Sek 4.3 i kompendiet
Hvordan tekst representeres på en datamaskin
Faktaboks 4.9:

På en datamaskin representeres tekst/tegn ved hjelp av heltallskoder. En tabell blir brukt for å assosiere ethvert tegn med en slik kode.

Vi trenger tre ting:
1. Et tegnsett (A, B, C, ..., a, b, c, ..., 1, 2, 3, ..., %, $, #)
2. En tabell med en kode for hver tegn → code point i kompendiet.
3. En regel for hvordan maskinen skal representere koden ved hjelp av bytes. (1 byte = 8 bits)

ASCII - tegnsettet:
128 - tegn
kodene fra 0 til 127 (trenger 7 bits)
maskinen bruker 1 byte per tegn, setter det mest signifikante bitet til 0. (0 1111111) 127

A, B, C, ...	:	65, 66, ... - 90
a, b, c, ...	:	97, 98, ... - 122
0, 1, 2, ..., 9	:	48 - 57
#, $, %, &	:	35, 36, 37, 38

Legg merke til at æ, ø, å IKKE er med i ASCII-tegnsettet!
Trenger et større tegnsett for at vi skal kunne skrive dokumenter på norsk.
ISO-Latin tegnsett (definert på 80-tallet) har forskjellige varianter for hvert språk
inneholder 256 tegn
ISO-Latin 1 (western) inneholder, æ, ø, å

197	Å	229	å
198	Æ	230	æ
216	Ø	248	ø

koderne for 0-127 er de samme som i ASCII
90-tallet: Formatet Unicode definerte tegnsett som inneholder de fleste tegn for de fleste språk.
100 000 tegn!
Unicode-tabellen: koder fra 0 til $1114\,111$
bare 10% av kodene for $1114\,111$ er i bruk.
Gjør det lett å legge til tegn senere. $10ffff_{16}$
ASCII, ISO-LATIN 1: Tegnet representeres med 8 bits = 1 byte ved hjelp av koden i tabellen.
Alle tegn i Unicode-tabellen kan representeres med 21 bits, og vi trenger da å bruke 3 bytes.
I praksis ville vi kaste bort plass hvis vi allokerte 3 bytes for alle tegn. Derfor har man laget "enkodinger" som kan kode tegn, ved å bruke forskjellig antall bytes per tegn

↓
regel for å skrive en kode om til et sett av bytes.

4.3.4 UTF-8 enkoding (av Unicode-tegnsettet).

Regel: Første bits i enhver byte er en av

0, 10, 110, 1110, 11110

- 0 → ASCII
- 10 → byte 2, 3 eller 4 i tegn som blir rep. med flere bytes.
- 110, 1110, 11110 → første bits når vi bruke 2 bytes (3, 4) til å rep. tegn

Eksempel 4.12

norsk Å: har Unicode-kode: 197

$$197 = 128 + 64 + 4 + 1 = \underbrace{1100}_{C_{16}}\ \underbrace{0101}_{5_{16}}{}_2 = C5_{16}$$
($2^7\ \ 2^6\ \ 2^2\ \ 2^0$)

vi ser at $128 \leq 197 \leq 2047$, slik at regel 2 for UTF-8 enkoding skal brukes ($\Rightarrow$ to bytes)

regel 2: må representeres med 11 bits: 000 1100 0101 ($d_{10} d_9 \ldots d_0$)

UTF-8 sier at dette skal kodes som

$$110\,d_{10}d_9d_8\ \ d_7d_6\ \ 10\,d_5d_4d_3\ \ d_2d_1d_0 = \underbrace{110\ 0001}_{8+4=12\ =C_{16}\ \ 3_{16}}\ \underbrace{10\ 000101}_{8_{16}\ \ 5_{16}}$$

$$= C3\,85_{16}$$

Viktigst for dere å huske:
1. UTF-8 koder med 1, 2, 3, eller 4 bytes.
2. UTF-16 koder med 2, eller 4 bytes.

Kontrollspørsmål: Hvis en tekst er kodet med 3413 bytes. Hvilke av UTF-8 eller UTF-16 kan denne ha vært kodet med?

python: # coding = UTF-8
Eks 4.4 4b 6e (75) (74)
Knut
Mørken

ø kan ikke lagres som ren ascii.
latin1 ø har unicode-kode f8
(hvis koder med latin1, så skriver vi f8 for ø).

UTF-8: f8 = 1111 1000 = 000 1111 1000

⇒ kodes som 110 00011 10 11 1000 = c3 b8$_{16}$
(c 3 b 8)

når koder med UTF-8: ø kodes som c3 b8,16
ellers kodes alt som med ISO-Latin1.
UTF-16: kodes alt som det står; med to bytes.
ø (f8) kodes som 00f8.