UNIVERSITETET I BERGEN

PRESEDENS

INF100

HØST 2023

Torstein Strømme

David Grellscheid

PRØVEEKSAMEN

- Fredag 10. november 12:15 – 16:00
- Nash Auditorium, VilVite
  - eller hjemmefra

- 100% frivillig
- Blir ikke rettet *av oss*
- Sensorveiledning blir publisert → bytt besvarelse med en venn

HJELP, DET ER EKSAMEN

- Øv på å programmere
- Det *vil* komme spørsmål der du skal skrive kode selv på eksamen.
- Eksamen blir ikke «lettere» i nytt format – men sensor vil ikke gi trekk for dustefeil i koden din (stavefeil, feil navn på metoder fra standardbiblioteket etc.)

- Øv på å lese, spore og forstå kode
- Øv på å forklare kode og konsepter (gjør en en innsats på quiz2)
- 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? Hjelpefunksjoner 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 godt 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”

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

www.menti.com
8287 0925

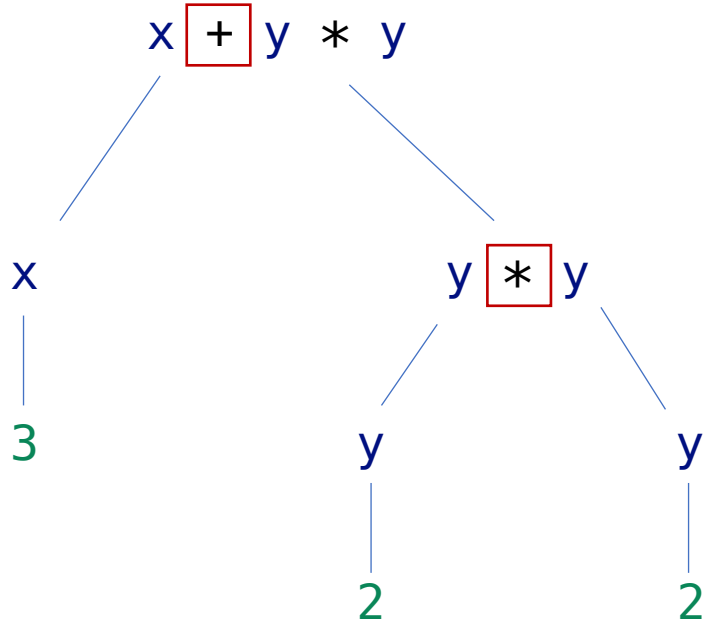
EVALUERING AV UTTRYKK

- 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 inne i en parentes, evaluer uttrykket inne i parentesene, og erstatt med evaluert verdi.
- Ellers:
  - Velg den operatoren med lavest presedens lengst til høyre★, og del uttrykket i to:
    - Evaluer venstresiden av uttrykket
    - Evaluer høyresiden av uttrykket
    - Kombiner verdiene med operatoren

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

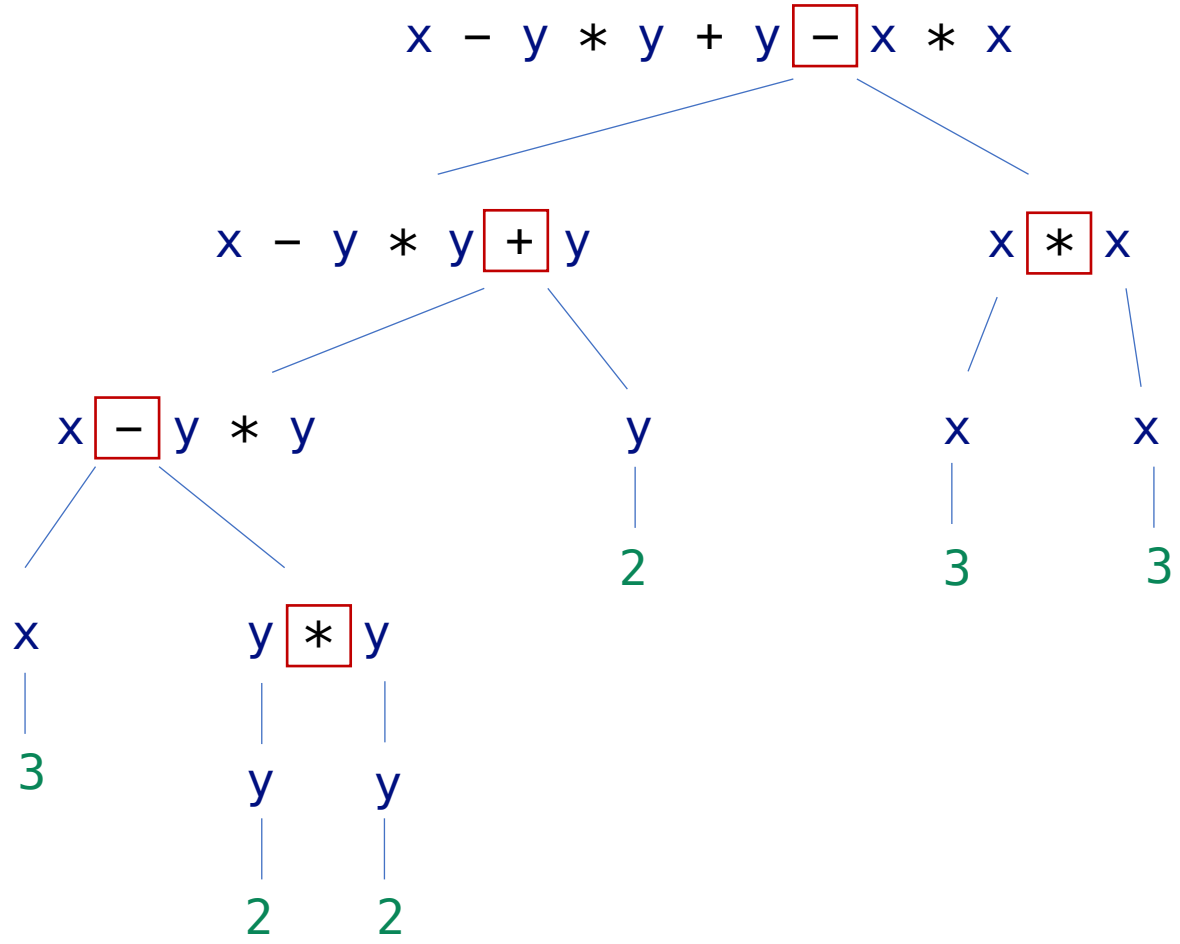
EKSEMPLER

x = 3
y = 2



EKSEMPLER

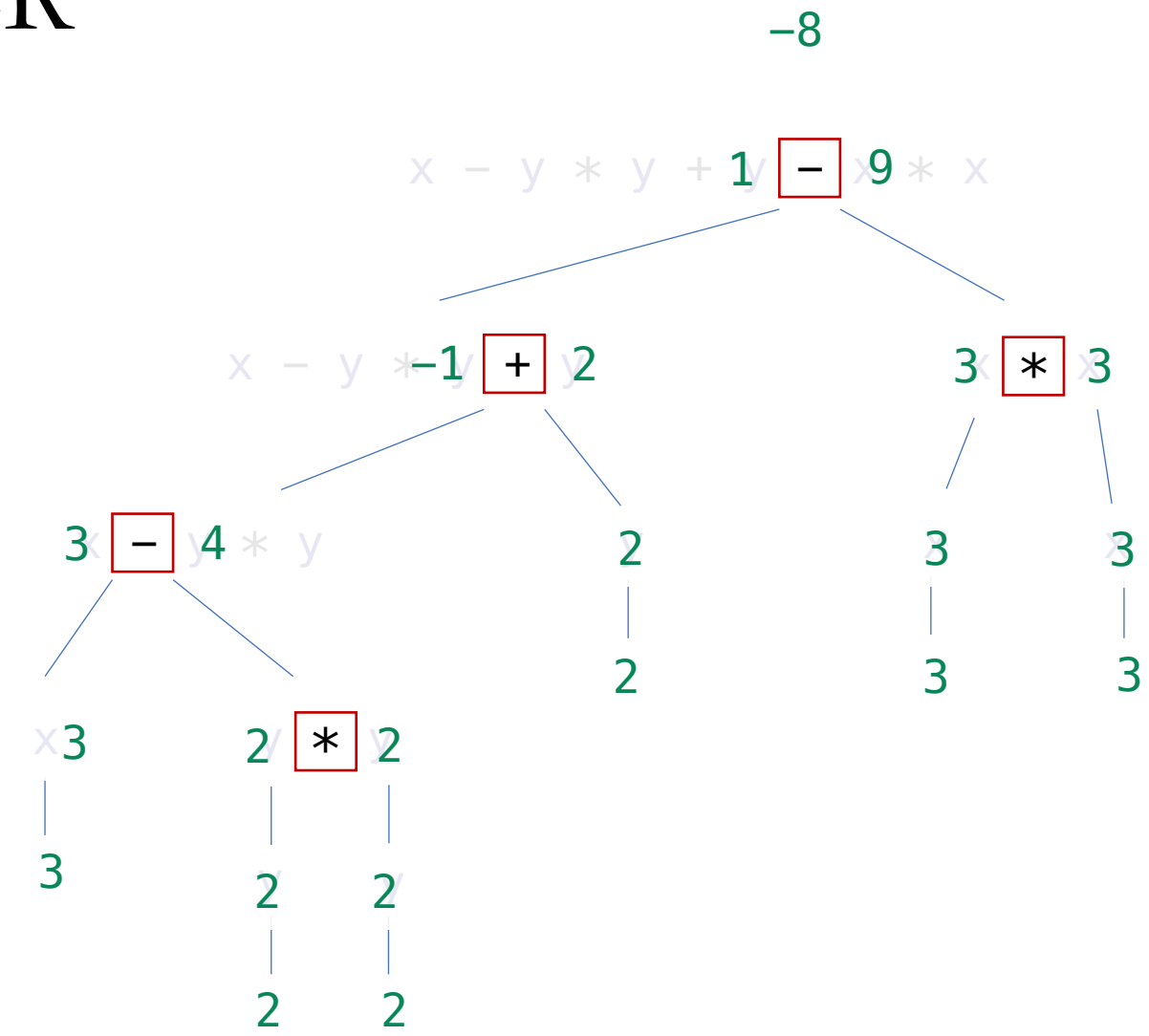
x = 3
y = 2



((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 funksjoner og variabler skjer fra venstre mot høyre (unntak: if else)

- 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

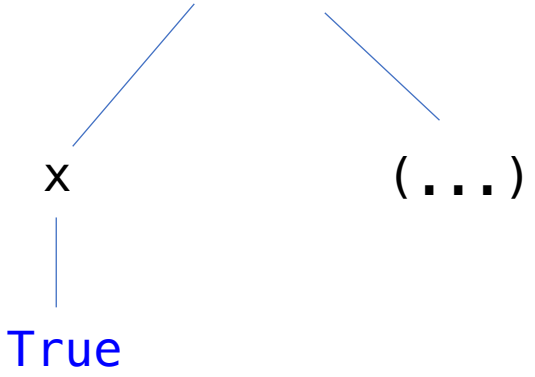
LOGISKE OPERATORER: SANNHETSTABELLER

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

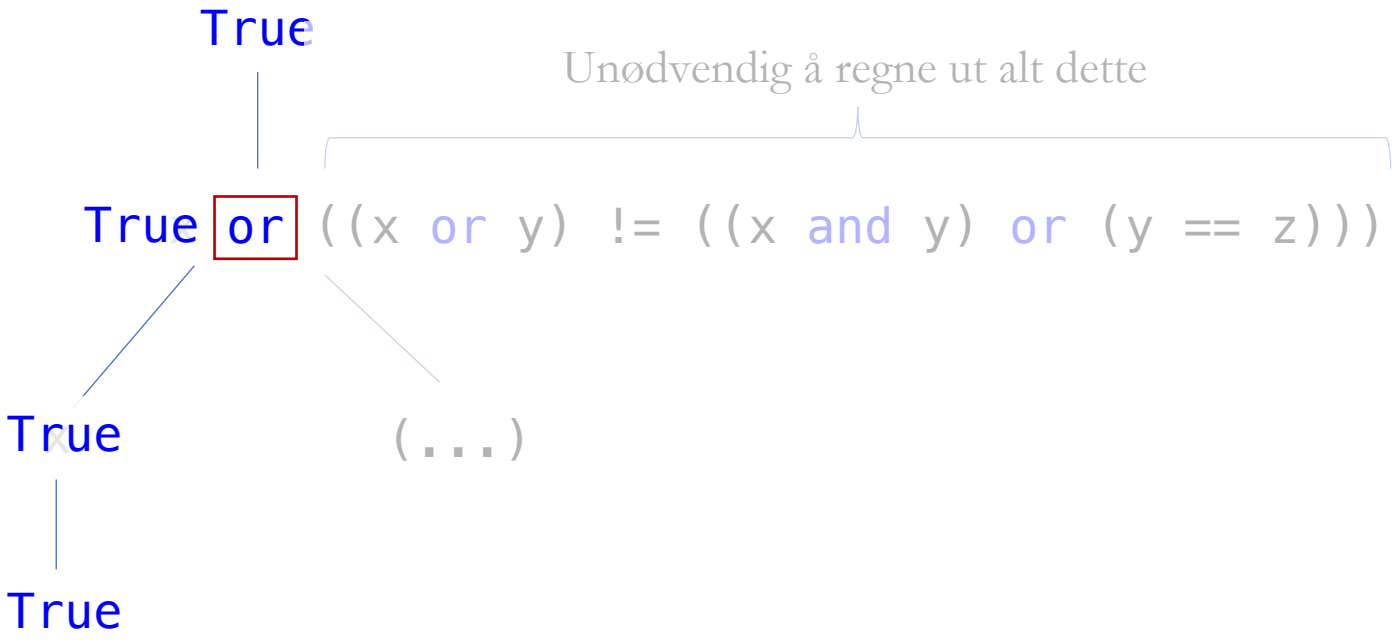
```
x = True
y = False
z = True
```

```
x or ((x or y) != ((x and y) or (y == z)))
```



KORTSLUTNINGSEVALUERING

```
x = True
y = False
z = True
```



KORTSLUTNINGSEVALUERING

- Venstre side av `or` –uttrykk er `True` → høyre side evlaueres 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

```python
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
```

```python
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
```

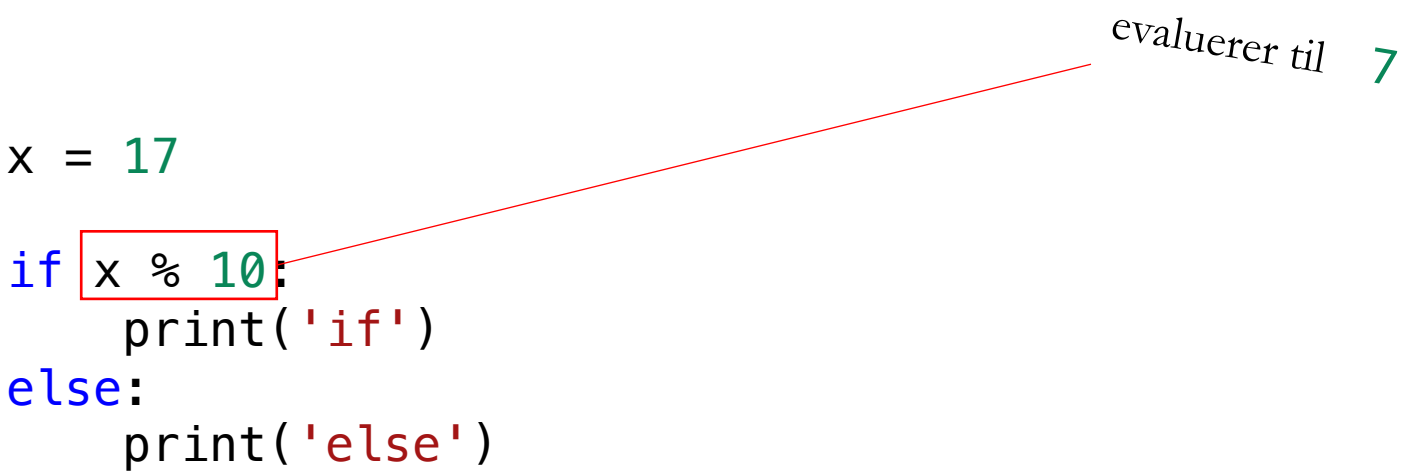
KORTSLUTNINGSEVALUERING

```
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



TRUTHY OG FALSY

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

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

TRUTHY OG FALSY

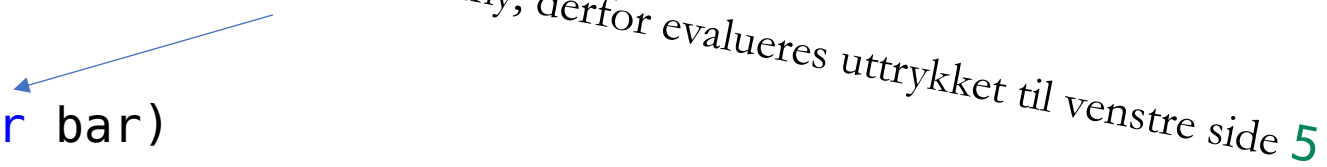
- Disse verdiene er falsy:

```
False
None
0       0.0
''
[]      ()      {}      set()
```

- (Nesten) alt annet er truthy.

TRUTHY OG FALSY

```
foo = 5
bar = True

x = (foo or bar)
```



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 STEMMER

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

PRESEDENS

www.menti.com

7447 8317

VANLIGE FEIL

```
def print_longest_words(w1, w2, w3):
    longest = max(w1, w2, w3)
    if len(w1) == longest:
        return w1
    elif len(w2) == longest:
        return w2
    elif len(w3) == longest:
        return w3
```

Feil: sammenligner feil typer ting

VANLIGE FEIL

```
def joker(x1, x2, x3):
    if x1 <= 4:
        print("opp")
    else:
        print("ned")
        if x2 <= 4:
            print("opp")
        else:
            print("ned")
            if x3 <= 4:
                print("opp")
            else:
                print("ned")
```

Feil: feil med innrykk gjør at kode får flere betingelser enn den skulle hatt.

VANLIGE FEIL

```python
def is_even_positive_int(x):
    return x==int and x >= 0 and x % 2 == 0
```

Feil: sammenligner feil typer ting

Tips: gjør én ting om gangen

VANLIGE FEIL

```
def are_all_ints(x1, x2, x3):
    if type(x1) == int:
        return True
    else:
        return False
    if type(x2) == int:
        return True
    else:
        return False
    if type(x3) == int:
        return True
    else:
        return False
```

```
def are_all_ints(x1, x2, x3):
    for x in (x1, x2, x3):
        if type(x) == int:
            return True
        else:
            return False
```

Feil: returnerer før man er ferdig å regne ut svaret

VANLIGE FEIL

```
def approx_area_under_g(x_lo, x_hi):
    running_total = 0
    for x in range(x_lo, x_hi):
        running_total += g(x)
        return running_total
```

Feil: returnerer inne i en løkke før man er ferdig å regne ut svaret

VANLIGE FEIL

```
def find_nth_occurrence(word, character, n):
    found_so_far = 0
    for i in word:
        if word[i] == character:
            found_so_far += 1
        if found_so_far == n:
            return i
    return -1
```

Feil: blander indekser og elementer i en løkke over lister