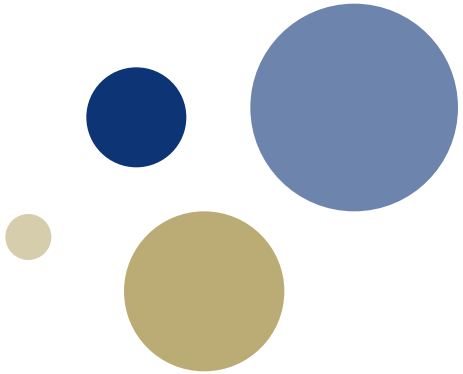
NTNU

Kunnskap for en bedre verden



Python: Valg og betingelser

TDT4110 IT Grunnkurs

Professor Guttorm Sindre

Læringsmål og pensum

- Mål
  - Kunne forstå og bruke if-setninger
  - … sammenlikning av strenger
  - … nøstede beslutningsstrukturer
  - … betingelser og uttrykk med logiske operatorer
  - … boolske variable
- Pensum
  - Starting out with Python: Chapter 4 / Chapter 3

  Decision Structures and Boolean Logic

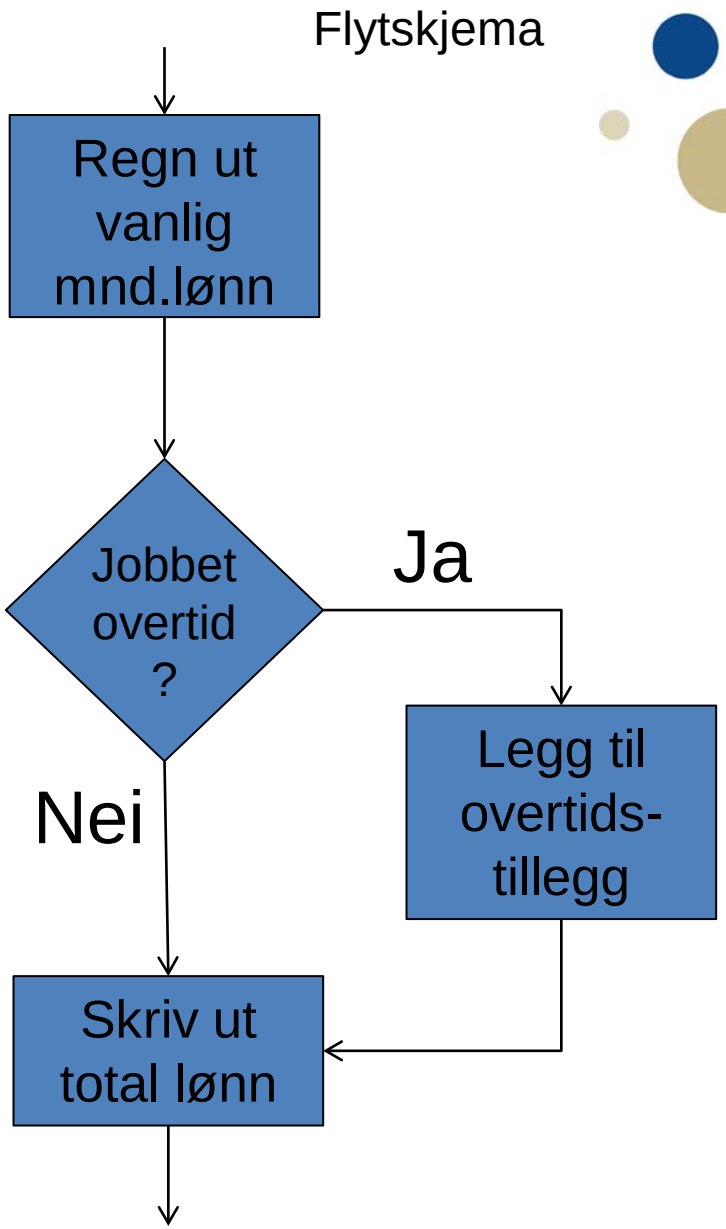
if-setningen

Kapittel 3.1

if-setninger

- HVORFOR trenger vi dette:
  - Ta beslutninger
  - Situasjonsbetingede handlinger
- HVORDAN virker if-setninger
  - Hvis en betingelse er tilfredsstilt, utføres handling (en eller flere kodelinjer)
  - Ellers utføres den ikke
  - Fortsetter deretter med kode som står etter if-setningen
  - **INNRYKK** viser hva som er del av if-setningen og hvor den slutter
- Syntaks:

```
if betingelse:
    kodelinje
    kodelinje
    etc.
```



Betingelser i Python

- *relasjonsoperatorer* brukes ofte i betingelser
  - A la de *aritmetiske* operatorene +, -, *, /
  - Kan sammenligne
    - to tall (eller uttrykk som resulterer i tall)
    - tegn, tekster, boolske variable, etc.
- NB: ”er lik” i Python: ==
  - (mens = betyr tilordning)

Python	Matematikk	Forklaring
< >	< >	Mindre enn Større enn
<= >=	≤ ≥	Mindre eller lik Større eller lik
==	=	Er lik
!=	≠	Er ulik

Eksempler på betingelser

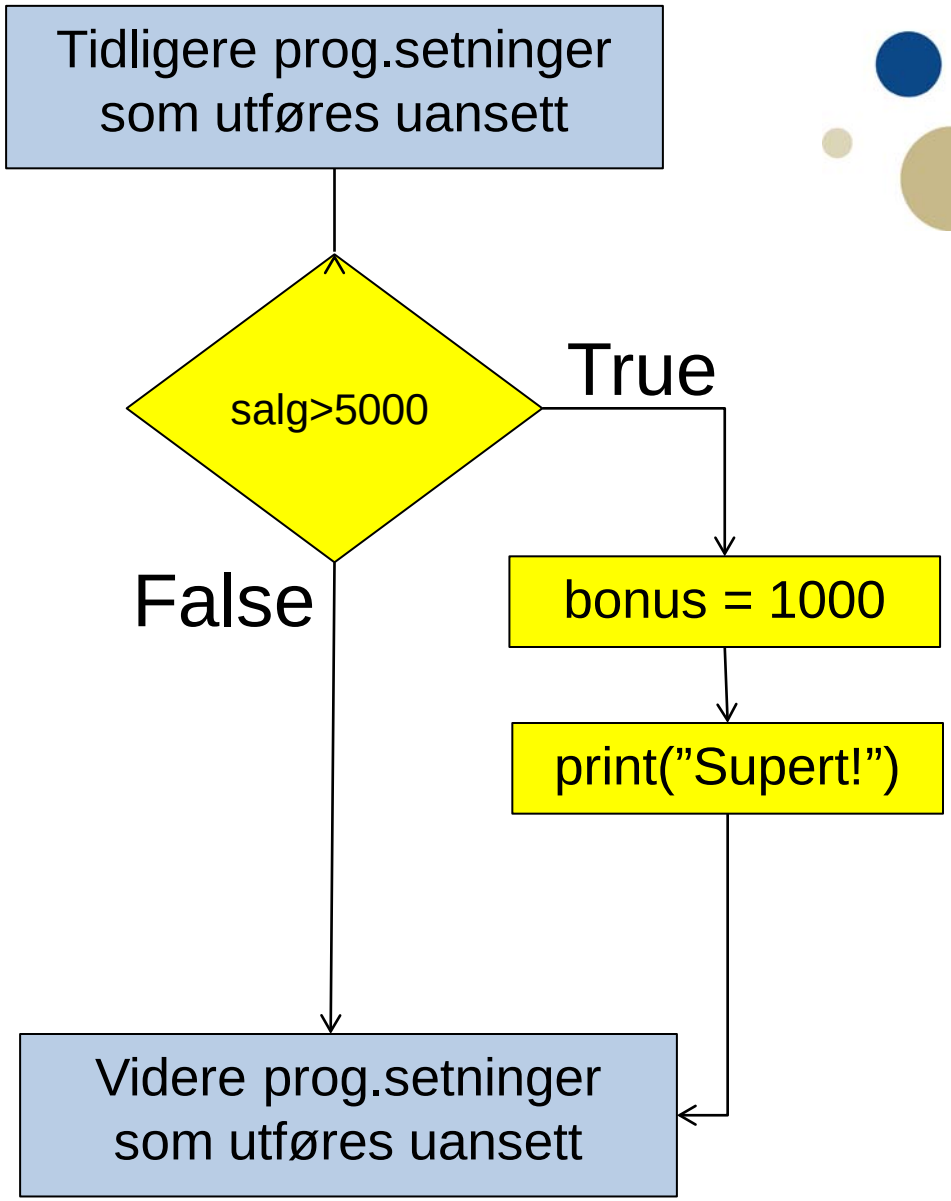
Betingelse	Verdi
4 < 3	usann (False)
4==4	sann (True)
3!=3	usann (False)
3 > 3	usann (False)
3>=3	sann (True)

if-setningen

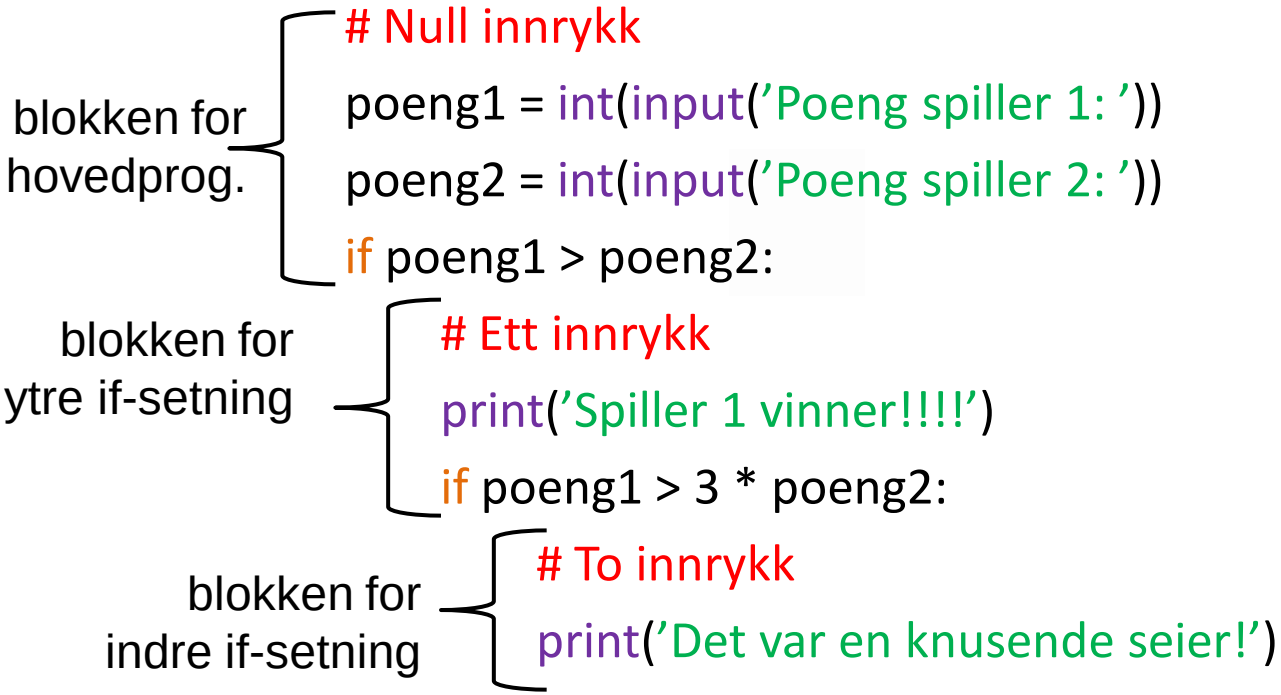
- if-eksempel:

```
bonus = 0
salg = int(input('Totalt salg: '))
if salg > 5000:
    bonus = 1000
    print('Supert!')
provisjon = salg * 0.3 + bonus
print('Provisjon:', provisjon)
```

Innrykk viser hva som tilhører if-setningen!



Nøstede blokker



if-else uttrykk

Kapittel 3.2

if-else uttrykk

- Et if-else uttrykk vil
  - kjøre en kodeblokk hvis betingelsen er sann (True)
  - en kodeblokk hvis usann (False).
- Velge mellom to alternative handlinger:
  - Det vi skal gjøre hvis betingelsen er sann:
    - puttes under if…
  - Det vi skal gjøre hvis betingelsen er usann:
    - puttes under else…
  - Det vi skal gjøre uansett om betingelsen er sann eller usann
    - Gjøres enten FØR hele if-else-setningen starter
    - Eller ETTER at hele if-else-setningen er slutt

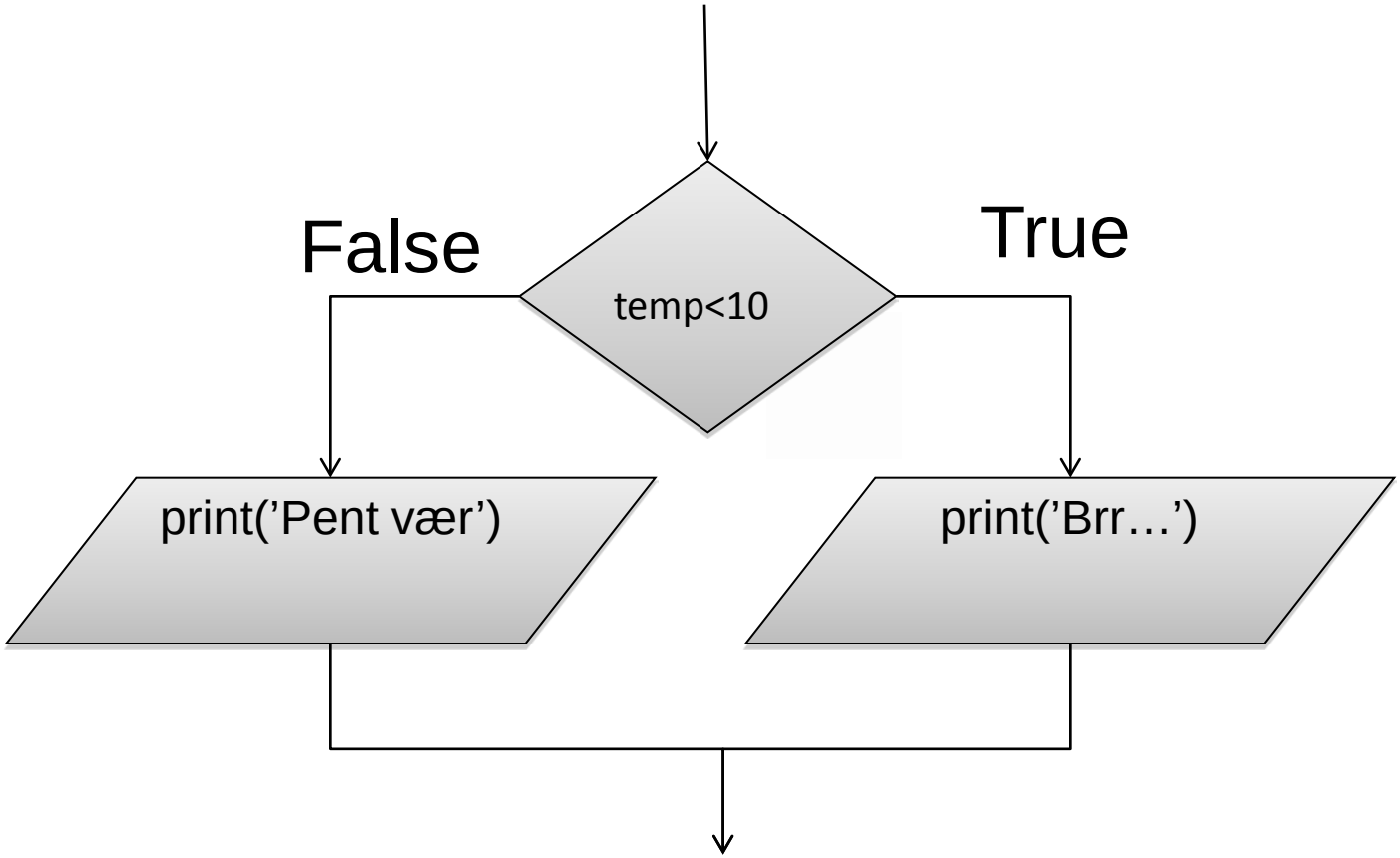
if-else generell kode

```
if betingelse:
    kode
    kode
    etc.
else:
    kode
    kode
    etc.
```

Denne kodeblokka blir utført hvis betingelsen er **sann**

Denne kodeblokka blir utført hvis betingelsen er **usann**

if-else flytskjema



if-else kodeeksempel

```
temp = int(input('Temperaturen er (heltall)? :'))
if temp < 10:
    # Betingelsen er sann
    # Innrykk for kodeblokka
    print('Brrr!')
else:
    # Betingelsen er usann
    print('Deilig!')
```

Oppgave: if…else - SKUDDÅR

Lag et program som sier om et år var skuddår

```
>>> ================================ RESTART =======
>>>
Skriv inn årstall som et helt tall: 1599
1599 var vanlig år.
Det hadde 365 dager.
Håper dette var en nyttig opplysning, takk for nå!
>>> ================================ RESTART =======
>>>
Skriv inn årstall som et helt tall: 1604
1604 var skuddår.
Det hadde 366 dager.
Håper dette var en nyttig opplysning, takk for nå!
>>>
```

LETT VARIANT:
Skuddår hvis årstall delelig på 4.
Ellers ikke.

(Disse reglene Gjaldt 8 e.Kr. – 1581)

MIDDELS VARIANT:
Dagens regler (1582-):
Skuddår hvis delelig på 4
Men ikke hvis delelig på 100
Men likevel hvis delelig på 400

VANSKELIG VARIANT:
Svar historisk korrekt for alle perioder:
- 46 f.Kr.: ingen skuddår
45 f.Kr. – 9 f.Kr.: delelig på 3
8 f.Kr. – 7 e.Kr.: ingen skuddår
8 e.Kr. – 1581: delelig på 4
1582 – : dagens regler

Bruk negative tall for f.Kr.

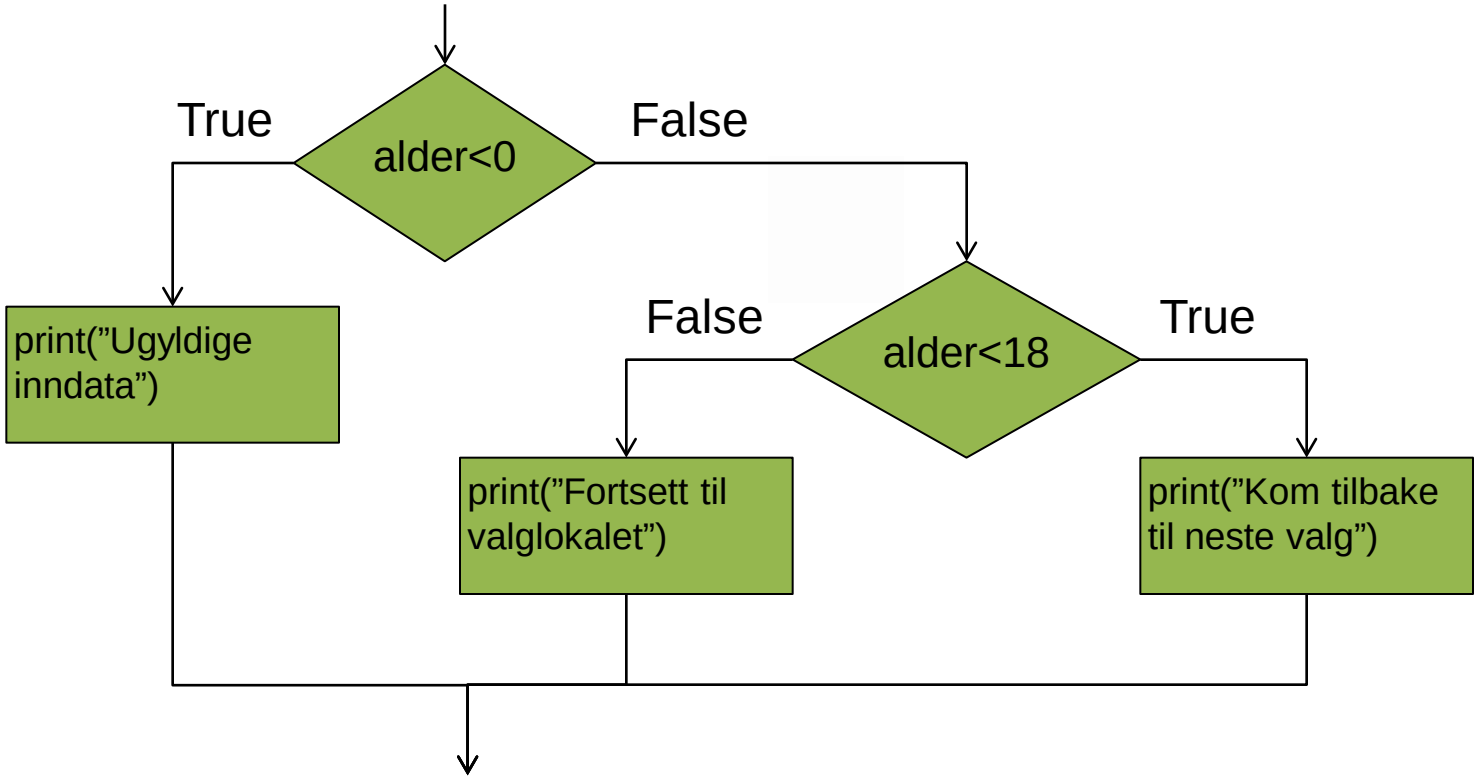
Startfil: skuddaar-uferdig.py

Nøstede betingelser og if-elif-else

Kapittel 3.4

Flytskjema for nøstede if-setninger

- Vi kan skrive flere if-setninger inne i hverandre (nøsting)



Nøsting av if-setninger

- Kodeeksempel på *nøstede* if-setninger:

```
if (alder < 0):
    print(”Ugyldige inndata”)
else:
    if (alder < 18):
        print(”Kom tilbake til neste valg”)
    else:
        print(”Forsett til valglokalet”)
```

- *innrykk* viser at de indre setningene hører til if-setningen.
- En if-setning avsluttes ved å fjerne innrykk.
- Kan ha flere nivåer med if-setninger inni hverandre.

Hvordan tenke når problemet har flere, komplekse betingelser

- Er betingelsene avhengige av hverandre eller uavhengige?
  - Uavhengige betingelser: Separate if-setninger
  - Gjensidig utelukkende: Ulike grener av if-(elif)-else
  - Valg kun relevant gitt et visst utfall av et foregående valg:
    - Nøstes inni den forrige if-setningen
- Hvilken rekkefølge må betingelsene stå i?
  - Feil rekkefølge / struktur kan gjøre at ingen / alle slår til
- Trade-off / smak og behag:
  - Enkle betingelser men gjenta samme utfall flere steder?
  - Eller: hvert utfall bare ett sted, men komplekse betingelser
- Mulige hjelpeverktøy: beslutningstrær / - tabeller

Bruk av if-elif-else

- Nøsting av setninger kan fort bli uoversiktlig
- Python har derfor **elif** for bedre lesbarhet.
  - elif er en forkortelse for **else if** (hvis ikke det ovenfor slår til, så….)
- Eksempel: Fartsbøter i 60-sone

```
if fart > 85:
    print('Inndragning av førerkort')
elif fart > 80:
    print('Bot 6500 kroner')
elif fart > 75:
    print('Bot 4200 kroner')
elif fart > 70:
    print('Bot 2900 kroner')
elif fart > 65:
    print('Bot 1600 kroner')
elif fart > 60:
    print('Bot 600 kroner')
else:
    print('Ingen bot')
```

Denne typen setningsstruktur passer hvis kun ett utfall skal velges. Pass på rekkefølgen!

Logiske uttrykk

- På samme vis som vi har sammensatte *aritmetiske* uttrykk kan vi sette sammen betingelser til vilkårlig store uttrykk
- Dette kaller vi *logiske uttrykk*
- Vi kaller ”limet” som binder disse sammen for *logiske operatorer*
- Python definerer de følgende logiske operatorene slik:

Operator i Python	Forklaring
and	Logisk og
or	Logisk eller
not	Logisk ikke, eller negasjon

Logiske uttrykk (fortsettelse)

- Hva betyr **`and`**, **`or`** og **`not`** i praksis:
  - **`and`** gir True kun hvis begge sider er True:

    `False and True` gir False

    `False and False` gir False

    `True and True` gir True
  - **`or`** gir True hvis minst en av sidene er True:

    `False or True` gir True

    `True or True` gir True

    `False or False` gir False
  - **`not`** gir motsatt sannhetsverdi av den vi hadde:

    `not True` gir False

    `not False` gir True

Eksempel på logiske uttrykk

- Vanlig bruk er å sjekke at en verdi ligger i et intervall:

```
x >= 5 and x <= 10
```

- kan også sjekke intervaller på følgende måte:

```
5<= x <= 10
```

- Parenteser for større uttrykk med flere betingelser:

```
(i>=1 and i<=N) or (j>=1 and j<=N)
```

For å få riktige betingelser husk presedens-rekkefølgen:

1. `**` # Eksponent (opphøyd)
2. `*, /, //, %` # heltallsdivisjon, rest
3. `+, -`
4. `<, <=, >, >=, <>, !=, ==`
5. `not`
6. `and`
7. `or`
8. `if – else`

- Bruk parenteser hvis nødvendig
  - De evalueres innenfra og ut

Betingelser og kortslutning

- Generelt evalueres betingelser fra venstre mot høyre
- Kortslutningsevaluering (”short circuit evaluation”):
  - Hvis resultat er åpenbart før vi når slutten
  - Python dropper å evaluere resten
- Eksempel, X er en boolsk variabel eller uttrykk
  - (True or X) vil bli True uansett verdi av X
  - (False and X) vil bli False uansett verdi av X

```
x = 5
d = 0
if d == 0 or x / d > 2:  # vil virke selv om deling på 0 gir tull
    print(’OK’)
```

Bestem True eller False

Anta at variablene har fått verdier som følger:

A = 5

B = 9

C = 12

D = 39

Hva blir den boolske verdien av uttrykkene?

- (B <= C)
- (A > 5 or B == 2)
- (A + B < C + D) and (D >= 39)
- (A > B or B > C or C > D or D > A)

Boolske variable

Kapittel 3.6

Boolske variable

- En boolsk variabel kan referere til en av to verdier:
  - True eller False.
- Kan lagre at en spesiell betingelse er sann eller ikke.

```
aar=int(input('Skriv inn årstall: '))
skuddaar = aar % 4 == 0          #blir True/False
# N linjer lenger nede i programmet…
if skuddaar:
        antall_dager=366
# ytterligere M linjer lenger nede i programmet…
if skuddaar:
        antall_d_feb=29
else:
        antall_d_feb=28
```

Avslutning if-setninger

- Unngå overflødig bruk av `not` – tungt å lese
- Ved if else,
  - skriv helst positiv utfall i if og negativt i else
- Vi kan ha flere setninger mellom if … else
  - Innrykk avgjør hvor koden hører til
- Typiske feil man kan gjøre
  - Noen setninger kan umulig slå til
    - feil betingelse, eller feil rekkefølge på betingelser
  - Mer enn en setning slår til
    - brukt flere if hvor det skulle vært elif?

Oppgave: Skuddår igjen

- Gjør programmet for skuddår kortere
  - Bruk **and, or, not**
    - Kan dermed klare oss med en enkel if-else-setning
    - Unngå nøsting eller bruk av **elif**

Sammenlikne strenger

Kapittel 3.3

Sammenlikne to strengvariable

- Variabler som inneholder tekststrenger kan sammenlignes
- Eksempel, sjekke om to variable har likt innhold:

```
ditt_navn = input('Skriv inn ditt fornavn: ')
fars_navn = input('Skriv inn din fars fornavn: ')
if ditt_navn == fars_navn:
    print('Dere har samme navn')
else:
    print('Dere har ulike navn')
```

Sammenlikne variabel og streng

- Sjekke om en variabel inneholder en tekst
  - Eksempel: Sjekke om bruker har svart Ja eller Nei
  - Utfordring: svar kan være 'JA', 'Ja', 'ja', 'J', 'j', …

```
svar = input('Vil du høre en gåte til? (Ja / Nei): ')
if svar == 'Ja':      # funker kun for akkurat 'Ja', ikke 'JA', 'J', …
        print('Hva er likheten mellom …')
        input('Ditt svar: ')
        print('Løsningen var: …')
else:
        print('Neivel. Takk for nå!')
```

– Bedre:

```
if svar[0].upper() == 'J':   # funker for alt som begynner med 'J', 'j'
        print('Hva er…')
        …
```

Sammenligne strenger alfabetisk

- Alle tegn i Python har en tilhørende tallverdi

```
if 'A' < 'B':
    print('Bokstaven A er alfabetisk foran bokstaven B')
    # Bokstaven A har tallverdi 65, B 66
```

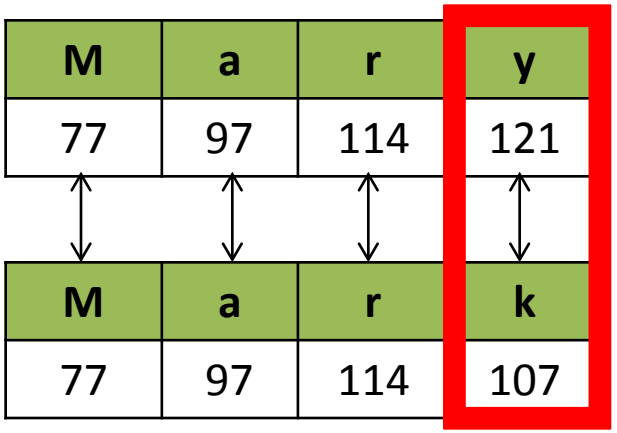
- Dette vil ofte funke for alfabetisk sammenligning
  - men ikke alltid!
    - noen norske bokstaver stemmer ikke
    - store og små bokstaver har ulik verdi
    - navn med mellomrom i kan skape komplikasjoner

ASCII tabellen – tegn representert som tall

1 ┌	33 !	65 A	97 a	129 ▯	161 ¡	193 Á	225 á
2 ┐	34 "	66 B	98 b	130 ‚	162 ¢	194 Â	226 â
3 └	35 #	67 C	99 c	131 ƒ	163 £	195 Ă	227 ă
4 ┘	36 $	68 D	100 d	132 „	164 ¤	196 Ä	228 ä
5 │	37 %	69 E	101 e	133 …	165 ¥	197 Å	229 å
6 ─	38 &	70 F	102 f	134 †	166 ¦	198 Æ	230 æ
7 •	39 '	71 G	103 g	135 ‡	167 §	199 Ç	231 ç
8 ◘	40 (	72 H	104 h	136 ˆ	168 ¨	200 È	232 è
9	41)	73 I	105 i	137 ‰	169 ©	201 É	233 é
10	42 *	74 J	106 j	138 Š	170 ª	202 Ê	234 ê
11 ♂	43 +	75 K	107 k	139 ‹	171 «	203 Ë	235 ë
12 □	44 ,	76 L	108 l	140 Œ	172 ¬	204 Ì	236 ì
13	45 -	77 M	109 m	141 ▯	173 -	205 Í	237 í
14 ♫	46 .	78 N	110 n	142 Ž	174 ®	206 Î	238 î
15 ☼	47 /	79 O	111 o	143 ▯	175 ¯	207 Ï	239 ï
16 ┼	48 0	80 P	112 p	144 ▯	176 °	208 Đ	240 ð
17 ◄	49 1	81 Q	113 q	145 '	177 ±	209 Ñ	241 ñ
18 ↕	50 2	82 R	114 r	146 '	178 ²	210 Ò	242 ò
19 ‼	51 3	83 S	115 s	147 "	179 ³	211 Ó	243 ó
20 ¶	52 4	84 T	116 t	148 "	180 ´	212 Ô	244 ô
21 ┴	53 5	85 U	117 u	149 •	181 µ	213 Ő	245 ő
22 ┬	54 6	86 V	118 v	150 –	182 ¶	214 Ö	246 ö
23 ┤	55 7	87 W	119 w	151 —	183 ·	215 ×	247 ÷
24 ↑	56 8	88 X	120 x	152 ˜	184 ¸	216 Ø	248 ø
25 ├	57 9	89 Y	121 y	153 ™	185 ¹	217 Ù	249 ù
26 →	58 :	90 Z	122 z	154 š	186 º	218 Ú	250 ú
27 ←	59 ;	91 [	123 {	155 ›	187 »	219 Û	251 û
28	60 <	92 \	124 |	156 œ	188 ¼	220 Ü	252 ü
29	61 =	93]	125 }	157 ▯	189 ½	221 Ý	253 ý
30	62 >	94 ^	126 ~	158 ž	190 ¾	222 Þ	254 þ
31	63 ?	95 _	127 ⌂	159 Ÿ	191 ¿	223 ß	255 ÿ
32	64 @	96 `	128 €	160	192 À	224 à	

Sammenlikning av to strenger

- Hva skjer her? Hva sammenliknes?
  - Sjekker bokstav for bokstav!




```
navn1 = 'Mary'
navn2 = 'Mark'
if navn1 > navn2:
    print('Mary er alfabetisk etter Mark')
else:
    print('Mary er alfabetisk før Mark')
```

Oppsummering

- Betingelser i Python: < , > , <= , >= , == , !=, <>
- Operatorer for logiske uttrykk: and, or , not
- Logiske uttrykk kan enten bli False eller True
- if-setninger:

```
if (<betingelse>):
    <utfør noe>          # HUSK INNRYKK!
elif (<betingelse):
    <utfør noe>          # HUSK INNRYKK!
else:
    <utfør noe annet>
```

- Vi kan også bruke nøstede if-setninger
- Innrykk er avgjørende for logikken i programmet