KODESPORING, BETINGELSER og FUNKSJONER

INF100

HØST 2026

Torstein Strømme

LAB

- Du kan levere så mange ganger du vil.
- Du kan se resultatene av auto-testene umiddelbart.

- Det er mange gruppeledere klare til å hjelpe!

- 5 poeng på lab1 ved å delta i din gruppe denne uken.

DROP-IN GRUPPETIMER

	Man	Tirs	Ons	Tors	Fre
8-10					kickstart
10-12					
12-14					
14-16	forelesning				

Bruk læringsfellesskapet!

PRESENTASJON

- Innen utgangen av onsdag: få tidspunkt fra din gruppeleder på mitt.uib
- Hvis ikke: send melding til din gruppeleder!

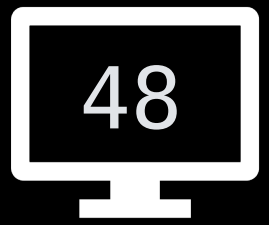
LITEN RECAP:

TYPER

```
int       42

float     1.5

str       "……"

dict      { …: …, …: … }

list      [ …, …, … ]

bool      True eller False

NoneType     None
```

typen bestemmer
hva operator betyr

```
x = 42
y = 6
total = x + y
print(total)
```




```
x = "42"
y = "6"
total = x + y
print(total)
```

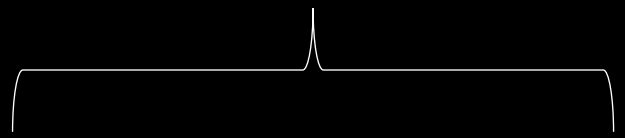
https://inf100.ii.uib.no/lab/1/?problem=addisjonskalkulator

LITEN RECAP:

FAGORD

ORDBOK

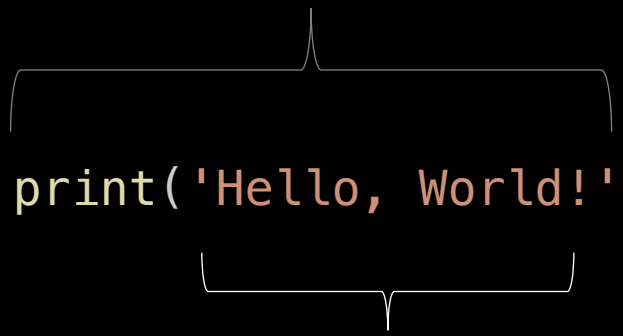
setning (engelsk: statement). Ett «steg» i et program, ofte én linje.



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

ORDBOK

setning (engelsk: statement). Ett «steg» i et program, ofte én linje.



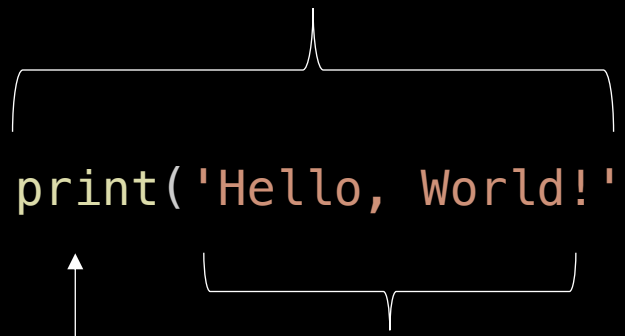
objekt. En eller annen form for verdi som benyttes i programmet.

Eksempler på objekter:

```
'Hello, World!'   '42'
 42     3.14     True
      [4, 3, 2, 3]
```

ORDBOK

setning (engelsk: statement). Ett «steg» i et program, ofte én linje.

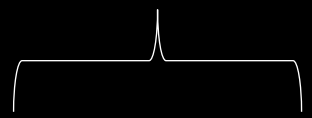


objekt. En eller annen form for verdi som benyttes i programmet.

funksjon. En «kommando» som kan få noe til å skje.

ORDBOK

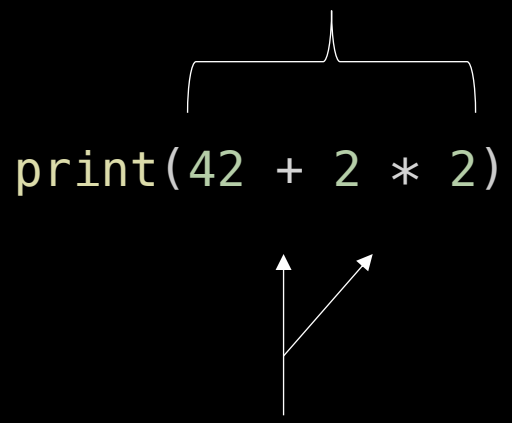
uttrykk (engelsk: expression). Et regnestykke som evaluerer til en verdi (et objekt).



```
print(42 + 2 * 2)
```

ORDBOK

uttrykk (engelsk: expression). Et regnestykke som evaluerer til en verdi (et objekt).



operasjon. En måte å kombinere to objekter for å produsere et nytt objekt.

Eksempler på operasjoner:

```
+  -  *  /  **  //  %
```

KODESPORING, BETINGELSER og FUNKSJONER

INF100

HØST 2026

Torstein Strømme

KODESPORING

- Koden utføres linje for linje
- Forutsi hva som skjer i neste steg

KODESPORING: EKSEMPEL

```
balance = 1000
org_balance = balance

# Første år: 5% rente
interest_rate = 0.05
interest_amount = balance * interest_rate
balance = balance + interest_amount

# Andre år: 10% rente
interest_rate = 0.10
interest_amount = balance * interest_rate
balance = balance + interest_amount

difference = balance - org_balance
print(f'Etter to år har du tjent {difference} kr i renter')
```

KODESPORING

```
balance = 1000
org_balance = balance

# Førs
intere
intere
balanc

# Andr
intere
intere
balanc

differ
print(
```

linje	balance	org_balance	interest_rate	interest_amount	difference	print
1	~~1000~~					
2		1000				
5			~~0.05~~			
6				~~50.0~~		
7	~~1050.0~~					
10			0.10			
11				105.0		
12	1155.0					
15					155.0	
16						Etter to år har du tjent 155.0 kr i renter

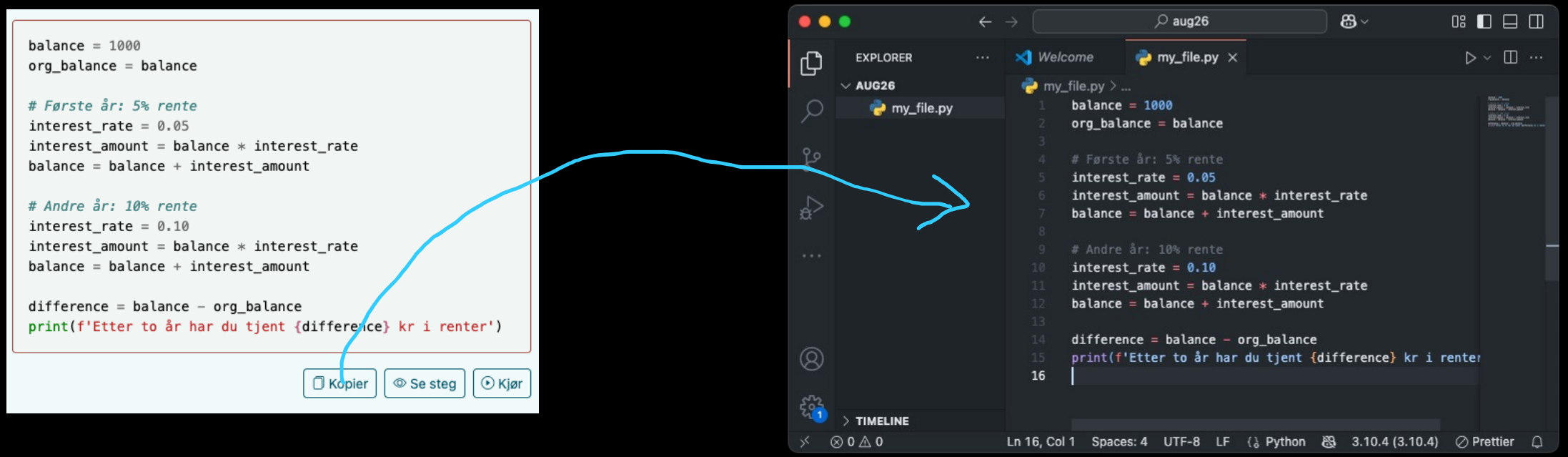
EKSAMENSOPPGAVE HØST 24

1(b)

```
a = 1
b = 2
a = a + b
b = a * b
a -= 1
print(b - a)
```

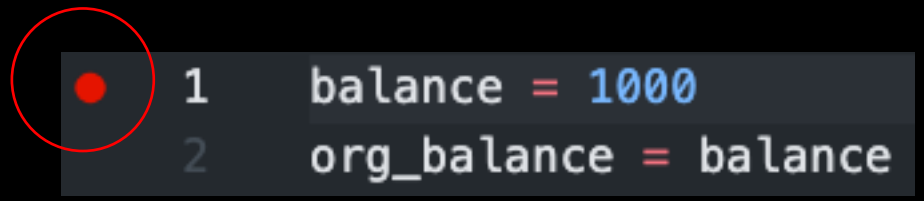
Kva skriv dette programmet ut? (hvis programmet krasjar, skriv berre Error)

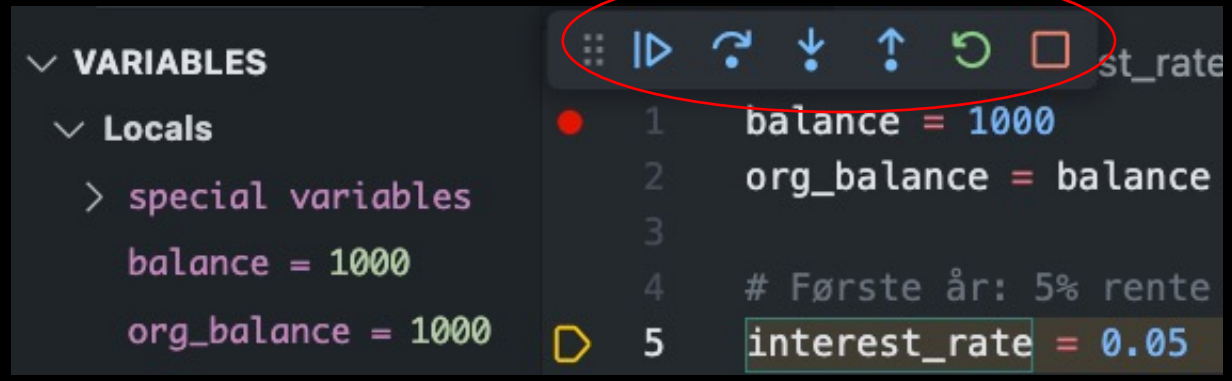
https://inf100.ii.uib.no/bin/saving/



KODESPORING MED DEBUGGER

```python
balance = 1000
org_balance = balance

# Første år: 5% rente
interest_rate = 0.05
interest_amount = balance * interest_rate
balance = balance + interest_amount

# Andre år: 10% rente
interest_rate = 0.10
interest_amount = balance * interest_rate
balance = balance + interest_amount

difference = balance - org_balance
print(f'Etter to år har du tjent
{difference} kr i renter')
```





LOTTERI

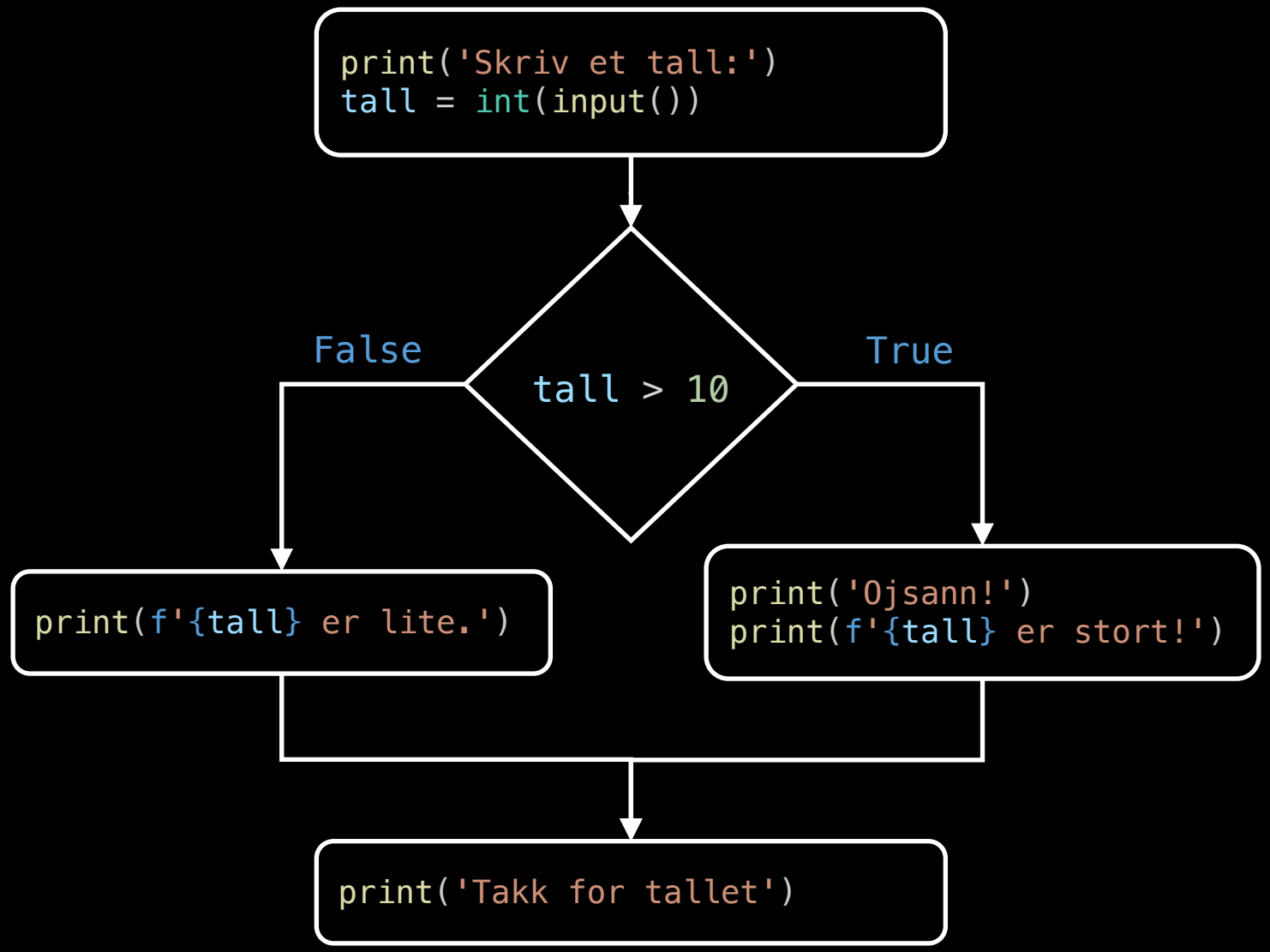
```python
from random import randrange

dice_roll = randrange(6) + 1
print(dice_roll)
```

Evaluerer til et «tilfeldig» heltall mindre enn 6

BETINGELSER

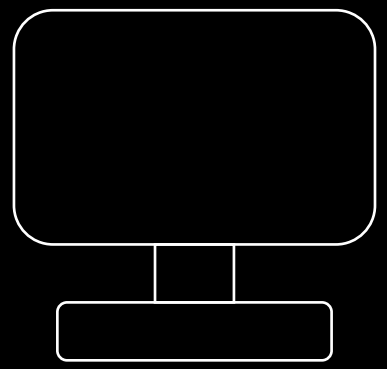
BETINGELSER

```python
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')
```



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')
```

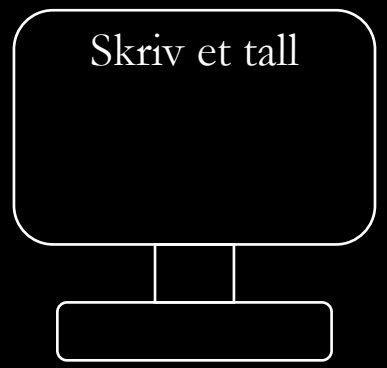


BETINGELSER

```
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')
```

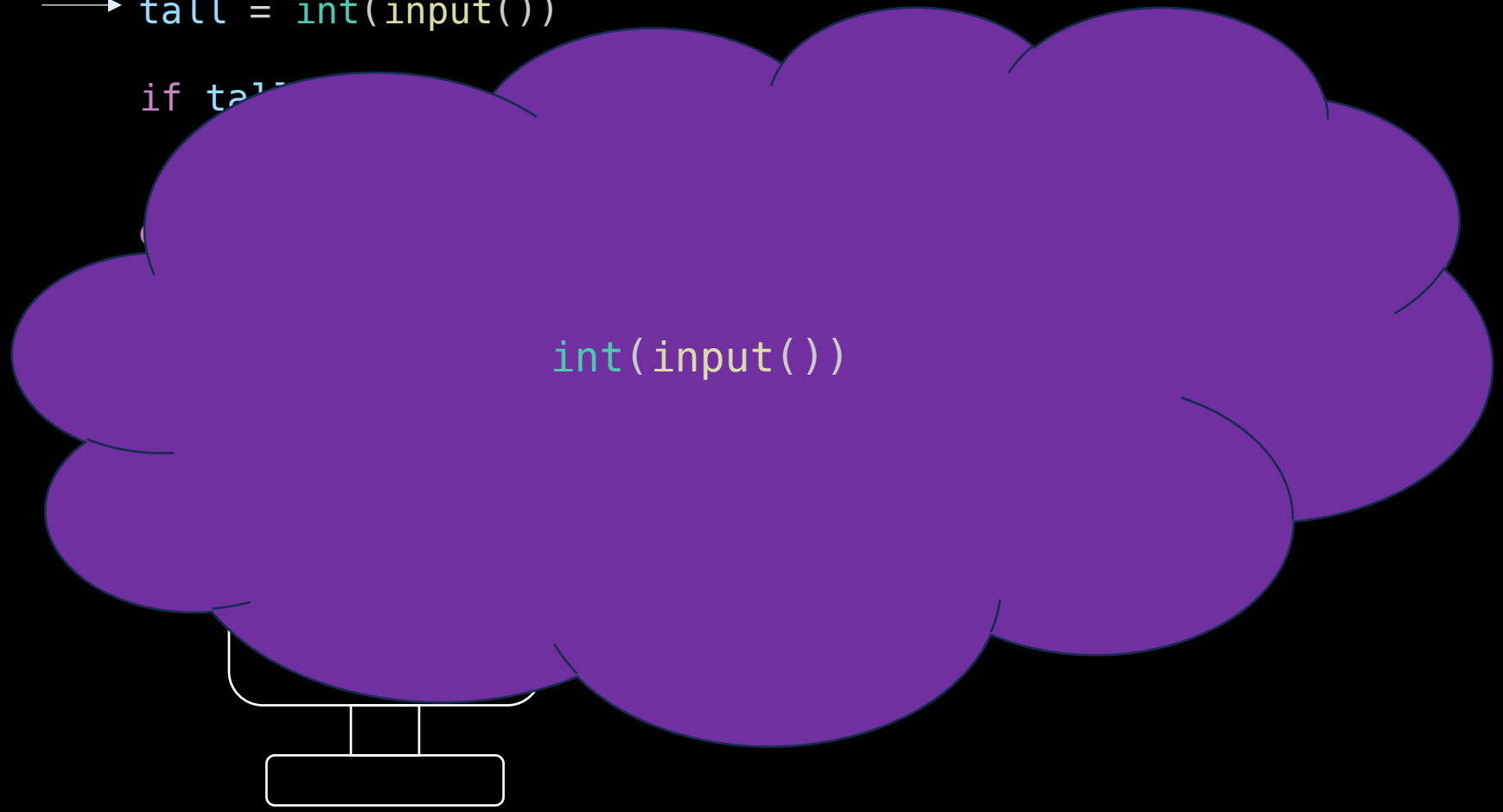
BETINGELSER

```
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')
```



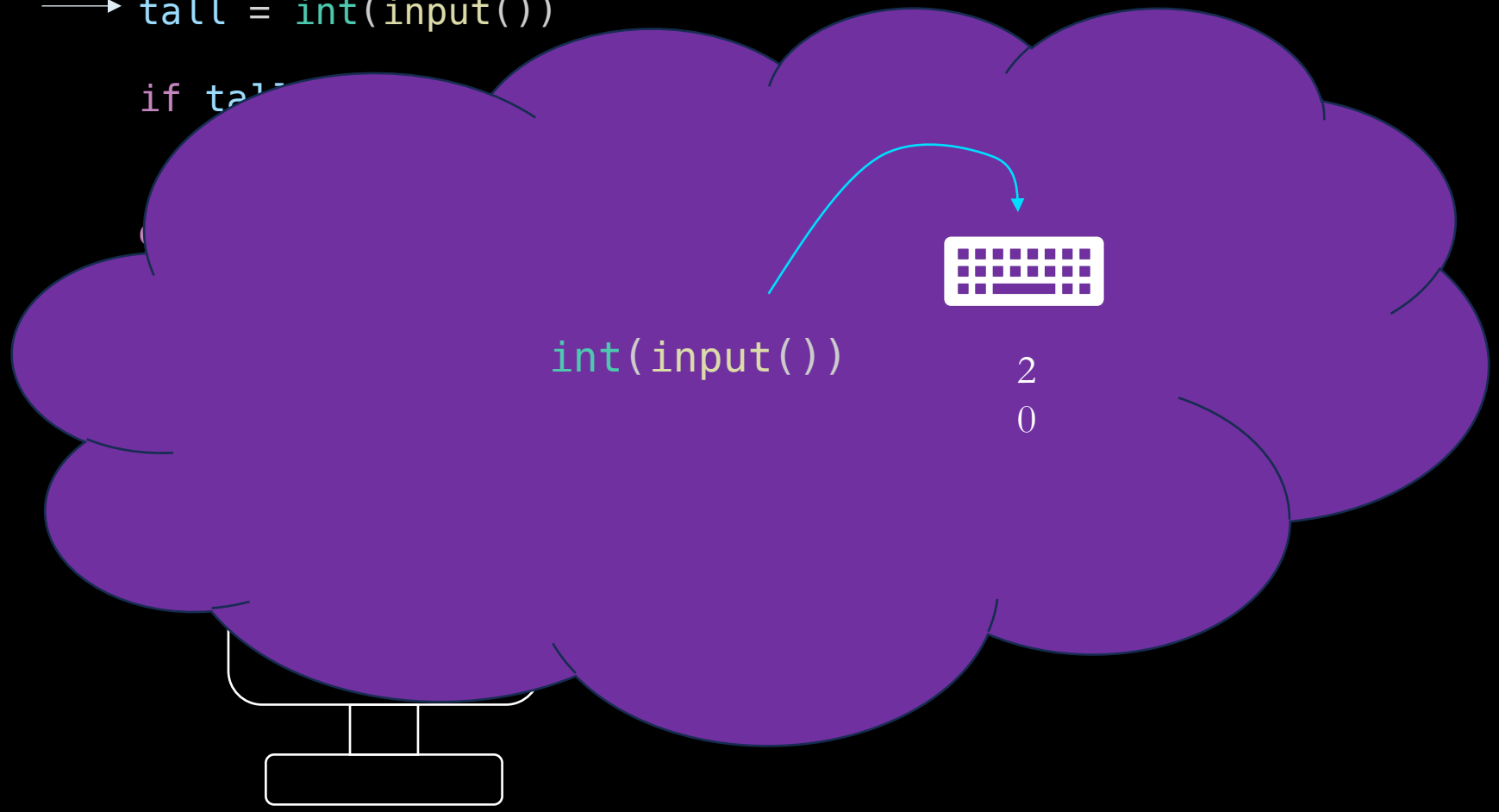
BETINGELSER

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

if tal
```



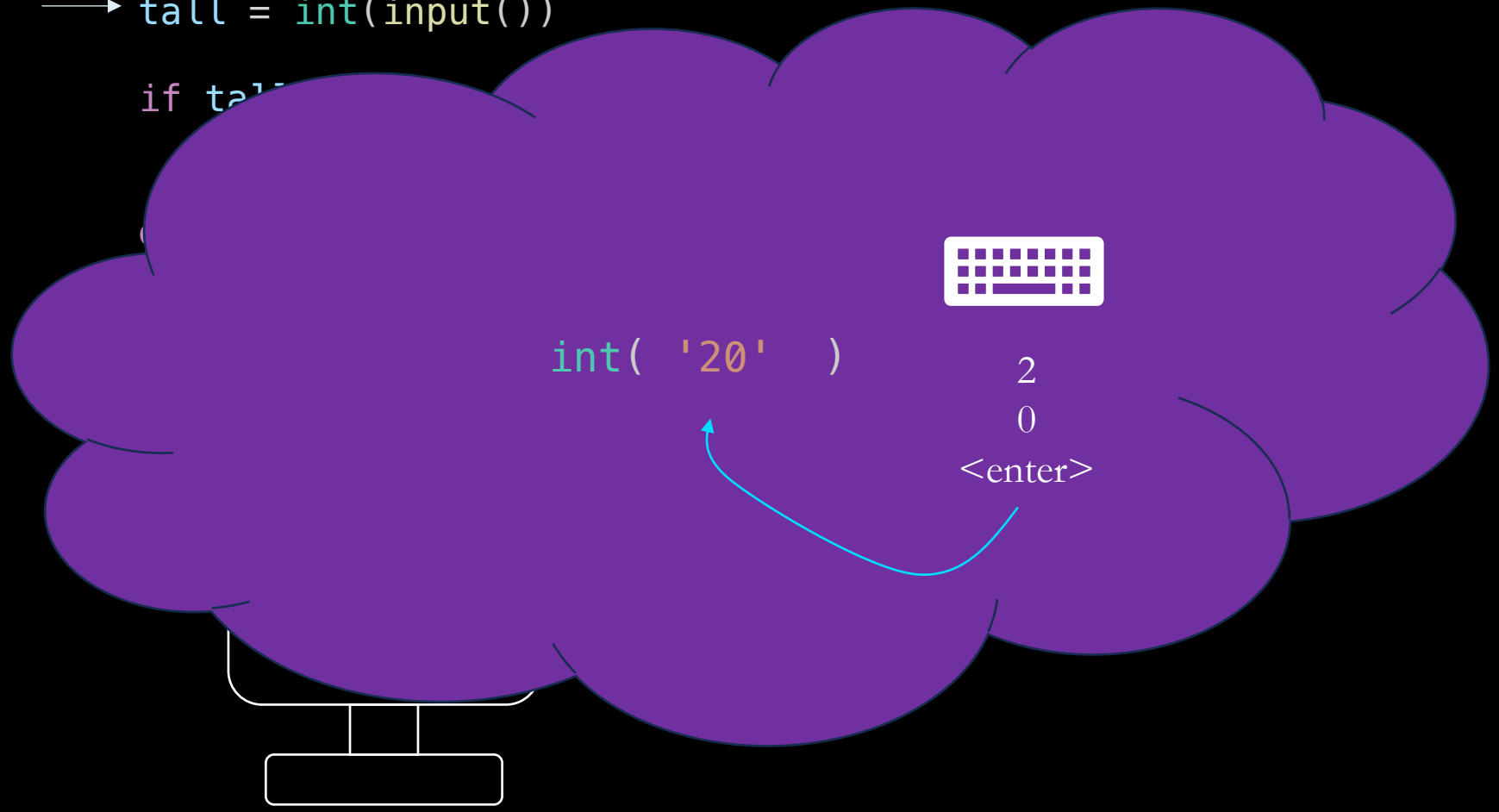
BETINGELSER

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

if tal
```



BETINGELSER

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

if tal
```

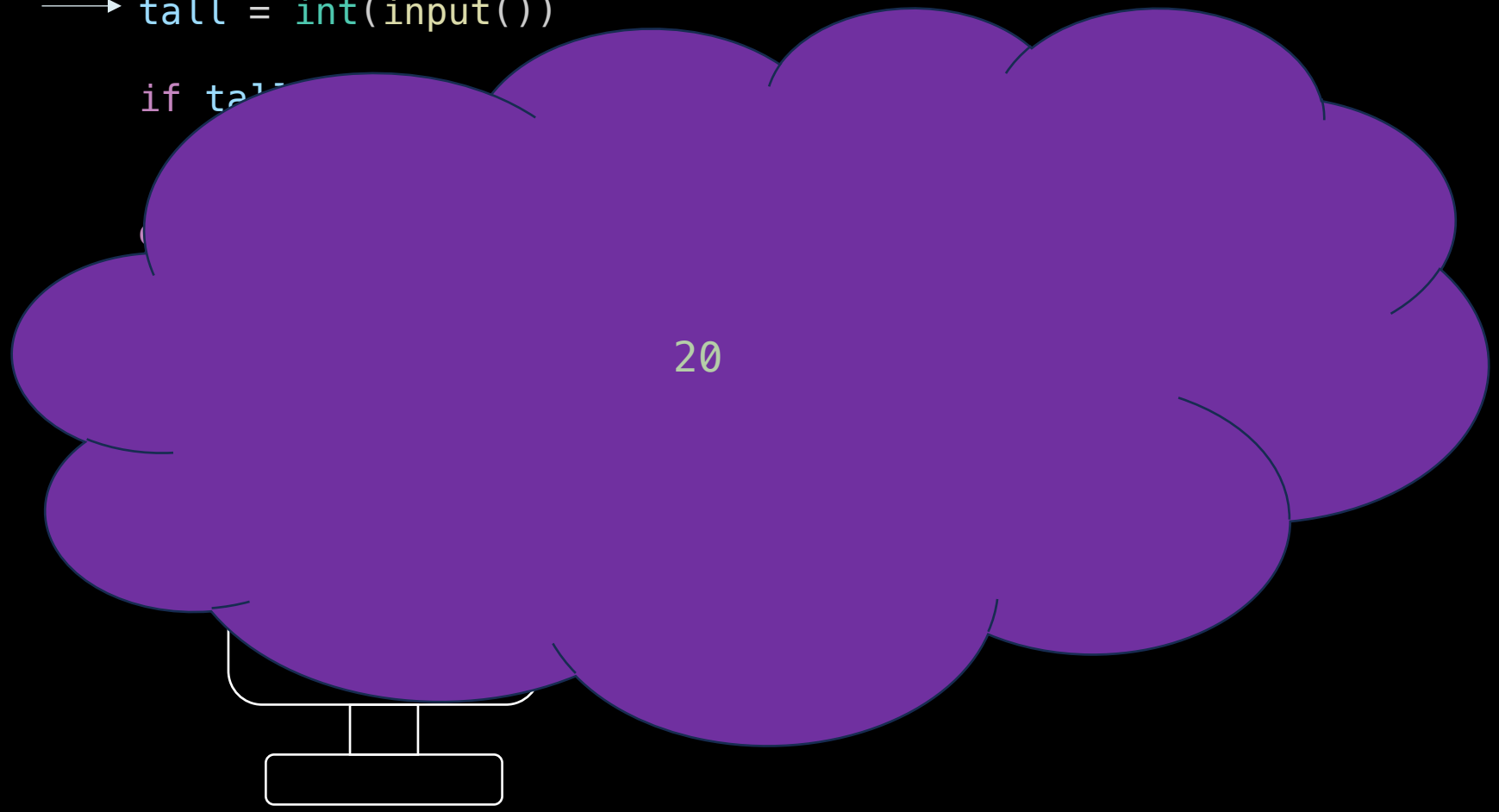


BETINGELSER

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

if tal
```

BETINGELSER

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

if tal
```



BETINGELSER

```
print('Skriv et tall:')
tall = 20

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

print('Takk for tallet')
```

VARIABLER OBJEKTER

tall → 20

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')
```

VARIABLER OBJEKTER

BETINGELSER

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

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

print('Takk for tal
```

tall > 10

VARIABLER OBJEKTER

tall → 20

BETINGELSER

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

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

print('Takk for tal
```

20 > 10

VARIABLER OBJEKTER

tall → 20

BETINGELSER

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

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

print('Takk for tal
```

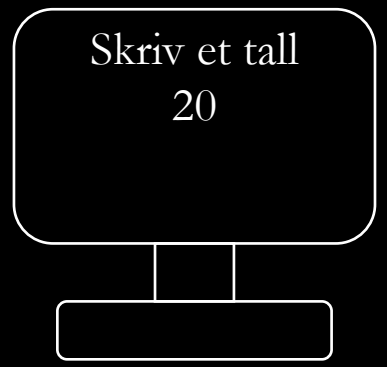
True

VARIABLER OBJEKTER

tall → 20

BETINGELSER

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

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

print('Takk for tallet')
```



VARIABLER OBJEKTER

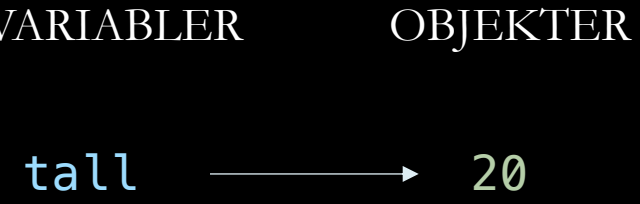
BETINGELSER

```
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')
```

VARIABLER OBJEKTER

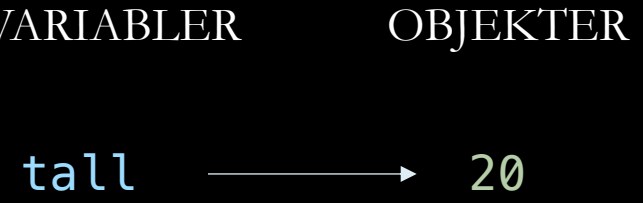
BETINGELSER

```
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')
```



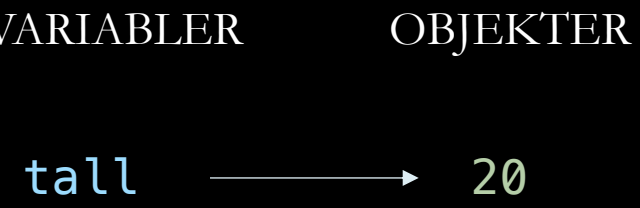
BETINGELSER

```
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')
```



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')
```



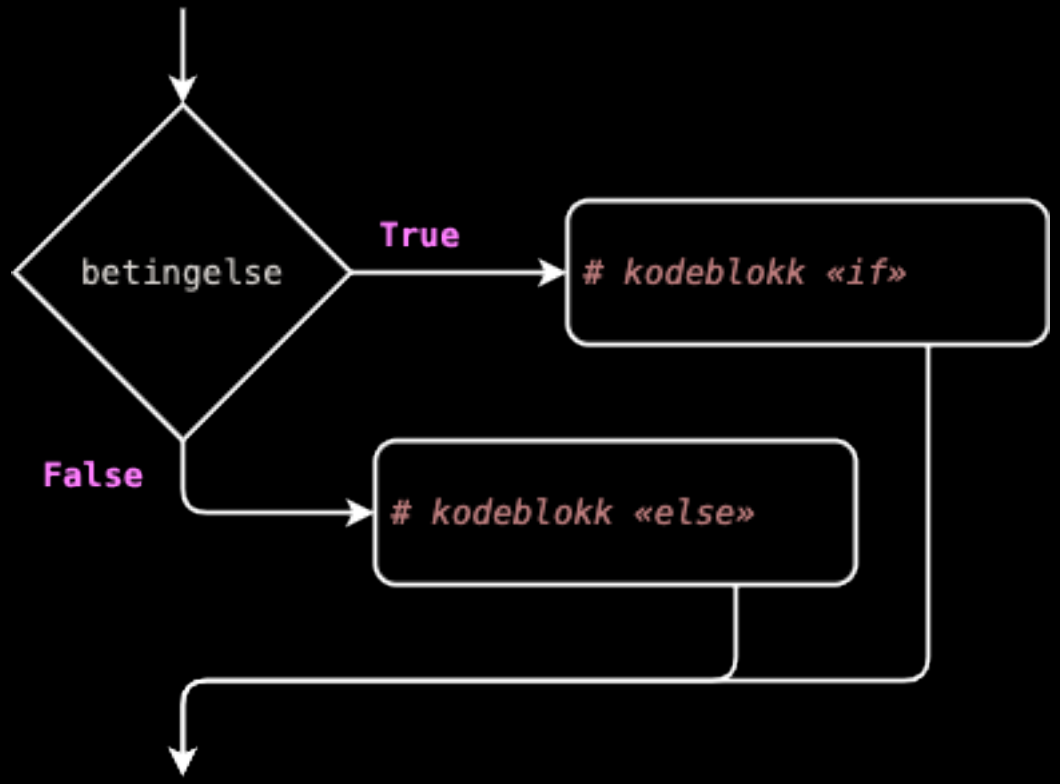
BETINGELSER

```
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')
```

VARIABLER OBJEKTER

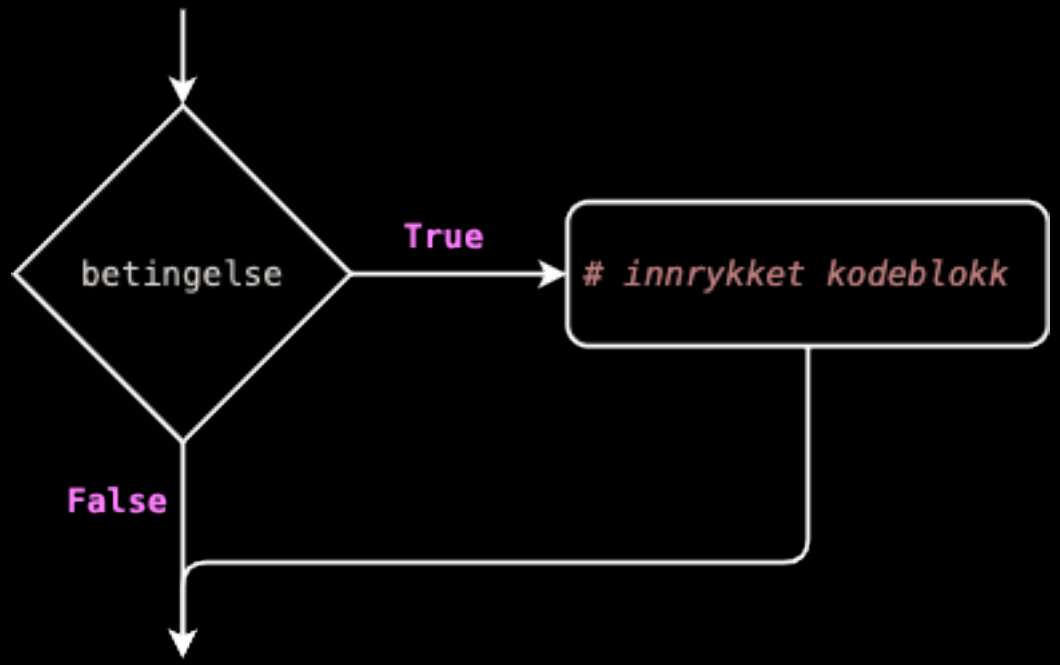
IF-ELSE

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

# ...
```



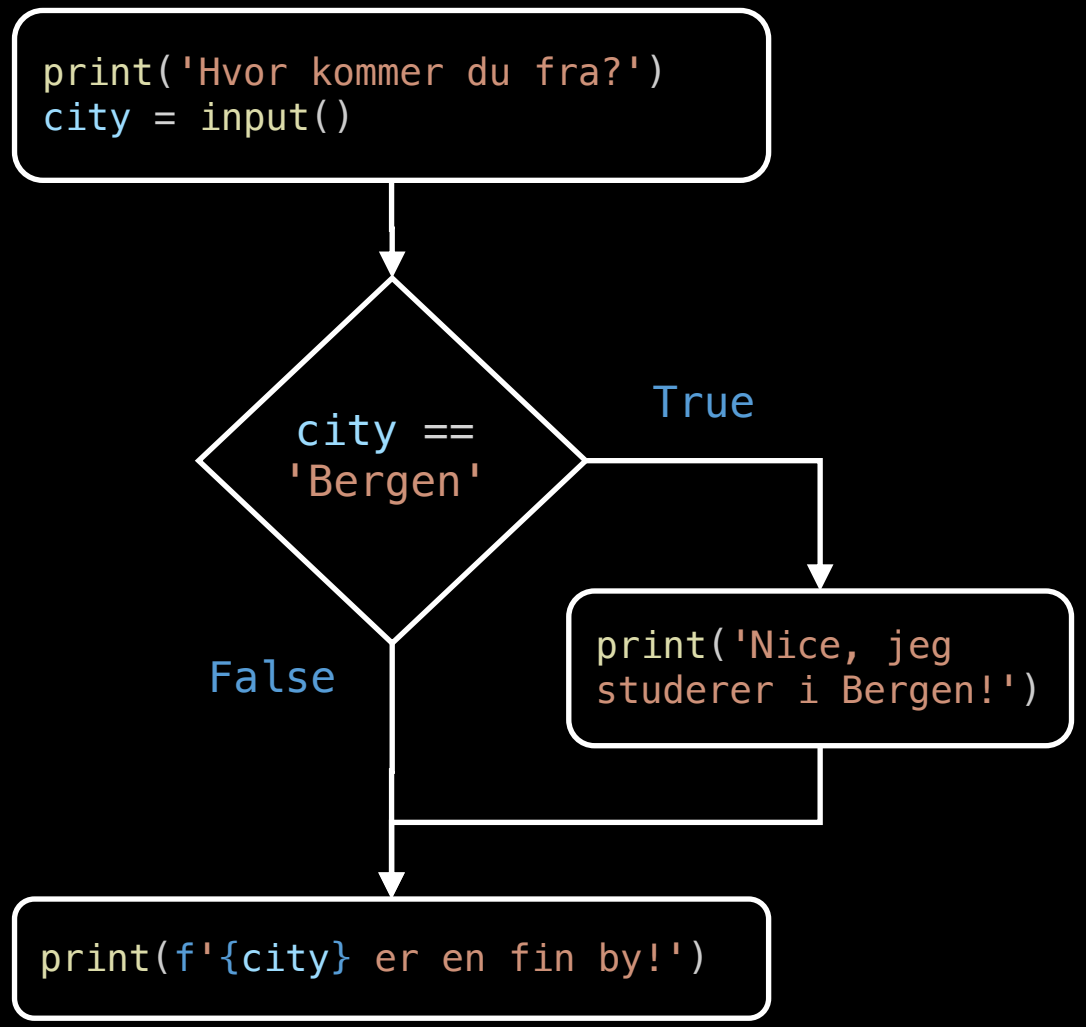
IF

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

# ...
```



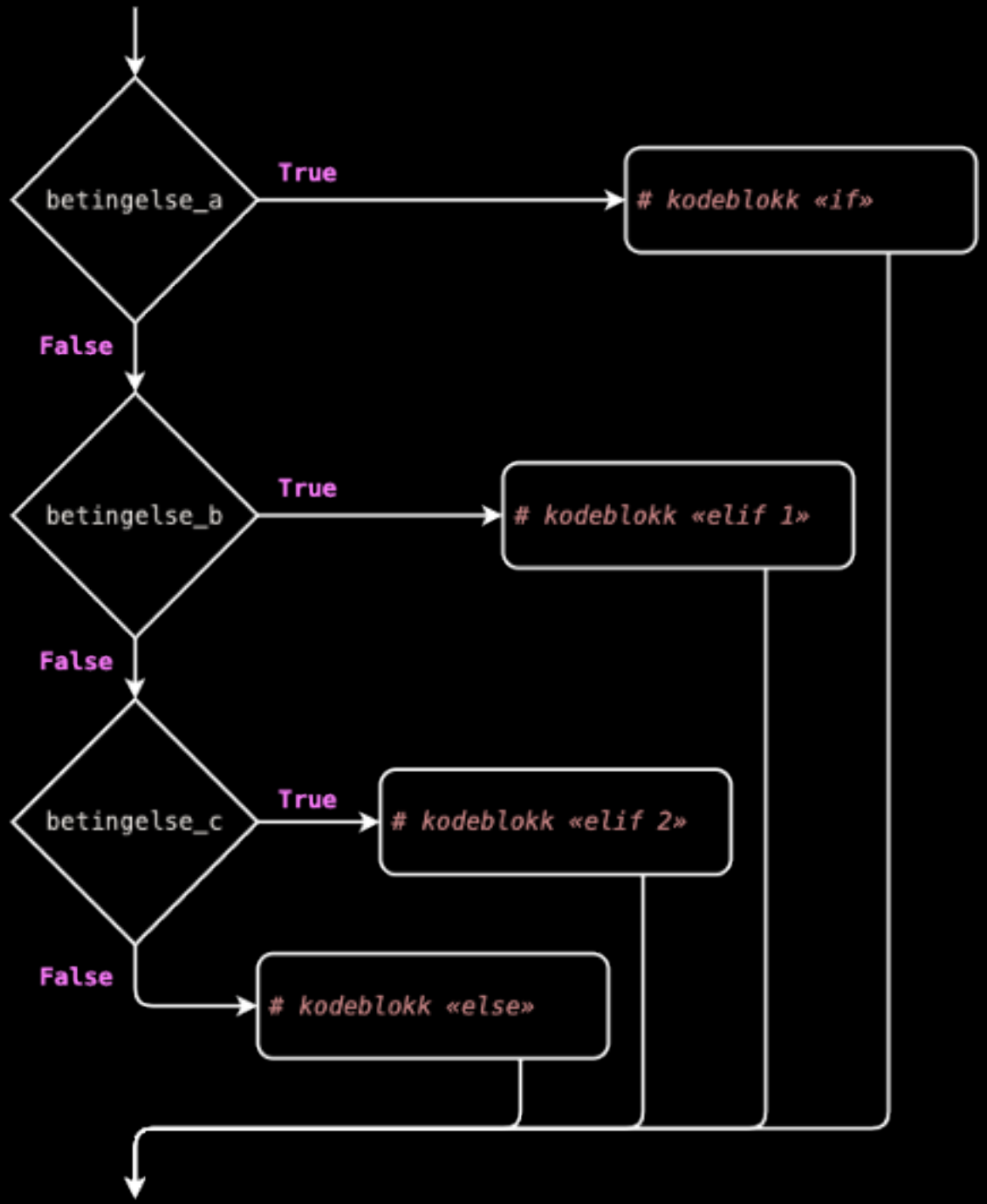
IF

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

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

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



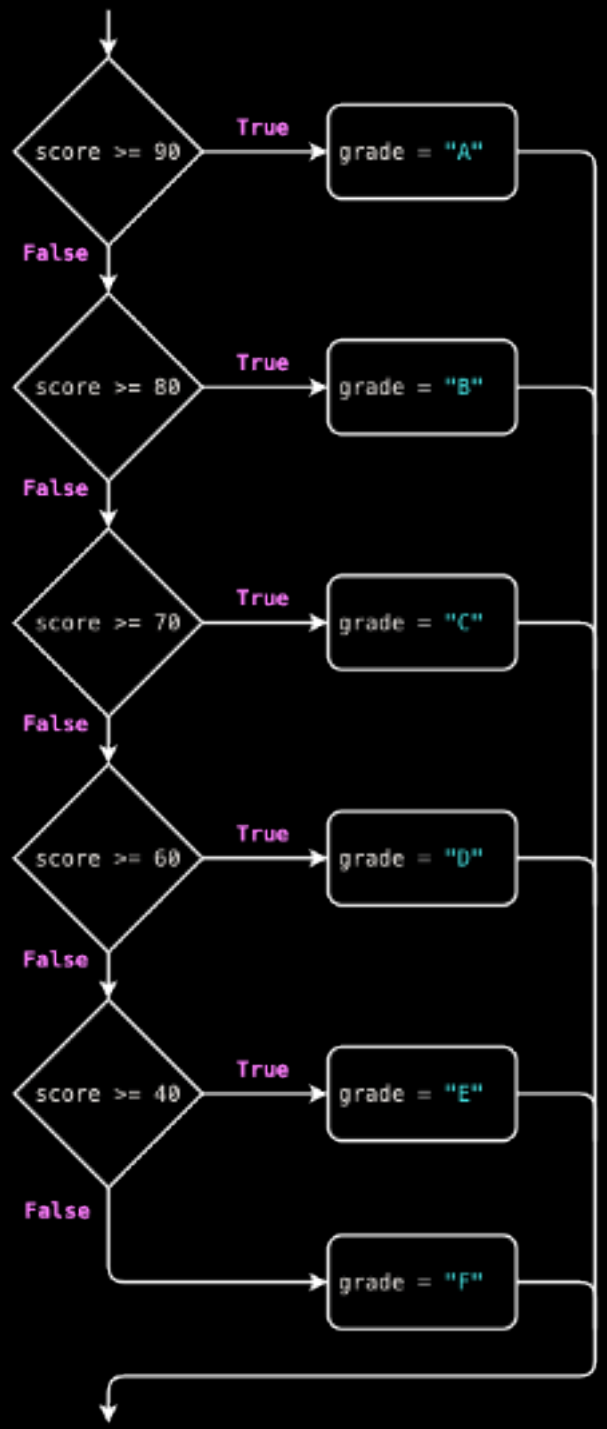
IF-ELIF-ELSE

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



IF-ELIF-ELSE

```python
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

BOOLSKE UTTRYKK

Uttrykk	Evaluerer til
`True`	`True`
`False`	`False`
`4 == 5`	`False`
`66 == 66`	`True`

BOOLSKE UTTRYKK

Uttrykk	Evaluerer til
`True`	`True`
`False`	`False`
`4 == 5`	`False`
`66 == 66`	`True`
`45 > 33`	`True`
`2.9 < 2.7`	`False`

> < >= <= == in

BOOLSKE UTTRYKK

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

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

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)
```

ORDBOK

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

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

Betingelse. Uttrykk som evaluerer til en boolsk verdi

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

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

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

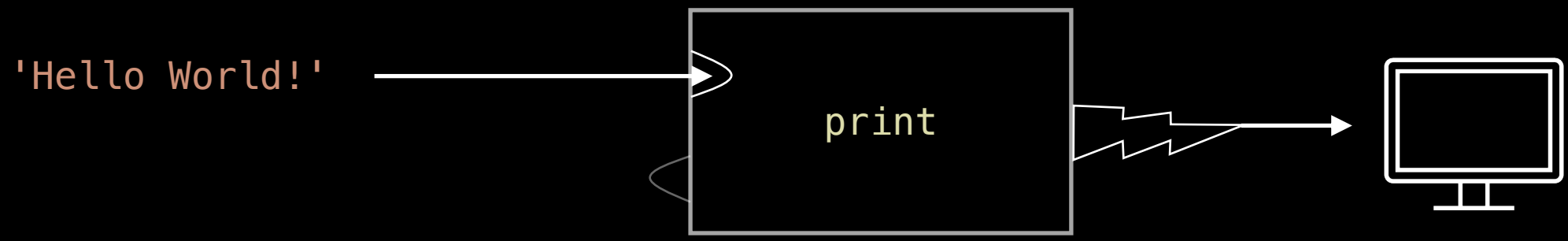
Innrykk. Antall mellomrom foran en kodeblokk.

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

FUNKSJONER

FUNKSJONER MED EFFEKT

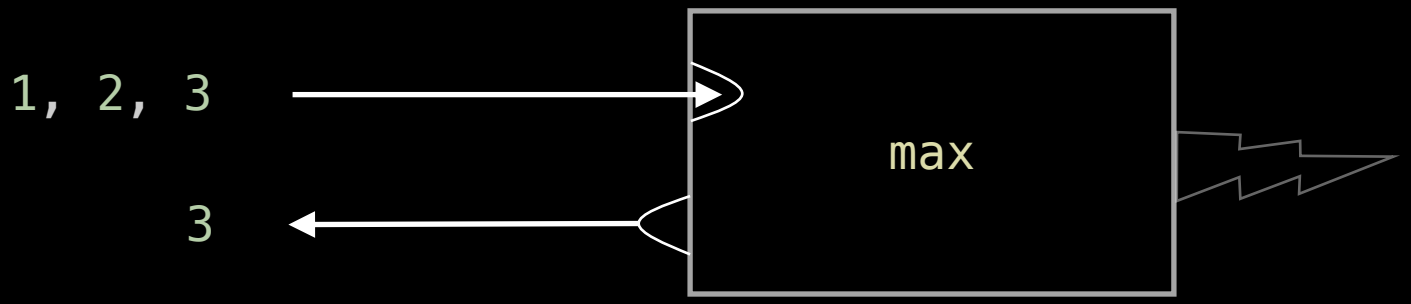
```
print('Hello World!')
```



eksempel på en «effekt:»
- noe skjer på skjermen
- noe vises i terminalen
- noe blir lagret i en fil

FUNKSJONER MED RETURVERDI

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

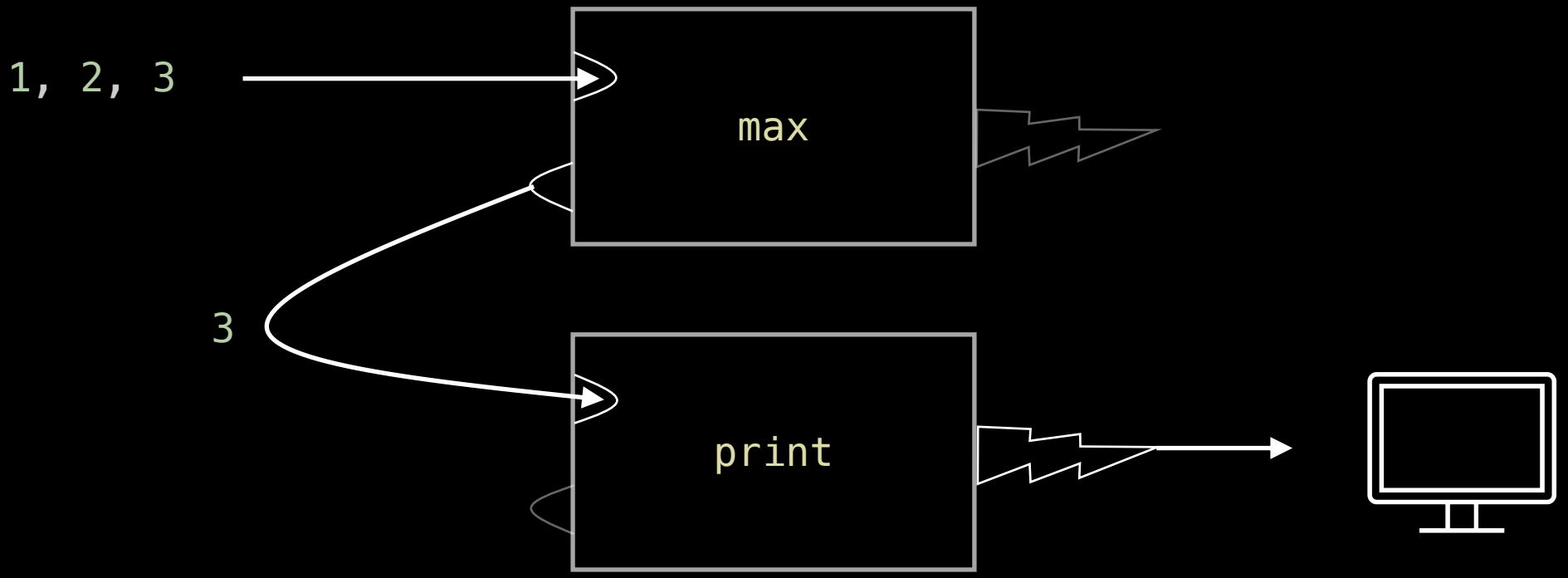


max-funksjonen har ingen effekter, kun returverdi

KOMPONERING AV FUNKSJONSKALL

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

evaluerer til sin returverdi



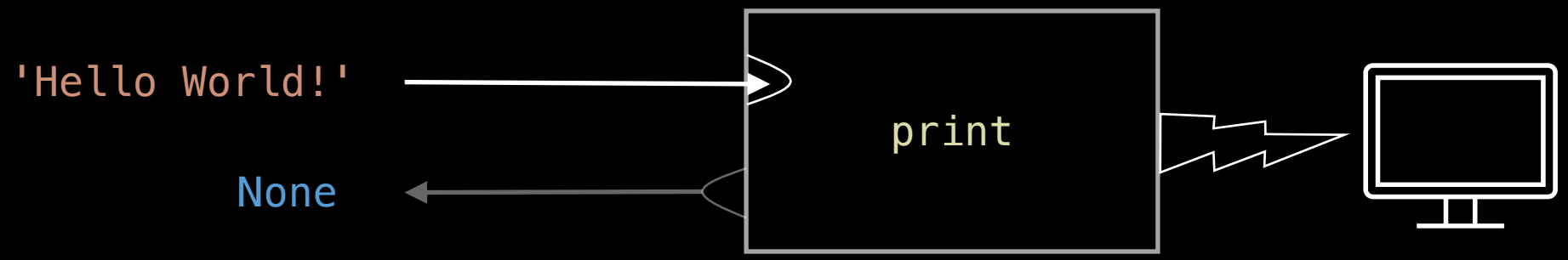
INNEBYGDE FUNKSJONER

```
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

Vi *definerer* en funksjon som heter incremented_thrice

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

```
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

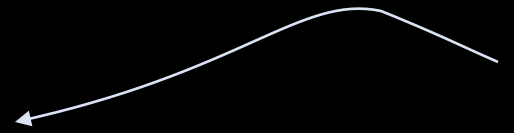
```
def incremented_thrice(x):
    x = x + 1
    x += 1
    return x + 1

a = 5
b = incremented_thrice(a)
c = incremented_thrice(b)

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

HVA MED DETTE?


```python
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
```