

BETINGELSER, FUNKSJONER OG FEIL

INF100

HØST 2024

Torstein Strømme

I DAG

- Betingelser
- Boolske uttrykk

- Funksjoner
- Krasj og feil
- Assert

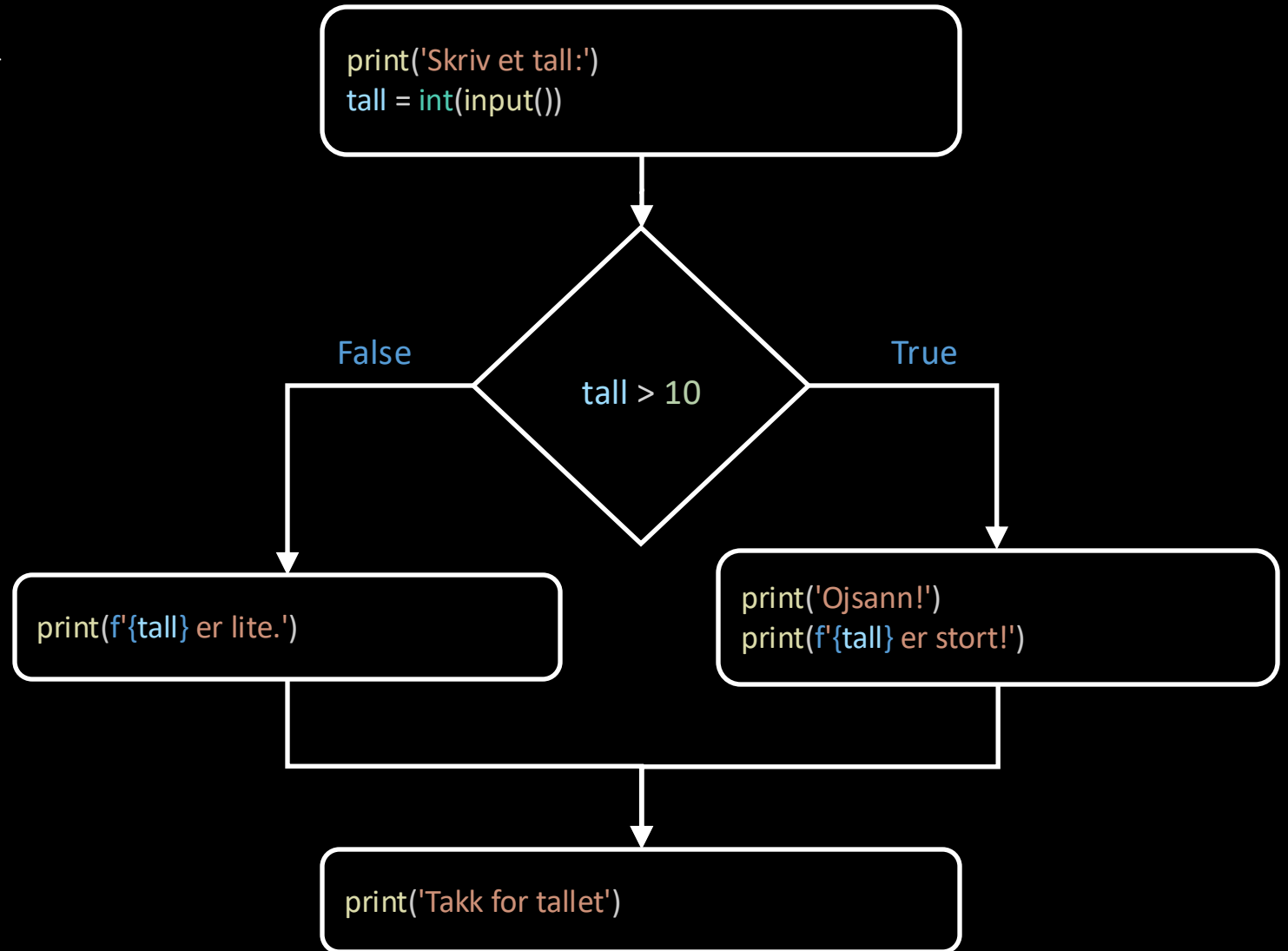
BETINGELSER

BETINGELSER

```
print('Skriv et tall:')  
tall = int(input())
```

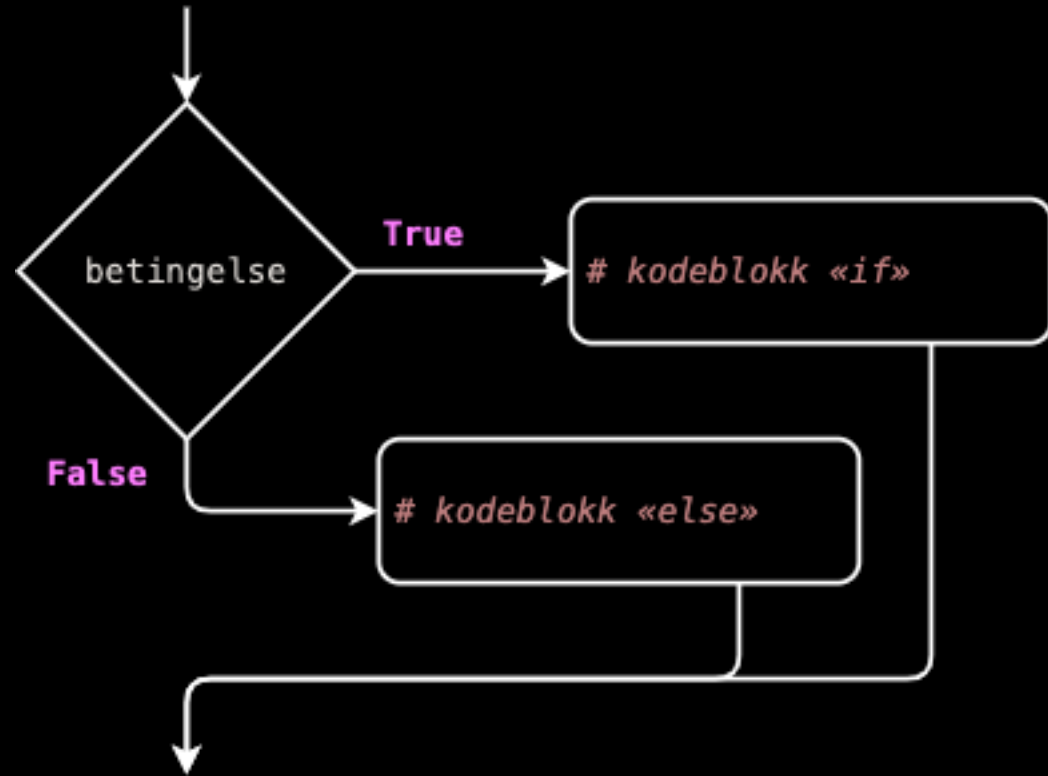
```
if tall > 10:  
    print('Ojsann!')  
    print(f'{tall} er stort!')  
else:  
    print(f'{tall} er lite.')
```

```
print('Takk for tallet')
```



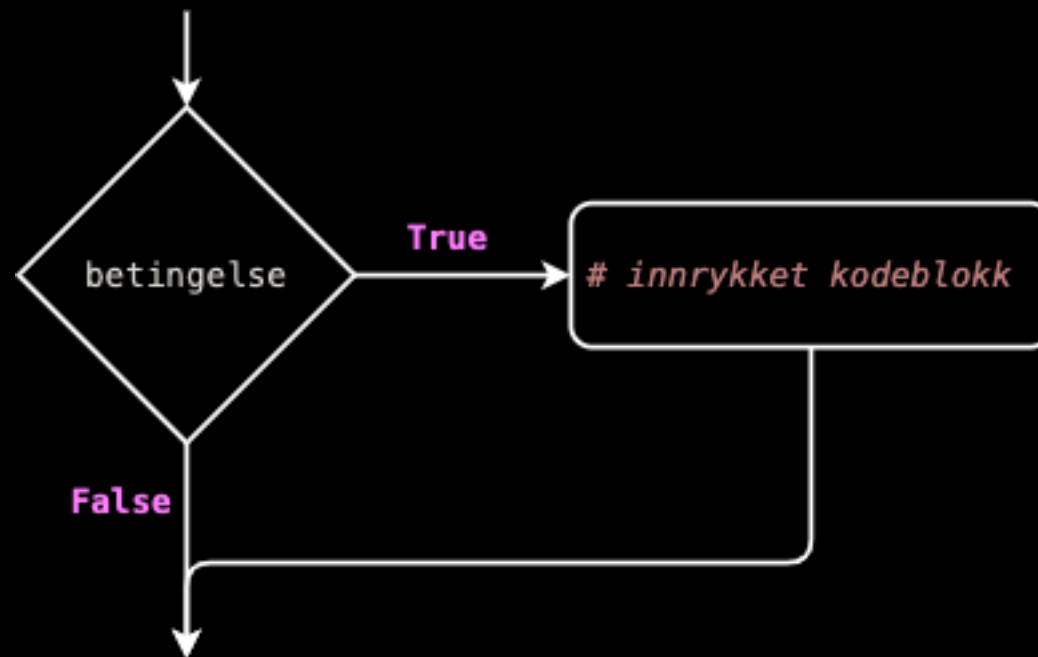
IF-ELSE

```
if (betingelse):  
    #  
    # kodeblokk «if»  
    #  
else:  
    #  
    # kodeblokk «else»  
    #  
  
# ...
```



IF

```
if (betingelse):  
    #  
    # kodeblokk «if»  
    #  
# ...
```

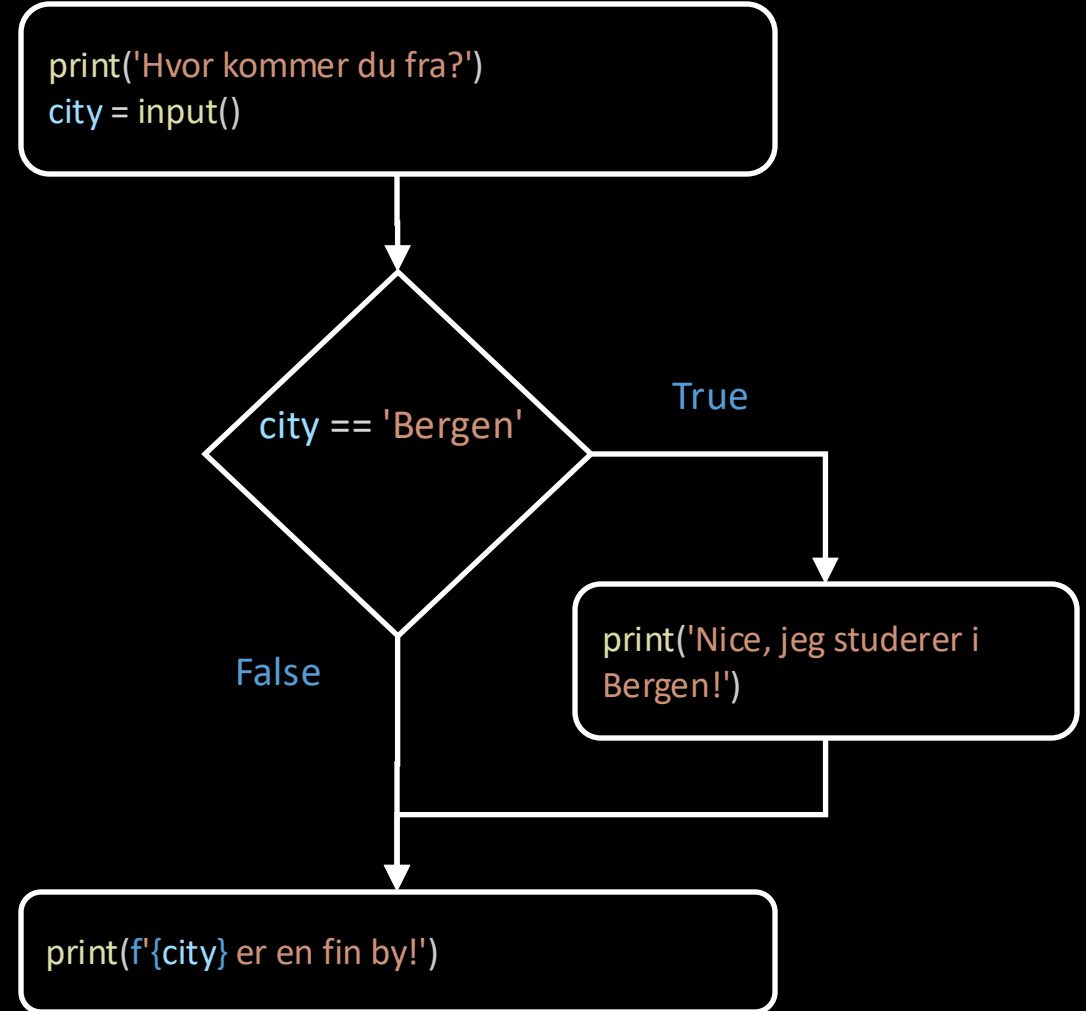


IF

```
print('Hvor kommer du fra?')  
city = input()
```

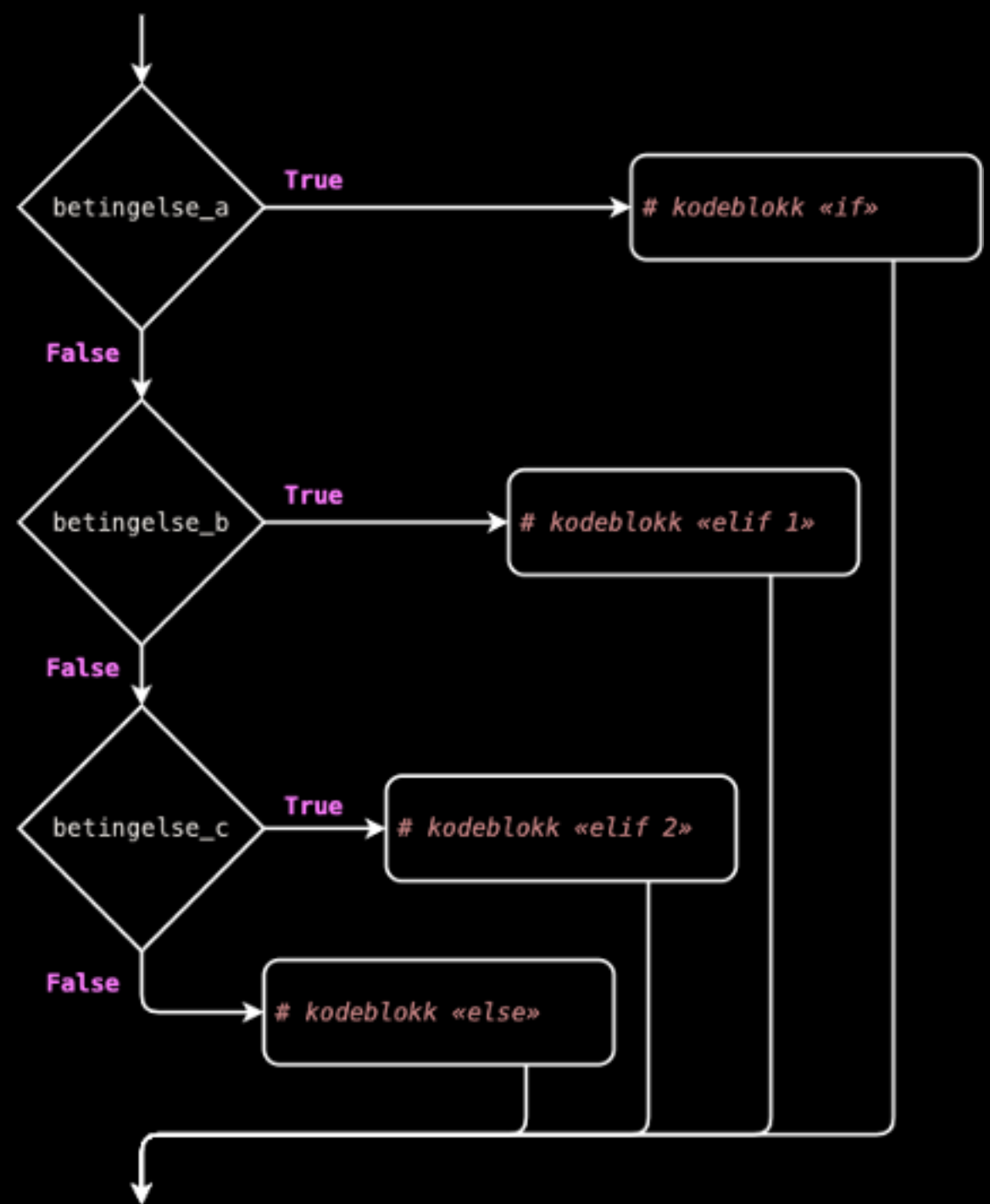
```
if (city == 'Bergen'):  
    print('Nice, jeg studerer i Bergen!')
```

```
print(f'{city} er en fin by!')
```



IF-ELIF-ELSE

```
if betingelse_a:  
    #  
    # kodeblokk «if»  
    #  
elif betingelse_b:  
    #  
    # kodeblokk «elif 1»  
    #  
elif betingelse_c:  
    #  
    # kodeblokk «elif 2»  
    #  
else:  
    #  
    # kodeblokk «else»  
    #  
# ...
```

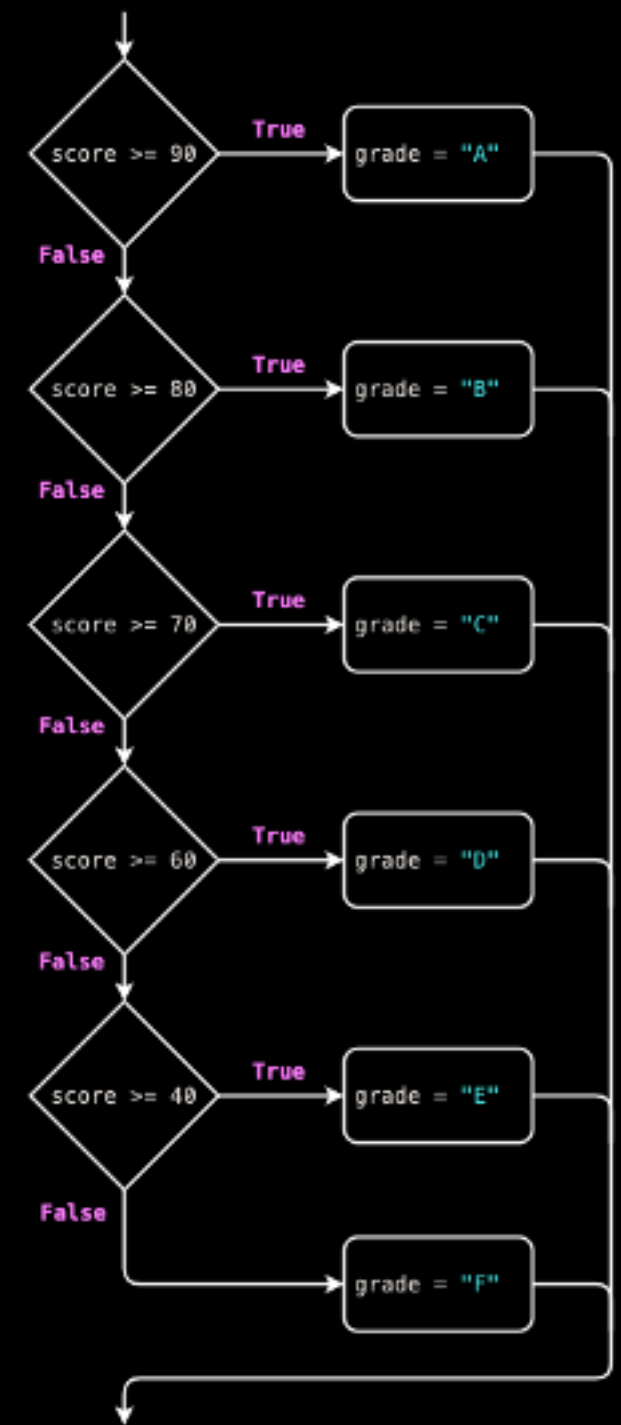


IF-ELIF-ELSE

```
print("Hvor mange poeng fikk du?")  
score = int(input())
```

```
if score >= 90:  
    grade = "A"  
elif score >= 80:  
    grade = "B"  
elif score >= 70:  
    grade = "C"  
elif score >= 60:  
    grade = "D"  
elif score >= 40:  
    grade = "E"  
else:  
    grade = "F"
```

```
print(f"Du fikk en {grade}.")
```



True

False

> < >= <= == in

BOOLSKE UTTRYKK

Expression / Uttrykk	Value / Verdi
True	True
False	False
4 == 5	False
66 == 66	True
45 > 33	True
2.9 < 2.7	False
'Abcd' < 'Abcz'	True
'ask' in 'oppvask'	True
0.2 + 0.3 == 0.5	True

ORDBOK

Boolsk verdi. En verdi som er enten True eller False.

```
x = int(input())
```

Betingelse. Uttrykk som evaluerer til en boolsk verdi

```
if x < 0:  
    print('flip it')  
    x = -x  
else:  
    print('leave it')
```

If-setning. Et avsnitt av kildekoden hvor kontrollflyten kan gå én av to (eller flere) veier avhengig av betingelse.

Kodeblokk. Et avsnitt av kildekoden gruppert sammen med samme innrykk. Starter alltid med kolon (:).

Innrykk. Antall mellomrom foran en kodeblokk.

```
print('absolute value of x is ', x)
```

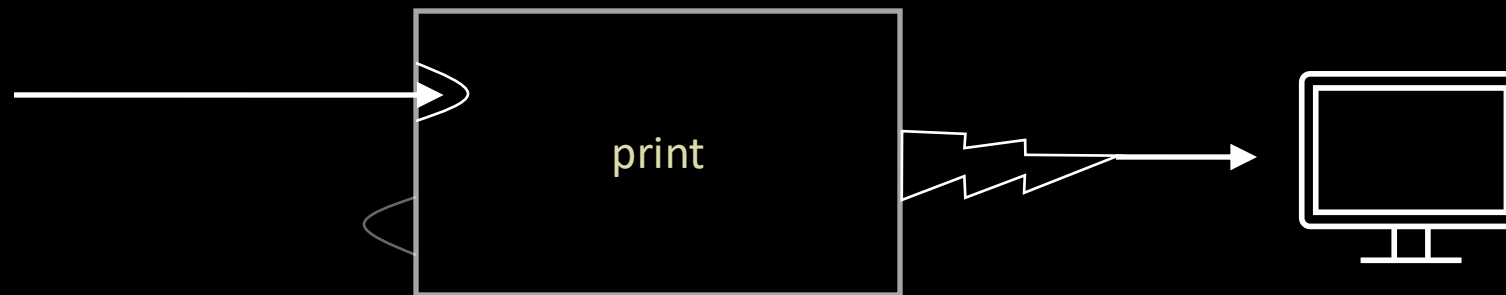
FUNKSJONER



FUNKSJONER MED EFFEKT

```
print('Hello World!')
```

'Hello World!'

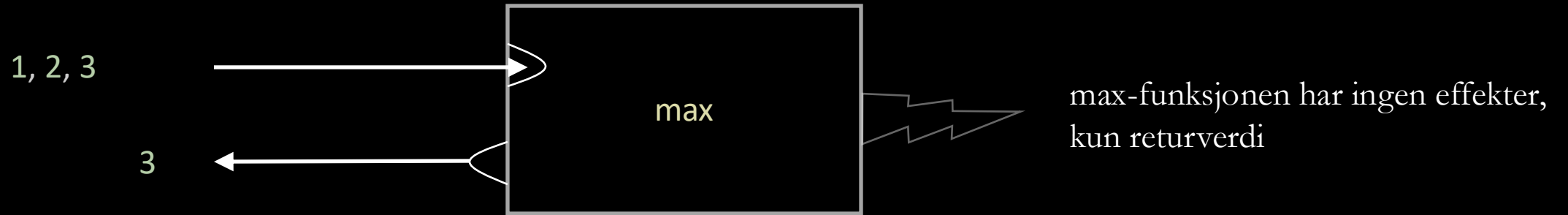


eksempel på en «effekt:»

- noe skjer på skjermen
- noe vises i terminalen

FUNKSJONER MED RETURVERDI

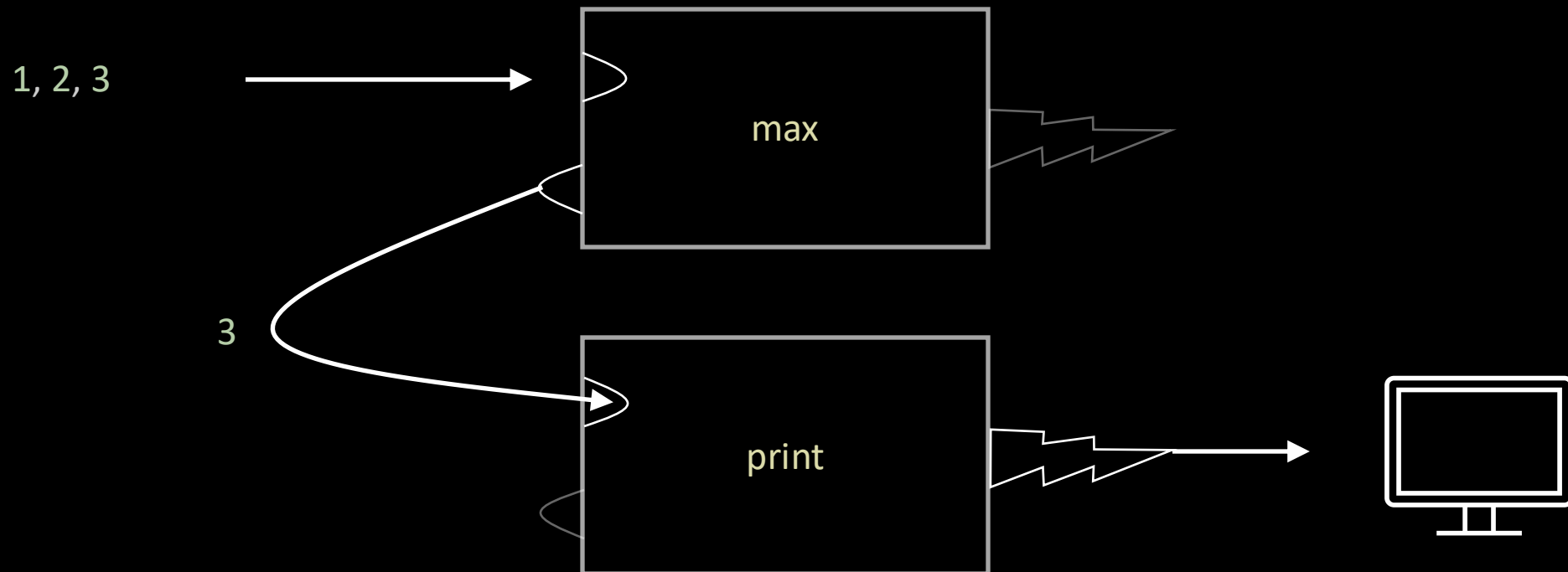
`max(1, 2, 3)`



KOMPONERING AV FUNKSJONSKALL

`print(max(1, 2, 3))`

evaluerer til sin returverdi



INNEBYGDE FUNKSJONER

```
print("Skriv ut noe til terminalen")
```

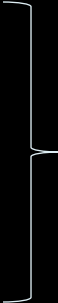
```
x = len("Lengden av en streng")
```

```
x = sum(1, 2, 3)
```

```
x = min(1, 2, 3)
```

```
x = max(1, 2, 3)
```

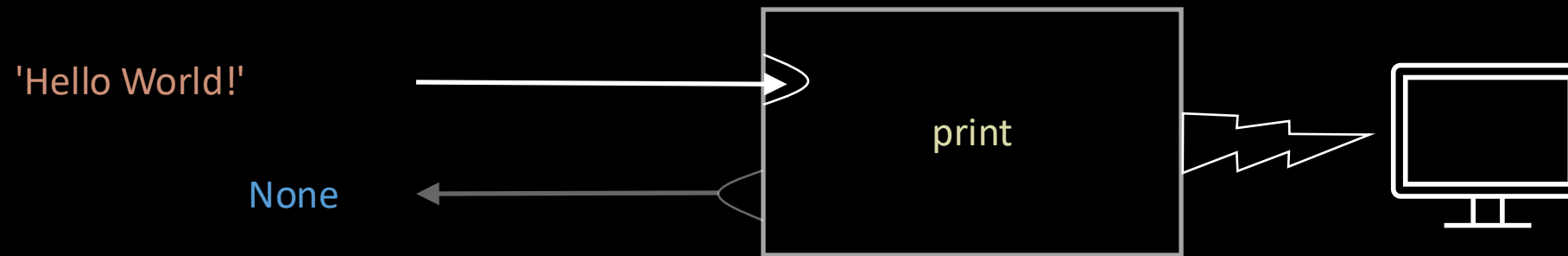
```
x = abs(-3)
```



Har returverdi




ALLE FUNKSJONER HAR RETURVERDI

```
print('Hello World!')
```



...selv om noen funksjoner alltid returnerer verdien `None`

SANT ELER USANT?

- Alle meningsfulle funksjoner må ha input  f. eks. `print()`
- Alle meningsfulle funksjoner har en effekt  f. eks. `max`-funksjonen
- Alle funksjoner har en returverdi  kan være `None`

Å DEFINERE EN FUNKSJON

Mellom parentesene er *parametre*.
Det kan være valgfritt antall.

Vi *definerer* en funksjon som heter `incremented_thrice`

```
def incremented_thrice(x):  
    x = x + 1  
    x += 1  
    return x + 1
```

Kolon

`x` er en *parameter* (en variabel)
som refererer til en verdi bestemt
av den som kaller funksjonen

Kropp. Kode etter kolon
som har *innrykk* blir utført
når funksjonen kalles

Retursetning. Avslutter funksjonen.
Returverdien bestemmes her.

hvis ingen retursetning,
vil `None` returneres

LIVE MED DEBUGGER

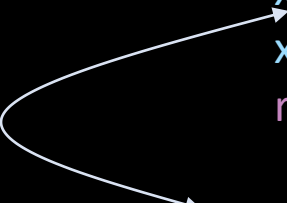
```
def incremented_thrice(x):  
    x = x + 1  
    x += 1  
    return x + 1
```

```
a = 5  
b = incremented_thrice(a)  
c = incremented_thrice(b)  
  
print(f"a: {a}, b: {b}, c: {c}")
```

HVA MED DETTE?

ikke
samme
variabel!

```
def incremented_thrice(x):  
    x = x + 1  
    x += 1  
    return x + 1  
  
x = 5  
y = incremented_thrice(x)  
y = incremented_thrice(y)  
  
print(f"x: {x}, y: {y}")
```



HVA MED DETTE?

```
def incremented_thrice(x):  
    x = x + 1  
    x += 1  
    print(x + 1)
```

```
x = 5  
y = incremented_thrice(x)  
y = incremented_thrice(y)
```

```
print(f"x: {x}, y: {y}")
```

uten retursetning vil None returneres



KRASJ OG FEIL

TRE FORMER FOR FEIL

- Syntaks
 - Programmet krasjer før det begynner å kjøre

```
def volume_of_box(x, y, z)  
    print(x * y + z)
```

```
print("Det er plass til " + volum_of_box(1, 2, 3) + " m3 i boksen
```

line 4

```
print("Det er plass til " + volum_of_box(1, 2, 3) + " m3 i boksen  
      ^
```

SyntaxError: unterminated string literal (detected at line 4)

TRE FORMER FOR FEIL

- Syntaks
 - Programmet krasjer før det begynner å kjøre

```
def volume_of_box(x, y, z)  
    print(x * y + z)
```

```
print("Det er plass til " + volum_of_box(1, 2, 3) + " m3 i boksen")
```

line 1

```
def volume_of_box(x, y, z)  
    ^
```

SyntaxError: expected ':'

TRE FORMER FOR FEIL

- Syntaks
 - Programmet krasjer før det begynner å kjøre

```
def volume_of_box(x, y, z):  
    print(x * y + z)
```

```
print("Det er plass til " + volum_of_box(1, 2, 3) + " m3 i boksen")
```

line 4

```
print("Det er plass til " + volum_of_box(1, 2, 3) + " m3 i boksen")  
    ^
```

SyntaxError: '(' was never closed

TRE FORMER FOR FEIL

- Krasj (engelsk: runtime error)
 - Programmet krasjer når det kjører

```
def volume_of_box(x, y, z):  
    print(x * y + z)
```

```
print("Det er plass til " + volum_of_box(1, 2, 3) + " m3 i boksen")
```

```
line 4, in <module>
```

```
    print("Det er plass til " + volum_of_box(1, 2, 3) + " m3 i boksen")
```

```
NameError: name 'volum_of_box' is not defined. Did you mean: 'volume_of_box'?
```

TRE FORMER FOR FEIL

- Krasj (engelsk: runtime error)
 - Programmet krasjer når det kjører

```
def volume_of_box(x, y, z):  
    print(x * y + z)
```

```
print("Det er plass til " + volume_of_box(1, 2, 3) + " m3 i boksen")
```

```
line 4, in <module>
```

```
    print("Det er plass til " + volume_of_box(1, 2, 3) + " m3 i boksen")
```

```
TypeError: can only concatenate str (not "NoneType") to str
```

TRE FORMER FOR FEIL

- Krasj (engelsk: runtime error)
 - Programmet krasjer når det kjører

```
def volume_of_box(x, y, z):  
    return x * y + z
```

```
print("Det er plass til " + volume_of_box(1, 2, 3) + " m3 i boksen")
```

```
line 4, in <module>
```

```
    print("Det er plass til " + volume_of_box(1, 2, 3) + " m3 i boksen")
```

```
TypeError: can only concatenate str (not "int") to str
```

TRE FORMER FOR FEIL

- Logisk feil
 - Programmet gir feil svar

```
def volume_of_box(x, y, z):  
    return x * y + z
```

```
print("Det er plass til " + str(volume_of_box(1, 2, 3)) + " m3 i boksen")
```

```
Det er plass til 5 m3 i boksen
```

TRE FORMER FOR FEIL

- Syntaks

- Programmet krasjer før det begynner å kjøre
- Feilmelding gir visuell indikasjon på hva som er feil

IndentationError
SyntaxError

- Krasj

- Programmet krasjer underveis i kjøring

AttributeError
IndexError
KeyError
NameError
TypeError
ZeroDivisionError

- Logiske feil

- Programmet gir galt svar

...

ASSERT

- Krasj programmet med vilje når noe ikke er som det skal
- Tester koden, og beskytter mot logiske feil

```
assert True # Gjør ingenting  
assert False # Krasjer
```

- Vi bruker prinsippet om assert når vi retter kode automatisk (CodeGrade)
- Det er mulig å slå av assert for å optimisere kjøretid (men: ikke gjør det)
- Sjekk at assert er aktivt: legg inn `assert False` og se at det krasjer

ASSERT

- Krasj programmet med vilje når noe ikke er som det skal
- Tester koden, og beskytter mot logiske feil

```
def volume_of_box(x, y, z):  
    return (x * y + z)
```

```
assert 6 == volume_of_box(1, 2, 3)  
print("Det er plass til " + str(volume_of_box(1, 2, 3)) + " m3 i boksen")
```

```
line 4, in <module>  
    assert 6 == volume_of_box(1, 2, 3)  
AssertionError
```