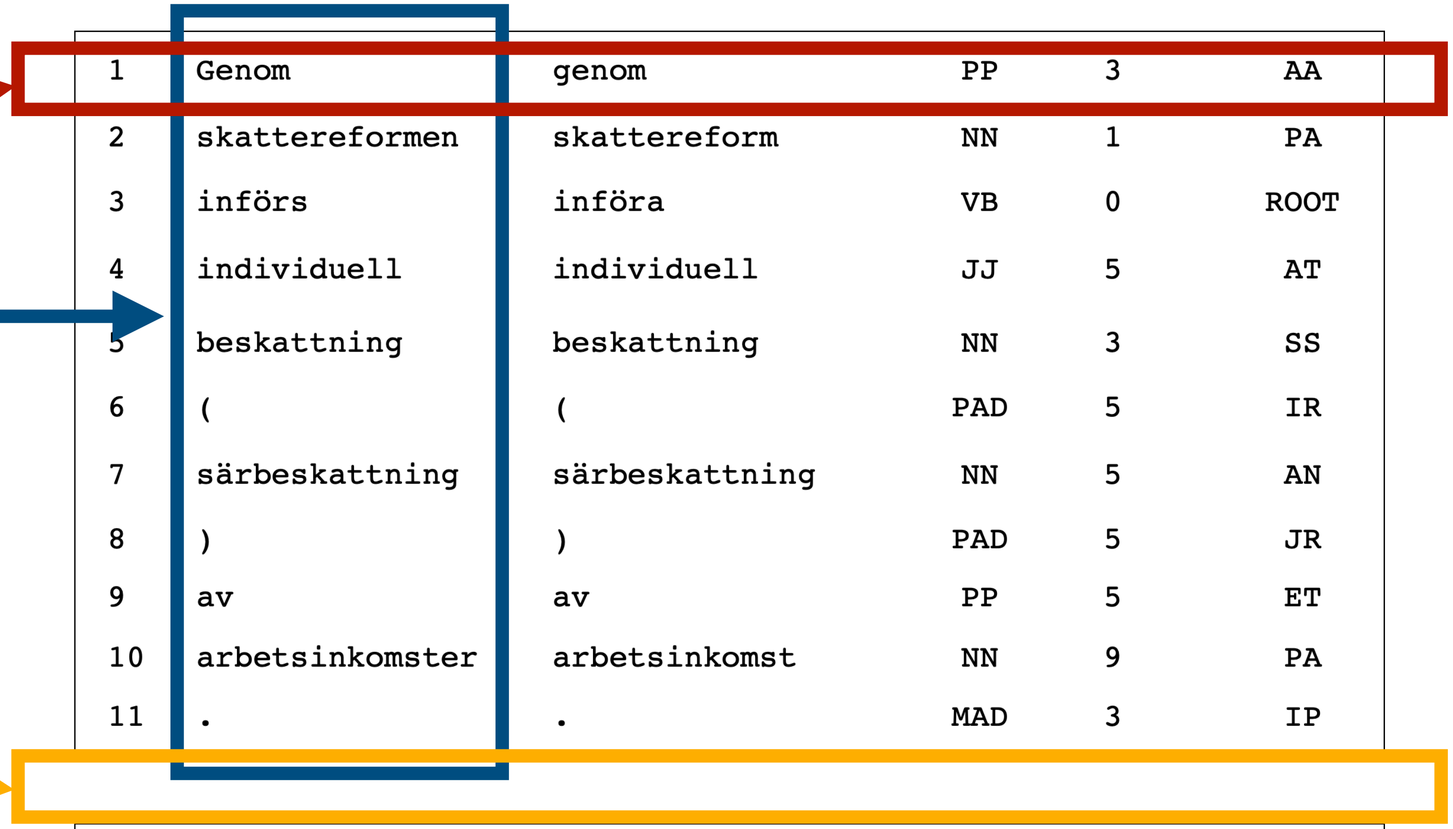
Tabulerade Data: CoNLL-Format (2)

- Filen struktureras i **rader** och **kolumner**.
- Rader skiljs åt med ett **nyrad-tecken**.
- Kolumner skiljs åt med ett **separatortecken** (ex. tab).

1	Genom	genom	PP	3	AA
2	skattereformen	skattereform	NN	1	PA
3	införs	införa	VB	0	ROOT
4	individuell	individuell	JJ	5	AT
5	beskattning	beskattning	NN	3	SS
6	(	(	PAD	5	IR
7	särbeskattning	särbeskattning	NN	5	AN
8	)	)	PAD	5	JR
9	av	av	PP	5	ET
10	arbetsinkomster	arbetsinkomst	NN	9	PA
11	.	.	MAD	3	IP

CoNLL-Formatet

- En **rad** representerar ett ord
- En **kolumner** är en viss typ av lingvistisk annotation (t.ex. graford)
- Slutet på en mening markeras med en blankrad.



1	Genom	genom	PP	3	AA
2	skattereformen	skattereform	NN	1	PA
3	införs	införa	VB	0	ROOT
4	individuell	individuell	JJ	5	AT
5	beskattning	beskattning	NN	3	SS
6	(	(	PAD	5	IR
7	särbeskattning	särbeskattning	NN	5	AN
8	)	)	PAD	5	JR
9	av	av	PP	5	ET
10	arbetsinkomster	arbetsinkomst	NN	9	PA
11	.	.	MAD	3	IP

1	Genom	genom	PP	3	AA
2	skattereformen	skattereform	NN	1	PA
3	införs	införa	VB	0	ROOT
4	individuell	individuell	JJ	5	AT

XML (1)

```
<sentence id="8f74-8115">
    <w pos="PP" lemma="genom|" ref="01" dephead="03" deprel="AA">Genom</w>
    <w pos="NN" lemma="skattereform|" ref="02" dephead="01" deprel="PA">skattereformen</w>
    <w pos="VB" lemma="införa|" dephead="" deprel="ROOT">införs</w>
    <w pos="JJ" lemma="individuell|" ref="04" dephead="05" deprel="AT">individuell</w>
    <w pos="NN" lemma="beskattning|" ref="05" dephead="03" deprel="SS">beskattning</w>
    <w pos="PAD" lemma="|" ref="06" dephead="05" deprel="IR">(</w>
    <w pos="NN" lemma="särbeskattning|" ref="07" dephead="05" deprel="AN">särbeskattning</w>
    <w pos="PAD" lemma="|" ref="08" dephead="05" deprel="JR">)</w>
    <w pos="PP" lemma="av|" ref="09" dephead="05" deprel="ET">av</w>
    <w pos="NN" lemma="arbetsinkomst|" ref="10" dephead="09" deprel="PA">arbetsinkomster</w>
    <w pos="MAD" lemma="|" ref="11" dephead="03" deprel="IP">.</w>
</sentence>
```

XML (2)

- Information struktureras hierarkiskt med hjälp av **element.**
- Ett element består av en starttagg och en sluttagg.
    - <w>*särbeskattning*</w>
- Varje element kan dessutom ha ett antal attribut-värde-par
    - <w pos=”NN”>*särbeskattning*</w>
- Ett element kan innehålla såväl text som andra element

Fördelar och Nackdelar

Tabulerade data

- Enkelt och platseffektivt.
- Implicit representation.
- För att ta ut en ordklasstagg behöver man **veta vilken kolumn den är i**.
- Svår att representera mer komplexa relationer.

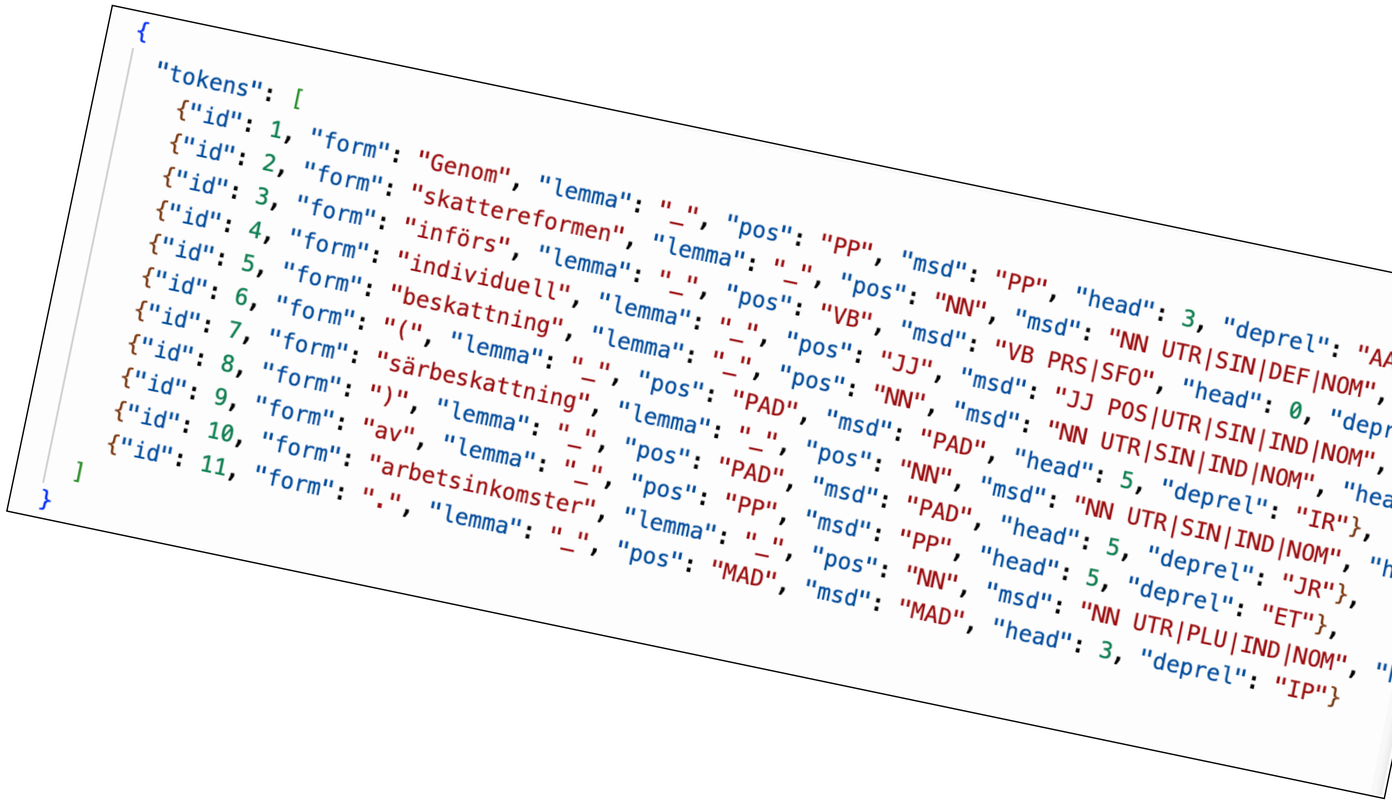
XML

- Mer platskrävande.
- Explicit representation.
- För att ta ut en ordklasstagg kan man **direkt efterfråga motsvarande attribut**.
- Hierarkisk och flexibel.

JSON (1)


```
{
  "tokens": [
    {"id": 1, "form": "Genom", "lemma": "_", "pos": "PP", "msd": "PP", "head": 3, "deprel": "AA"},
    {"id": 2, "form": "skattereformen", "lemma": "_", "pos": "NN", "msd": "NN UTR|SIN|DEF|NOM", "hea
    {"id": 3, "form": "införs", "lemma": "_", "pos": "VB", "msd": "VB PRS|SFO", "head": 0, "deprel":
    {"id": 4, "form": "individuell", "lemma": "_", "pos": "JJ", "msd": "JJ POS|UTR|SIN|IND|NOM", "he
    {"id": 5, "form": "beskattning", "lemma": "_", "pos": "NN", "msd": "NN UTR|SIN|IND|NOM", "head":
    {"id": 6, "form": "(", "lemma": "_", "pos": "PAD", "msd": "PAD", "head": 5, "deprel": "IR"},
    {"id": 7, "form": "särbeskattning", "lemma": "_", "pos": "NN", "msd": "NN UTR|SIN|IND|NOM", "hea
    {"id": 8, "form": ")", "lemma": "_", "pos": "PAD", "msd": "PAD", "head": 5, "deprel": "JR"},
    {"id": 9, "form": "av", "lemma": "_", "pos": "PP", "msd": "PP", "head": 5, "deprel": "ET"},
    {"id": 10, "form": "arbetsinkomster", "lemma": "_", "pos": "NN", "msd": "NN UTR|PLU|IND|NOM", "h
    {"id": 11, "form": ".", "lemma": "_", "pos": "MAD", "msd": "MAD", "head": 3, "deprel": "IP"}
  ]
}
```

JSON (2)



- Mycket populär idag.
- Enkel representation: Objekt (i Javascript) med nyckel-värde-struktur.
    - Data i JSON består alltid av ett namn och ett värde: *”id”: 7* .
- Mindre platskrävande än XML, men mer explicit än tabulerade data.
- Skiljer inte data från metadata (som XML).

Ordklasser

Skolans 9 Ordklasser

Tagg	Kategori	Exempel
VB	verb	kasta
NN	substantiv	pudding
PN	pronomen	hon
JJ	adjektiv	glad
AB	adverb	inte
KN	konjunktion	och
PP	preposition	över
RG	räkneord	tre
IN	interjektion	aj

Finkornighet i Beskrivningen

- Varje korpus måste bestämma sig för ett system med vars hjälp de lingvistiska datan ska beskrivas.
    - 9 ordklasser eller 22?
    - Mer finkornighet tillåter fler analyser men blir mer komplex att annotera och att jobba med.
- Förutom själva korpusen måste det därför finnas en definition av de kategorier som använts.
    - Ofta tillsammans med konkreta exempel.

Ordklasser i SUC (1)

Tagg	Kategori	Exempel
NN	substantiv	*pudding*
VB	verb	*kasta*
PP	preposition	*över*
AB	adverb	*inte*
JJ	adjektiv	*glad*
PN	pronomen	*hon*
DT	determinerare	*denna*
KN	konjunktion	*och*
PM	egennamn	*Evelina*

https://spraakbanken.gu.se/parole/Docs/SUC2.0-manual.pdf

Ordklasser i SUC (2)

Tagg	Kategori	Exempel
PC	particip	*utsänd*
SN	subjunktion	*att*
RG	räkneord (grundtal)	*tre*
HP	frågande/relativt pronomen	*som*
IE	infinitivmärke	*att*
PL	partikel	*ut*

https://spraakbanken.gu.se/parole/Docs/SUC2.0-manual.pdf

Ordklasser i SUC (3)

Tagg	Kategori	Exempel
PS	possessivt pronomen	*hennes*
HA	frågande/relativt adverb	*när*
UO	utländskt ord	*the*
RO	räkneord (ordningstal)	*tredje*
IN	interjektion	*ja*
HD	frågande/relativ determinerare	*vilken*
HS	frågande/relativt possessivt pronomen	*vars*

https://spraakbanken.gu.se/parole/Docs/SUC2.0-manual.pdf

Generatorfunktioner

Vad är en Generatorfunktion?

- Ett sätt att använda data **utan att data lagras**.
    - Istället för att lagra en lista i minnet genereras värden vid behov.
- Ett sätt att **minska utnyttjandet av minne.**
    - Man kan iterera över stora dataset.

- På kurshemsidan finns en notebook med exempel och förklaringar.

Generatorfunktioner i Python

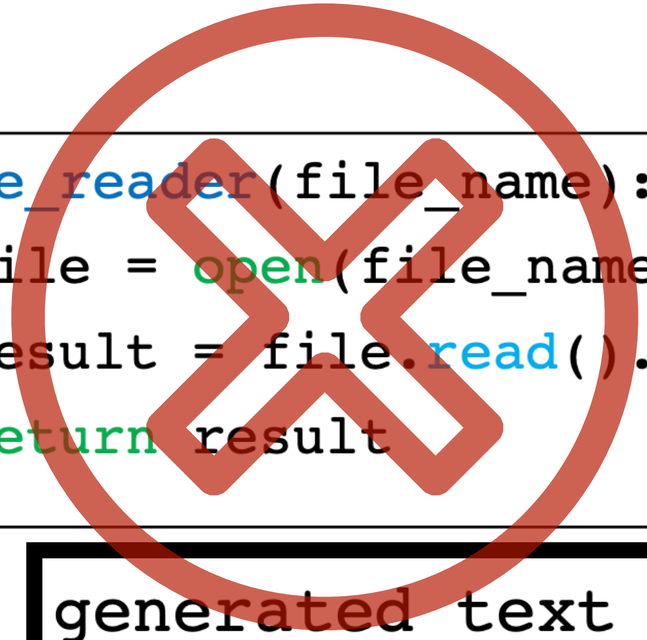
- När den kallas på så returnerar den en objekt men den exekveras inte automatiskt.
- Använder nyckelordet ***yield*** för att leverera relevanta data.
    - Kan innehålla ett eller fler *yield*.
- När funktionen når *yield* så pausas exekveringen.
    - **Lokala variabler och deras tillstånd behålls i minnet** mellan anropen.

```
def file_reader(file_name):
    file = open(file_name)
    result = file.read().split('\n')
    return result
```

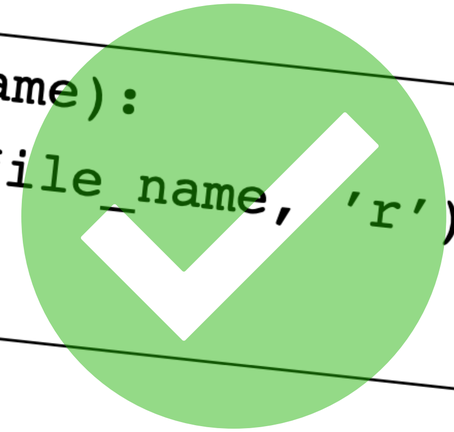
```
def file_reader(file_name):
    for row in open(file_name, 'r'):
        yield row
```

```
generated_text = file_reader('some_big.txt')


row_count = 0
for row in generated_text:
    row_count += 1


print('Row count is ' + str(row_count))
```

```
def file_reader(file_name):
        file = open(file_name)
        result = file.read().split('\n')
        return result
```



```
def file_reader(file_name):
        for row in open(file_name, 'r'):
                yield row
```



```
generated_text = file_reader('some_big.txt')


row_count = 0
for row in generated_text:
        row_count += 1


print('Row count is ' + str(row_count))
```

Notebook: Generatorfunktioner

L2

Tack för idag!