

Kunnskap for en bedre verden

Kunnskap for en bedre verden

TDT4110 Informasjonsteknologi grunnkurs:  
Tema: Betingelser og logiske uttrykk

Dag Olav Kjellemo

## Noen nyttige ting om print og strenger: Endring av prints standard oppførsel

- Når vi har flere argumenter til print, kan vi endre mellomrommet som er «standardverdi» (default value):

```
print('hei','hå',sep='_og_')          # hei_og_hå  
print(4,5,sep='')                   # 45
```

- Og vi kan endre linjeskiften som er default:

- `print('Python kan tilpasse seg dine behov.', end='')`
- `Print('Dette kommer uten linjeskift etter forrige print.')`

- **sep** og **end** er såkalte navngitte parametre til print. Mange parametre har standardverdier for å spare oss litt skriving når vi bare vil ha det «vanlige».

\n	Gir linjeskift
\t	Hopper til neste tabulator
\'	Skriver ut tegnet '
\"	Skriver ut tegnet "
\\	Skriver ut tegnet \

## Escape-tegn

- Escape-karakterer er spesialkarakterer som kan skrives inn i tekststrenger som gjør spesielle operasjoner:

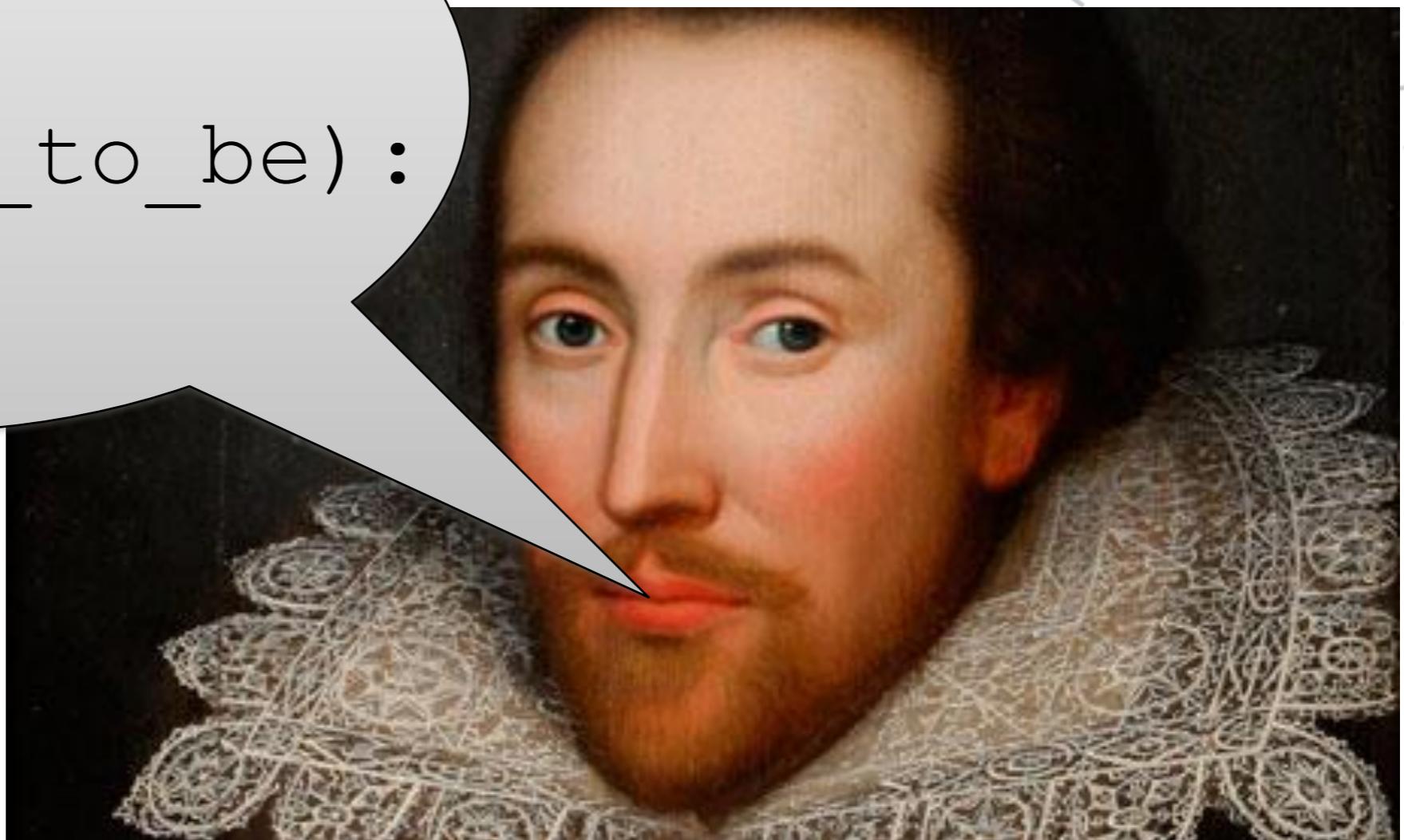
\n	Gir linjeskift
\t	Hopper til neste tabulator
\'	Skriver ut tegnet ’
\”	Skriver ut tegnet ”
\\"	Skriver ut tegnet \

-Eks:

```
print ('\tInnrykk er kult med\nNy linje')
```

# To be or not to be

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



# 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

- Velger hva som utføres avhengig av verdier på variabler (kontrollstruktur).
- If-setning brukes til å utføre en kodeblokk hvis noen betingelser er sanne.
- Kode:

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

Screenshot of Thonny IDE showing Python code for a simple if-statement.

```

# Eksempel på enkel if-setning
salg = int(input('\nHvor mye har du solgt for (i hele kr.): '))
if salg > 50000:
    bonus = 1000
    print(f'\nFlott! Du får bonus på kr {bonus} for god jobb.')
print('\nTakk for nå.\n')

```

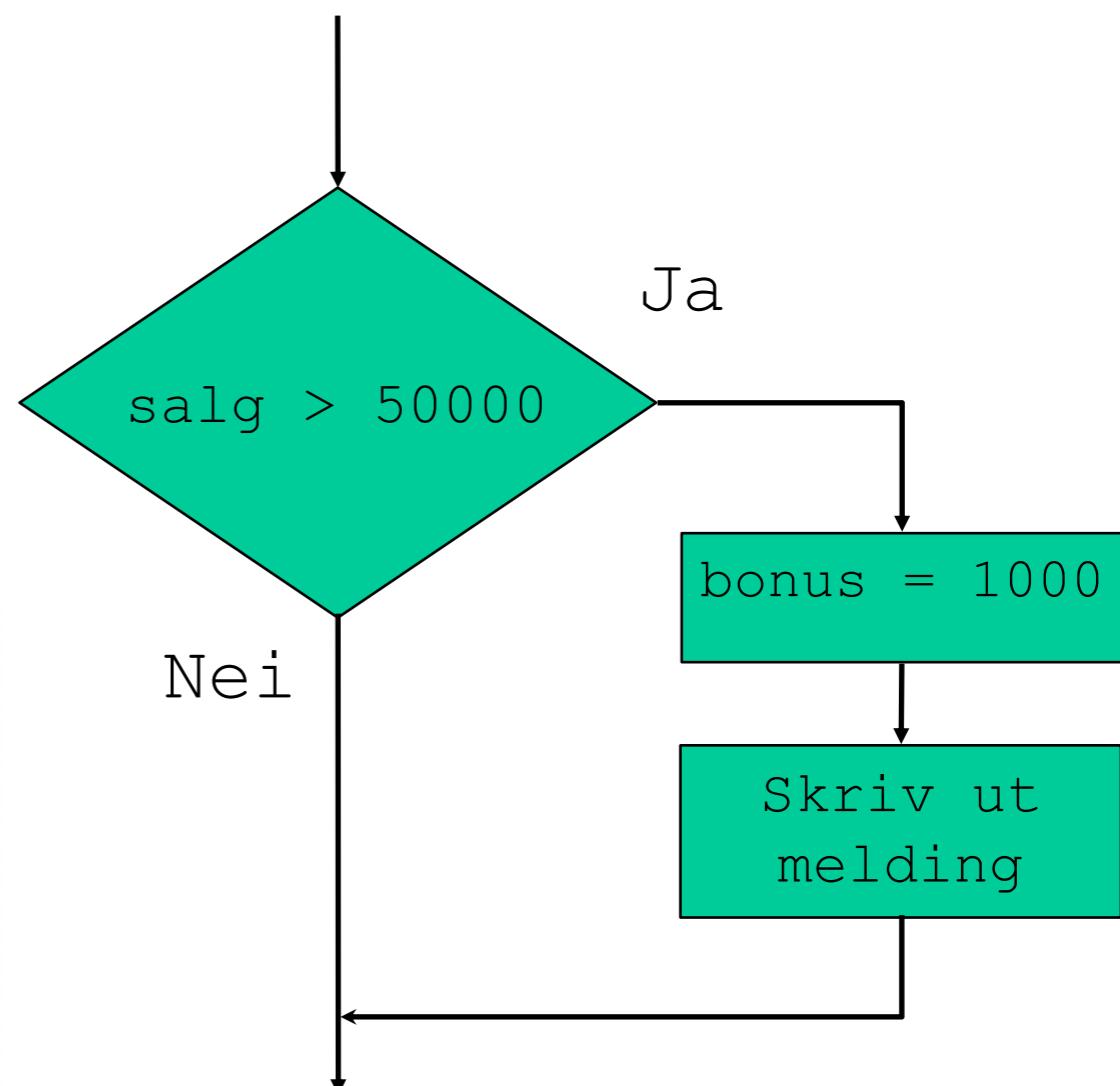
The Shell tab shows the program's output:

```

Hvor mye har du solgt for (i hele kr.): 57000
Flott! Du får bonus på kr 1000 for god jobb.
Takk for nå.

```

### Flytskjema



## Hva kan man teste på?

- Python har spesielle *relasjonsoperatorer* for å sammenligne uttrykk
- Mest mulig lik tegnene vi kjenner fra matematikken

Matematikk	Python	Forklaring
<	<	Mindre enn
>	>	Større enn
$\leq$	$\leq$	Mindre eller lik
$\geq$	$\geq$	Større eller lik
=	==	Er lik
$\neq$	!=	Er ulik

NB !

Betingelse	Verdi
$4 < 3$	False
$4 == 4$	True
$3 != 3$	False
$3 \geq 3$	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

The screenshot shows the Thonny Python IDE interface. The code editor window has a tab labeled '<untitled>' containing the number '1'. Below it is a 'Shell' window with the following Python session:

```
Python 3.7.4 (/Library/Frameworks/Python.framework/Versions/3.7/bin/python3)
>>> a = 8
>>> if a = 8:
        print(8)

File "<pyshell>", line 1
    if a = 8:
        ^
SyntaxError: invalid syntax
>>> if a == 8:
        print('Nå er det riktig.')

Nå er det riktig.
>>> |
```

A blue oval highlights the incorrect assignment statement 'if a = 8'. Another blue oval highlights the resulting 'SyntaxError: invalid syntax' message. The rest of the session shows a correct comparison 'if a == 8' followed by a print statement and its output.

# Vi kan formulere koden på mange måter

The screenshot shows the Thonny Python IDE interface. The top window is titled "Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK2019/1Forelesninger/Uke 37/Lysark 6-2.py @ 9 : 8". It contains two tabs: "Lysark 6.py \*" and "Lysark 6-2.py". The "Lysark 6-2.py" tab is active and displays the following Python code:

```
1 # Eksempel på enkel if-setning
2
3 salg = int(input('\nHvor mye har du solgt for (i hele kr.): '))
4
5 if salg > 50000:
6     bonus = 1000
7     print(f'\nFlott! Du får bonus på kr {bonus} for god jobb.')
8
9 print('\nTakk for nå.\n')
10
11
```

Below the code editor is a "Shell" window. It shows the command ">>> %Run 'Lysark 6-2.py'" followed by the program's output:

```
>>> %Run 'Lysark 6-2.py'

Hvor mye har du solgt for (i hele kr.): 56000
Flott! Du får bonus på kr 1000 for god jobb.

Takk for nå.

>>> %Run 'Lysark 6-2.py'

Hvor mye har du solgt for (i hele kr.): 40000
Takk for nå.

>>>
```

# Vi kan formulere koden på mange måter

The image shows two side-by-side Thonny IDE windows. Both windows have tabs for 'Lysark 6.py' and 'Lysark 6-2.py'. The left window shows a simple if-statement where the condition is checked before the assignment. The right window shows a more complex approach where the condition is checked after the assignment, and the result is then used to construct a string.

```

Left Window (Lysark 6-2.py):
1 # Eksempel på enkel if-setning
2
3 salg = int(input('\nHvor mye har du solgt for (i hele kr.): '))
4
5 if salg > 50000:
6     bonus = 1000
7     print(f'\nFlott! Du får bonus på kr {bonus} for god jobb.')
8
9 print('\nTakk for nå.\n')
10
11

Right Window (Lysark 6.py):
1 # Eksempel på enkel if-setning
2
3 # Brukeren skriver inn beløp fra tastaturet. Det lagres som et
4 # tall i variabelen salg
5 salg = int(input('\nHvor mye har du solgt for (i hele kr.): '))
6
7 # Legger teksten som skal skrives ut i en variabel
8 tekst = '\n\nDu må selge for over 50 000 kr. for å få bonus.'
9
10 if salg > 50000:
11     bonus = 1000
12     # Endrer teksten som skal skrives ut
13     tekst = f'\nFlott! Du får bonus på kr {bonus} for god jobb.'
14
15 # Skriver ut innholdet i tekst-variabelen og en sluttmelding
16 print(f'{tekst}\nTakk for nå.\n')

Shell:
>>> %Run 'Lysark 6-2.py'
Hvor mye har du solgt for (i hele kr.): 56000
Flott! Du får bonus på kr 1000 for god jobb.
Takk for nå.

>>> %Run 'Lysark 6-2.py'
Hvor mye har du solgt for (i hele kr.): 40000
Takk for nå.

>>>

```

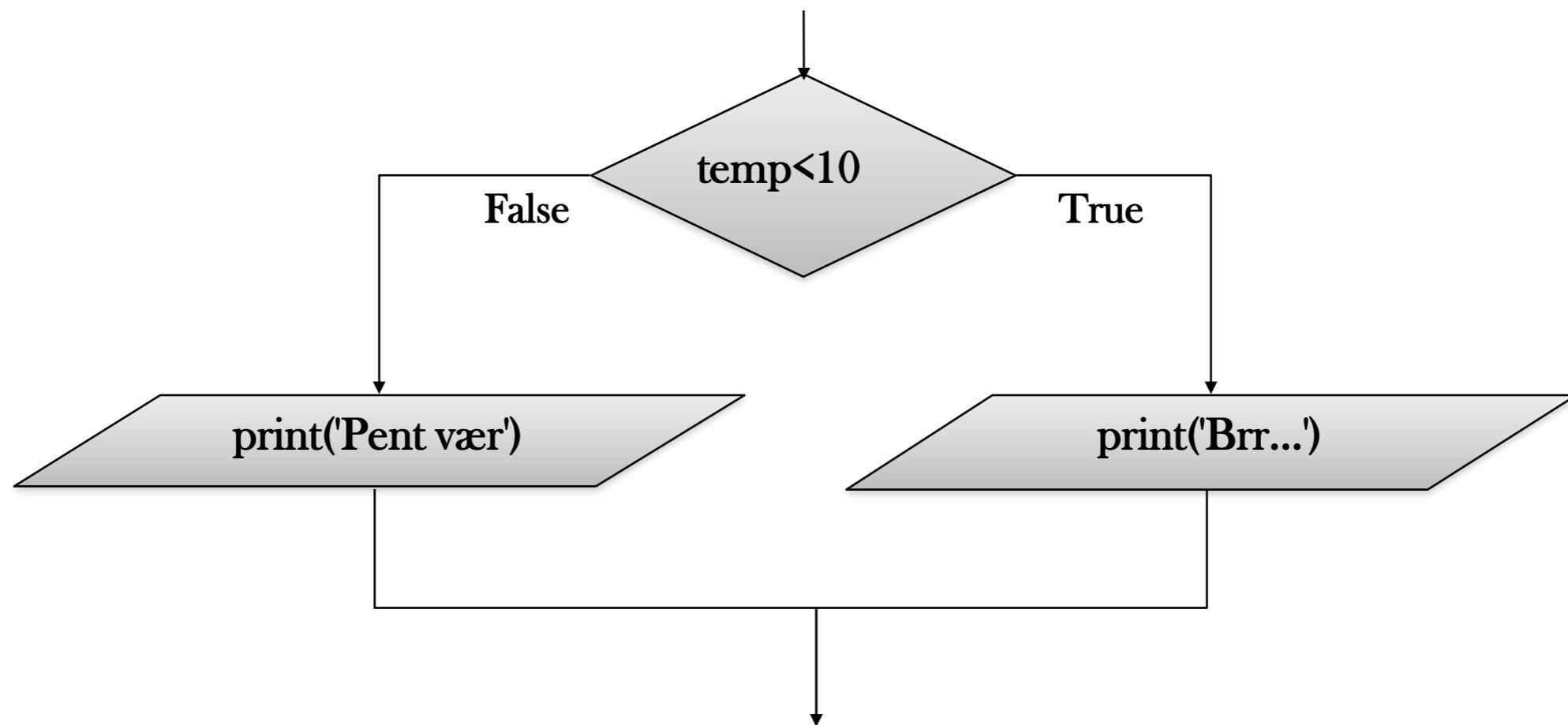
```

Shell:
>>> %Run 'Lysark 6.py'
Hvor mye har du solgt for (i hele kr.): 55000
Flott! Du får bonus på kr 1000 for god jobb.
Takk for nå.

```

## if-else flytskjema

- 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 uttrykk

# Kapittel 3.2

```

1 # Eksempel på enkel if-else setning
2
3 salg = int(input('\nHvor mye har du solgt for (i hele kr.): '))
4
5 if salg > 50000:
6     bonus = 1000
7     print(f'\nFlott! Du får bonus på kr {bonus} for god jobb.')
8 else:
9     print('\nDu må selge for over 50 000 kr. for å få bonus.')
10
11 print('\nTakk for nå.\n')

```

Shell >>> %Run 'Lysark 8.py'

Hvor mye har du solgt for (i hele kr.): 46000

Du må selge for over 50 000 kr. for å få bonus.

Takk for nå.

>>> %Run 'Lysark 8.py'

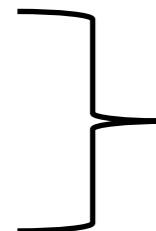
Hvor mye har du solgt for (i hele kr.): 56000

Flott! Du får bonus på kr 1000 for god jobb.

Takk for nå.

if betingelse:

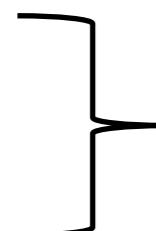
kode  
kode  
etc.



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

else:

kode  
kode  
etc.



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

# if-else kodeeksempel

The screenshot shows the Thonny Python IDE interface. The top bar displays the title "Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2020/1 Forelesninger/Uke 37/1 Ny versjon/1 Eksempler/5 Peppes Pizza.py @ 15 : 37". Below the title, there is a toolbar with various icons. The main window contains several tabs: "untitled", "Enkel if.py", "Lysark 6.py \*", "Lysark 6-2.py", "Kvantumrabatt.py \*", and "5 Peppes Pizza.py" (which is currently selected). The code editor shows a script named "5 Peppes Pizza.py" with the following content:

```
1 # Du kjøper en pizza på restaurant. Restauranten gir studentrabatt på
2 # 20 %. Det er vanlig å gi 8% tips
3 # Beløpet skal fordeles likt på antall som deltar
4
5 pizza = int(input('Hva kostet herligheten: '))
6 Studentrabatt = 0.20
7 Tips = 0.08
8
9 student = input('Er du student (J/N): ')
10 if student == 'J':
11     totalt = pizza*(1-Studentrabatt)*(1+Tips)
12 else:
13     totalt = pizza*(1+Tips)
14
15 print(f'Total pris: {totalt:.2f} kr.')
16
17 antall = int(input('Hvor mange deltok på middagen? '))
18 prisPerPers = totalt/antall
19 print(f'Siden dere var {antall} personer, så må hver person betale {prisPerPers:.2f} kroner.')
20
```

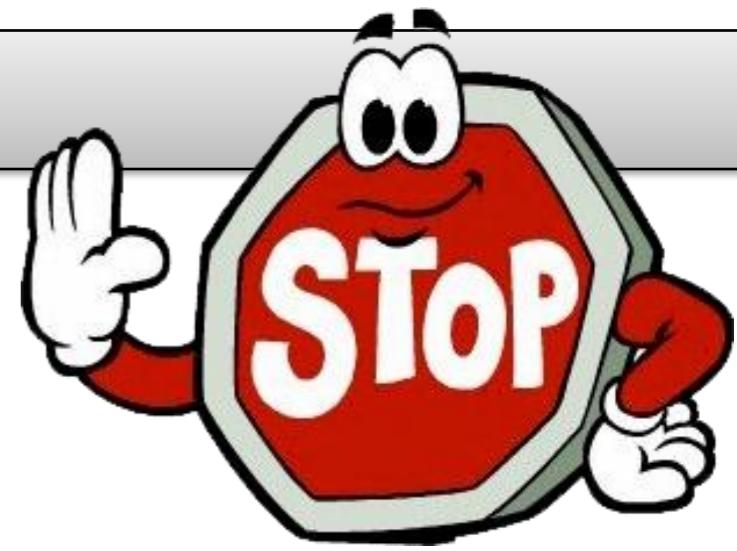
Below the code editor is a "Shell" tab containing the output of the program's execution:

```
Hva kostet herligheten: 1258
Er du student (J/N): J
Total pris: 1086.91 kr.
Hvor mange deltok på middagen? 3
Siden dere var 3 personer, så må hver person betale 362.30 kroner.
```

A command prompt "">>>> |" is visible at the bottom of the shell window.

## Oppgave: if else

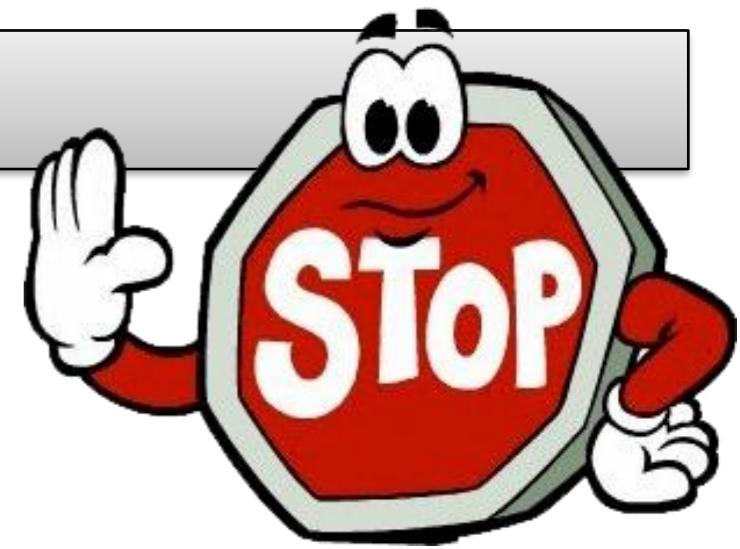
- Skriv koden til funksjonen sjekk\_puls:
  - En input-parameter: pulsslag per min
  - Hvis puls er høyere eller lik 80:
    - Skriv til konsoll: Ro deg ned!
  - Hvis puls er lavere enn 80:
    - Skriv til konsoll: Bare slapp av



## 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



The screenshot shows the Thonny Python IDE interface. The top bar indicates the window is titled 'Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK2019/1Forele...'. Below the title bar is a toolbar with various icons. The main area contains two tabs: 'Enkel if 3.py' and 'puls.py', with 'puls.py' currently selected. The code in 'puls.py' is as follows:

```
1 puls = int(input('Hva er pulsen: '))
2 if puls > 80:
3     print('Ro deg ned!')
4 else:
5     if puls < 40:
6         print('Kontakt lege snarest.')
7     else:
8         print('Bare slapp av.')
```

Below the code editor is a 'Shell' tab. The shell output shows the following interaction:

```
Hvor mye har du solgt for (i hele kr.): 5500
Flott! Du får bonus for god jobb!
Takk for nå.

>>> %Run puls.py
Hva er pulsen: 36
Kontakt lege snarest.

>>> |
```

## Sammenlikning av en variabel og en tekststreng

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

The screenshot shows the Thonny IDE interface. The top window is titled 'Lysark 13 Passord.py' and contains the following code:

```
1 # Eksempel på sammenligning av tekst
2
3 passord = input('Skriv inn passord: ')
4 if passord == '1234':
5     print('Riktig passord.')
6 else:
7     print('Feil passord.)
```

The bottom window is titled 'Shell' and shows the execution of the script:

```
>>> %Run 'Lysark 13 Passord.py'
Skriv inn passord: 4321
Feil passord.

>>> %Run 'Lysark 13 Passord.py'
Skriv inn passord: 1234
Riktig passord.

>>> |
```

# 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:

The image shows two side-by-side screenshots of the Thonny Python IDE interface. Both screenshots display a code editor window and a shell window below it.

**Left Screenshot (Navn.py):**

```

1 # Eksempel på sammenligning av tekst
2
3 navn1 = 'Peter'
4 navn2 = 'Pelle'
5 if navn1 == navn2:
6     print(f'Samme navn')
7 else:
8     print(f'Forskjellige navn')

```

**Shell Output:**

```

>>> %Run Navn.py
Forskjellige navn
>>>

```

**Text overlay:** case-sensitivt

**Right Screenshot (Navn2.py):**

```

1 # Eksempel på sammenligning av tekst
2
3 navn1 = 'Peter'
4 navn2 = 'PETer'
5 if navn1 == navn2:
6     print(f'Samme navn')
7 else:
8     print(f'Forskjellige navn')
9
10

```

**Shell Output:**

```

>>> %Run Navn2.py
Forskjellige navn
>>>

```

## 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
```

NB: Hvis du sammenligner store og små bokstaver må du være klar over at de små bokstavene har høyere tallverdi enn de store, slik at 'a' < 'B' vil gi **False**.

# 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	[	\	]	^	_	`	á	í	ç
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!

```
navn1 = "Mary"
```

```
navn2 = "Mark"
```

```
if navn1 > navn2:
    print(navn1, 'er større enn', navn2)
else:
    print(navn1, 'er ikke større enn', navn2)
```

- Hva sammenliknes?

M	a	r	y
77	97	114	121
↑	↓	↓	↓
M	a	r	k
77	97	114	107

## Eksempel: sjekk strenger

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

The screenshot shows the Thonny Python IDE interface. The top bar displays the title "Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK/ITGK 2020/1 Forelesninger/Uke 36/1 03 - Eksempler/Ly...". Below the title is a toolbar with various icons. The main window has tabs for "Lysark 13 Passord.py" and "Lysark 18 2 strenger.py", with "Lysark 18 2 strenger.py" currently selected. The code editor contains the following Python script:

```
1 # Eksempel på sammenligning av tekst
2
3 str1 = input('Skriv inn en streng: ')
4 str2 = input('Skriv inn en streng: ')
5 if str1 == str2:
6     print(f'Begge strengene er {str1}.')
7 if str1 < str2:
8     print(f'{str1} kommer før {str2}.')
9 if str1 > str2:
10    print(f'{str2} kommer før {str1}.')
```

Below the code editor is a "Shell" window with the following interaction:

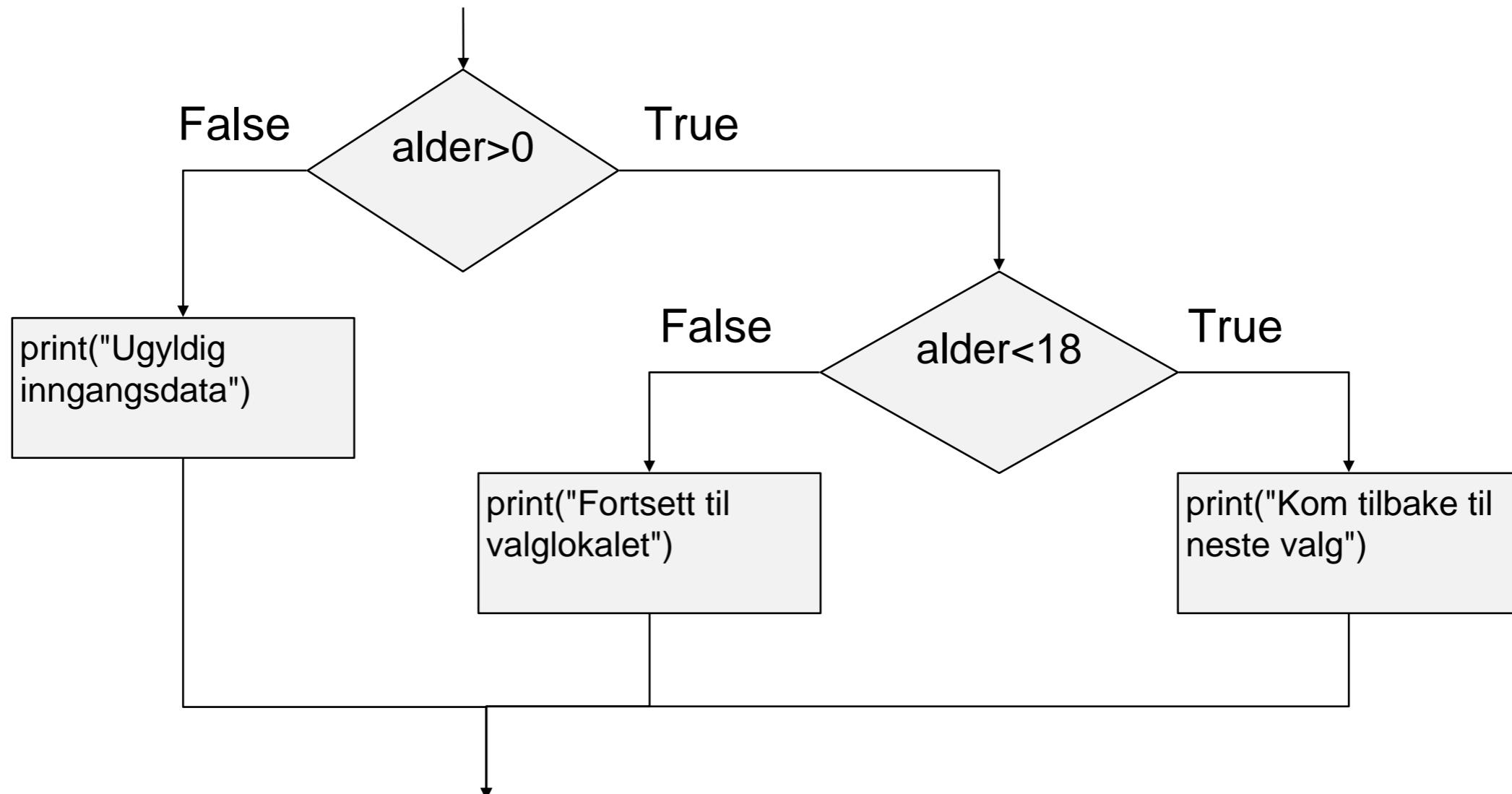
```
>>> %Run 'Lysark 18 2 strenger.py'
Skriv inn en streng: abc
Skriv inn en streng: Abc
Abc kommer før abc.
>>> |
```

Ser noen noe som kan forbedres med løsningen?

## Flytskjema for nøstede if-setninger

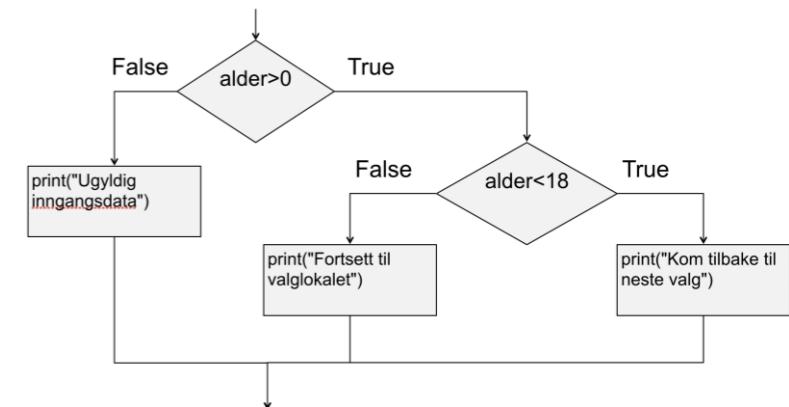
## Kapittel 3.4

- 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.



Screenshot of the Thonny Python IDE showing nested if statements and their execution.

The code in the editor is:

```

1 alder = int(input('Hvor gammel er du: '))
2 if alder < 0:
3     print('Ugyldig inngangsdata.')
4 else:
5     if alder < 18:
6         print('Kom tilbake til neste valg.')
7     else:
8         print('Fortsett til valglokalet.')
9
# Legg merke til innrykk!
  
```

The Shell tab shows the output of running the script:

```

>>> %Run 'Lysark 20.py'
Hvor gammel er du: 23
Fortsett til valglokalet.

>>> %Run 'Lysark 20.py'
Hvor gammel er du: 12
Kom tilbake til neste valg.

>>>
  
```

# Forbedring av tidligere eksempel

The screenshot shows the Thonny IDE interface. The top bar displays the path: Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK2019/1Forelesninger/Uke 37/Lysark 18.py @ 10 : 36. The code editor tab for Lysark 18.py is selected. The code itself is:

```

1 # Eksempel på sammenligning av tekst
2
3 str1 = input('Skriv inn en streng: ')
4 str2 = input('Skriv inn enda en streng: ')
5 if str1 == str2:                      # Kolon!
6     print(f'Begge strengene er {str1}')
7 if str1 < str2:
8     print(f'{str1} kommer før {str2}')
9 if str2 < str1:
10    print(f'{str2} kommer før {str1}')
11

```

The shell window below shows the execution of the script:

```

>>> %Run 'Lysark 18.py'
Skriv inn en streng: Peter
Skriv inn enda en streng: Pelle
Pelle kommer før Peter
>>> |

```

The screenshot shows the Thonny IDE interface. The top bar displays the path: Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK2019/1Forelesninger/Uke 37/Lysark 21.py @ 12 : 47. The code editor tab for Lysark 21.py is selected. The code is identical to Lysark 18.py:

```

1 # Eksempel på sammenligning av tekst
2
3 str1 = input('Skriv inn en streng: ')
4 str2 = input('Skriv inn enda en streng: ')
5 if str1 == str2:                      # Kolon!
6     print(f'Begge strengene er {str1}')
7 else:
8     if str1 < str2:
9         print(f'{str1} kommer før {str2}')
10    else:
11        if str2 < str1:
12            print(f'{str2} kommer før {str1}')
13

```

The shell window below shows the execution of the script:

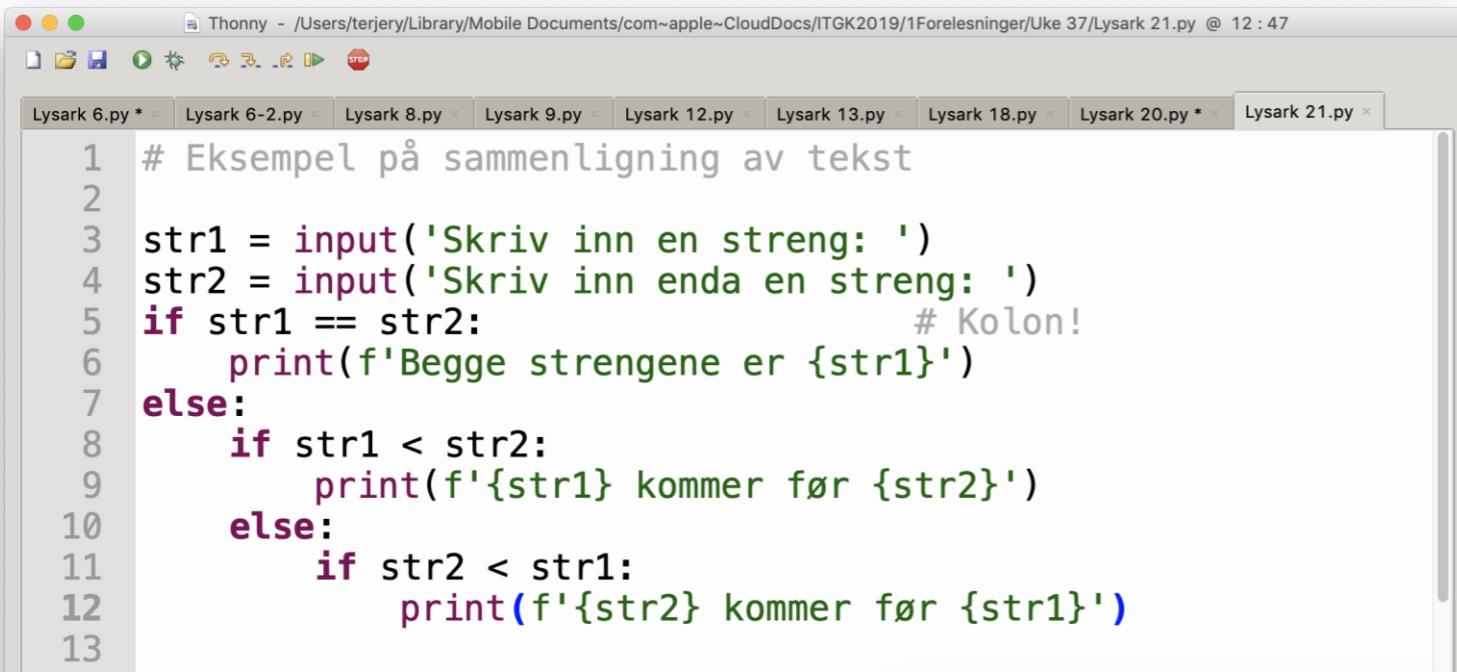
```

>>> %Run 'Lysark 21.py'
Skriv inn en streng: Petter
Skriv inn enda en streng: Pelle
Pelle kommer før Petter
>>> |

```

Her gjøres alle sammenligningene, selv om den første if-setningen gir True.

# Enda en forbedring



```

1 # Eksempel på sammenligning av tekst
2
3 str1 = input('Skriv inn en streng: ')
4 str2 = input('Skriv inn enda en streng: ')
5 if str1 == str2:                      # Kolon!
6     print(f'Begge strengene er {str1}')
7 else:
8     if str1 < str2:
9         print(f'{str1} kommer før {str2}')
10    else:
11        if str2 < str1:
12            print(f'{str2} kommer før {str1}')
13

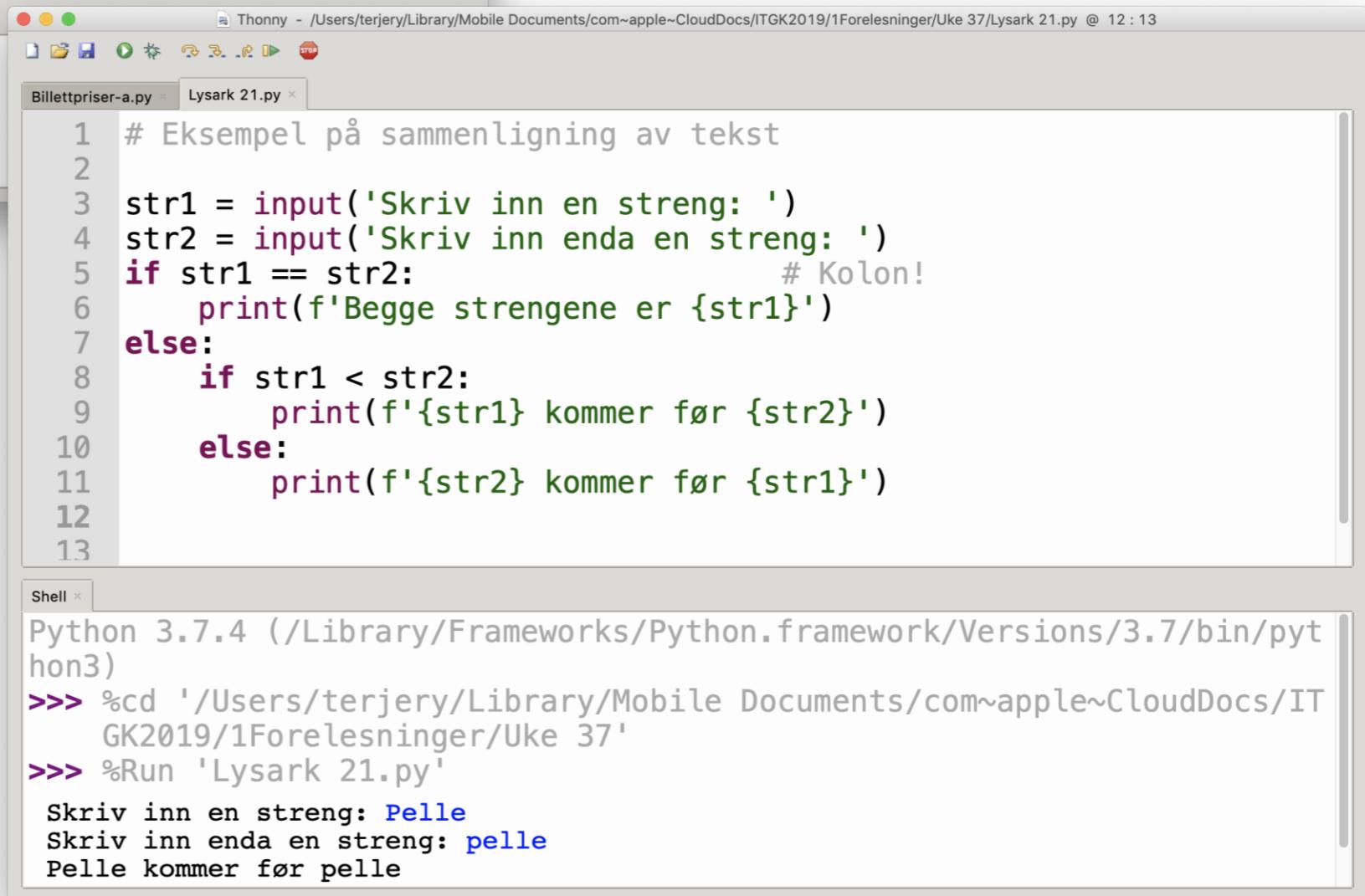
```

Shell x

```

>>> %Run 'Lysark 21.py'
Skriv inn en streng: Petter
Skriv inn enda en streng: Pelle
Pelle kommer før Petter
>>> |

```



```

Billetpriser-a.py x Lysark 21.py x
1 # Eksempel på sammenligning av tekst
2
3 str1 = input('Skriv inn en streng: ')
4 str2 = input('Skriv inn enda en streng: ')
5 if str1 == str2:                      # Kolon!
6     print(f'Begge strengene er {str1}')
7 else:
8     if str1 < str2:
9         print(f'{str1} kommer før {str2}')
10    else:
11        print(f'{str2} kommer før {str1}')
12
13

```

Shell x

```

Python 3.7.4 (/Library/Frameworks/Python.framework/Versions/3.7/bin/python3)
>>> %cd '/Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/IT
GK2019/1Forelesninger/Uke 37'
>>> %Run 'Lysark 21.py'
Skriv inn en streng: Pelle
Skriv inn enda en streng: pelle
Pelle kommer før pelle

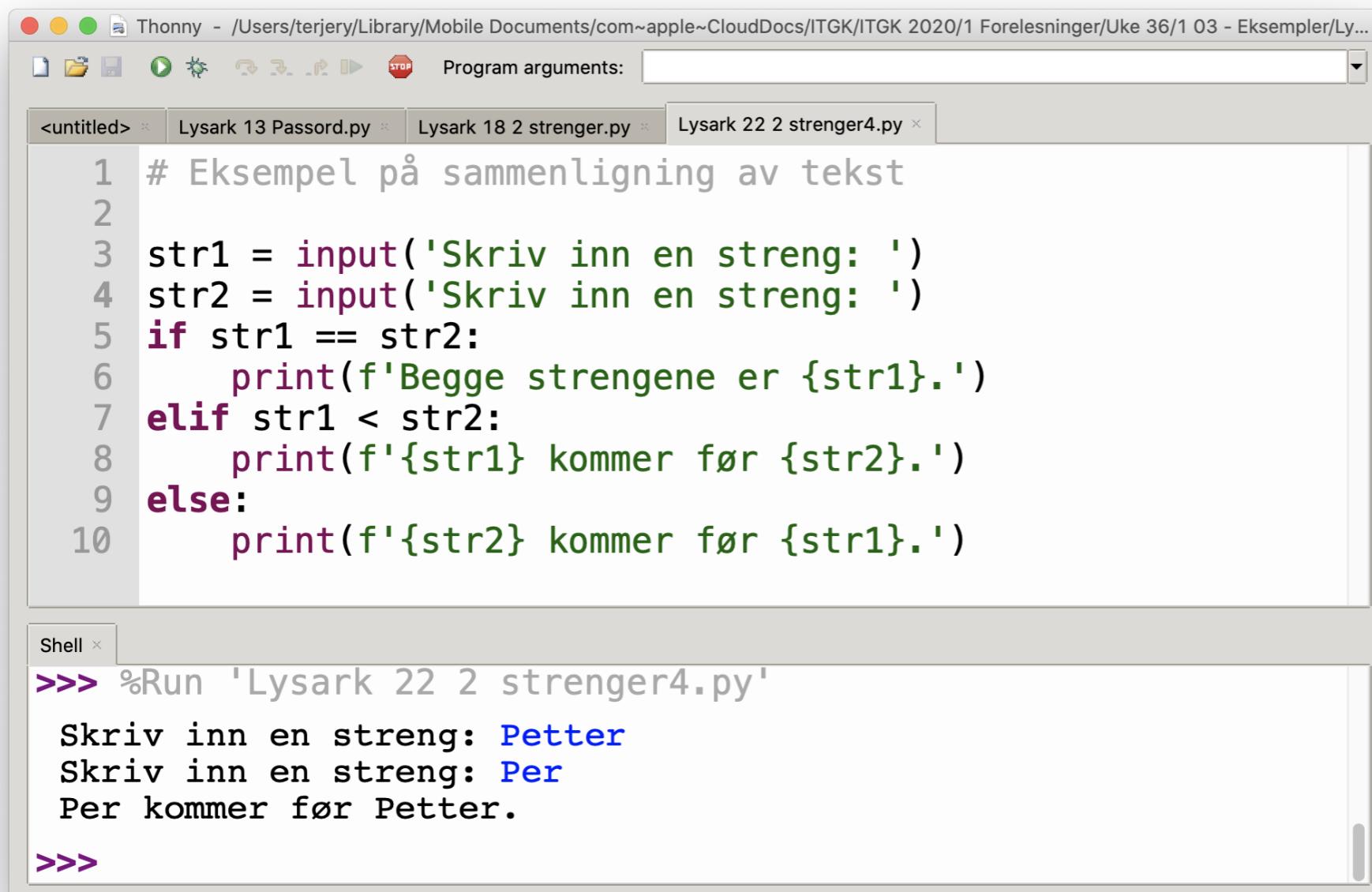
```

Siste if er unødvendig.

Hvis programmet kommer inn  
her vet vi allerede at  $\text{str2} < \text{str1}$

## 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....)



The screenshot shows the Thonny Python IDE interface. The top bar displays the title "Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK/ITGK 2020/1 Forelesninger/Uke 36/1 03 - Eksempler/Ly...". Below the title is a toolbar with various icons. The main area has tabs for "untitled", "Lysark 13 Passord.py", "Lysark 18 2 strenger.py", and "Lysark 22 2 strenger4.py", with "Lysark 22 2 strenger4.py" currently selected. The code editor contains the following Python script:

```

1 # Eksempel på sammenligning av tekst
2
3 str1 = input('Skriv inn en streng: ')
4 str2 = input('Skriv inn en streng: ')
5 if str1 == str2:
6     print(f'Begge strengene er {str1}.')
7 elif str1 < str2:
8     print(f'{str1} kommer før {str2}.')
9 else:
10    print(f'{str2} kommer før {str1}.')

```

Below the code editor is a "Shell" window with the following content:

```

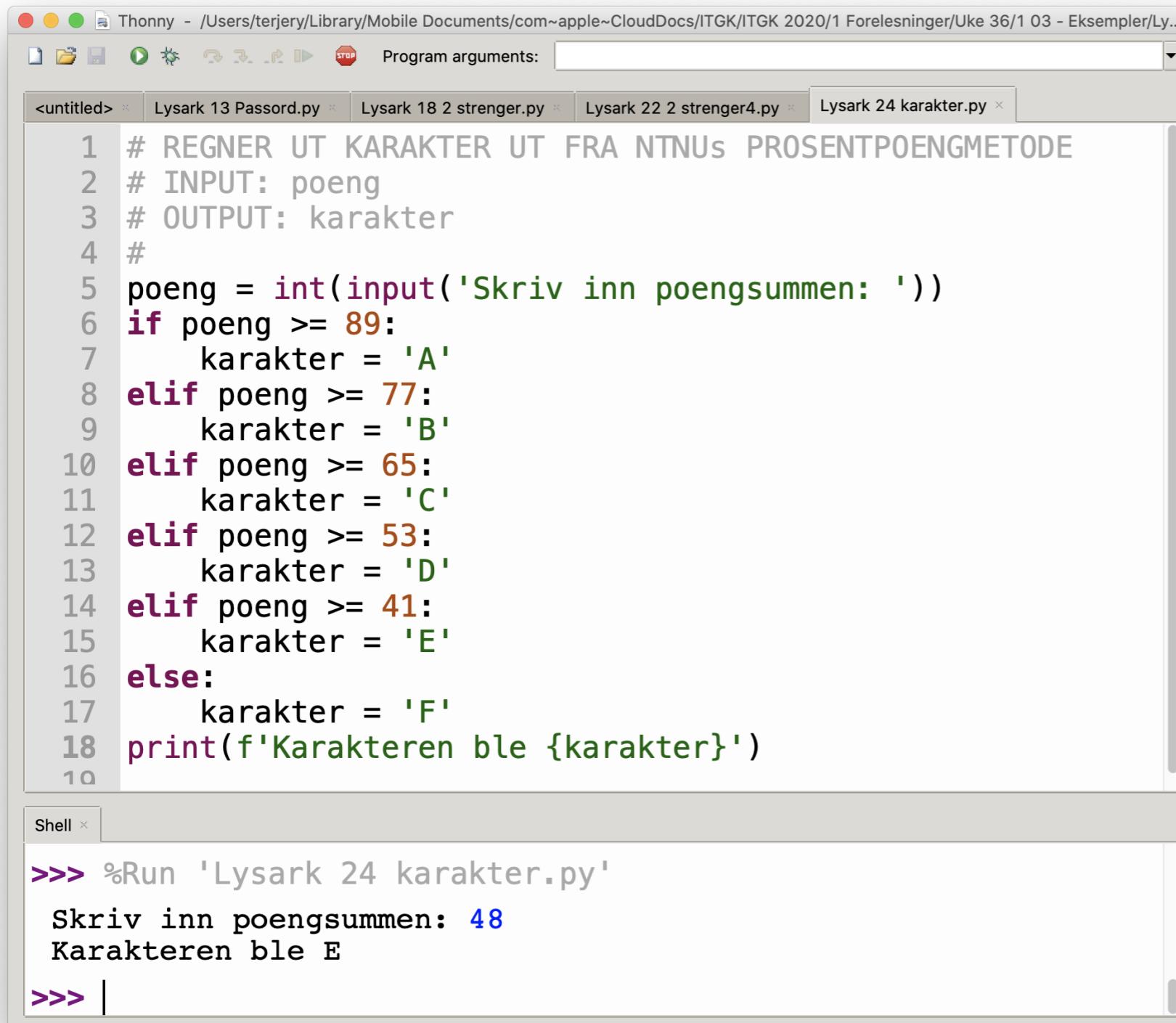
Shell ×
>>> %Run 'Lysark 22 2 strenger4.py'
Skriv inn en streng: Petter
Skriv inn en streng: Per
Per kommer før Petter.
>>>

```

- **NB!** Kun en av betingelsene vil slå til!

# 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....)



The screenshot shows the Thonny Python IDE interface. The top bar displays the title "Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK/ITGK 2020/1 Forelesninger/Uke 36/1 03 - Eksempler/Ly...". Below the title is a toolbar with various icons. The main window has tabs for "untitled", "Lysark 13 Passord.py", "Lysark 18 2 strenger.py", "Lysark 22 2 strenger4.py", and "Lysark 24 karakter.py". The "Lysark 24 karakter.py" tab is active. The code editor contains the following Python script:

```

1 # REGNER UT KARAKTER UT FRA NTNUs PROSENTPOENGMETODE
2 # INPUT: poeng
3 # OUTPUT: karakter
4 #
5 poeng = int(input('Skriv inn poengsummen: '))
6 if poeng >= 89:
7     karakter = 'A'
8 elif poeng >= 77:
9     karakter = 'B'
10 elif poeng >= 65:
11     karakter = 'C'
12 elif poeng >= 53:
13     karakter = 'D'
14 elif poeng >= 41:
15     karakter = 'E'
16 else:
17     karakter = 'F'
18 print(f'Karakteren ble {karakter}')
19

```

Below the code editor is a "Shell" window showing the output of running the script:

```

>>> %Run 'Lysark 24 karakter.py'
Skriv inn poengsummen: 48
Karakteren ble E
>>> |

```

- NB! Kun en av betingelsene vil slå til!

## Skuddåreksempel

- Regler for skuddår
  - Må være delelig med 4
  - Kan **ikke** være delelig med 100 med mindre det også er delelig med 400
- Med andre ord:
  - Hvis det ikke er delelig med 4 er det helt sikkert ikke et skuddår
  - Hvis det er delelig med 4 må man sjekke om det er delelig på 100
    - I så fall er det ikke et skuddår
  - Hvis ingen av disse betingelsene er oppfylt må man sjekke om der er delelig med 400
    - I så fall er det et skuddår
  - Hvis ingen av de over er oppfylle er det et skuddår

## Skuddåreksempel - pseudokode

- Gi info til bruker som forteller hva programmet gjør
- Les inn årstall fra tastatur
- Konverter streng til heltall
- Hvis året er  $\geq 1582$ 
  - Hvis året er et skuddår
    - skrive skuddårmelding
    - ellers
      - skriv melding om at det ikke er et skuddår

Hvordan gjør vi det?



## Skuddåreksempel

- Hvis året er et skuddår
  - Mange mulige løsninger
    - Nøstet if...else struktur
    - Kombinasjon av logiske operatorer (sammenligninger og and/or/not)
    - if...elif...else struktur

## Pseudokode for Nøstet if...else struktur

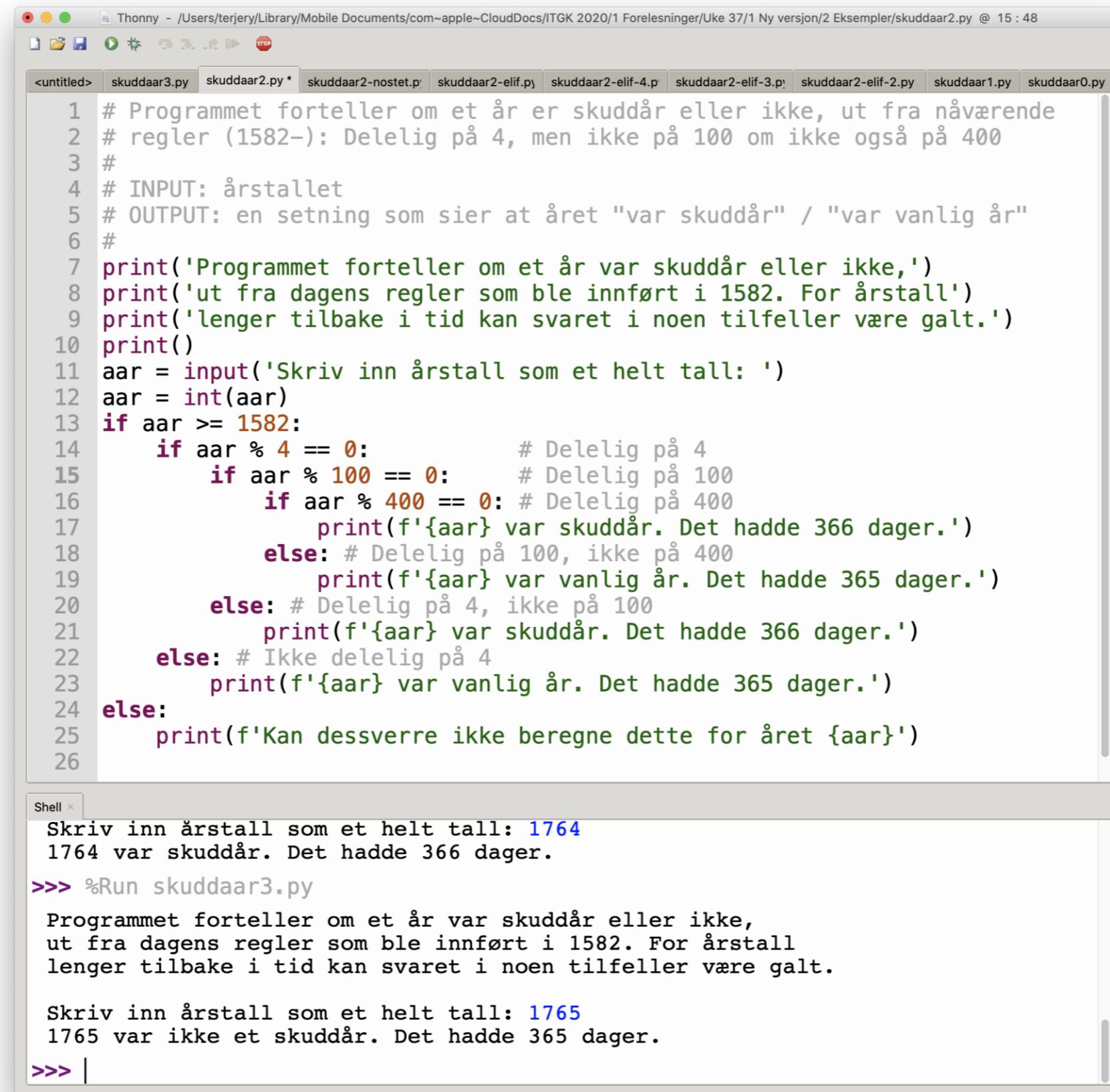
- **Er det et skuddår?**

- Hvis året er delelig med 4
  - Hvis året er delelig med 100
    - Hvis året er delelig med 400
      - Det er et skuddår
    - ellers
      - det er ikke et skuddår
  - ellers
    - det er et skuddår
- ellers
  - det er ikke et skuddår

Vi ser allerede her at dette ikke er en god løsning.  
For komplisert!

# Skuddåreksempl

- Nøstet if...else struktur



The screenshot shows the Thonny Python IDE interface. The code editor window displays a script named `skuddaar3.py` containing a nested if...else structure to determine if a year is a leap year according to the Gregorian calendar rules (1582). The code includes comments explaining the logic and handles input from the user.

```

1 # Programmet forteller om et år er skuddår eller ikke, ut fra nåværende
2 # regler (1582-): Delelig på 4, men ikke på 100 om ikke også på 400
3 #
4 # INPUT: årstallet
5 # OUTPUT: en setning som sier at året "var skuddår" / "var vanlig år"
6 #
7 print('Programmet forteller om et år var skuddår eller ikke,')
8 print('ut fra dagens regler som ble innført i 1582. For årstall')
9 print('lenger tilbake i tid kan svaret i noen tilfeller være galt.')
10 print()
11 aar = input('Skriv inn årstall som et helt tall: ')
12 aar = int(aar)
13 if aar >= 1582:
14     if aar % 4 == 0:          # Delelig på 4
15         if aar % 100 == 0:    # Delelig på 100
16             if aar % 400 == 0: # Delelig på 400
17                 print(f'{aar} var skuddår. Det hadde 366 dager.')
18             else: # Delelig på 100, ikke på 400
19                 print(f'{aar} var vanlig år. Det hadde 365 dager.')
20         else: # Delelig på 4, ikke på 100
21             print(f'{aar} var skuddår. Det hadde 366 dager.')
22     else: # Ikke delelig på 4
23         print(f'{aar} var vanlig år. Det hadde 365 dager.')
24 else:
25     print(f'Kan dessverre ikke beregne dette for året {aar}')
26

```

The shell window below shows the execution of the script and its output for the year 1764 and 1765.

```

Shell <
Skriv inn årstall som et helt tall: 1764
1764 var skuddår. Det hadde 366 dager.

>>> %Run skuddaar3.py
Programmet forteller om et år var skuddår eller ikke,
ut fra dagens regler som ble innført i 1582. For årstall
lenger tilbake i tid kan svaret i noen tilfeller være galt.

Skriv inn årstall som et helt tall: 1765
1765 var ikke et skuddår. Det hadde 365 dager.

>>> |

```

## Pseudokode for if...elif...else struktur

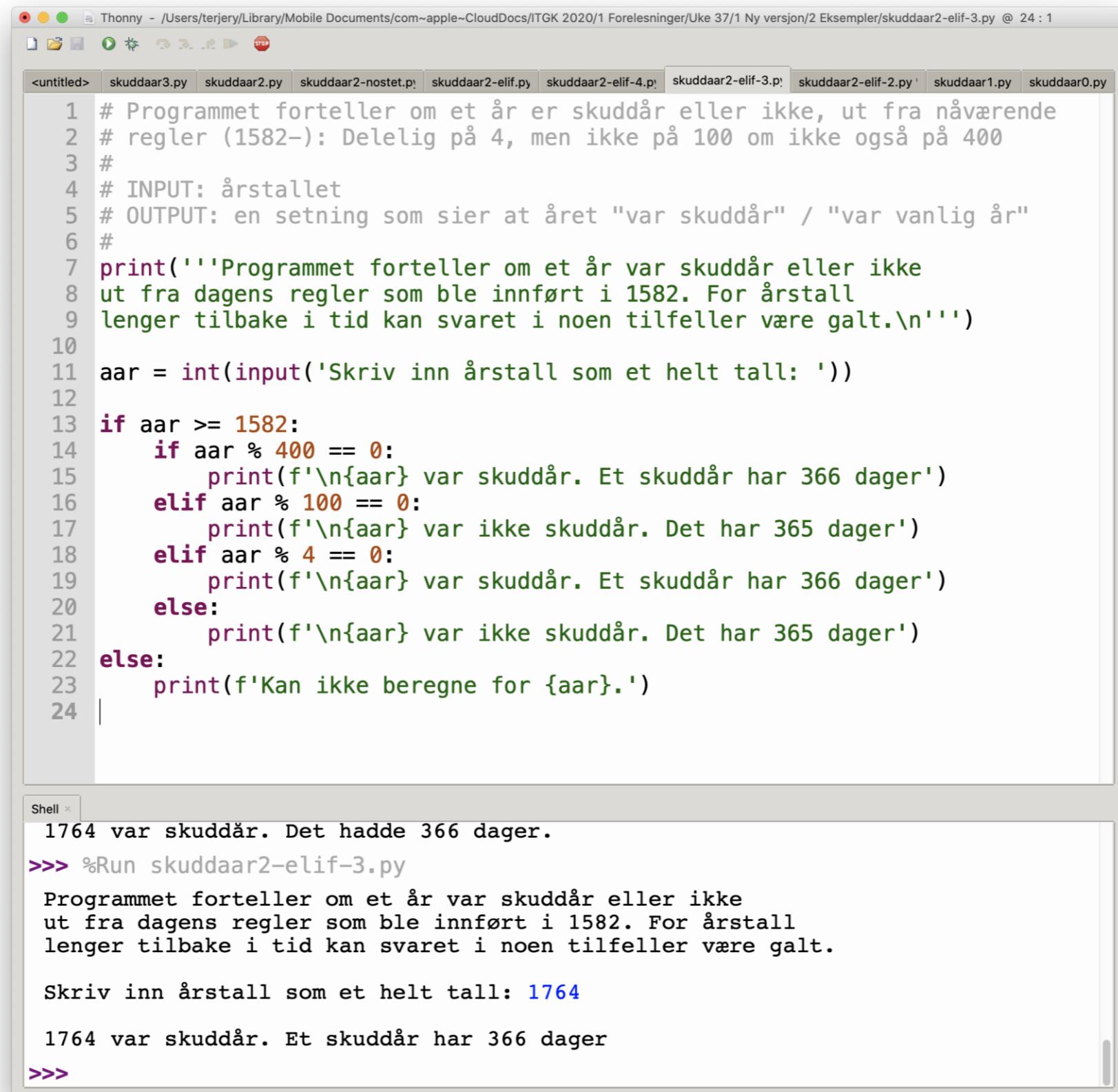
- **Er det et skuddår?**

- Hvis året er delelig med 400
  - Det er et skuddår
- ellers hvis året er delelig med 100
  - Det er ikke et skuddår
- ellers hvis året er delelig med 4
  - det er et skuddår
- ellers
  - det er ikke et skuddår

Mye enklere!

# Skuddåreksempl

- if...elif...else struktur - alternativ 2



The screenshot shows the Thonny Python IDE interface. The code editor window displays a script named 'skuddaar2-elif-3.py' which calculates whether a given year is a leap year or not based on the Gregorian calendar rules. The script includes a docstring explaining the rules, prompts for a year input, and uses an if-elif-else structure to determine if the year is a leap year. The shell window below shows the execution of the script with the year 1764, resulting in the output '1764 var skuddår. Det hadde 366 dager.'

```

1 # Programmet forteller om et år er skuddår eller ikke, ut fra nåværende
2 # regler (1582-): Delelig på 4, men ikke på 100 om ikke også på 400
3 #
4 # INPUT: årstallet
5 # OUTPUT: en setning som sier at året "var skuddår" / "var vanlig år"
6 #
7 print('''Programmet forteller om et år var skuddår eller ikke
8 ut fra dagens regler som ble innført i 1582. For årstall
9 lenger tilbake i tid kan svaret i noen tilfeller være galt.\n''')
10
11 aar = int(input('Skriv inn årstall som et helt tall: '))
12
13 if aar >= 1582:
14     if aar % 400 == 0:
15         print(f'\n{aar} var skuddår. Et skuddår har 366 dager')
16     elif aar % 100 == 0:
17         print(f'\n{aar} var ikke skuddår. Det har 365 dager')
18     elif aar % 4 == 0:
19         print(f'\n{aar} var skuddår. Et skuddår har 366 dager')
20     else:
21         print(f'\n{aar} var ikke skuddår. Det har 365 dager')
22 else:
23     print(f'Kan ikke beregne for {aar}.')
24

```

Shell x

```

1764 var skuddår. Det hadde 366 dager.
>>> %Run skuddaar2-elif-3.py
Programmet forteller om et år var skuddår eller ikke
ut fra dagens regler som ble innført i 1582. For årstall
lenger tilbake i tid kan svaret i noen tilfeller være galt.

Skriv inn årstall som et helt tall: 1764
1764 var skuddår. Et skuddår har 366 dager
>>>

```

# Skuddåreksempel

- if...elif...else struktur (alternativ, men dårligere løsning)

The screenshot shows the Thonny Python IDE interface. The top window displays the source code for `skuddaar2-elif.py`. The code implements the Gregorian calendar leap year rules starting from 1582. It prompts the user for a year, checks if it's divisible by 4, then 100, then 400, and finally falls back to an else case. The bottom window, titled "Shell", shows the execution of the script and its output for the years 1765 and 1764.

```

2 # regler (1582-): Delelig på 4, men ikke på 100 om ikke også på 400
3 #
4 # INPUT: årstallet
5 # OUTPUT: en setning som sier at året "var skuddår" / "var vanlig år"
6 #
7 print('Programmet forteller om et år var skuddår eller ikke,')
8 print('ut fra dagens regler som ble innført i 1582. For årstall')
9 print('lenger tilbake i tid kan svaret i noen tilfeller være galt.')
10 print()
11 aar = input('Skriv inn årstall som et helt tall: ')
12 aar = int(aar)
13 if aar >= 1582:
14     if aar % 4 != 0: # Ikke delelig på 4
15         print(f'{aar} var ikke skuddår. Det hadde 365 dager.')
16
17     elif aar % 100 != 0: # Delelig på 4, ikke på 100
18         print(f'{aar} var skuddår. Det hadde 366 dager.')
19
20     elif aar % 400 != 0: # Delelig på 100, ikke på 400
21         print(f'{aar} var ikke skuddår. Det hadde 365 dager.')
22
23     else: #Delelig på 400
24         print(f'{aar} var skuddår. Det hadde 366 dager.')
25 else:
26     print(f'Kan dessverre ikke beregne dette for året {aar}')
27

```

Shell ×

```

Skriv inn årstall som et helt tall: 1765
1765 var ikke et skuddår. Det hadde 365 dager.
>>> %Run skuddaar2-elif.py
Programmet forteller om et år var skuddår eller ikke,
ut fra dagens regler som ble innført i 1582. For årstall
lenger tilbake i tid kan svaret i noen tilfeller være galt.

Skriv inn årstall som et helt tall: 1764
1764 var skuddår. Det hadde 366 dager.
>>>

```

## Skuddåreksempel - pseudokode

- Sammenligning av de 2 siste løsningene

```

13 if aar >= 1582:
14     if aar % 4 != 0: # Ikke delelig på 4
15         print(f'{aar} var ikke skuddår. Det hadde 365 dager.')
16
17 elif aar % 100 != 0: # Delelig på 4, ikke på 100
18     print(f'{aar} var skuddår. Det hadde 366 dager.')
19
20 elif aar % 400 != 0: # Delelig på 100, ikke på 400
21     print(f'{aar} var ikke skuddår. Det hadde 365 dager.')
22
23 else: #Delelig på 400
24     print(f'{aar} var skuddår. Det hadde 366 dager.')
25 else:
26     print(f'Kan dessverre ikke beregne dette for året {aar}')

```

Tester på !=

Tester på ==

```

13 if aar >= 1582:
14     if aar % 400 == 0:
15         print(f'\n{aar} var skuddår. Et skuddår har 366 dager')
16     elif aar % 100 == 0:
17         print(f'\n{aar} var ikke skuddår. Det har 365 dager')
18     elif aar % 4 == 0:
19         print(f'\n{aar} var skuddår. Et skuddår har 366 dager')
20     else:
21         print(f'\n{aar} var ikke skuddår. Det har 365 dager')
22 else:
23     print(f'Kan ikke beregne for {aar}.')
24

```

- 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
<b>and</b>	Logisk og
<b>or</b>	Logisk eller
<b>not</b>	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:

<b>False and True</b>	gir	False
<b>False and False</b>	gir	False
<b>True and True</b>	gir	True

- Et uttrykk med **or** blir True hvis minst en av sidene er True:

<b>False or True</b>	gir	True
<b>True or True</b>	gir	True
<b>False or False</b>	gir	False

- **not** snur uttrykket:

<b>not True</b>	gir	False
<b>not False</b>	gir	True

```

1 a = int(input('Oppgi et heltall: '))
2 b = int(input('Oppgi et heltall til: '))
3 if a > 5 and b < 7:
4     print('a var større enn 5 og b var mindre enn 7')
5 else:
6     print('Enten var a mindre eller lik 5 eller så var b større eller lik 7')

>>> %Run 'Lysark 35.py'
Oppgi et heltall: 4
Oppgi et heltall til: 4
Enten var a mindre eller lik 5 eller så var b større eller lik 7
>>>

```

# Sannhetsverditabell

a : Det regner i dag

b : Det blåser

	b	
a and b	<b>False</b>	<b>True</b>
<b>False</b>	False	False
<b>True</b>	False	True

	b	
a or b	<b>False</b>	<b>True</b>
<b>False</b>	False	True
<b>True</b>	True	True

a	not a
<b>False</b>	True
<b>True</b>	False

## Eksempel på logiske uttrykk

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

$x \geq 5 \text{ and } x \leq 10$

det kan være lurt å skrive  $(x \geq 5) \text{ and } (x \leq 10)$

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

$5 \leq x \leq 10$

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

$(i \geq 1 \text{ and } i \leq N) \text{ or } (j \geq 1 \text{ and } j \leq N)$

## 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 (oppføyd)
3. \*, /, //, % # heltallsdivisjon, rest
4. +, -
5. <, <=, >, >=, <>, !=, ==
6. not
7. and
8. or
9. if – else
10. Ved lik prioritet: fra venstre mot høyre
  - Bruk parenteser for å få uttrykkene riktig, eller for å tydeliggjøre rekkefølgen.

## 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:

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

## En liten test... Bestem **true** eller **false**

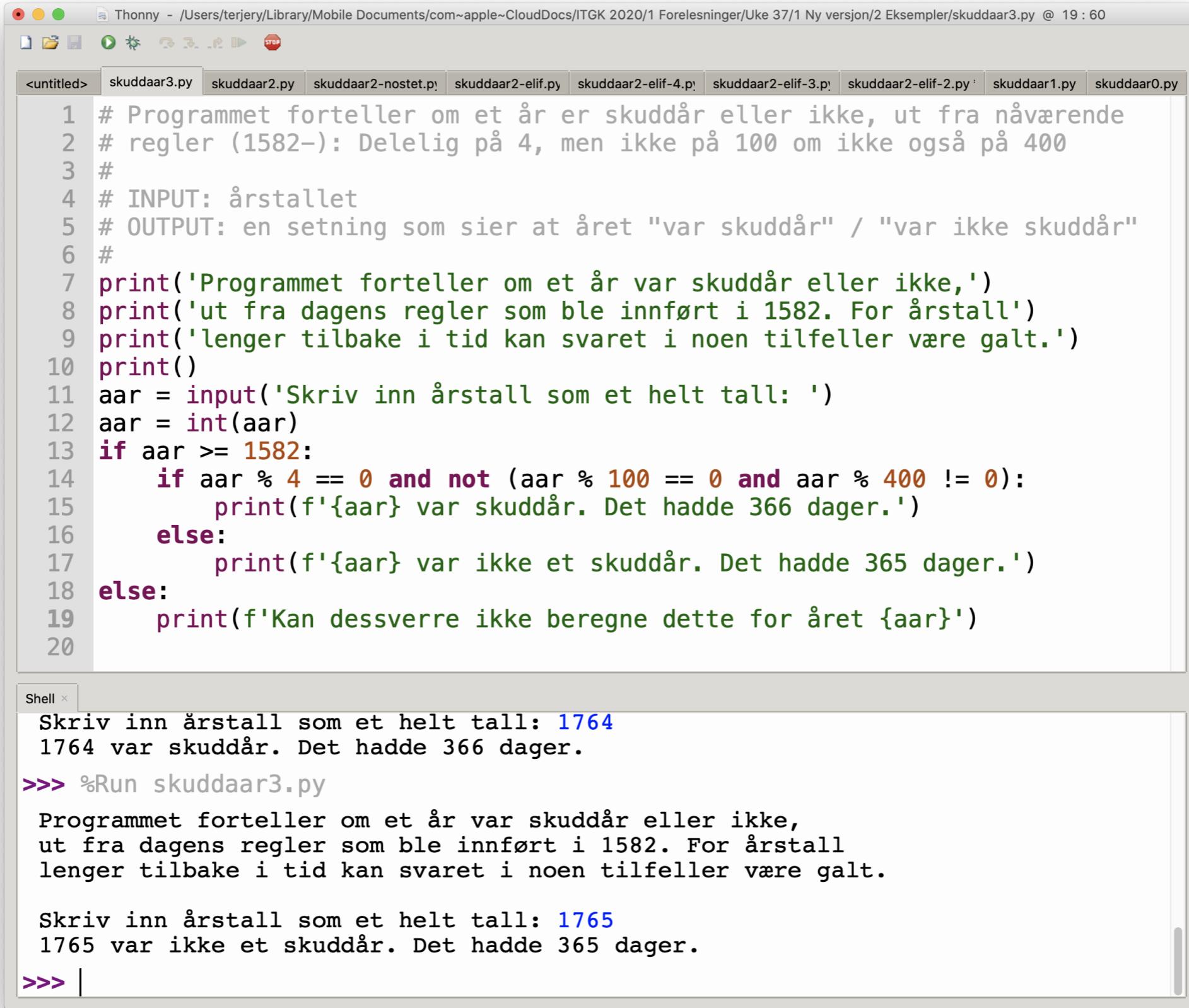
Bruk disse verdiene på variablene:

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

# Skuddåreksempel

- Kombinasjon av logiske operatorer (sammenligninger og and/or/not)



The screenshot shows the Thonny IDE interface. The top window displays the code for `skuddaar3.py`, which checks if a given year is a leap year based on the Gregorian calendar rules (divisible by 4, not divisible by 100, unless divisible by 400). The bottom window, titled "Shell", shows the interaction with the program:

```

Shell ×
Skriv inn årstall som et helt tall: 1764
1764 var skuddår. Det hadde 366 dager.

>>> %Run skuddaar3.py
Programmet forteller om et år var skuddår eller ikke,
ut fra dagens regler som ble innført i 1582. For årstall
lenger tilbake i tid kan svaret i noen tilfeller være galt.

Skriv inn årstall som et helt tall: 1765
1765 var ikke et skuddår. Det hadde 365 dager.

>>> |

```

## 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

Kan også skrives slik:

```
svar = input('Skriv svar: ')  
riktig = svar == 'tulling'
```

Sammenligning som gir resultatet **True** eller **False**  
som så tilordnes variabelen riktig

## Avslutning if-setninger

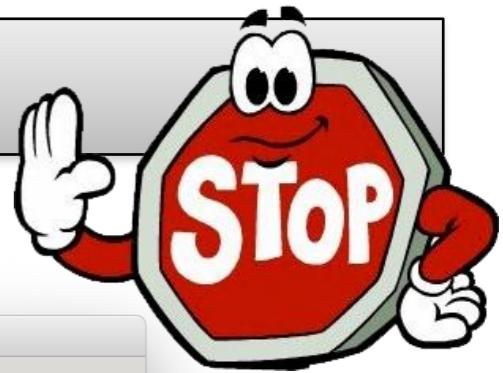
- Unngå overflødig bruk av negasjoner (**not**) - tungt å lese
- Ved `if else`, skriv helst positiv utfall i `if` og negativt i `else`
- Vi kan ha flere setningen mellom `if ... else`

## 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

# Oppgave: Dørvaktsimulator



Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK2019/1Forelesninger/Uke 37/Eksempler/doorman.py @ 13 : 1

Oppg 10.py ab.py Apostrof-og-anførselstegn.py Lysark 12.py doorman.py

```
1 print('Hei jeg er dørvakta!')
2 navn=input('Navnet ditt? ')
3 alder=int(input('Alder? '))
4
5 # Sjekker betingelsene
6
7 if alder>=18:
8     print('Du er gammel nok')
9     full=input(f'Er du full {navn}? ')
10    if full=='ja' or full=='Ja':
11        print('Du slipper ikke inn for du er full')
12    else:
13        print(f'Velkommen inn {navn}')
14 else:
15     print(f'Du er for ung {navn}')
16
```

Shell

```
Du er for ung Terje
>>> %Run doorman.py
Hei jeg er dørvakta!
Navnet ditt? Terje
Alder? 22
Du er gammel nok
Er du full Terje?
```

## Oppsummering

- Betingelser i Python: `<` , `>` , `<=` , `>=` , `==` , `!=` , `<>`
- Operatorer for logiske uttrykk: `and`, `or` , `not`
- Logiske uttrykk kan enten bli `False` eller `True`
- Vi kan også bruke nøstede if-setninger
- Husk : etter det logiske uttrykket i if, og etter else.

## 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.