



UNIVERSITETET I BERGEN

PRESEDENS

INF100

HØST 2024

Torstein Strømme

HJELP, DET ER EKSAMEN

- Øv på å programmere
- Øv på å lese, spore og forstå kode
- Øv på å forklare kode og konsepter (f. eks. quiz2/presentasjoner)
- Øv på å programmere

VANLIGE FEIL

- Å ikke prøve
- Manglende identifikasjon av delproblemer
 - For mange ting på samme linje
 - For mange ting i samme funksjon (manglende bruk av hjelpefunksjoner)
- Dårlige variabelnavn
 - `for i in a:` vs `for i in range(len(a)):`

VANLIGE FEIL

- Presedens
 - x and y in z
 - x == 3 or 4
- Funksjoner
 - print vs return
 - skop for variabler
 - return i destruktive funksjoner
- Lister
 - indeks vs. element
 - løkker over indeks vs elementer
 - modifisering av lister i en løkke
- Løkker
 - For tidlig return
 - Hva skal gjøres én gang, hva skal gjentas flere ganger?

EKSAMENSSTRATEGI

- Les gjennom alle oppgavene
 - Ikke kast bort tid på et problem du ikke har en klar plan for før du har lest nøye gjennom alle oppgavene.
- Første gang du leser problemet: identifiser delproblemer
 - Skriv ned kommentarer/idéer. Hva skal gjentas flere ganger? Hjelpesfunksjoner gjør hva?
 - Selv om du ikke klarer løse hele problemet, kjenner du kanskje igjen en del av det du klarer å løse.
- Prioriter oppgaver du kan løse raskt og oppgaver med mye poeng.

EKSAMENSSTRATEGI

- Bruke selvbeskrivende variabelnavn
 - Lettere for deg selv å forstå hva du driver med
 - Lettere for sensor å forstå hva du driver med
- Spis godt før du starter
- Sov godt før du starter
- Gå en lang tur dagen før (men ikke så sent på kvelden at du ikke får sove)

“99% of people fail to solve this problem”

$$6/2(1 + 2)$$

RECAP: UTTRYKK

$(x - y) * 2 + \text{foo}(x) > 42$

RECAP: UTTRYKK

```
res = (x - y) * 2 + foo(x) > 42
```

RECAP: UTTRYKK

```
if (x - y) * 2 + foo(x) > 42 :  
    ...
```

RECAP: UTTRYKK

```
while (x - y) * 2 + foo(x) > 42 :  
    ...
```

RECAP: UTTRYKK

((x - y) * 2 + foo(x) > 42) and foo(y) < 95

RECAP: UTTRYKK

$$(x - y) * 2 + \text{foo}(x) > 42$$

også et uttrykk

også et uttrykk

RECAP: UTTRYKK

$$(x - y) * 2 + \text{foo}(x) > 42$$


også et uttrykk

også et uttrykk

RECAP: UTTRYKK

$(x - y) * 2 + \text{foo}(x) > 42$

også et uttrykk



HVA ER ET UTTRYKK?

- Verdier er uttrykk
- Variabler er uttrykk
- Funksjonskall er uttrykk

42 True 'hello'

i number_of_students

max(x, y) a.sort()

- Hvis A og B er uttrykk og Ω er en operator så er A Ω B også et uttrykk

42 + foo(x)

(42 + foo(x)) * 95

- Hvis A er et uttrykk, så er (A) også et uttrykk

(42 + foo(x))

+ - * / // %
< > == != in
and or not* ...

*det er egne regler for «unary»-operatorer som ikke beskrives her

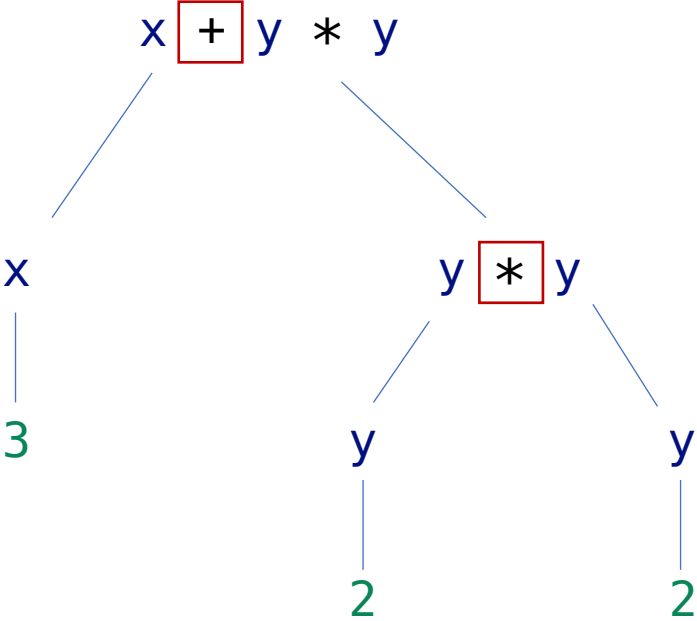
EVALUERING AV UTTRYKK

- Hvis et uttrykk er en verdi, er uttrykket ferdig evaluert.
- Hvis uttrykket er en variabel, evalueres uttrykket til den verdien variabelen referer til.
- Hvis uttrykket er et funksjonskall, evalueres uttrykket til returverdien
- Hvis uttrykket er i en parentes, evaluer uttrykket inne i parentesene
- Ellers:
 - Velg operatoren utenfor en parentes med lavest presedens lengst til høyre[★], og klipp uttrykket i to:
 - Evaluer venstresiden av uttrykket
 - Evaluer høyresiden av uttrykket
 - Kombiner verdiene med operatoren

★ unntak for **-operatøren (eksponentiering), der velges den lengst til venstre.

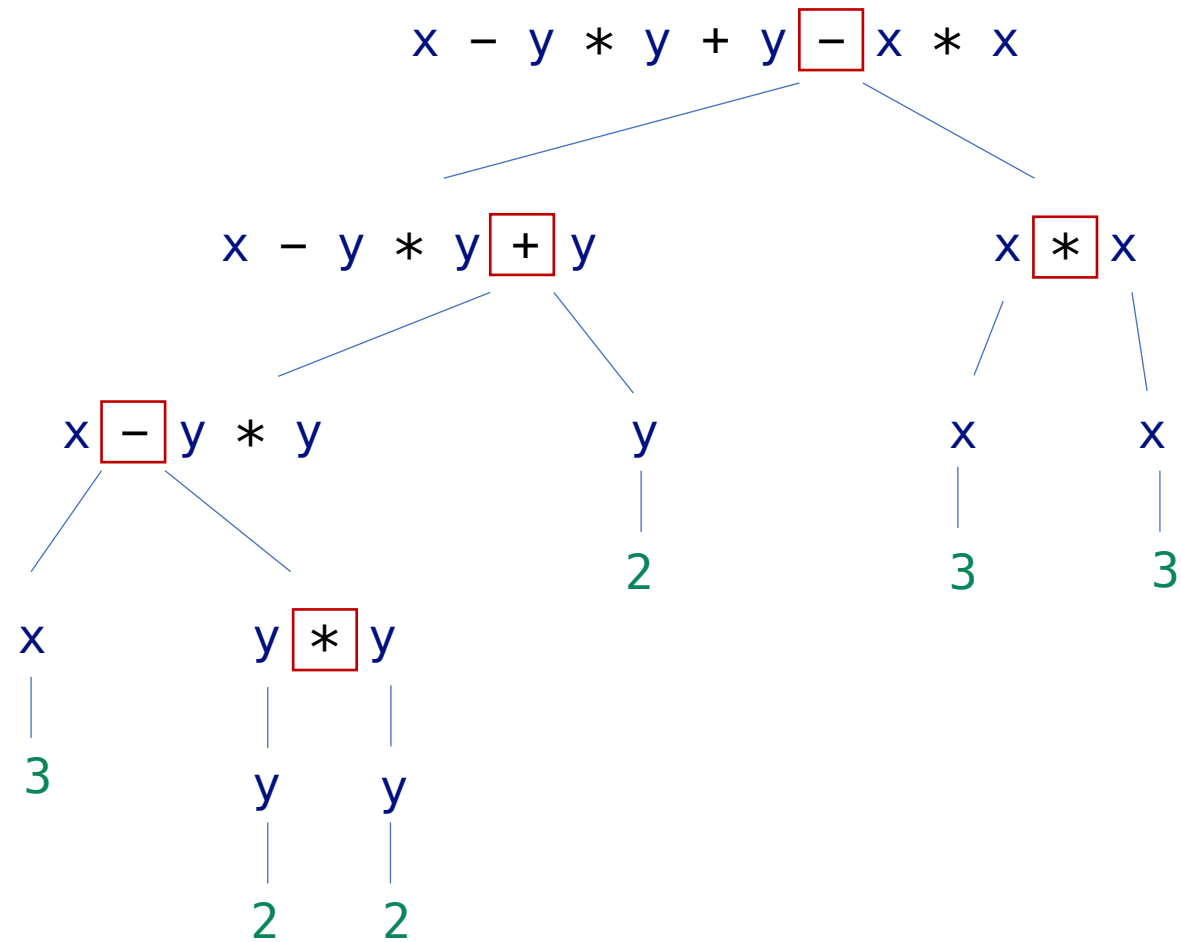
EKSEMPLER

$x = 3$
 $y = 2$



EKSEMPLER

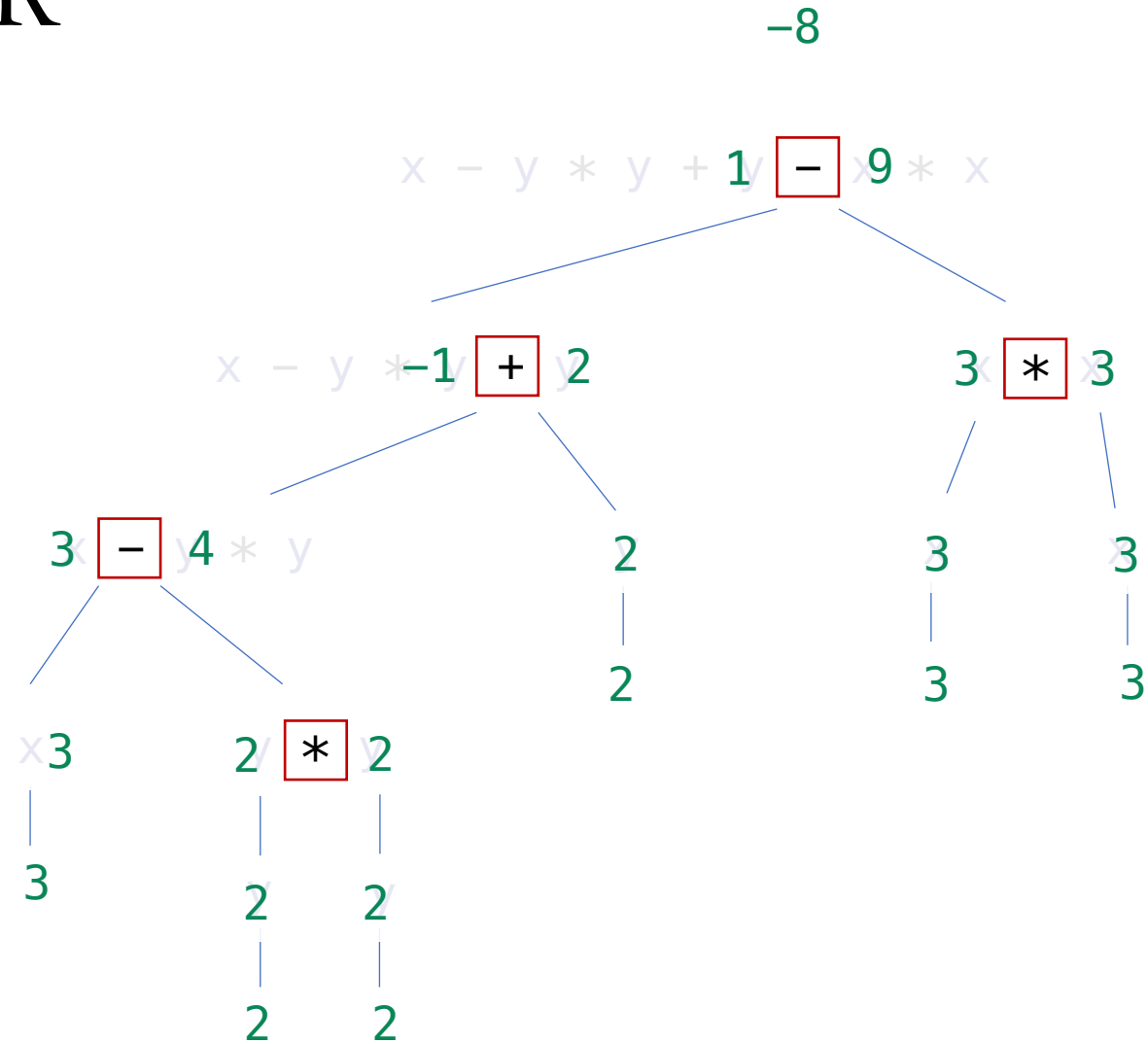
$$x = 3$$
$$y = 2$$



$$((x - (y * y)) + y) - (x * x)$$

EKSEMPLER

$x = 3$
 $y = 2$



$$((x - (y * y)) + y) - (x * x)$$

EVALUERING AV UTTRYKK

- Tommelfingerregel:
 - Operasjoner med høy presedens utføres først
 - Operasjoner med lik presedens utføres fra venstre mot høyre (unntak: **)
 - Evaluering av uttrykk på «samme nivå» evalueres fra venstre mot høyre

- Takeaway:
 - Benytt parenteser!

LOGISKE OPERATORER: SANNHETSTABELLER

x	y	x or y
True	True	True
True	False	True
False	True	True
False	False	False

LOGISKE OPERATORER: SANNHETSTABELLER

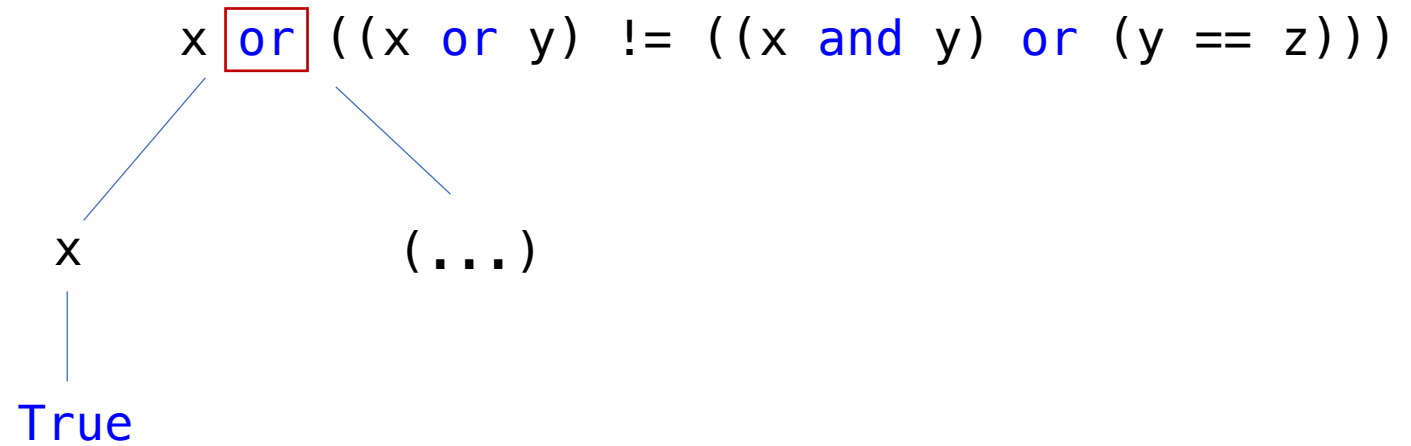
x	y	x or y	x and y
True	True	True	True
True	False	True	False
False	True	True	False
False	False	False	False

LOGISKE OPERATORER: SANNHETSTABELLER

x	y	x or y	x and y	(x and y) or y
True	True	True	True	True
True	False	True	False	False
False	True	True	False	True
False	False	False	False	False

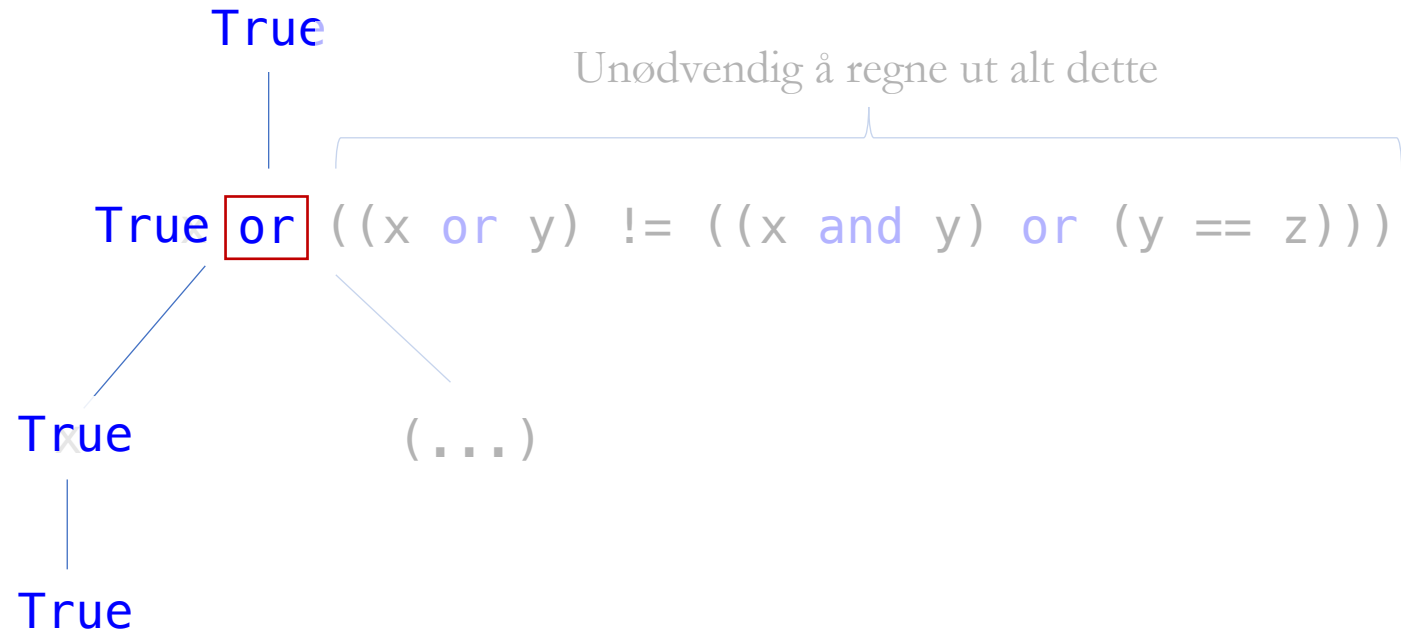
KORTSLUTNINGSEVALUERING

x = True
y = False
z = True



KORTSLUTNINGSEVALUERING

x = True
y = False
z = True



KORTSLUTNINGSEVALUERING

- Venstre side av **or** –uttrykk er **True** → høyre side evalueres ikke
- Venstre side av **and** –uttrykk er **False** → høyre side evalueres ikke

KORTSLUTNINGSEVALUERING

```
def true1():  
    print("true1", end=" ")  
    return True
```

```
def true2():  
    print("true2", end=" ")  
    return True
```

```
def false1():  
    print("false1", end=" ")  
    return False
```

```
def false2():  
    print("false2", end=" ")  
    return False
```

```
if true1():  
    print("johoo")
```

```
if false1() or true1():  
    print("johoo")
```

```
if true1() or true2():  
    print("johoo")
```

KORTSLUTNINGSEVALUERING

```
def is_large_int_wrong(x):  
    return x > 5 and type(x) == int
```

```
print(is_large_int_wrong(3))  
print(is_large_int_wrong(10))  
print(is_large_int_wrong("johoo")) # Krasjer
```

```
def is_large_int(x):  
    return type(x) == int and x > 5
```

```
print(is_large_int(3))  
print(is_large_int(10))  
print(is_large_int("johoo")) # False
```

KORTSLUTNINGSEVALUERING

```
def starts_with_x(s):  
    return len(s) > 0 and s[0] == 'x'
```

```
s = 'xfoo'
```

```
if starts_with_x(s):  
    print('yay!')
```

KORTSLUTNINGSEVALUERING

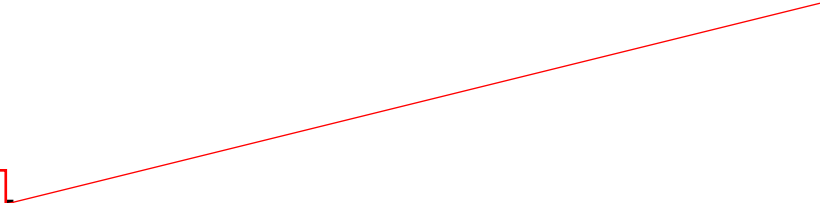
```
def starts_with_x(s):  
    return len(s) > 0 and s[0] == 'x'  
  
s = ''  
if starts_with_x(s): # Krasjer ikke!  
    print('yay!')
```

TRUTHY OG FALSY

```
x = 17
```

```
if x % 10:  
    print('if')  
else:  
    print('else')
```

evaluerer til 7



TRUTHY OG FALSY

```
if 7:  
    print('if')  
else:  
    print('else')
```

7 er en truthy verdi
(derfor skrives «if» ut
til skjermen)

TRUTHY OG FALSY

- Disse verdiene er falsy:

False

None

0 0.0

''

[] () {} set()


- (Nesten) alt annet er truthy.

TRUTHY OG FALSY

```
foo = 5  
bar = True
```

```
x = (foo or bar)
```

venstresiden er truthy; derfor evalueres uttrykket til venstre side 5



Hva `lft or rgt` *egentlig* evaluerer til:

Hvis `lft` er truthy, evaluer til `lft`;
ellers evaluer til `rgt`

TRUTHY OG FALSY

```
foo = 5  
bar = True
```

```
x = (bar and foo)
```

venstresiden er truthy; derfor evalueres uttrykket til høyre side 5

Hva `lft and rgt` *egentlig* evaluerer til:

Hvis `lft` er falsy, evaluer til `lft`;
ellers evaluer til `rgt`

SANNHETSTABELLEN STEMME

x	y	x or y	x and y
Truthy	Truthy	Truthy	Truthy
Truthy	Falsy	Truthy	Falsy
Falsy	Truthy	Truthy	Falsy
Falsy	Falsy	Falsy	Falsy

SANNHETSTABELLEN STEMME

x	y	x or y	x and y
42	'foo'	42	'foo'
42	''	42	''
0	'foo'	'foo'	0
0	''	''	0