INF120: Oblig 3

Yngve Mardal Moe

Nov 24, 2021

Contents

I disse to obligene skal du lage din helt egne søkemotor for å søke i bøker. Søkemotoren skal være en Python funksjon som tar to input, en søkeindeks, og en søkestreng. Hvordan søkeindeksen er bygd opp står beskrevet senere. Det denne søkemotorfunksjonen (`søk_i_indeks_med_streng`) skal gjøre er å finne de indekserte bøkene som inneholder alle ordene i søkestrengen. Nedenfor er et eksempel, hvor vi søker etter strengen `Sherlock holmes, scarlet`. Ut får vi mengden med bøker som inneholder alle disse tre ordene.

```
>>> søk_i_indeks_med_streng(søkeindeks, "Sherlock holmes, scarlet")
{'Chronicles_of_Martin_Hewitt.bok', 'The_Hound_of_the_Baskervilles.bok'}
```

I andre del av oppgaven skal dere lage en slik søkeindeks fra en mappe med mange bøker i.

Chapter 1

Hva trenger dere for denne oppgaven

I mappen "bøker" ligger en rekke bøker som tekstfiler. Disse er hentet fra Project Gutenberg. Din jobb er å lage et verktøy som lar brukeren skrive inn en søkestreng. Programmet skal så printe filnavnene til bøkene som inneholder alle ordene i søkestrengen.

For å gjøre dette trenger du tre verktøy vi har lært om i forelesning:

- Dictionary
- Mengder (set)
- Strenger

I tillegg kommer vi til å bruke Path objektet i Python for å iterere over alle bokfilene.

Chapter 2

Hvordan skal dere arbeide med denne oppgaven

Last ned zip filen som inneholder oppgaven, i den filen ligger det en fire Python-filer: `indeks_søk.py`, `lag_indeks.py`, `__init__.py` og `ferdig_indeks.py`. Dere skal kun modifisere `indeks_søk.py` og `lag_indeks.py`. `__init__.py` er der for å gjøre jobben vår lettere, mens `ferdig_indeks` inneholder kode som laster inn en ferdig søkeindeks slik at dere kan søke i en indeks før dere har lagd kode for å lage deres egen søkeindeks. I tillegg ligger en mappe med bokfiler i `bøker` mappa, disse bokfilene skal dere indeksere selv.

Chapter 3

En søkeindeks

Søkemotoren er basert på en indeks som må bygges først. Å lage denne indeksen kan ta litt tid, men når indeksen er bygd en gang er det veldig raskt å søke etter ord. La oss starte med å forklare hvordan indeksen er bygd opp.

Vår søkeindeks skal bestå av nøkkel-verdi par hvor hver nøkkel er unike ord (f.eks. `"sherlock"`). Hver verdi er filnavnet til alle bøkene som innehodler det ordet (f.eks. `{'Chronicles_of_Martin_Hewitt.txt', 'The_Hound_of_the_Baskervilles.txt', 'In_the_Fog.txt'}`).

Chapter 4

Å slå opp i en søkeindeks

Når vi slår opp i en søkeindeks bruker vi en søkestreng. Eksempel på dette er `"Holmes, hounds"`. I vår indeks skal vi kun bruke små bokstaver og ingen spesialtegn, derfor må vi transformere søkestrengen slik at disse fjernes. Da vil eksempel strengen vår se slik ut: `"holmes hounds"`. Når strengen kun inneholder ord på samme format som søkeindeksen vår deler vi strengen opp i en liste med ord. I vårt eksempel blir det `["holmes", "hounds"]`. Så slår vi opp i indeksen vår etter disse ordene og får en mengde for hvert ord. I vårt eksempel får vi disse mengdene:

```
{'Chronicles_of_Martin_Hewitt.txt', 'The_Hound_of_the_Baskervilles.txt',
'In_the_Fog.txt'}
```

og

```
{'The_Hound_of_the_Baskervilles.txt', 'Martin_Hewitt_Investigator.txt'}
```

Det søkemotoren skal gi ut er snittet (intersection) av disse mengdene, det vil si mengden av elementer som er del av alle (begge i dette tilfellet) mengdene. I vårt eksempel er det singleton mengden:

```
{'The_Hound_of_the_Baskervilles.txt'}
```

Chapter 5

Å utvide en søkeindeks

La oss se for oss at vi har en søkeindeks. Altså en dictionary hvor hver nøkkel er enkeltord og verdiene er mengden av alle tekstfiler hvor enkeltordene dukker opp. La oss videre se for oss at vi vil legge til en ny tekstfil i denne indeksen.

For å legge til en ny tekstfil i en indeks må vi først lage en mengde med alle ord som oppstår i filen. For hvert ord i denne mengden så gjør vi dette

- **Sjekk om ordet allerede er en nøkkel i indeksen.**
  - **Hvis ordet er en nøkkel i indeksen:** Legg til det nåværende filnavnet til den korresponderende mengden.
  - **Hvis ordet ikke er en nøkkel i indeksen:** Lag et nytt nøkkel-verdi par i indeksen hvor nøkkelen er ordet og verdien er en mengde som kun inneholder det nåværende filnavnet.

Chapter 6

Å lage en ny søkeindeks

Når vi lager en ny søkeindeks så starter vi med en tom dictionary. Dette er startsindeksen vår. Deretter itererer vi over alle filene vi ønsker å indeksere og legger de til i søkeindeksen vår.

Chapter 7

Funksjoner dere skal lage

Her har dere en liste med funksjoner dere skal lage. Det anbefales at dere oppretter funksjoner i rekkefølge slik de er skrevet her. Grunnen til det er at senere funksjoner bygger på de du har lagd tidligere.

I *Notes* delene har vi skrevet hint dere kan lese hvis dere står fast med noen oppgaver og i *Examples* delene har dere eksempel input og output for funksjonene.

7.1 Oblig 2: Søke i en ferdig indeks

`indeks_søk.fjern_spesialtegn(`*streng*`)`

Fjern de følgende spesialtegnene fra input strengen: `[',', '.', '"', '\'', ':', ';', '(', ')', '-', '?', '!', '\n']`

Mellomrom på slutten og starten av linjer skal også fjernes.

Parameters `streng` (*str*) – Input strengen som spesialtegn skal fjernes fra

Returns **ren_streng** – Strengen etter at spesialtegn er fjernet.

Return type str

Notes

Vi kan bruke `replace` funksjonen for å bytte substrenger med andre strenger.

```
>>> streng = "abc123abc"
>>> streng.replace('a', 'e')
ebc123ebc
```

Examples

```
>>> streng = "abc1, hei på deg. Hva heter du?"
>>> fjern_spesialtegn(streng)
"abc1 hei på deg Hva heter du"
```

```
>>> streng = "  Hei på deg!!!\n"
>>> fjern_spesialtegn(streng)
"Hei på deg"
```

`indeks_søk.finn_unike_ord_i_streng(`*streng*`)`

Lag en mengde med alle ordene som dukker opp i strengen.

Pass på at strengen kun inneholder små bokstaver (hvis store bokstaver er med i strengen skal de bli gjort om til små bokstaver). Mellomrom på slutten og starten av linjer skal også fjernes.

De følgende spesialtegn må og fjernes: `[',', '.', '"', '\'', ':', ';', '(', ')', '-', '?', '!', '\n']`

Parameters `streng` (*str*) – Strengen vi vil finne unike tegn i.

Returns **ord_i_streng** – Mengden med unike ord i strengen.

Return type Set[str]

Notes

Vi kan splitte opp strenger med `split` funksjonen.

```
>>> tekst = 'abc 123'
>>> print(tekst.split(' '))
['abc', '123']
```

Examples

```
>>> streng = "Nå arbeider vi med INF120. Faktisk arbeider vi med siste␣
↪oblig i INF120!"
>>> finn_unike_ord_i_streng(streng)
{'nå', 'arbeider', 'vi', 'med', 'inf120', 'faktisk', 'siste', 'oblig', 'i
↪'}
```

Merk at rekkefølgen på ordene ikke spiller noen rolle!

`indeks_søk.finn_felles_ellement_i_flere_mengder(`*liste_av_mengder*`)`

Lag en funksjon som finner felles element i en samling av mengder (set).

Parameters `liste_av_mengder` (*List[Set]*) – En liste hvor hvert element er en mengde (set på engelsk).

Returns **snitt_av_mengder** – En mengde som kun inneholder de elementene som er del av ALLE mengdene i `liste_av_mengder`.

Return type Set[str]

Notes

Vi kan finne snittet mellom to mengder med `intersection` funksjonen.

```
>>> mengde1 = {1, 2, 3}
>>> mengde2 = {2, 3, 4}
>>> mengde1.intersection(mengde2)
{2, 3}
```

Om vi tar snittet mellom en mengde og seg selv så endres ingen ting.

```
>>> mengde1 = {1, 2, 3}
>>> mengde1 = mengde1.intersection(mengde1)
>>> print(mengde1)
{1, 2, 3}
```

Examples

```
>>> mengde1 = {1, 2, 3}
>>> mengde2 = {2, 3, 4}
>>> liste_av_mengder = [mengde1, mengde2]
>>> finn_felles_element_i_flere_mengder(liste_av_mengder)
{2, 3}
>>> mengde3 = {3, 4, 5}
>>> liste_av_mengder = [mengde1, mengde2, mengde3]
>>> finn_felles_element_i_flere_mengder(liste_av_mengder)
{3}
```

`indeks_søk.søk_i_indeks_med_mengde`(*indeks*, *mengde_av_søkeord*)

Finn alle dokument som inneholder alle søkeordene i `mengde_av_søkeord`.

Denne funksjonen skal ta to argument som input, en søkeindeks og en mengde med søkeord.

Søkeindeksen er en dictionary med engelske ord som nøkler og mengden med alle dokument som inneholder det ordet som verdi.

Mengden med søkeord er en mengde (set på Engelsk) som beskriver hva som søkes etter.

Det som returneres er mengden med dokument som inneholder ALLE søkeordene.

Parameters

- `indeks` (*dict[str] -> Set[str]*) – Søkeindeksen.
- `mengde_av_søkeord` (*Set[str]*) – Mengden med søkeord.

Returns **relevante_bøker** – Mengden med bøker som inneholder alle ordene i `mengde_av_søkeord`.

Return type Set[str]

Notes

Husk `finn_felles_ellement_i_flere_mengder` funksjonen din.

Vi kan teste om et element er en nøkkel i en dictionary med `in` nøkkelordet

```
>>> d = {'a': 1, 'b': 2}
>>> 'a' in d
True
>>> 1 in d
False
```

Vi kan lage en tom mengde med `set` funksjonen.

```
>>> tom_mengde = set()
>>> print(tom_mengde)
{}
```

Examples

```
>>> indeks = last_inn_indeks()
>>> søk_i_indeks_med_mengde(indeks, {"sherlock", "holmes", "scarlet")
{'Chronicles_of_Martin_Hewitt.bok', 'The_Hound_of_the_Baskervilles.bok'}
>>> søk_i_indeks_med_mengde(indeks, {"Dette", "er", "ikke", "i", "indeksen
↪"})
{}
```

`indeks_søk.klargjør_søkestreng(`*søkestreng*`)`

Ta inn en søkestreng og klargjør den for å søke i en søkeindeks.

Strengen skal behandles på samme måte som vi behandler nye strenger som skal indekseres. Store bokstaver skal gjøres om til små, spesialtegn skal fjernes og "whitespace" tegn på starten og slutten av strengen skal fjernes. Til slutt skal strengen splittes ved alle mellomrom og duplikatord skal fjernes.

Parameters **søkestreng** (*str*) – Strengen vi vil finne unike tegn i.

Returns **ord_i_streng** – Mengden med unike ord i strengen.

Return type Set[str]

Notes

Kan du gjenbruke en funksjon du lagde tidligere i obligen?

Examples

```
>>> streng = "abc1, hei på deg. Hva heter du?"
>>> klargjør_søkestreng(streng)
"abc1 hei på deg Hva heter du"
```

```
>>> streng = "  Hei på deg!!!\n"
>>> klargjør_søkestreng(streng)
"Hei på deg"
```

`indeks_søk.søk_i_indeks_med_streng`(*indeks*, *søkestreng*)

Finn alle dokument som inneholder alle søkeordene i `søkestreng`.

Denne funksjonen skal ta to argument som input, en søkeindeks og en mengde med søkeord.

Søkeindeksen er en dictionary med engelske ord som nøkler og mengden med alle dokument som inneholder det ordet som verdi.

Søkestrengen skal først klargjøres. Dette gjøres ved å gjøre strengen til små bokstaver og å fjerne spesialtegn. I tillegg skal og whitespace på starten og slutten av strengen fjernes. Deretter skal hvert ord i søkestrengen hentes ut. Disse ordene brukes når det skal søkes i de indekserte dokumentene.

Det som returneres er mengden med dokument som inneholder ALLE søkeordene.

Parameters

- `indeks` (*dict*) – Søkeindeksen.
- `mengde_av_søkeord` (*str*) – Mengden med søkeord.

Returns **relevante_bøker** – Mengden med bøker som inneholder alle ordene i `mengde_av_søkeord`.

Return type Set[str]

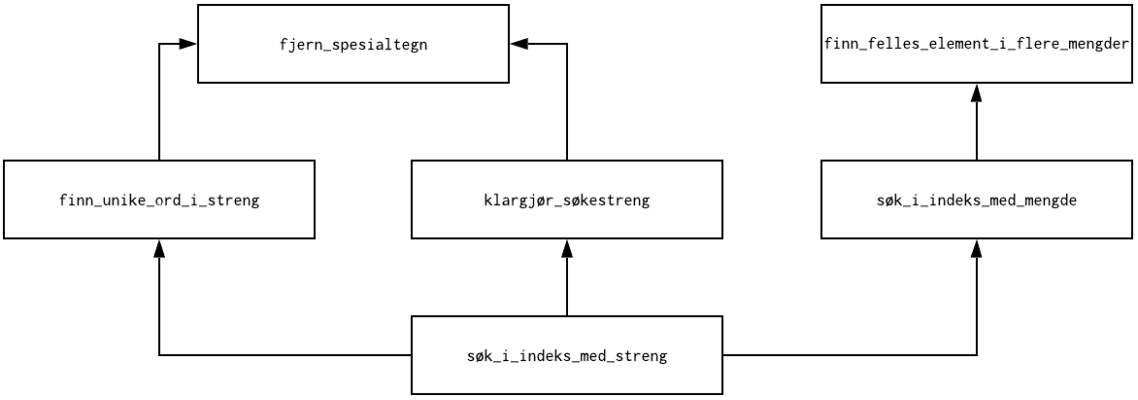
Notes

Husk `søk_i_indeks_med_mengde` og `klargjør_søkestreng` funksjonene dine.

Examples

```
>>> indeks = last_inn_indeks()
>>> søk_i_indeks_med_streng(indeks, "Sherlock Holmes, scarlet")
{'Chronicles_of_Martin_Hewitt.bok', 'The_Hound_of_the_Baskervilles.bok'}
>>> søk_i_indeks_med_streng(indeks, "Dette er ikke i indeksen")
{}
```

7.1.1 Funksjonskalldiagram



Her ser dere et bilde som viser hvilke funksjoner som skal brukes i hver funksjon dere lager. Hvis en pil starter i funksjon *a* og ender i funksjon *b*, så skal funksjon *a* bruke funksjon *b*.

7.2 Oblig 3: Lage en søkeindeks

`lag_indeks.finn_alle_unike_ord_i_liste_av_strenger(`*liste_av_strenger*`)`

Lag en mengde med alle ordene som dukker opp i strengene i lista.

Pass på at strengene kun inneholder små bokstaver (hvis store bokstaver er med i strengene skal de bli gjort om til små bokstaver). Mellomrom på slutten og starten av strengene skal også fjernes.

De følgende spesialtegn må og fjernes: “[‘,’, ‘.’, “”’, ‘”, ‘:’, ‘;’, ‘(’, ‘)’, ‘-’, ‘?’, ‘!’, ‘

‘]“

liste_av_strenger [str] Strengene vi vil finne unike tegn i.

ord_i_streng [Set[str]] Mengden med unike ord i alle strengene.

Vi kan oppdatere en tom mengde med `set` funksjonen og legge til mange element om gangen med `update` funksjonen.

```
>>> mengde = set()
>>> print(mengde)
{}
>>> mengde.update([1, 2, 'a'])
>>> print(mengde)
{1, 2, 'a'}
>>> mengde.update([2, 'a', 'b', 1])
```

(continues on next page)

(continued from previous page)

```
>>> print(mengde)
{'b', 1, 2, 'a'}
```

Se forelesningvideoen for 13. mars og 19. mars.

Husk `finn_unike_ord_i_streng` du lagde før denne!

```
>>> strenger = ["Nå arbeider vi med INF120.", "Faktisk arbeider␣
↪vi med siste oblig i INF120!"]
>>> finn_alle_unike_ord_i_liste_av_strenger(strenger)
{'nå', 'arbeider', 'vi', 'med', 'inf120', 'faktisk', 'siste',
↪'oblig', 'i'}
```

Merk at rekkefølgen på ordene ikke spiller noen rolle!

`lag_indeks.finn_unike_ord_i_bok(`*bokfil*`)`

Lag en mengde med alle ordene som dukker opp i en bok.

Pass på at strengen kun inneholder små bokstaver, og at mellomrom på slutten og starten av linjer er fjernet.

De følgende spesialtegnene skal og fjernes: “[‘,’, ‘.’, ””’, ”, ‘:’, ‘;’, ‘(’, ‘)’, ‘-’, ‘?’, ‘!’, ‘

‘]“

bokfil [str eller pathlib.Path] Filen som inneholder boka.

ord_i_bok [Set[str]] Mengden med unike ord i boka

Om du prøver å gjøre et Path objekt om til en Path så endrer du ingen ting.

```
>>> fil = 'bøker/The_Works_of_Edgar_Allan_Poe.txt'
>>> fil = Path(fil)
>>> fil
WindowsPath('bøker/The_Works_of_Edgar_Allan_Poe.txt')
>>> fil = Path(fil)
>>> fil
WindowsPath('bøker/The_Works_of_Edgar_Allan_Poe.txt')
```

MERK: Om du bruker Mac eller Linux vil det være en UnixPath istedenfor en WindowsPath.

Husk: Du lagde en funksjon som finner unike ord i en liste av strenger!

```
>>> indeks_søk.finn_unike_ord_i_streng("Hei, hei, på deg.")
{'hei', 'på', 'deg'}
```

`lag_indeks.legg_til_bok_i_indeks(`*indeks*, *bokfil*`)`

Denne funksjonen skal legge til en ny bok i indeksen.

Måten denne nye boken skal legges til i indeksen er følgende

1. Finn de unike ordene i boka.
2. Iterer gjennom hvert unikt ord og sjekk om det er del av indeksen allerede.
   - Hvis det er del av indeksen skal tittelen på denne bokfilen legges til i den korresponderende mengden med filnavn.
   - Hvis det ikke er del av søkeindeksen, så skal legges til som nøkkel i søkeindeksen, hvis korresponderende verdi skal være mengden som inneholder tittelen på denne bokfilen.

Parameters

- `indeks` (*Dictionary[str] -> Set[str]*) – En søkeindeks. Hver nøkkel er engelske ord som forekommer i minst en av bøkene som er indeksert. Nøkkelen "sherlock" peker på en mengde som inneholder filnavnet til alle bøker som inneholder ordet "sherlock".
- `bokfil` (*Path or str*) – Filplasseringen til bokfilen som skal legges til i søkeindeksen.

Returns **indeks** – En oppdatert søkeindeks. Hver nøkkel er engelske ord som forekommer i minst en av bøkene som er indeksert. Nøkkelen "sherlock" peker på en mengde som inneholder filnavnet til alle bøker som inneholder ordet "sherlock".

Return type Dictionary[str] -> Set[str]

Notes

Vi kan legge til nye element i en mengde ved å bruke `mengde.add` funksjonen.

```
>>> mengde = set(['a', 'hei', 'hei', 'a'])
>>> print(mengde)
{'a', 'hei'}
>>> mengde.add('b')
>>> print(mengde)
{'a', 'hei', 'b'}
>>> mengde.add('a')
{'a', 'hei', 'b'}
```

Når du sjekker om et element er en del av en dictionary så sjekker du om det elementet er en nøkkel i den dictionaryen.

```
>>> indeks = {'a': {'bok1.bok', 'bok2.bok'}, 'is': {'bok1.bok'}}
>>> 'a' in indeks
True
>>> 'c' in indeks
False
```

Vi kan få filnavnet til en path ved å se på `name` attributten.

```
>>> bokfil = Path('bøker/Benefactor.bok')
>>> bokfil.name
Benefactor.bok
```

`lag_indeks.indekser_bøker(`*mappe*`)`

Lag en søkeindeks med alle filene i den spesifiserte mappen.

Her skal du starte med å lage en tom søkeindeks, det vil si en tom dictionary. Så skal du iterere gjennom hver bokfil i den spesifiserte mappen og legge alle til i indeksen.

Parameters `mappe` (*`Path or str`*) – Mappen vi skal søke etter filer i. Hvis denne er en streng skal den gjøres om til en Path.

Returns **indeks** – En søkeindeks over alle bokfiler i den spesifiserte mappen.

Return type Dictionary[str] -> Set[str]

Notes

Se forelesningsvideoen den 27. mars.

Om du prøver å gjøre et Path objekt om til en Path så endrer du ingen ting.

```
>>> fil = 'bøker/The_Works_of_Edgar_Allan_Poe.bok'
>>> fil = Path(fil)
>>> fil
WindowsPath('bøker/The_Works_of_Edgar_Allan_Poe.bok')
>>> fil = Path(fil)
>>> fil
WindowsPath('bøker/The_Works_of_Edgar_Allan_Poe.bok')
```

MERK: Om du bruker Mac eller Linux vil det være en UnixPath istedenfor en WindowsPath, men funksjonaliteten er den samme.

Du kan iterere gjennom alle filer med en spesifikk filtype med `Path.glob` funksjonen

```
>>> mappe = Path('bøker')
>>> for bokfil in mappe.glob('*.bok'):
...     print(bokfil)
bøker\A_Journey_to_the_Centre_of_the_Earth.bok
bøker\Benefactor.bok
...
bøker\Vulcans_Workshop.bok
```

Python Module Index

Index