

TDT4110 Informasjonsteknologi grunnkurs:

Tema: Betingelser og logiske uttrykk

Utgave 3: Kap. 3

Terje Rydland - IDI/NTNU

2

```
if (be):  
    ...  
else (not_to_be):  
    ...
```



3

Læringsmål og pensum

- Mål
 - Lære å bruke og forstå if-setninger
 - Lære å bruke og forstå sammenlikning av strenger
 - Lære å bruke og forstå nestede beslutningsstrukturer
 - Lære å bruke og forstå logiske operatorer
 - Lære å bruke og forstå boolske variabler
- Pensum
 - Starting out with Python: Chapter 3
 - Decision Structures and Boolean Logic

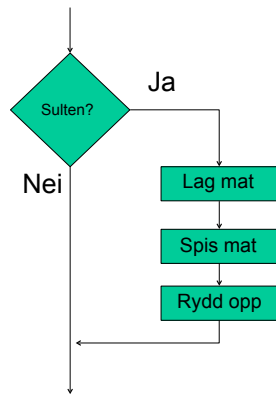
If-setningen

Kap. 3.1

Flytskjema

- If-setninger brukes til å bestemme at spesielle deler av programmet utføres avhengig av verdier på variabler (kontrollstruktur).
- If-setning brukes typisk til å utføre en kodeblokk hvis noen betingelser er sanne.
- Kode:

```
if betingelse:  
    uttrykk  
    uttrykk  
    etc.
```



Betingelser i Python

- Python har spesielle tegn eller **operatorer** for å uttrykke betingelser
- På lik linje med de **aritmetiske** operatorene +, -, *, /
- Mest mulig lik tegnene vi kjenner fra matematikken
- Kalles også **relasjonsoperatore**

Python	Matematikk	Forklaring
<	<	Mindre enn
>	>	Storre enn
<=	≤	Mindre eller lik
>=	≥	Storre 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	sann (True)

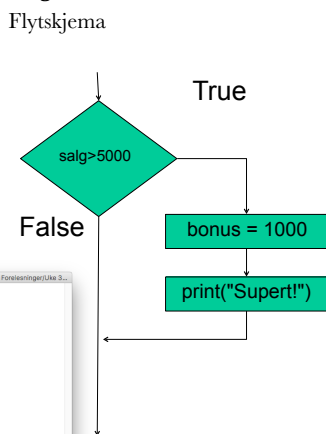
OBS! Forskjell på tilordning og sammenligning

- Det er lett å blande sammen tilordning og evaluering av logiske uttrykk:
- Dette er tilordning: **a=5**
 - **Tilordner** variabelen **a** verdien 5 (lagrer tallet 5 i variabelen *a*)
- Dette er evaluering: **a==5**
 - **Tester** om variabelen **a** har verdien 5
- Python vil si ifra hvis du prøver å gjøre en tilordning når du skal teste en om en variabel har en verdi

If-setningen

- Eksempel på if-setning:

```
if salg > 50000:  
    bonus = 1000  
    print('Supert')
```



```
# Eksempel på en enkel if-setning  
salg = int(input("Hvor mye har du solgt for (i hele kr): "))  
if salg > 5000:  
    bonus = 1000  
    print("Flott! du får en bonus for god jobb!")  
print("Takk for nå.")
```

if-else uttrykk

Kapittel 3.2

- Et if-else uttrykk vil kjøre en blokk av kode hvis betingelsen er sann (True) og en annen blokk av kode hvis betingelsen er usann (False).
- if-else uttrykk skal brukes i koden der det er to mulige alternativer av kode som skal utføres.

if-else generell kode

```
if betingelse:
```

```
    kode
```

```
    kode
```

```
    etc.
```

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

```
else:
```

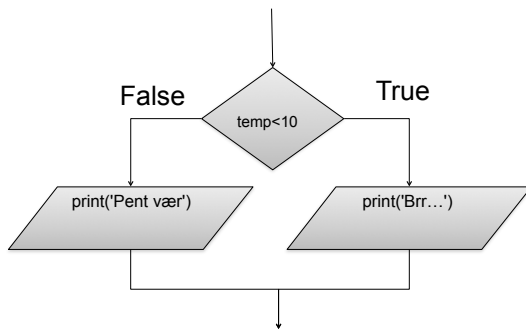
```
    kode
```

```
    kode
```

```
    etc.
```

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

if-else flytskjema



if-else kodeeksempel

```
# if-else kodeeksempel

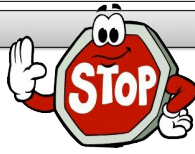
temp = int(input('Temperaturen er: '))

if temp < 10:
    # Legg merke til kolon
    print('Det var kaldt, du!') # Innrykk!
else:
    # Legg merke til kolon
    print('OK. Det kan vi leve med.') # Innrykk!

print('Det var dagens værmelding.') # Slutt på innrykk
```

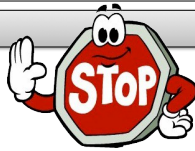
```
Python 3.5.2 Shell
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 26 2016, 10:47:25)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: /Users/terjery/Library/Mobile Documents/com-apple-Cloud
Docs/ITGK 2016/Python/1Forelesninger/Uke 37/if-else.py
Temperaturen er: 8
Det var kaldt, du!
Det var dagens værmelding.
>>> |
```

Oppgave: if else



- Skriv koden til funksjonen sjekk_puls:
 - En input-parameter: pulsslag per min
 - Hvis puls er høyere eller lik 80:
 - Skriv til konsoll: Ro deg ned!
 - Hvis puls er lavere enn 80:
 - Skriv til konsoll: Bare slapp av

Oppgave: if else



- Skriv koden til funksjonen sjekk_puls:
 - En input-parameter: pulsslag per min
 - Hvis puls er høyere eller lik 80:
 - Skriv til konsoll: Ro deg ned!
 - Hvis puls er lavere enn 80:
 - Skriv til konsoll: Bare slapp av

```
puls = int(input('Hva er pulsen: '))
if puls >= 80:
    print('Ro deg ned!')
else:
    print('Bare slapp av')
```

Ln: 5 Col: 26

```
Python 3.5.1 Shell
>>>
RESTART: /Users/terjery/Library/Mobile Documents/
com-apple-CloudDocs/ITGK 2016/Python/PDF/02eks.py
Oppgi antall pulsslag pr. min.: 18
Bare slapp av
>>> |
```

Ln: 59 Col: 4

Sammenlikning av to variabler som inneholder strenger Kapittel 3.3

- Variabler som inneholder tekststrenger kan sammenlignes på lik linje med tall.
- Eksempel på å sjekke om to variabler er like:

```
navn1 = 'Peter'
navn2 = 'Pelle'
if navn1 == navn2:
    print("Samme navn!")
else:
    print("Forskjellige navn")
```

Ln: 6 Col: 30

Sammenlikning av en variabel og en tekststreng

- Man kan også sjekke om en variabel med tekststreng er lik en spesifisert tekst:

```
02eks.py - /Users/terjery/Library/Mobile Documents/com-apple-CloudDocs/ITGK 2016...
password = input("Skriv inn passord: ")
if password == "Nuff":
    print("Riktig passord")
else:
    print("Feil passord")
Ln: 5 Col: 4
```

```
Python 3.5.1 Shell
>>>
RESTART: /Users/terjery/Library/Mobile Documents/
com-apple-CloudDocs/ITGK 2016/Python/PDF/02eks.py
Skriv inn passord: øki fh
Feil passord
>>>
Ln: 66 Col: 4
```

Sjekke om en streng er større enn en annen streng

- I Python kan du også sjekke om en streng er større (eller mindre) enn en annen streng.
 - Dvs. at en tekststreng har tegn som er representert med mindre eller større verdier enn i den andre strengen.
 - Alle tegn i Python representerer en tallverdi

```
if 'A' < 'B':
    print('Bokstaven A er mindre enn bokstaven B')
# Bokstaven A representeres med tallet 65 og B med 66
```

ASCII tabellen – tegn representert som tall

Kun de første 128 (7-bit: 2^7) er standard

	0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB
10	LF	VT	FF	CR	SO	SI	DLE	DC1	DC2	DC3
20	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
30	RS	US	SPACE	!	"	#	\$	%	&	'
40	()	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[\]	^	_	~	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{		}	~	DEL		

Med 8-bit (2^8) får vi 256 plasser (nummerert fra 0 til 255), men her er det ikke en felles standard i ASCII

Løsning: UNICODE

Sammenlikning av to strenger

- Hva skjer her?
 - Sjekker bokstav for bokstav!

Hva sammenliknes?

M	a	r	y
77	97	114	121
↕	↕	↕	↕
M	a	r	k
77	97	114	107

```
navn1 = "Mary"
```

```
navn2 = "Mark"
```

```
if navn1 > navn2:
```

```
    print(navn1, 'er større enn', navn2)
```

```
else:
```

```
    print(navn1, 'er ikke større enn', navn2)
```

Eksempel: sjekk strenger

- Lag et program der bruker kan skrive inn to tekststrenger.
- Sjekk om strengene er like eller om en kommer før i alfabetet enn den andre og kommenter resultatet til skjerm.

streng_sjekk.py

Eksempel: sjekk strenger

- Skal lage et program med en main-funksjon hvor bruker kan skrive inn to tekststrenger.
- Lag en funksjon som tar inn to tekststrenger og sjekker om strengene er like eller om en kommer før i alfabetet enn den andre og kommenterer resultatet til skjerm.

streng_sjekk.py

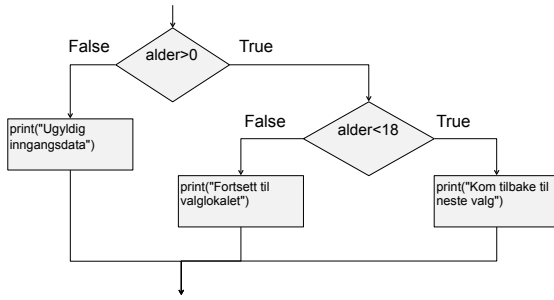
```

str1 = input('Skriv inn en streng: ')
str2 = input('Skriv inn enda en streng: ')
if (str1 == str2):
    print('Begge strengene er', str1)
if (str1 < str2):
    print(str1, 'kommer før', str2)
if (str2 < str1):
    print(str2, 'kommer før', str1)
  
```

Ln: 9 Col: 0

Ser noen noe som kan forbedres med løsningen?

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



Nøsting av if-setninger

- Man bruker **innrykk** for å si at de indre setningene hører til if-setningen. En if-setning avsluttes ved å fjerne innrykk.
- Kan ha flere nivåer med if-setninger inne i hverandre.

```

*02eks.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/...
if (alder < 0):
    print("Ugyldige inngangsdata")
else:
    if (alder < 18):
        print("Kom tilbake til neste valg")
    else:
        print("Fortsett til valglokalet")
Ln: 7 Col: 8
  
```

Forbedring av tidligere eksempel

```

streng_sjekk.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/...
str1 = input('Skriv inn en streng: ')
str2 = input('Skriv inn enda en streng: ')
if (str1 == str2):
    print('Begge strengene er', str1)
if (str1 < str2):
    print(str1, 'kommer før', str2)
if (str2 < str1):
    print(str2, 'kommer før', str1)
Ln: 9 Col: 0

streng_sjekk-V1.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITOK...
str1 = input('Skriv inn en streng: ')
str2 = input('Skriv inn enda en streng: ')
if (str1 == str2):
    print('Begge strengene er', str1)
else:
    if (str1 < str2):
        print(str1, 'kommer før', str2)
    else:
        if (str2 < str1):
            print(str2, 'kommer før', str1)
Ln: 11 Col: 0
  
```


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....)

```
*02eks.py - /Users/terjery/Library/Mobil...
if (poeng>=90):
    karakter = 'A'
elif (poeng>=80):
    karakter = 'B'
elif (poeng>=60):
    karakter = 'C'
elif (poeng>=50):
    karakter = 'D'
elif (poeng>=40):
    karakter = 'E'
else:
    karakter = 'F'

Ln: 12 Col: 4
```

- NB! Kun en av betingelsene vil slå til!

Enda en forbedring

```
streng_sjekk-V1.py - /Users/terjery/Library/Mobile Documents/com-apple-CloudDocs/ITOK...
str1 = input('Skriv inn en streng: ')
str2 = input('Skriv inn enda en streng: ')
if (str1 == str2):
    print('Begge strengene er',str1)
else:
    if (str1<str2):
        print(str1,'kommer før',str2)
    else:
        if (str2<str1):
            print(str2,'kommer før',str1)

Ln: 11 Col: 0
```

```
streng_sjekk-V2.py - /Users/terjery/Library/Mobile Documents/com-apple-Cloud...
str1 = input('Skriv inn en streng: ')
str2 = input('Skriv inn enda en streng: ')
if (str1 == str2):
    print('Begge strengene er',str1)
elif (str1<str2):
    print(str1,'kommer før',str2)
else:
    print(str2,'kommer før',str1)

Ln: 9 Col: 0
```

Karaktereksempel

```
karakter.py - /Users/terjery/Library/Mobile Documents/com-apple-CloudDocs/ITGK 2016/Python1...
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# REGNER UT KARAKTER UT FRA NTNUs PROSENTPOENGMETODE
# INPUT: poeng
# OUTPUT: karakter
#
poeng = int(input('Skriv inn poengsummen: '))
if (poeng >= 89):
    karakter = 'A'
elif (poeng >= 77):
    karakter = 'B'
elif (poeng >= 65):
    karakter = 'C'
elif (poeng >= 53):
    karakter = 'D'
elif (poeng >= 41):
    karakter = 'E'
else:
    karakter = 'F'
print('Karakteren ble', karakter)

Ln: 21 Col: 0
```

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

Operator i Python	Forklaring
<code>and</code>	Logisk og
<code>or</code>	Logisk eller
<code>not</code>	Logisk ikke, eller negasjon

Logiske uttrykk (fortsettelse)

- Hva betyr `and`, `or` og `not` i praksis:

– Et uttrykk med `and` blir True hvis begge sider er True:

```
False and True    gir    False
False and False   gir    False
True and True     gir    True
```

– Et uttrykk med `or` blir True hvis minst en av sidene er True:

```
False or True     gir    True
True or True      gir    True
False or False    gir    False
```

– `not` snur uttrykket:

```
not True          gir    False
not False         gir    True
```

Sannhetsverditabell

a : Det regner i dag

b : Det blåser

		b				b	
		False	True			False	True
a	False	False	False	a	False	False	True
	True	False	True		True	True	True

a	not a
False	True
True	False

Eksempel på logiske uttrykk

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

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

det kan være lurt å skrive `(x >= 5) and (x <= 10)`

- I Python kan man også sjekke intervaller på følgende måte (fungerer sjeldent i andre programmeringsspråk):

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

- Vi lager større uttrykk og sjekke flere betingelser ved bruk av parenteser:

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

Resultatet av logiske utregninger

- En enkel eller sammensatt betingelse kalles et **logisk uttrykk**
- Resultatet av en logisk beregning (evaluering) er enten **True** eller **False** i Python (med stor forbokstav!)
- ... IKKE sann eller usann!

For å få riktig betingelser husk Pythons Operatorhierarki:

1. ()
 2. ** # Eksponent (opphøyd)
 3. *, /, //, % # heltallsdivisjon, rest
 4. +, -
 5. <, <=, >, >=, <>, !=, ==
 6. not
 7. and
 8. or
 9. if - else
 10. Lik prioritet: fra venstre mot høyre
- Bruk parenteser for å få uttrykkene riktig

Betingelser

- Oppgave: Er denne betingelsen **True** eller **False**?

$4 < 7$ and not ($3 > 1$ or $8 >= 9$)

$4 < 7$ and not (True or False)

$4 < 7$ and not (True)

True and not (True)

True and False

False

- Begynn innenfra og jobb utover (inne i parenteser)

En liten test... Bestem true eller false

Bruk disse verdiene på variablene:

A = 5, B=9, C=12, D=39:

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

En liten test... Bestem true eller false

Bruk disse verdiene på variablene:

A = 5, B=9, C=12, D=39:

- | | |
|--|--------|
| • $(A > 5$ or $B == 2)$ | •False |
| • $(A + B < C + D)$ and $(D >= 39)$ | •True |
| • $(A > B$ or $B > C$ or $C > D$ or $D > A)$ | •True |
| • $(B <= C)$ | •True |

Boolske variabler - kap. 3.6

- En boolsk variabel kan referere til en av to verdier: True eller False.
- Boolske variabler brukes typisk som flagg for å lagre at en spesiell betingelse er sann eller ikke.

```
riktig = False
svar = input('Skriv svar: ')
if (svar == 'tulling'):
    riktig = True
...
# sjekker om riktig == True lengre nede i programmet
```

Avslutning if-setninger

- Unngå overflødig bruk av negasjoner (**not**) – tungt å lese
- Ved **if else**, skriv helst positiv utfall i **if** og negativt i **else**
- Vi kan ha flere setningen mellom **if ... else**

Oppgave: Dørvaktsimulator



- Det skal lages et dørvaktsimulator program som skal benytte seg av: print, input(), int(), if og else.
- Programmet skal skrive til skjerm "Hei jeg er dørvakta!" og så spørre etter navn, alder og om du er full (ja eller nei)
 - Hvis du er gammel nok (18+), så skal det skrives ut "Du er gammel nok".
 - Hvis du ikke er gammel nok, skal det skrives ut "Du er for ung "+ navnet
 - Hvis du er gammel nok, men er også full skal det skrives ut "Du slipper ikke inn for du er full!"
 - Hvis du er gammel nok og ikke full, skal det skrives ut "Velkommen inn "+ navnet

doorman.py

Oppgave: Dørvaktsimulator



```
doorman.py - /Users/terjery/Library/Mobile Documents/com-apple-CloudDocs/ITGK 2016/Python/1Forelesning...
print("Hei jeg er dørvakta!")
navn=input("Navnet ditt? ")
alder=int(input("Alder? "))

# Sjekker betingelsene
if(alder>=18):
    print("Du er gammel nok")
    full=input("Er du full? ")
    if(full=="ja" or full=="Ja"):
        print("Du slipper ikke inn for du er full")
    else:
        print("Velkommen inn",navn)
else:
    print("Du er for ung",navn)
```

Ln: 15 Col: 0

Alternativ løsning m/funksjon



```
doorman 2.py - /Users/terjery/Library/Mobile Documents/com-apple-CloudDocs/ITGK 2016/Py...
def sjekkOgSkrivUt(navn,alder,full):
    if (alder>=18):
        print("Du er gammel nok!")
        if (full=="ja" or full=="Ja" or full=="JA"):
            print("Du slipper ikke inn!")
        else:
            print("Velkommen inn",navn)
    else:
        print("Du er for ung",navn)

print("Hei, jeg er dørvakta!")
# Henter inn data fra bruker
navn = input("Navnet ditt? ")
alder = int(input("ALDER? "))
full = input("Er du full? ")

sjekkOgSkrivUt(navn,alder,full)
```

Ln: 5 Col: 38

Oppsummering

- Betingelser i Python: < , > , <= , >= , == , != , <>
- Operatorer for logiske uttrykk: and, or , not
- Logiske uttrykk kan enten bli **False** eller **True**
- 3 varianter
 - if, if...else, if...elif...elif...elif...else
- 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
- Husk : etter det logiske uttrykket i if, else og elif.