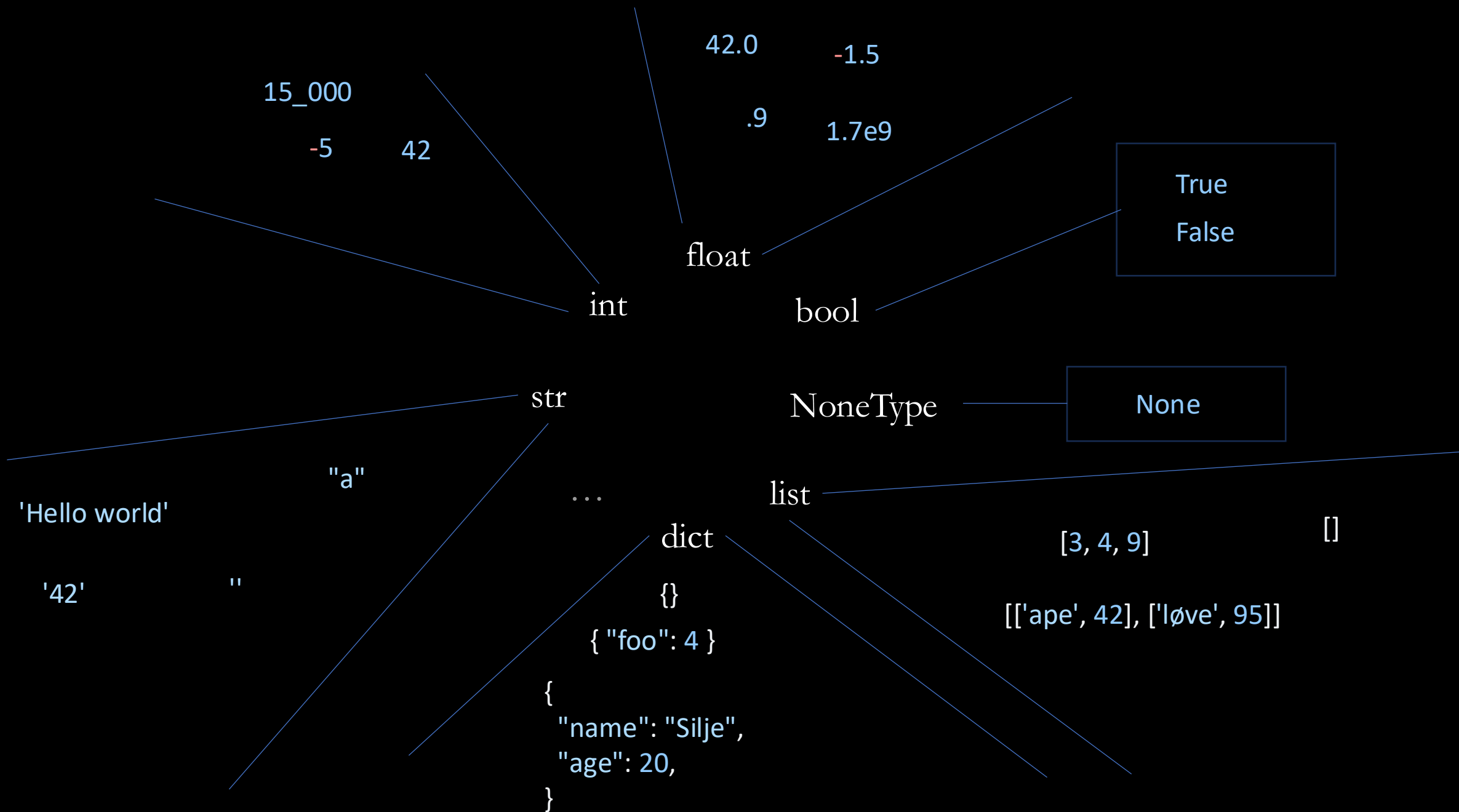




Gå til
www.menti.com

Skriv inn koden
3347 7691



eller bruker QR kode

LØKKER

INF100

HØST 2025

Crystal Chang Din

Med bidrag fra: Torstein Strømme

2 260 000

1 627 000

994 000

746 000

565 000

452 000



Joker

I denne oppgaven skal vi skrive en kunstig intelligens for å spille [Joker](#) fra Norsk Tipping. I dette spillet får man på forhånd vite fem grunntall; for hvert av grunntallene må man velge om man skal gå «opp» eller «ned» før et hemmelig vippetall avsløres. Dersom man valgte «opp» og det avslørte vippetallet er høyere eller lik grunntallet, vinner man en større premie. Det samme skjer dersom man velger «ned» og vippetallet er lavere eller lik grunntallet. Vi antar at de hemmelige vippetallene er tilfeldig valgt mellom 0 og 9.

Strategien vi skal implementere er å si «opp» dersom grunntallet er 4 eller lavere, og si «ned» ellers.

I filen *joker.py*, skriv et program som leser inn 5 tall fra en fil [joker.json](#) som inneholder en liste med 5 heltall, på formen:

```
{  
  "grunntall": [ 3, 4, 5, 6, 1 ]  
}
```

Disse representerer grunntallene vi får oppgitt når vi begynner å spille Joker. Deretter skal programmet skrive ut enten «opp» eller «ned», for hvert av de fem grunntallene. En kjøring av programmet kan se slik ut for filen over:

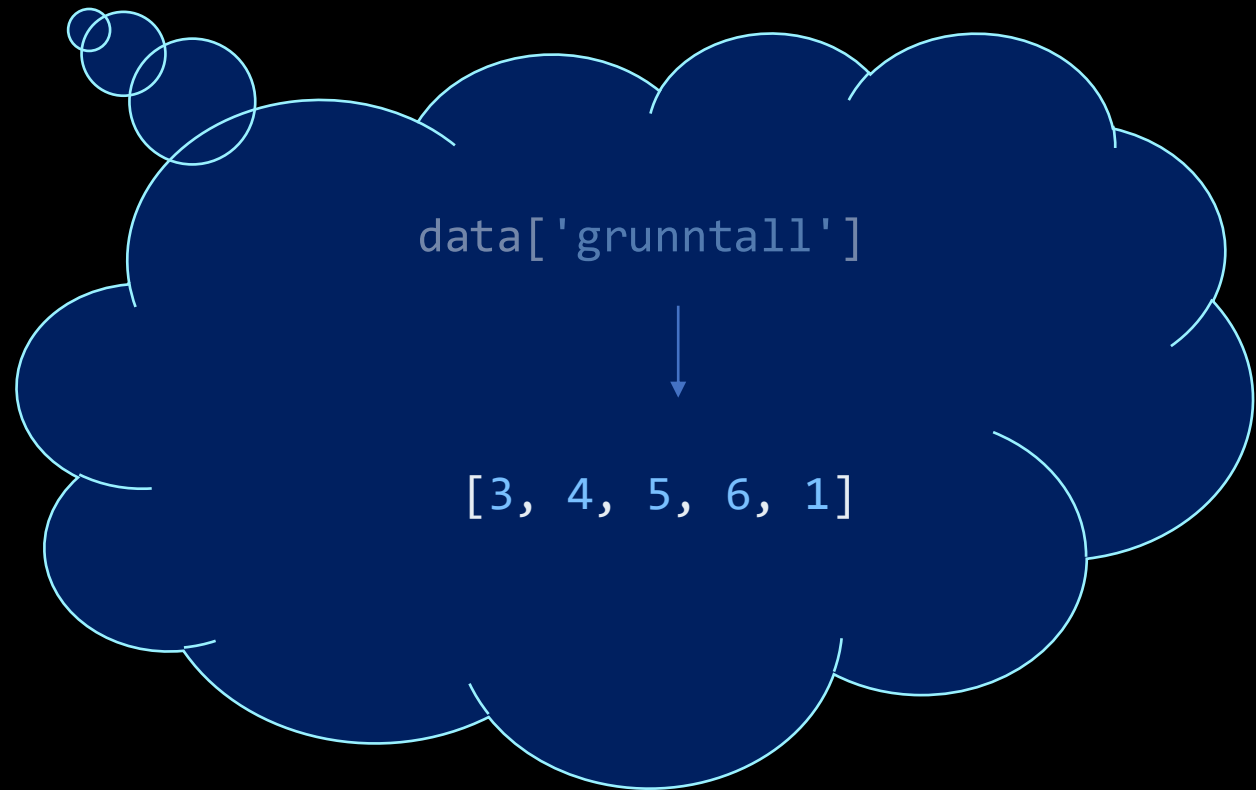
```
opp  
opp  
ned  
ned  
opp
```

```
for num in data['grunntall1']:
    if num <= 4:
        print('opp')
    else:
        print('ned')
```



data['grunntall1']

```
for num in data['grunntall1']:
    if num <= 4:
        print('opp')
    else:
        print('ned')
```



```
for num in data['grunntall']:  
    if num <= 4:  
        print('opp')  
    else:  
        print('ned')
```



```
for num in [3, 4, 5, 6, 1]:  
    if num <= 4:  
        print('opp')  
    else:  
        print('ned')
```


iterand. variabel som viser til et nytt element i den iterable for hver iterasjon av løkken

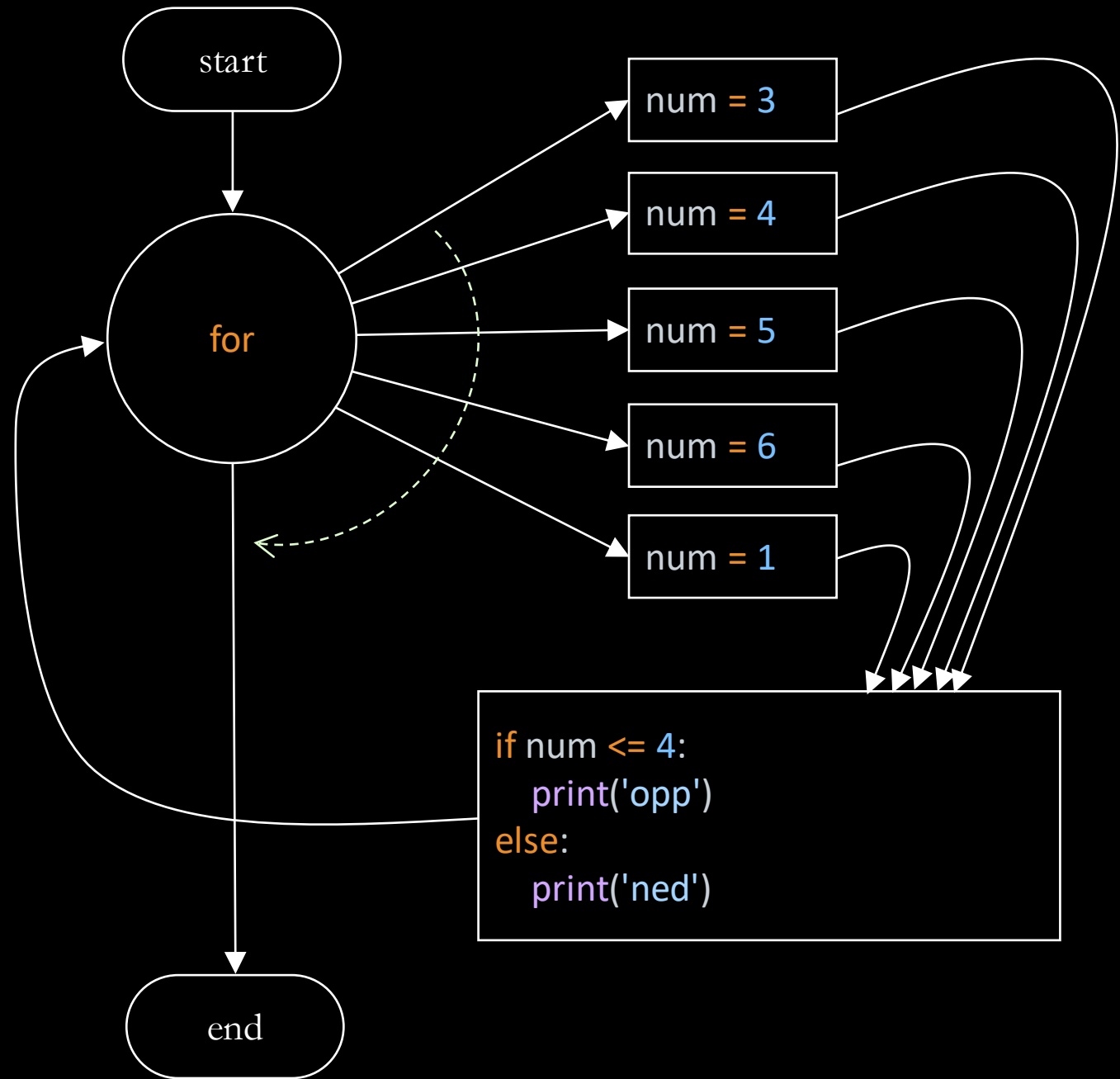
iterabel. en samling av elementer (f. eks. en liste) som skal itereres over

```
for num in [3, 4, 5, 6, 1]:  
    if num <= 4:  
        print('opp')  
    else:  
        print('ned')
```

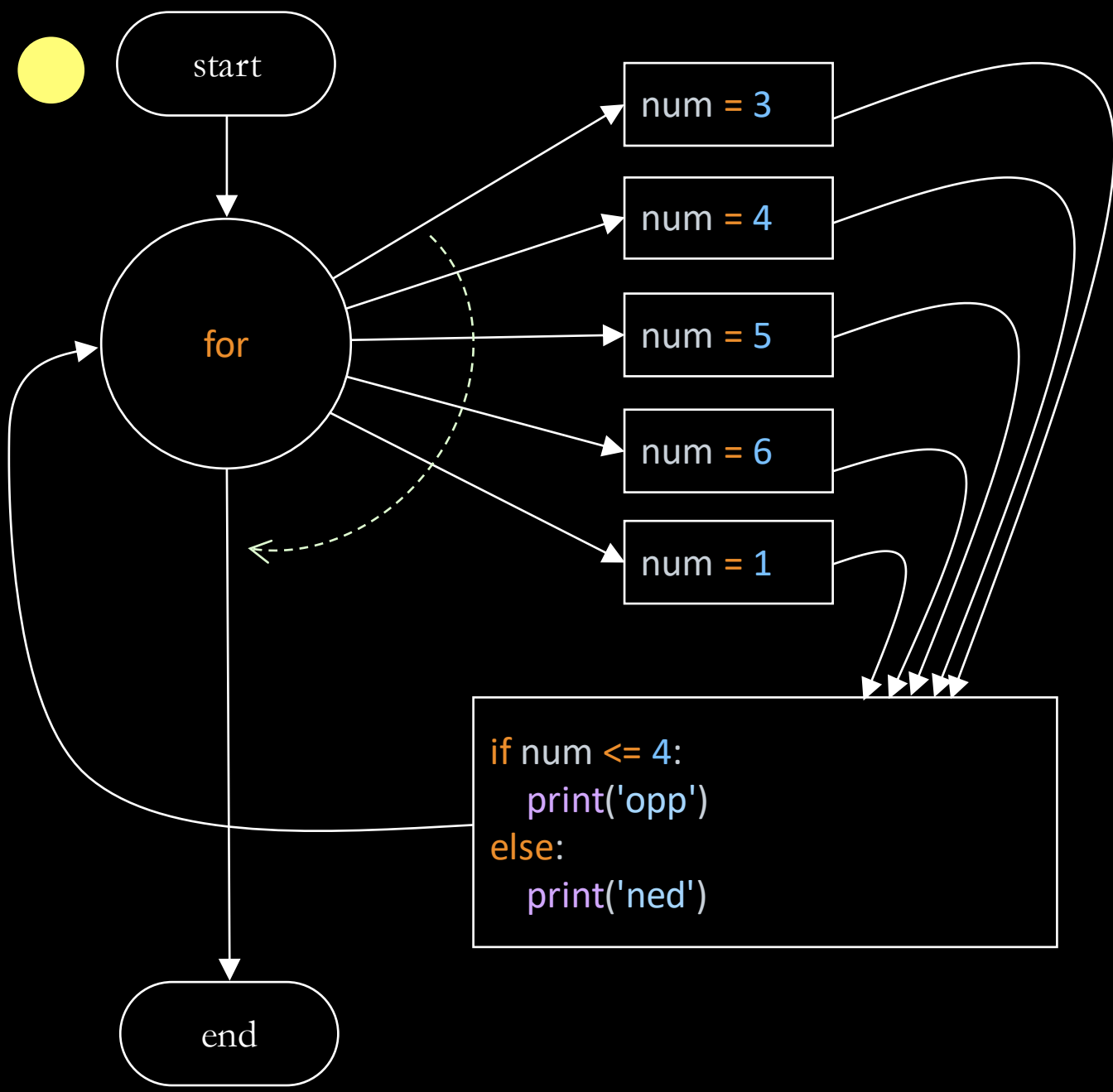
for-løkke. en setning med en løkke kropp som utføres én gang for hvert element i en iterabel.

løkke kropp. innrykket blokk av setninger som utføres én gang for hvert element i iterabelen.

```
for num in [3, 4, 5, 6, 1]:  
    if num <= 4:  
        print('opp')  
    else:  
        print('ned')
```



```
for num in [3, 4, 5, 6, 1]:  
    if num <= 4:  
        print('opp')  
    else:  
        print('ned')
```

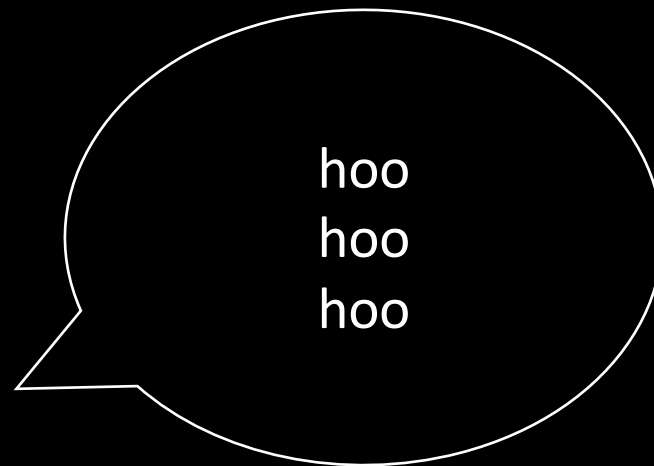


HVA PRINTER DETTE PROGRAMMET?

```
print('start')  
for x in [0, 1, 2, 3]:  
    print('loop', end=' ')  
    print(x)  
print('end')
```

HVA PRINTER DETTE PROGRAMMET?

```
x = 99
print(x)
for x in [3, 3, 2]:
    print('loop', end=' ')
    print(x)
print('huzzah')
print(x)
```



```
print('hoo')  
print('hoo')  
print('hoo')
```

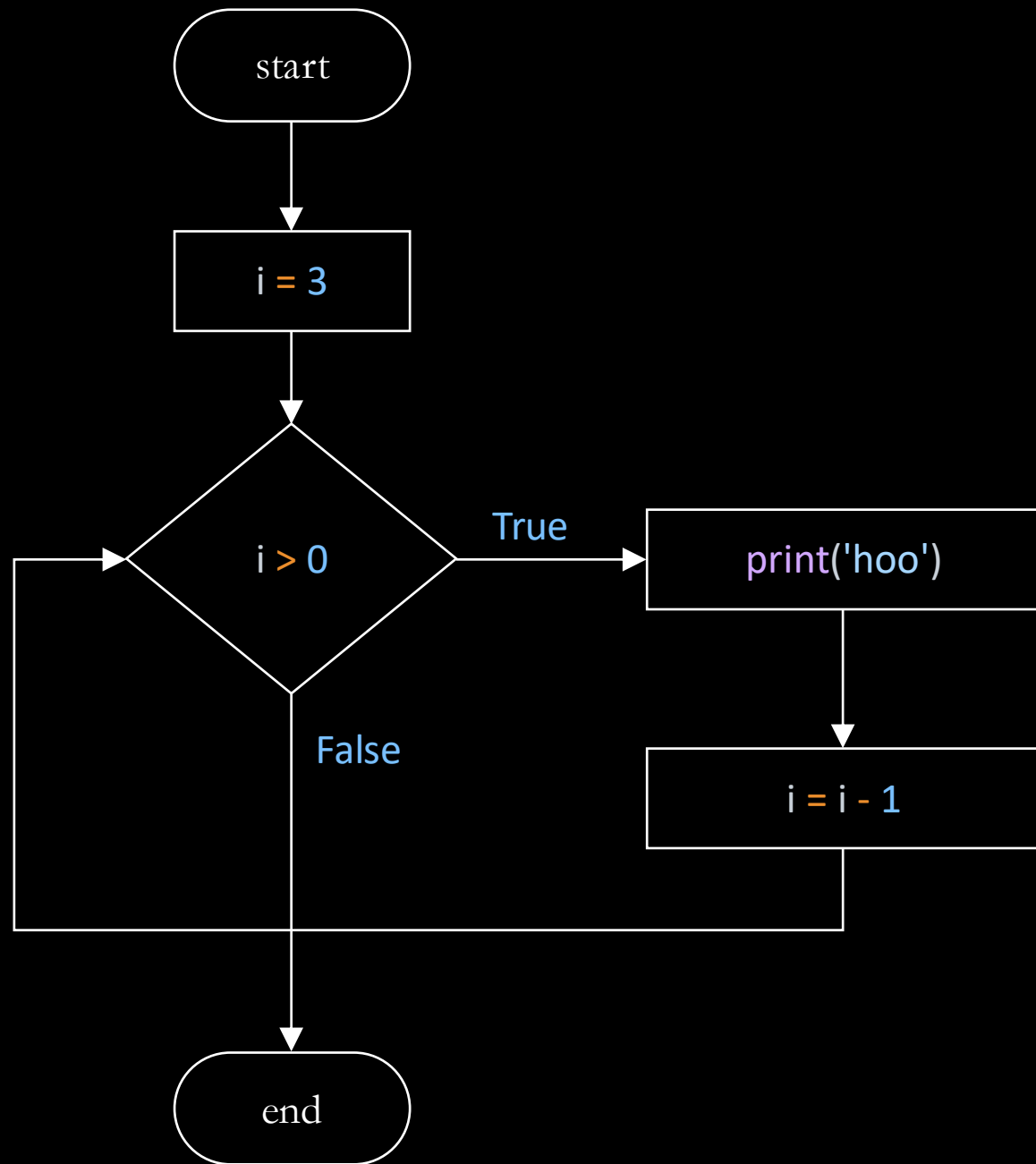
betingelse. et uttrykk som evaluerer til en boolsk verdi (True eller False)

```
i = 3
while i > 0:
    print('hoo')
    i = i - 1
```

} while-setning

løkke kropp. innrykket blokk av setninger som utføres så lenge betingelsen er True.


```
i = 3
while i > 0:
    print('hoo')
    i = i - 1
```

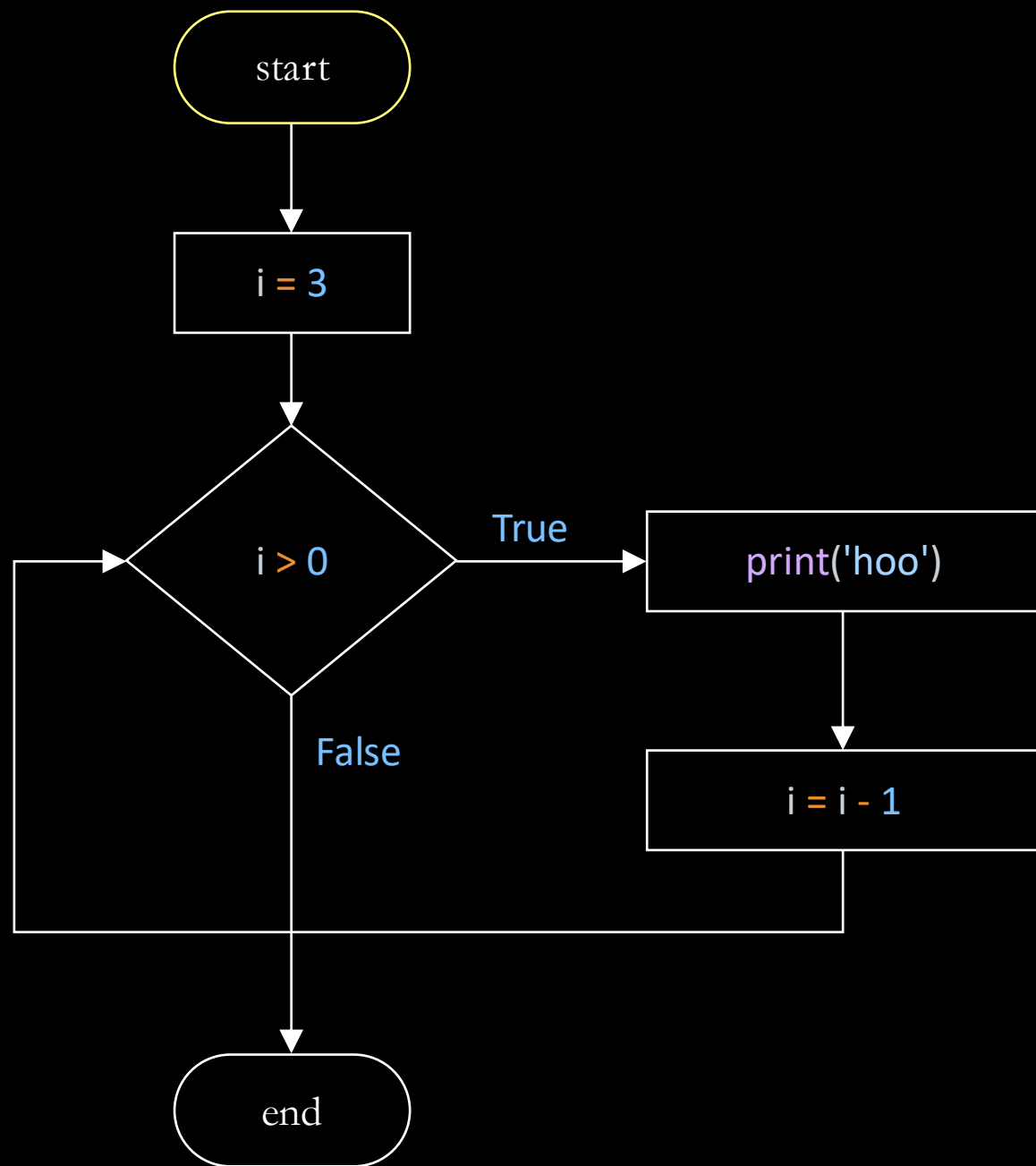
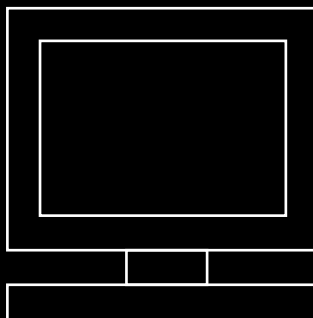




```
i = 3  
while i > 0:  
    print('hoo')  
    i = i - 1
```

Variabler

Objekter

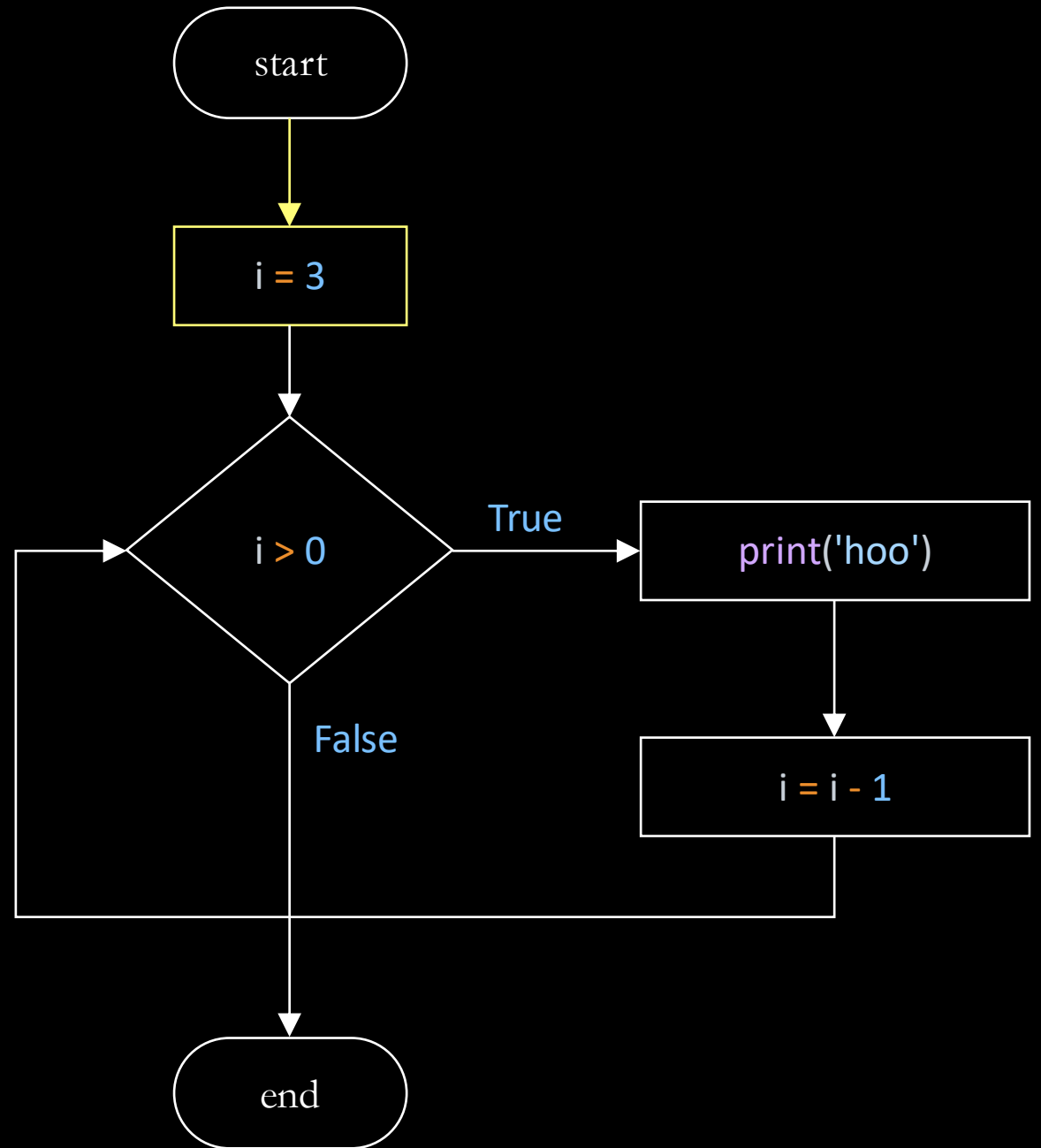
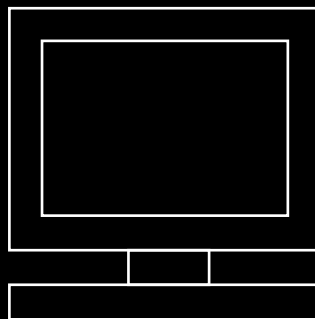




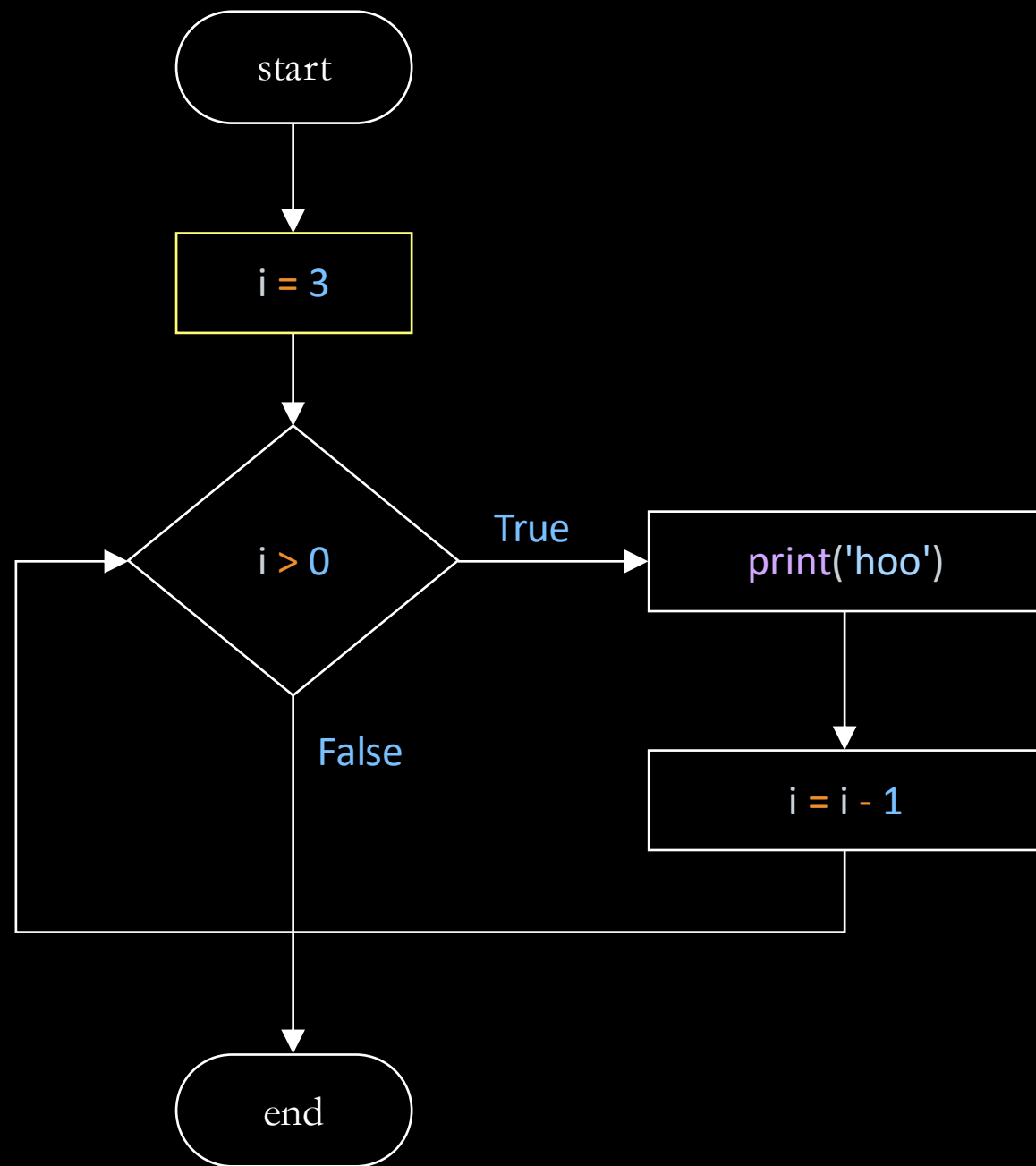
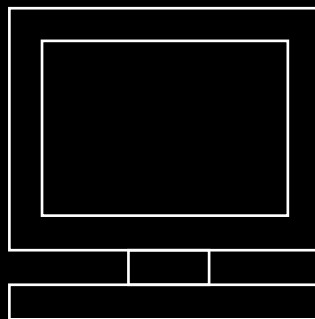
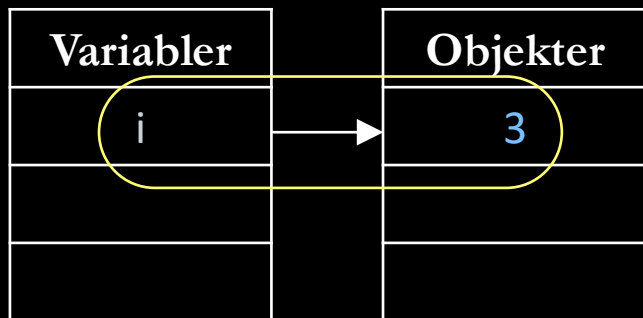
```
i = 3
while i > 0:
    print('hoo')
    i = i - 1
```

Variabler

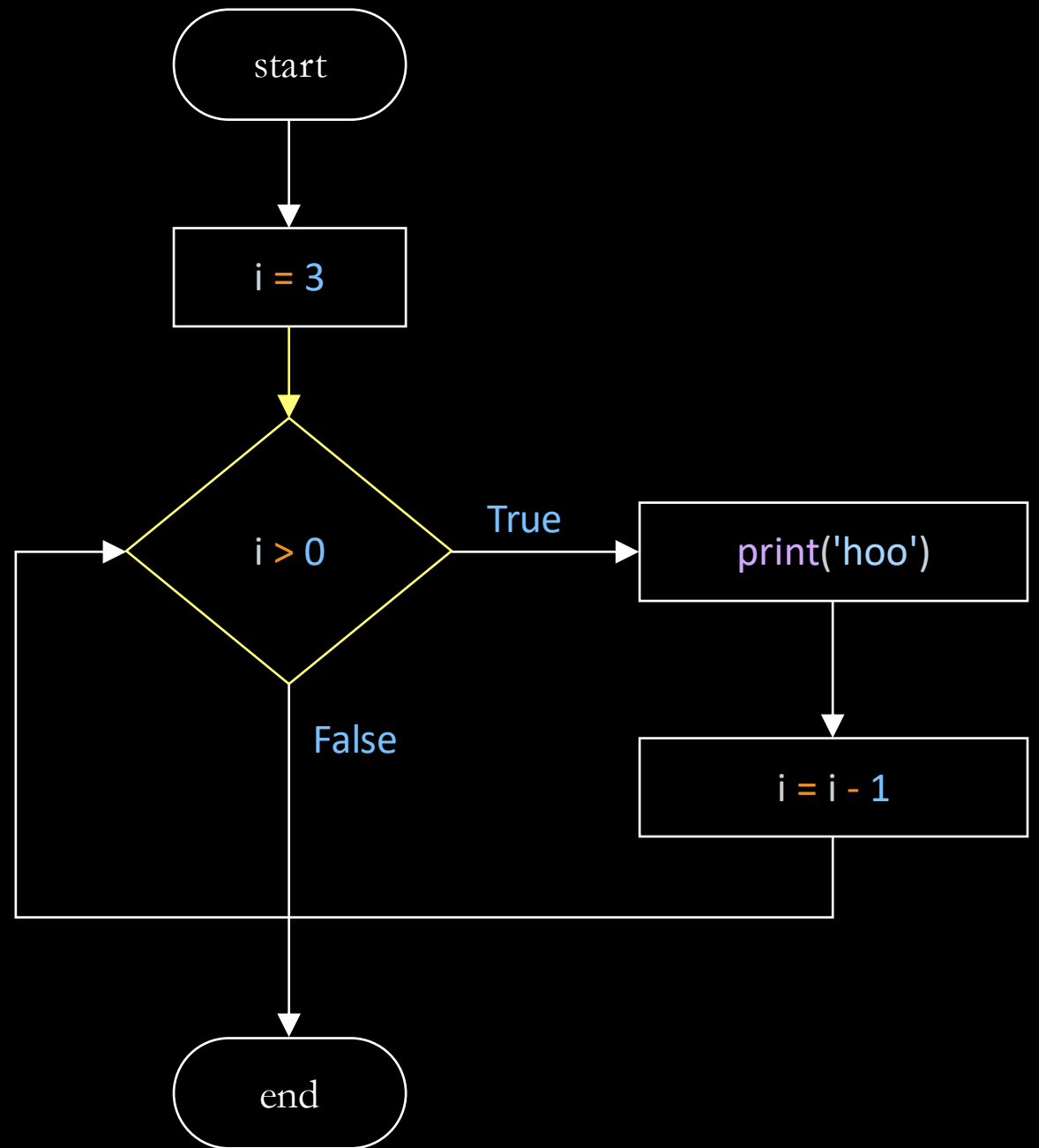
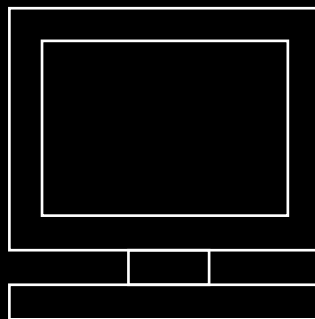
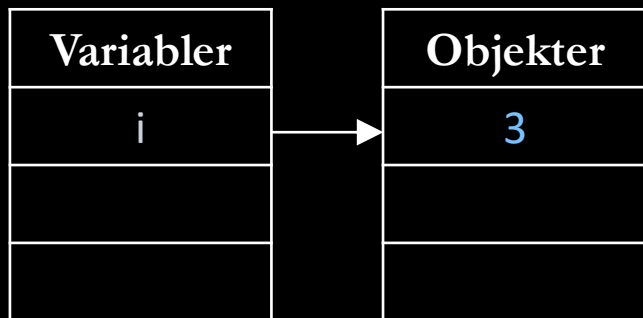
Objekter



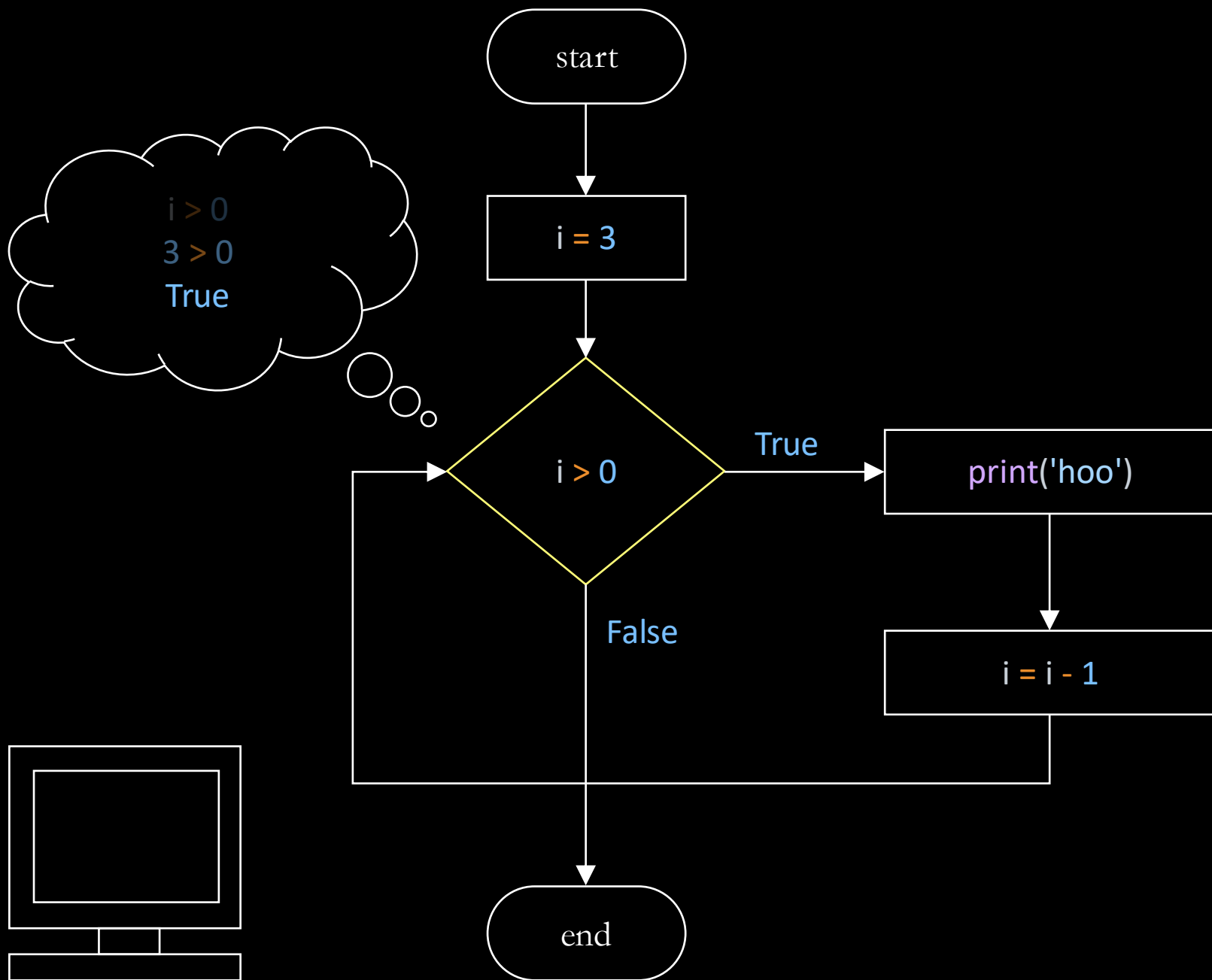
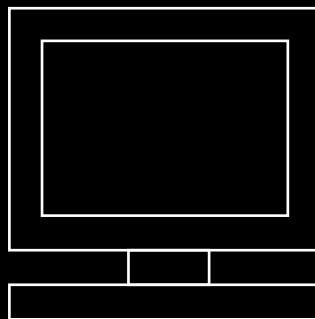
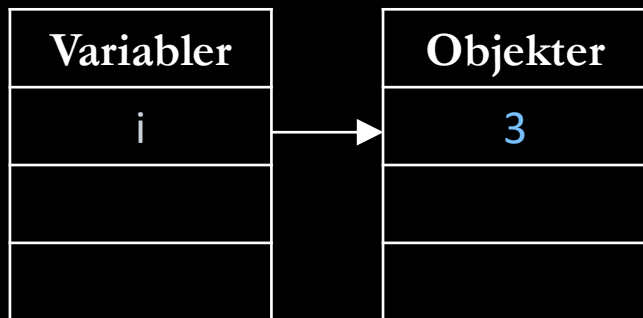
→ `i = 3`
`while i > 0:`
 `print('hoo')`
 `i = i - 1`



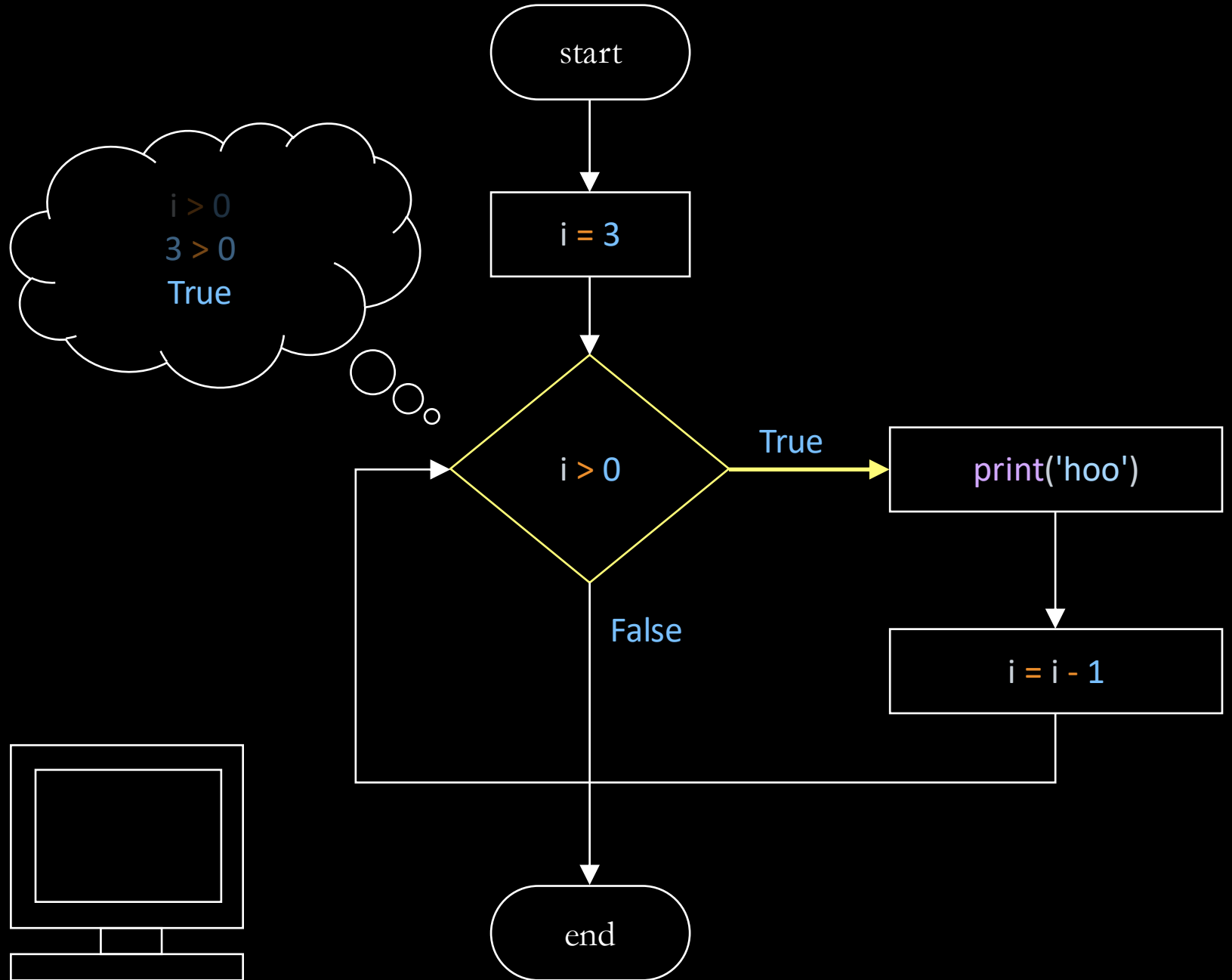
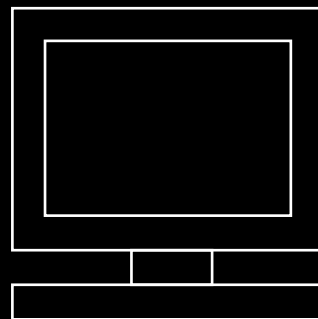
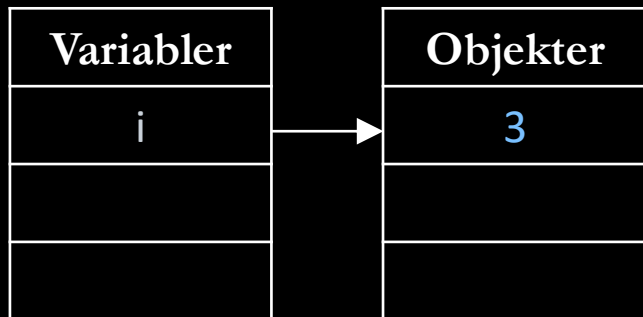
→ `i = 3`
`while i > 0:`
 `print('hoo')`
 `i = i - 1`



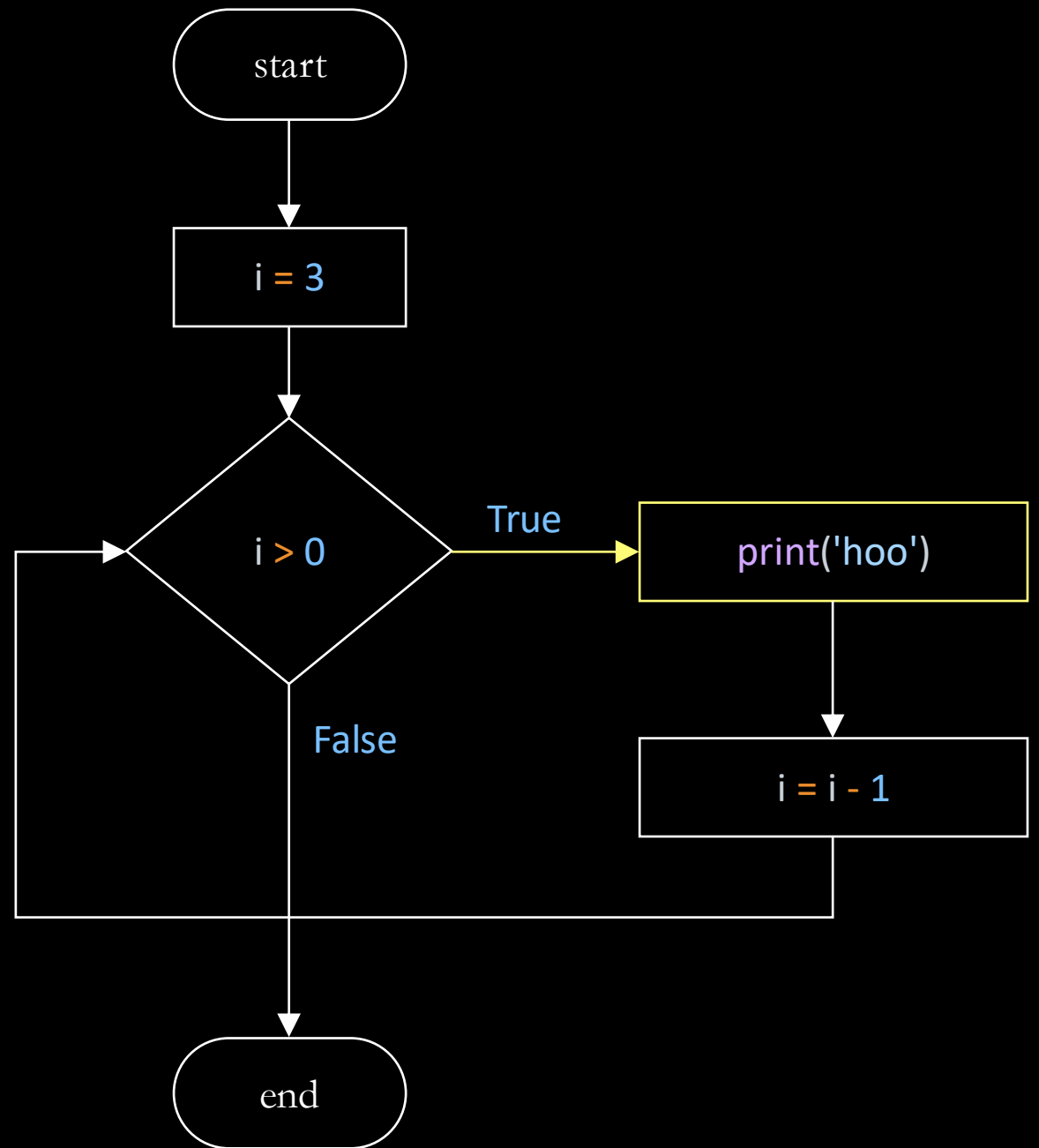
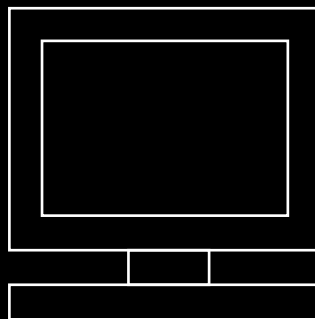
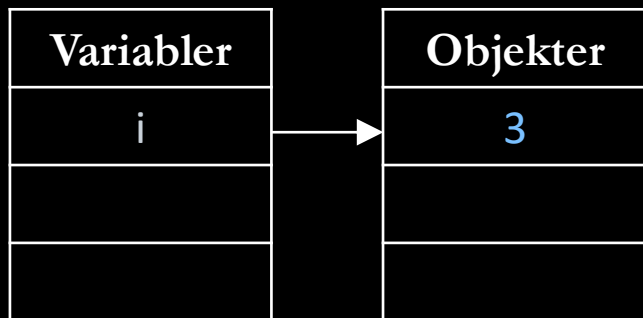
→ `i = 3`
`while i > 0:`
 `print('hoo')`
 `i = i - 1`



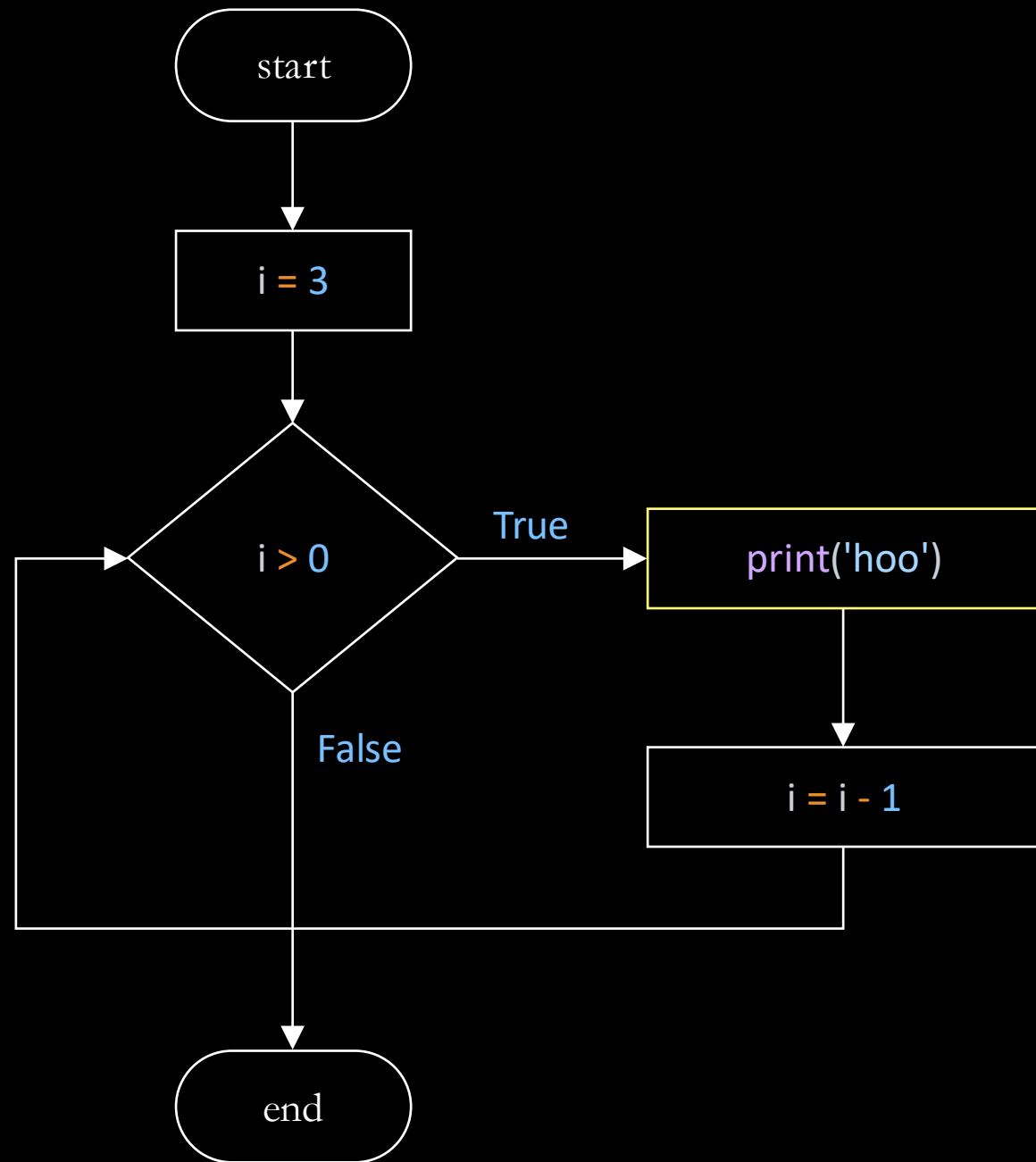
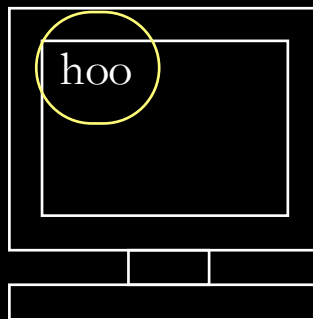
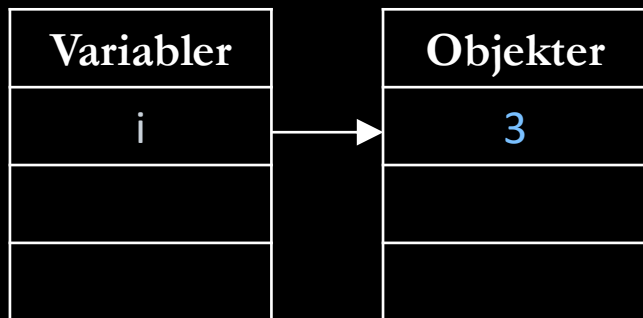
→ `i = 3`
`while i > 0:`
 `print('hoo')`
 `i = i - 1`



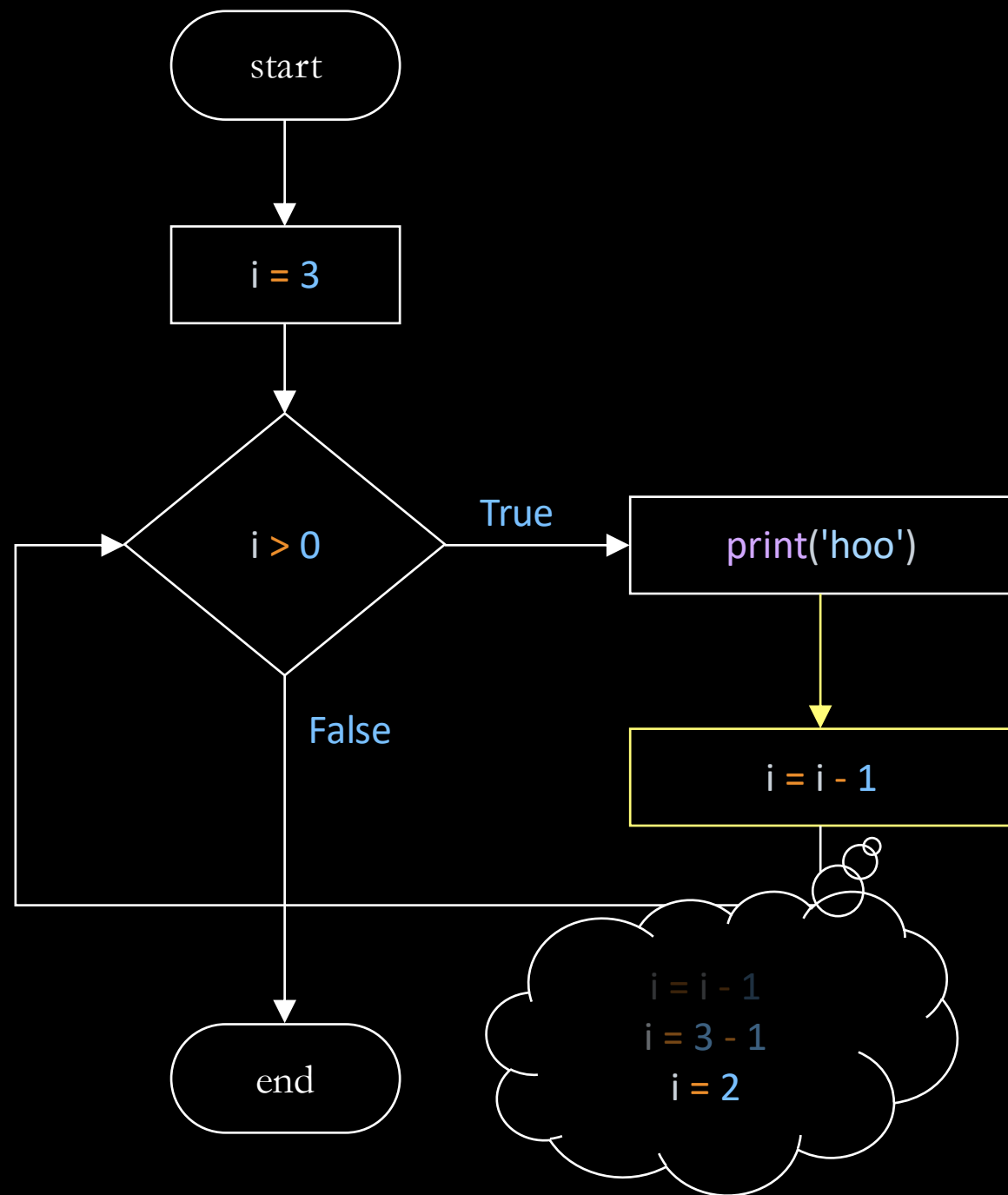
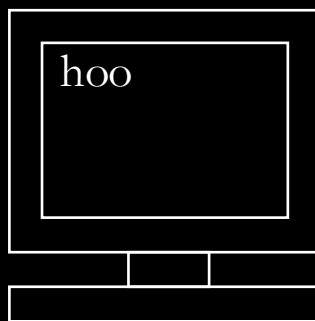
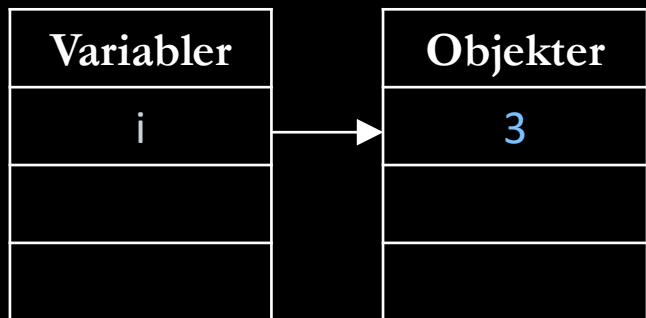
→
i = 3
while i > 0:
 print('hoo')
 i = i - 1



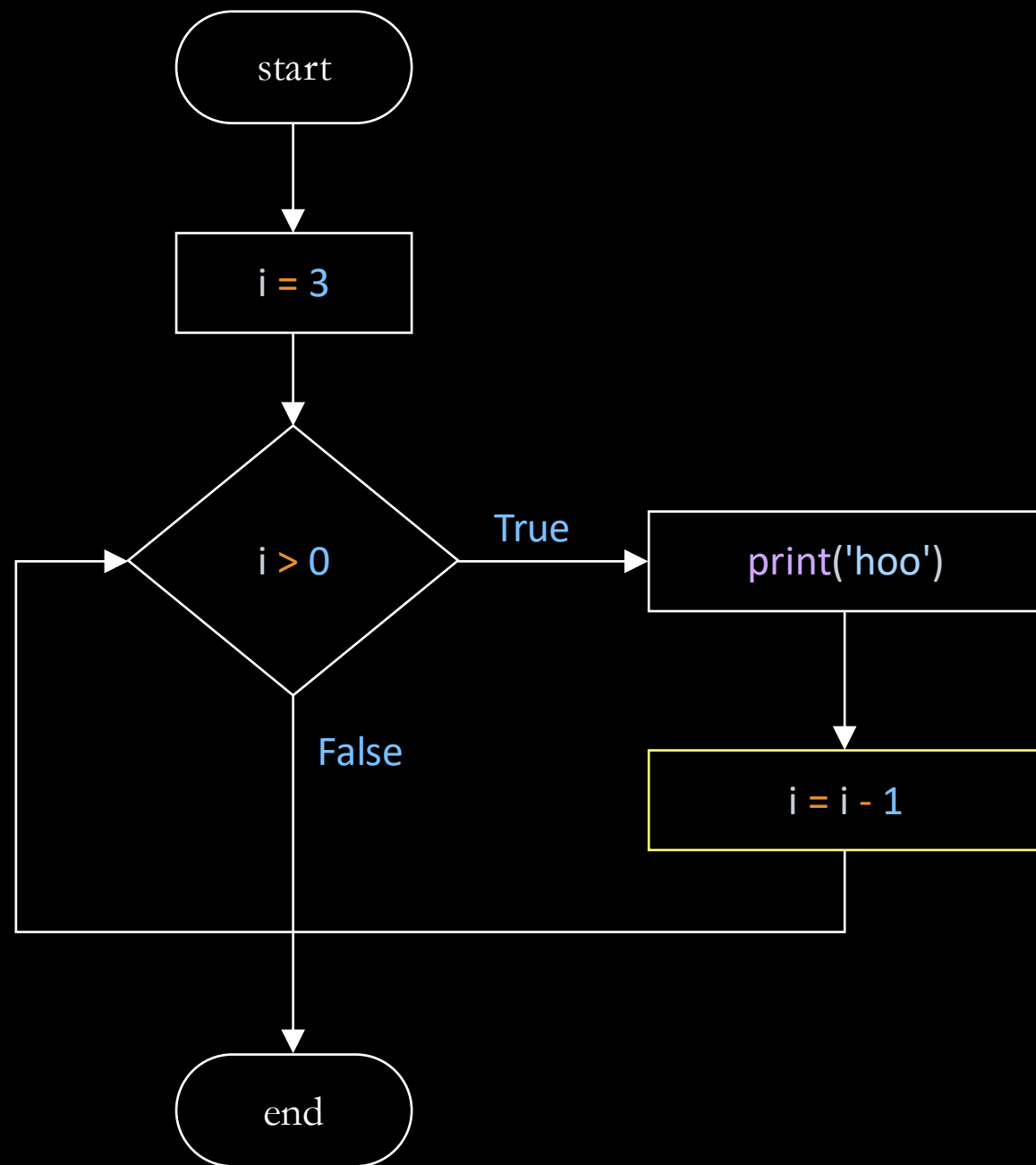
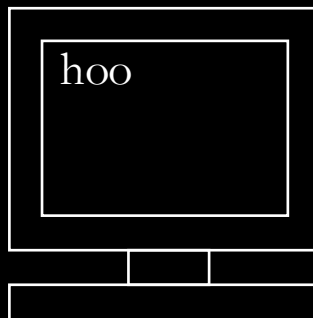
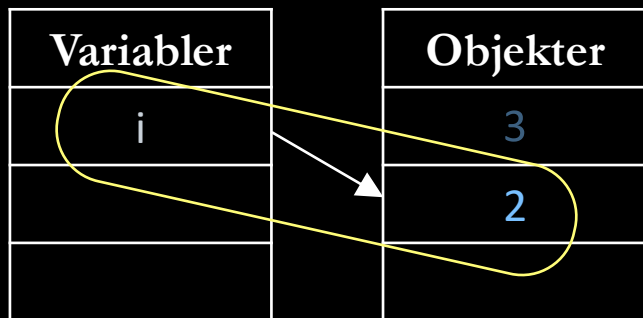
→
i = 3
while i > 0:
 print('hoo')
 i = i - 1



→
`i = 3`
`while i > 0:`
 `print('hoo')`
 `i = i - 1`

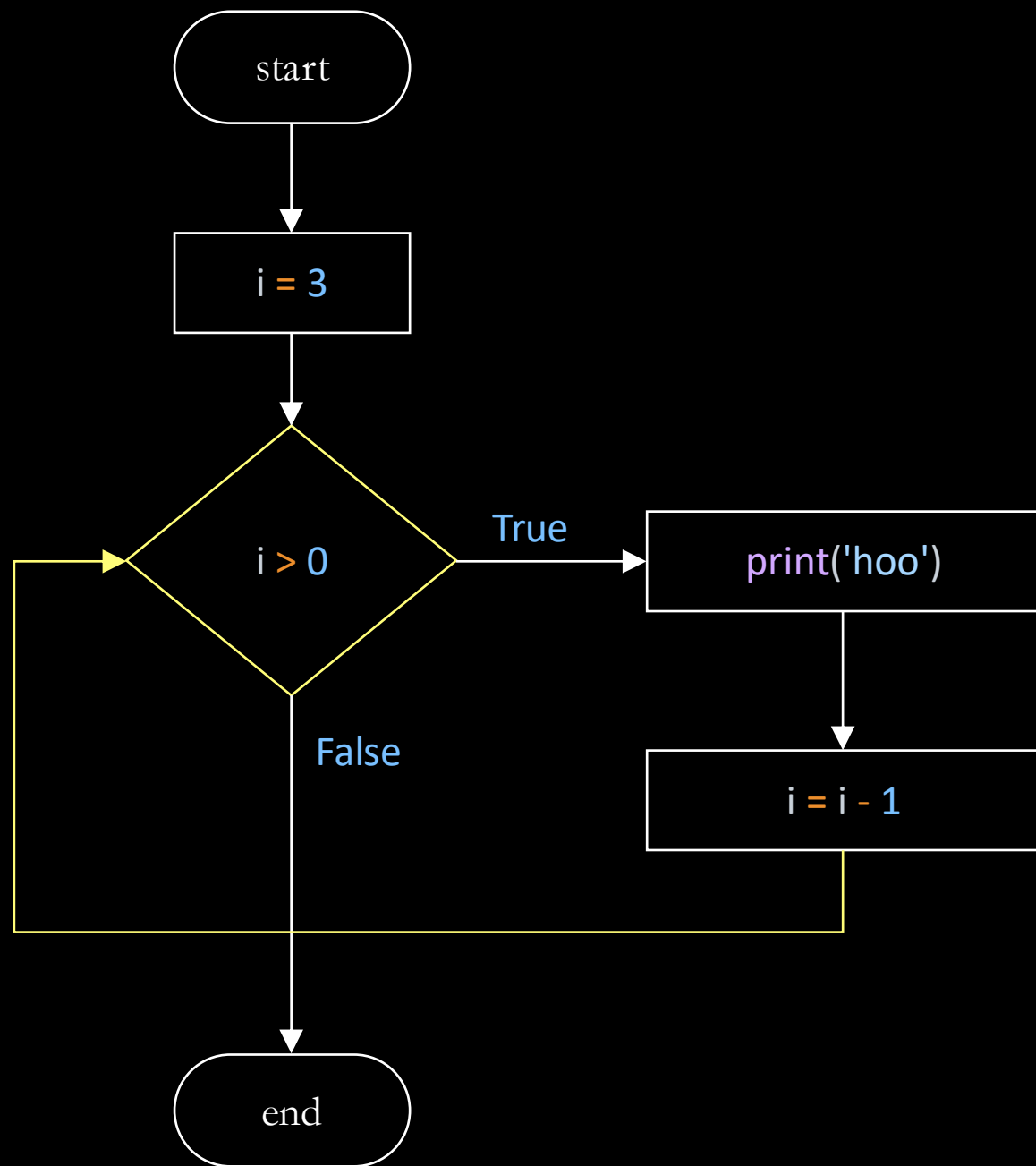
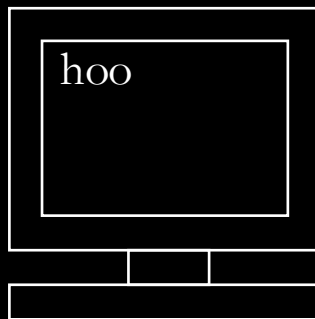


→
`i = 3`
`while i > 0:`
 `print('hoo')`
 `i = i - 1`



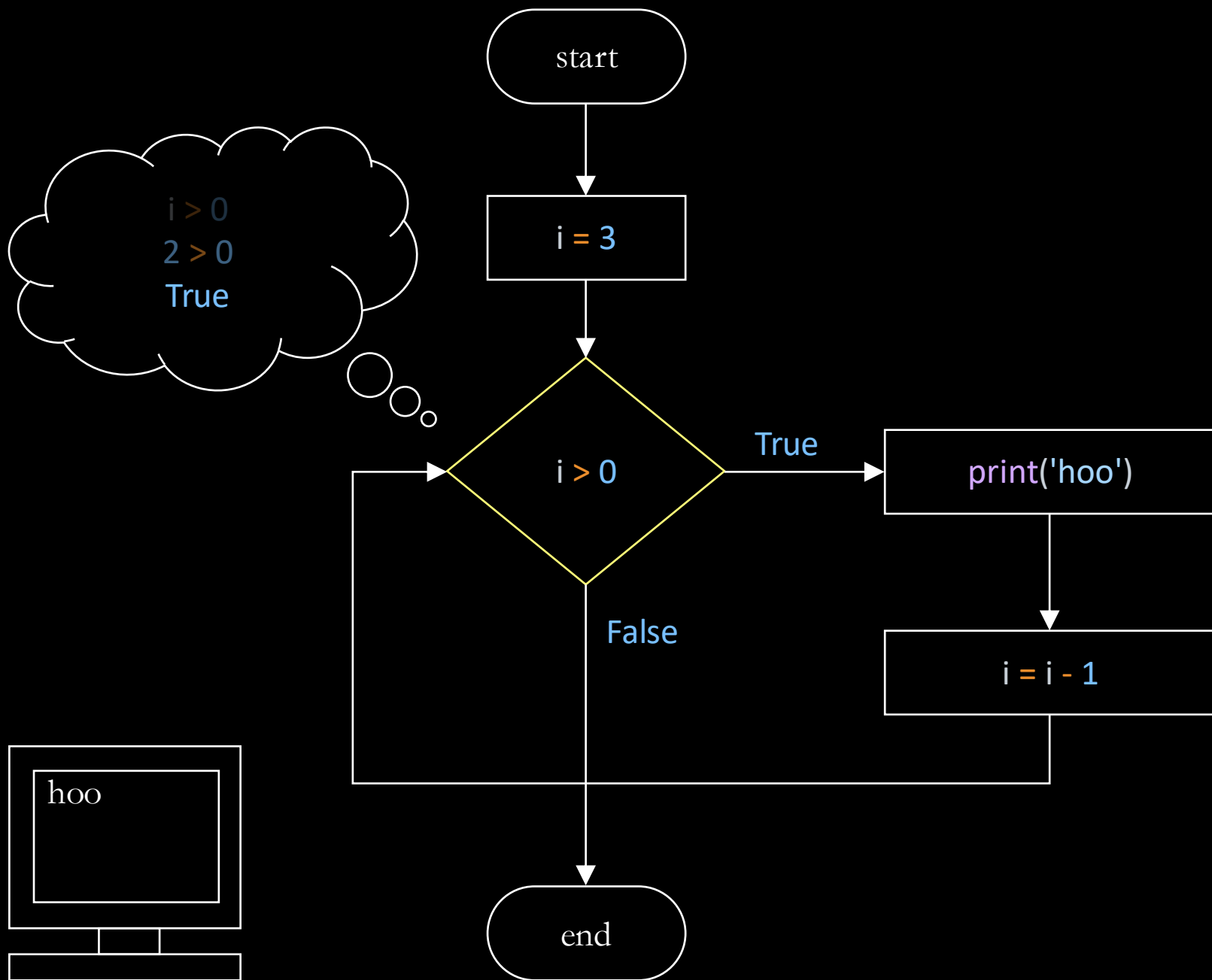
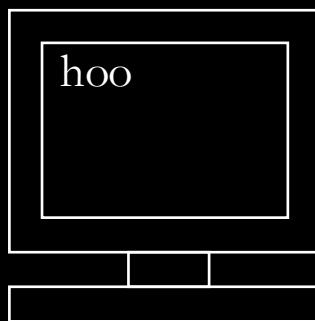
→ `i = 3`
`while i > 0:`
 `print('hoo')`
 `i = i - 1`

Variabler	Objekter
i	3
	2



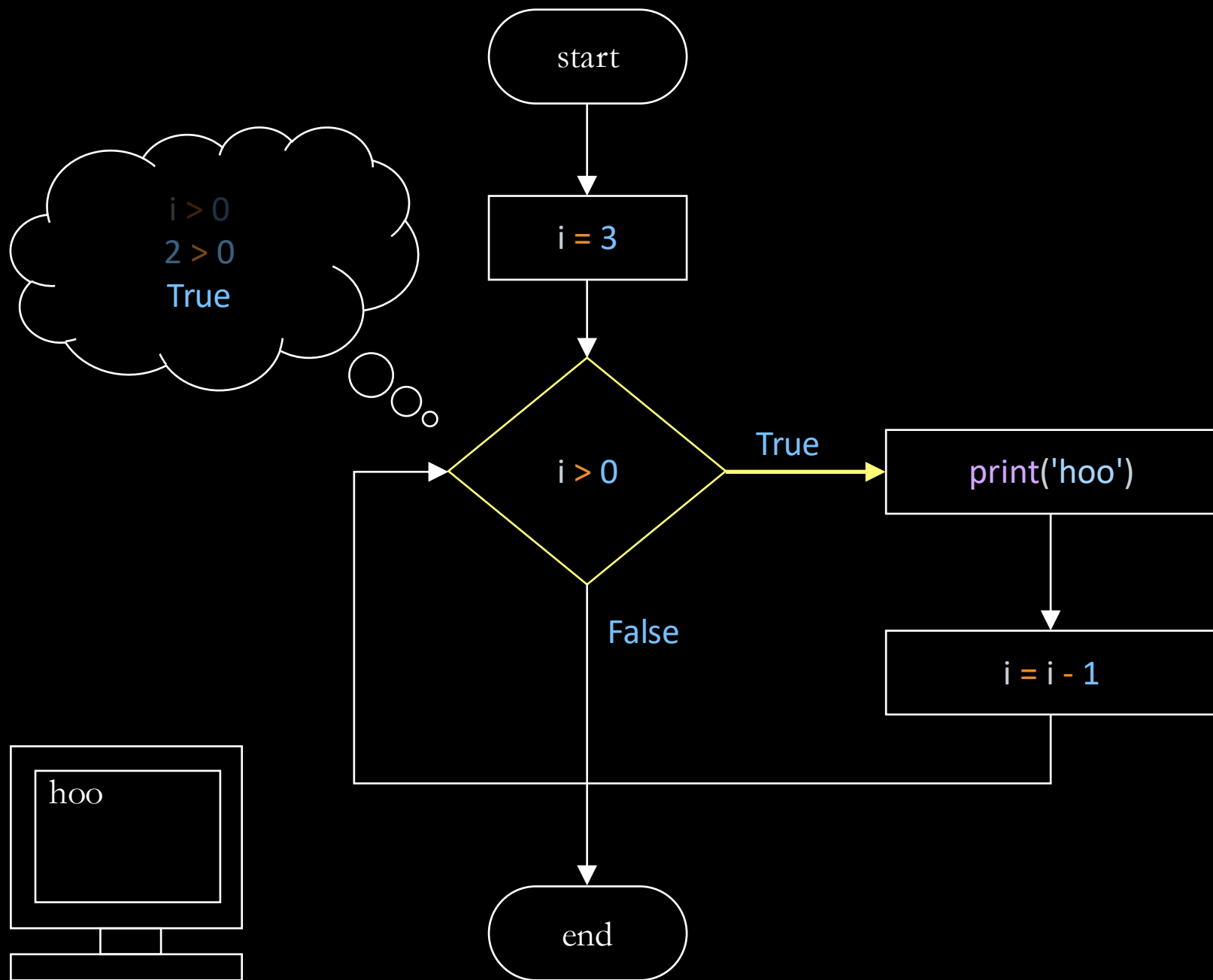
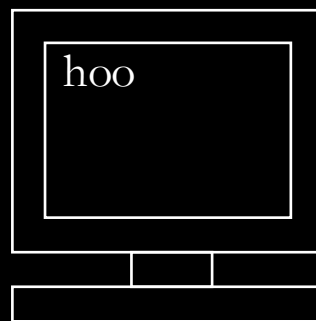
→ `i = 3`
`while i > 0:`
 `print('hoo')`
 `i = i - 1`

Variabler	Objekter
i	3
	2



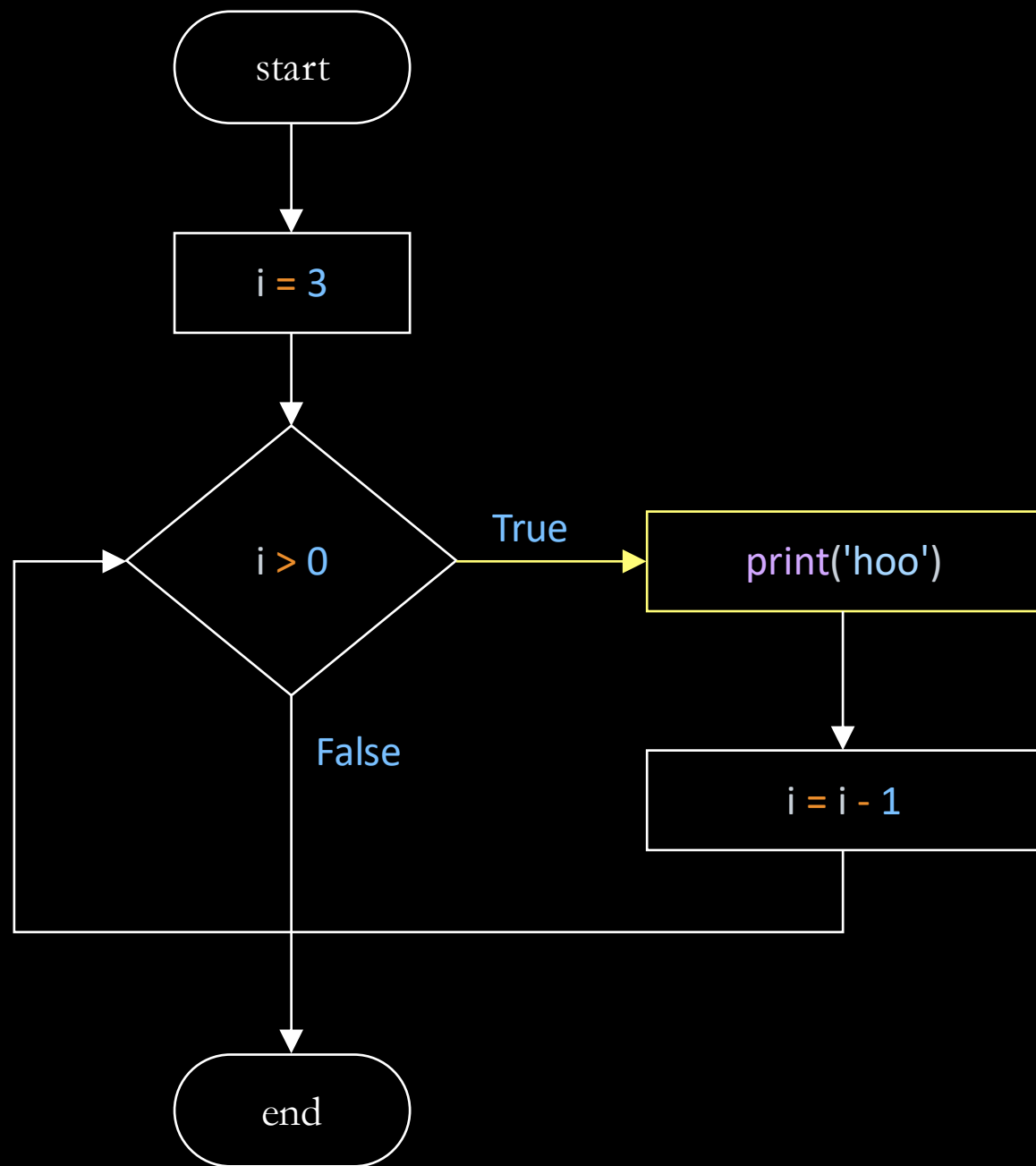
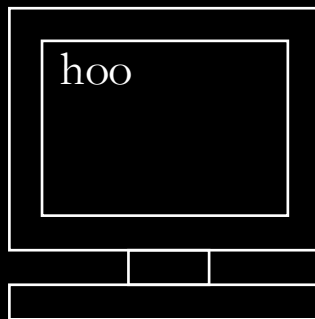
→ `i = 3`
`while i > 0:`
 `print('hoo')`
 `i = i - 1`

Variabler	Objekter
i	3
	2



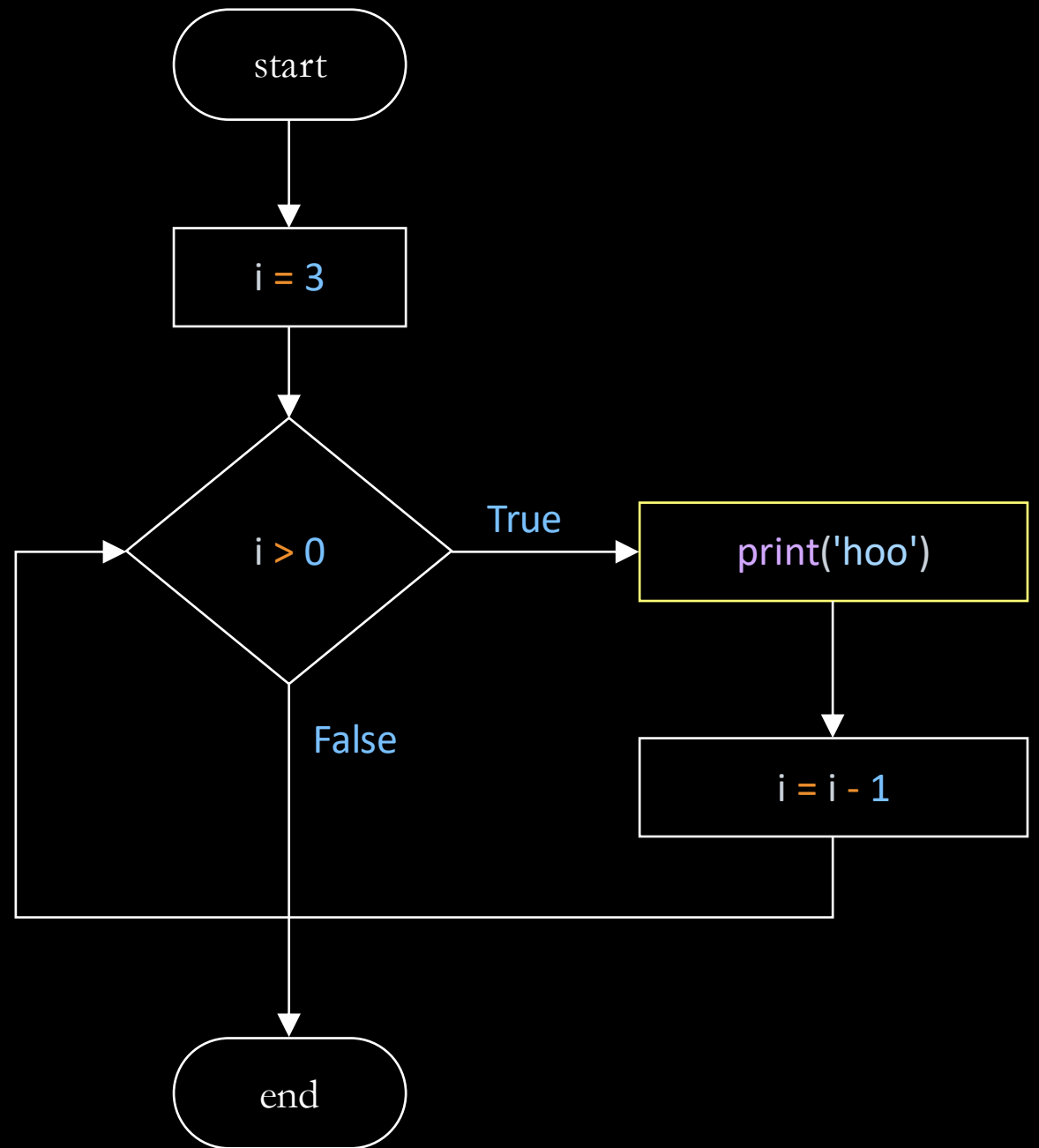
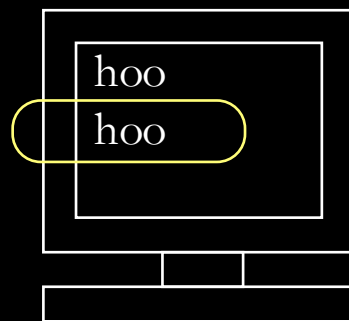
→
i = 3
while i > 0:
 print('hoo')
 i = i - 1

Variabler	Objekter
i	3
	2



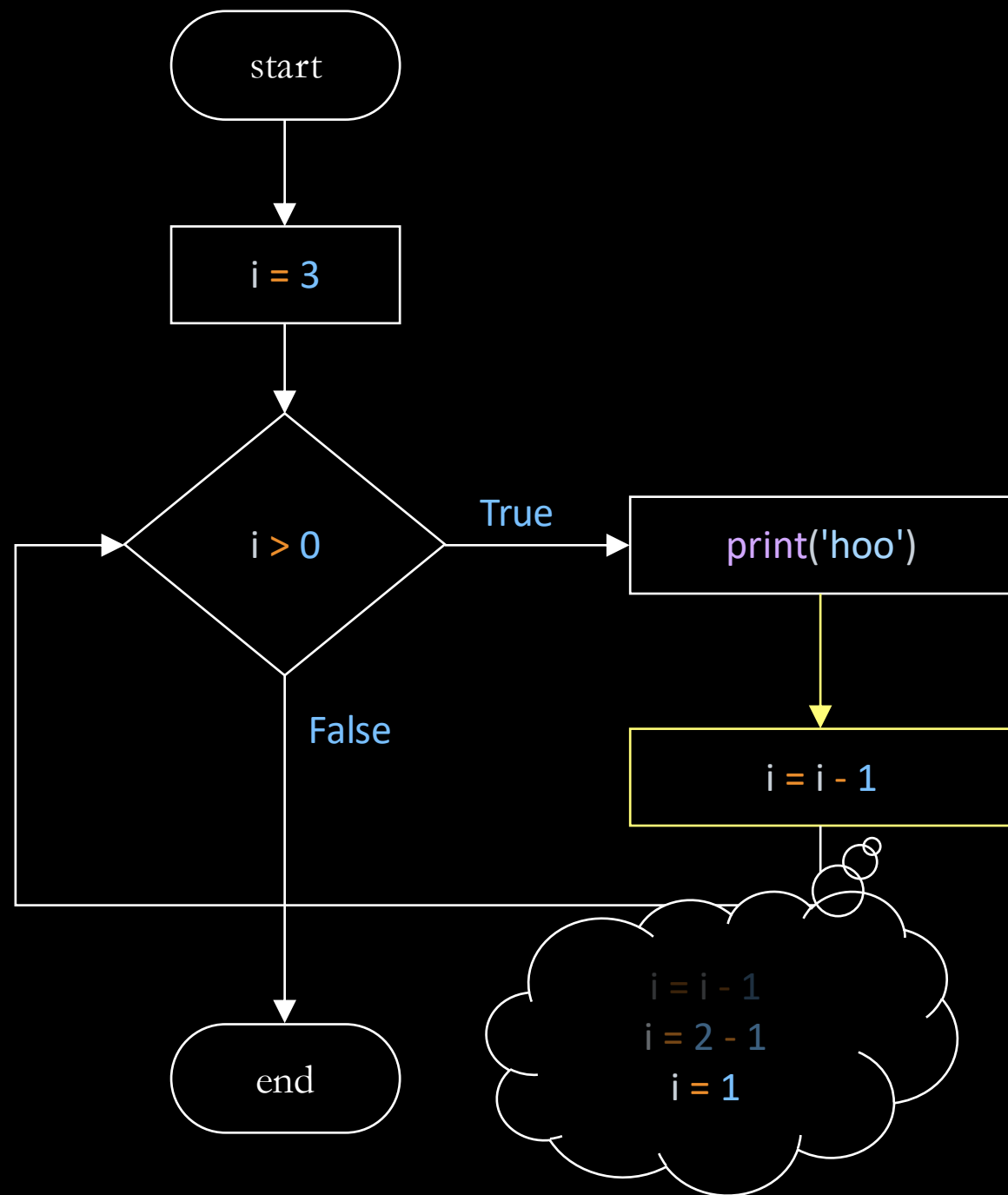
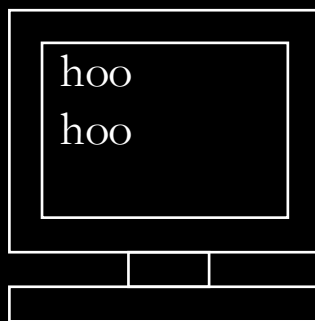
→
i = 3
while i > 0:
 print('hoo')
 i = i - 1

Variabler	Objekter
i	3
	2

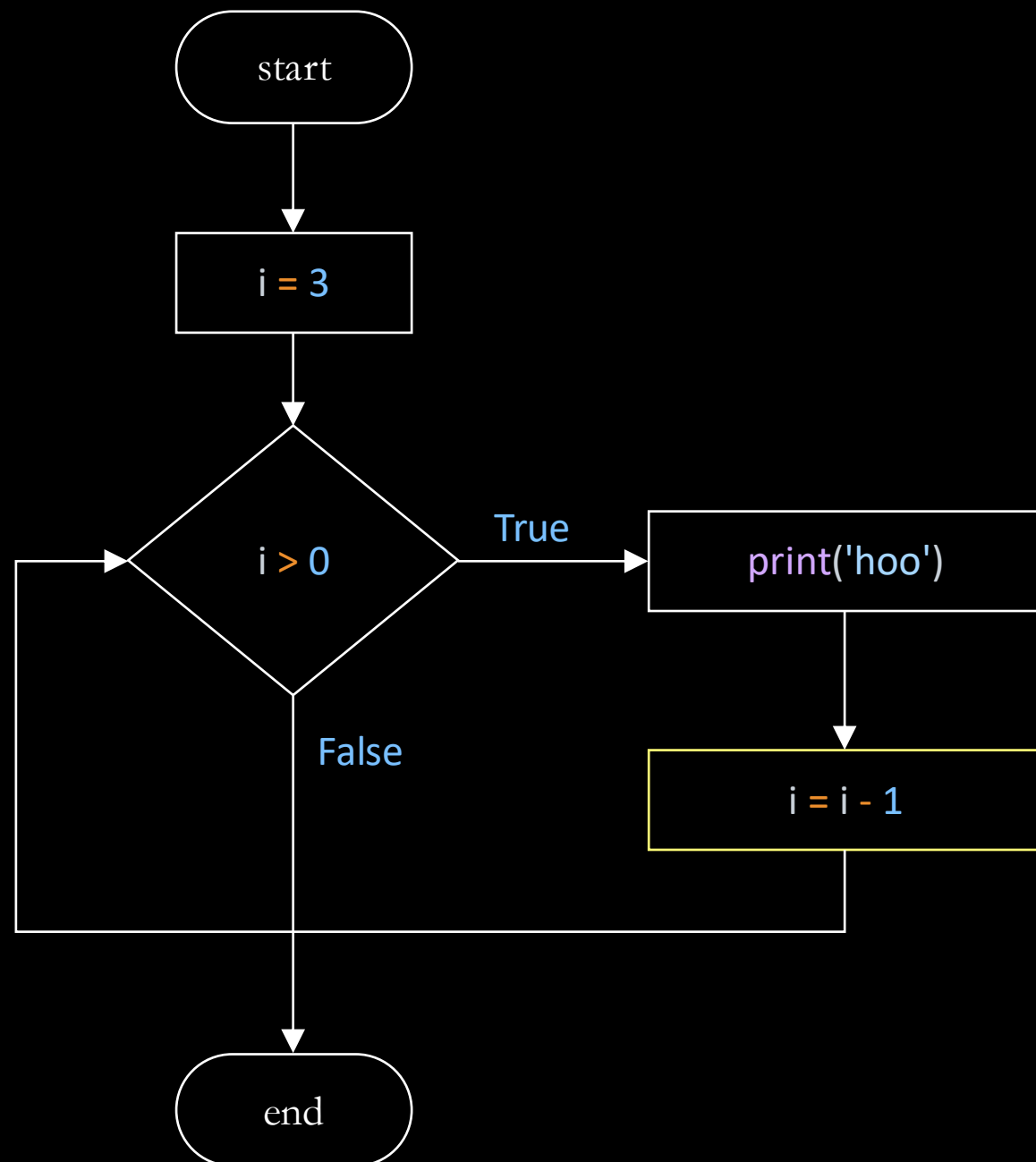
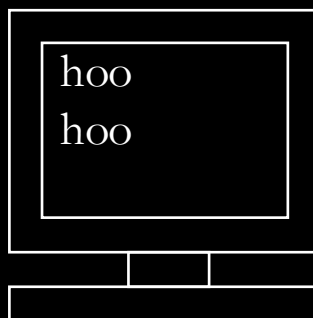
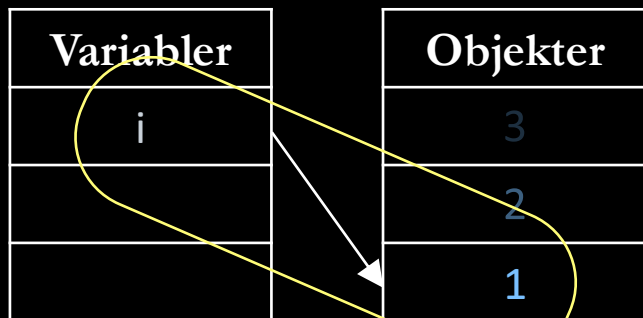


→
i = 3
while i > 0:
 print('hoo')
 i = i - 1

Variabler	Objekter
i	3
	2

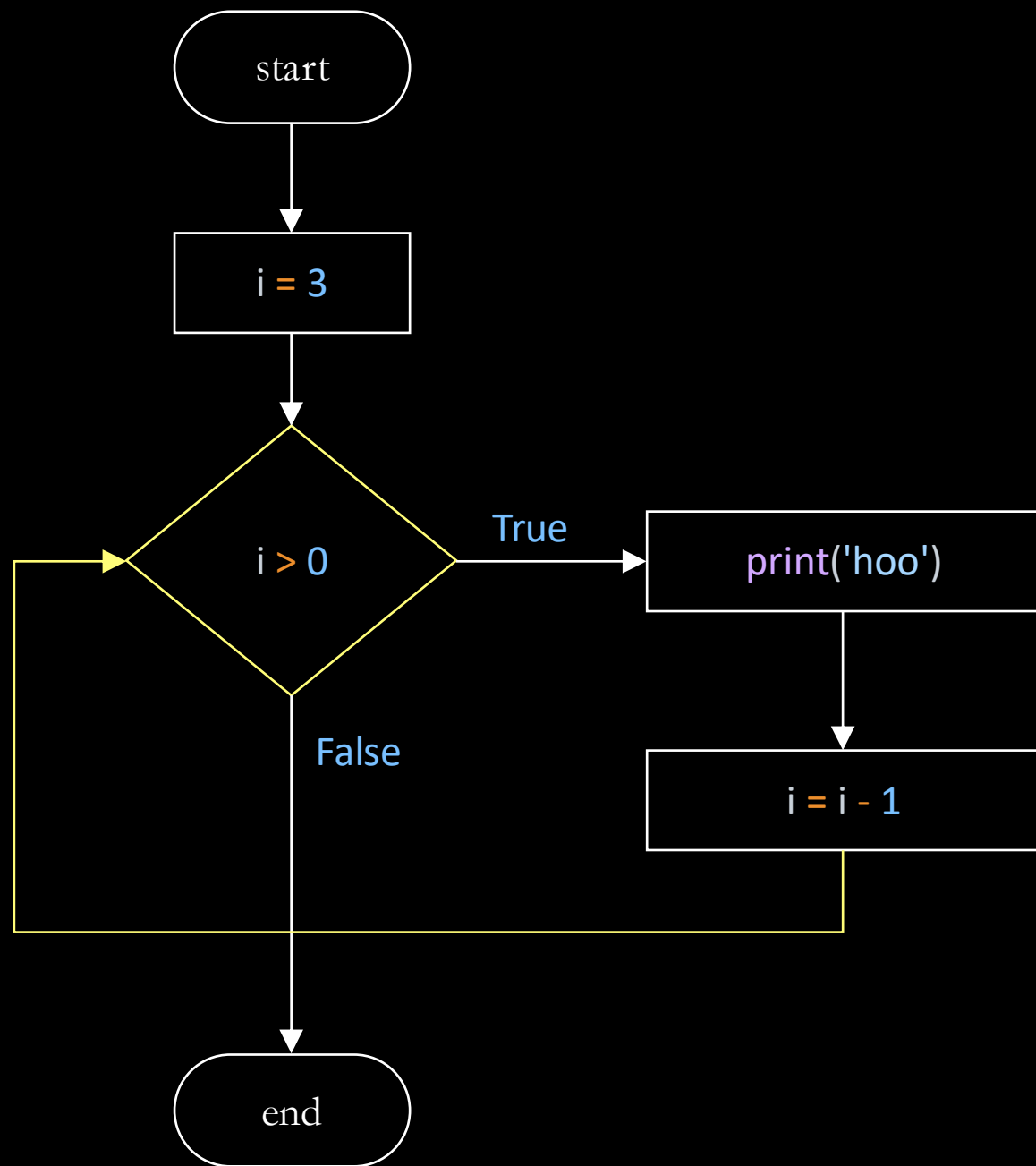
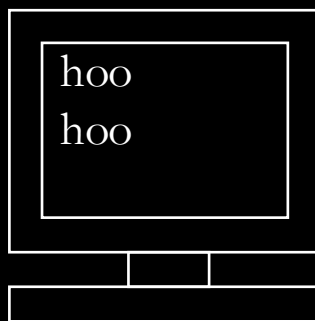


→
`i = 3`
`while i > 0:`
 `print('hoo')`
 `i = i - 1`



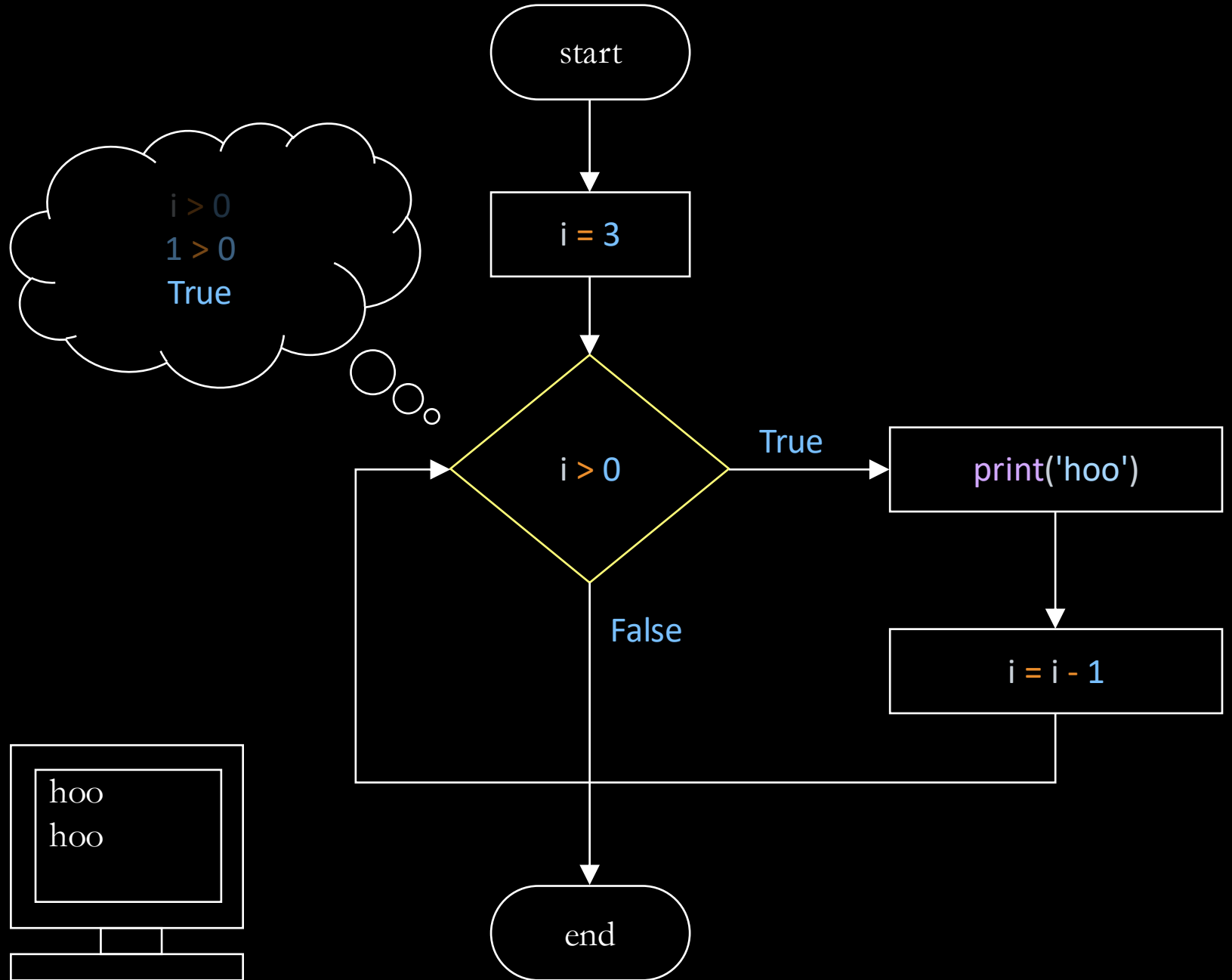
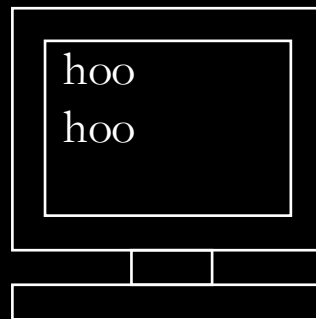
→ `i = 3`
`while i > 0:`
 `print('hoo')`
 `i = i - 1`

Variabler	Objekter
i	3
	2
	1



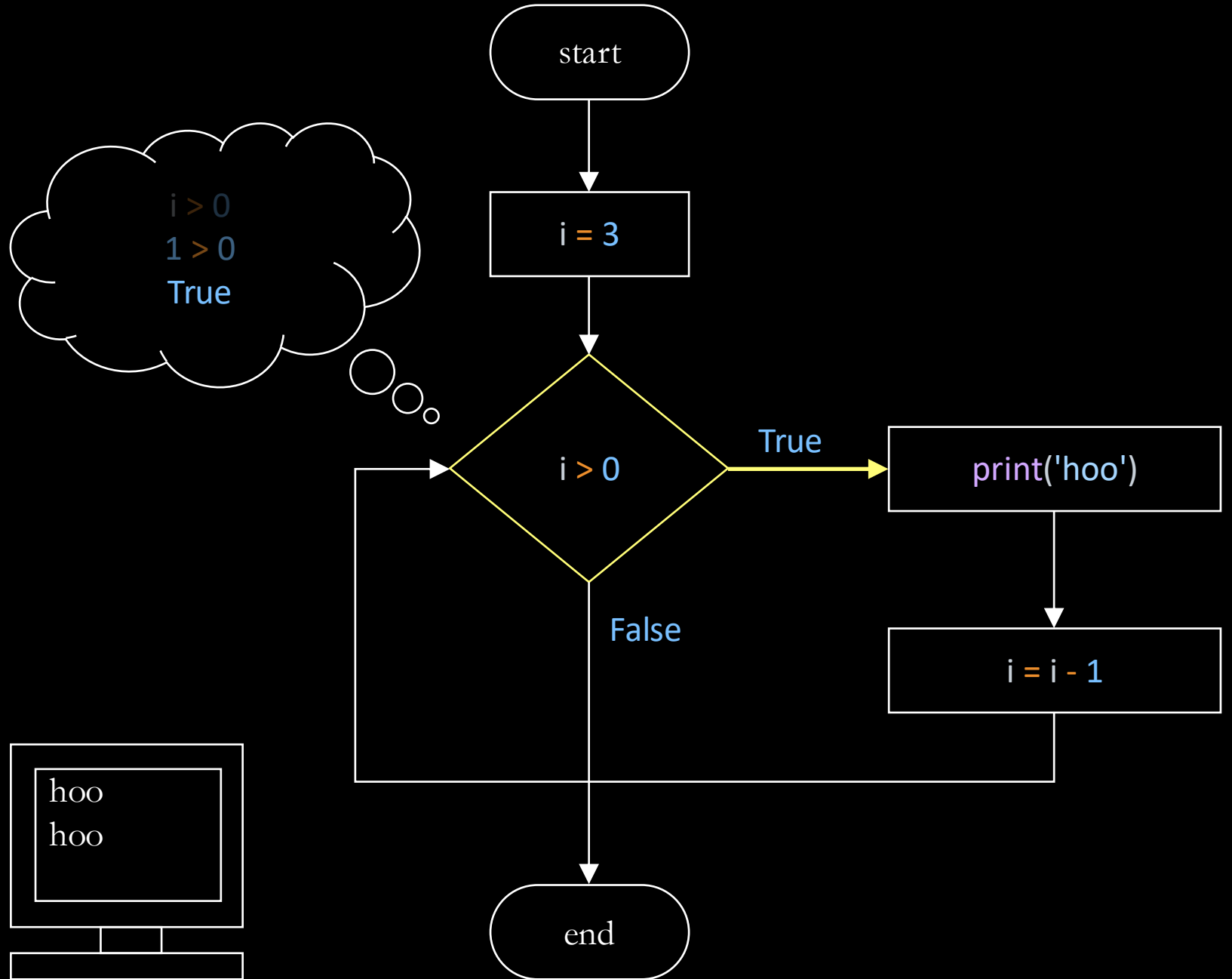
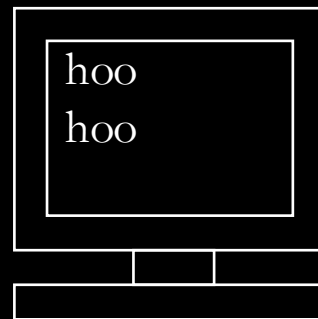
→ `i = 3`
`while i > 0:`
 `print('hoo')`
 `i = i - 1`

Variabler	Objekter
i	3
	2
	1



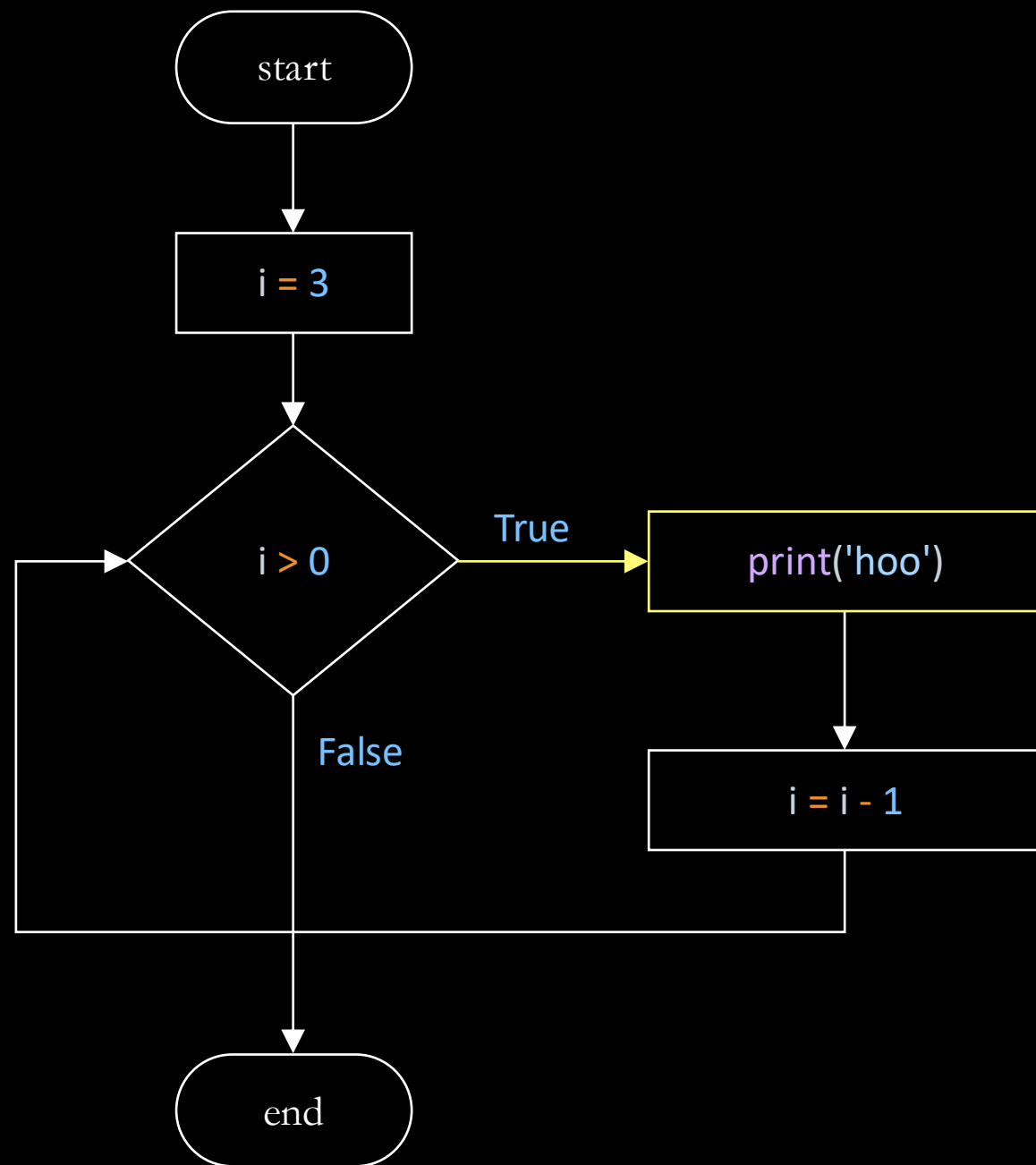
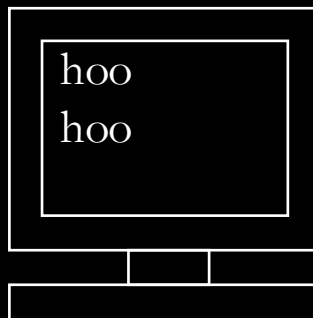
→ `i = 3`
`while i > 0:`
 `print('hoo')`
 `i = i - 1`

Variabler	Objekter
i	3
	2
	1



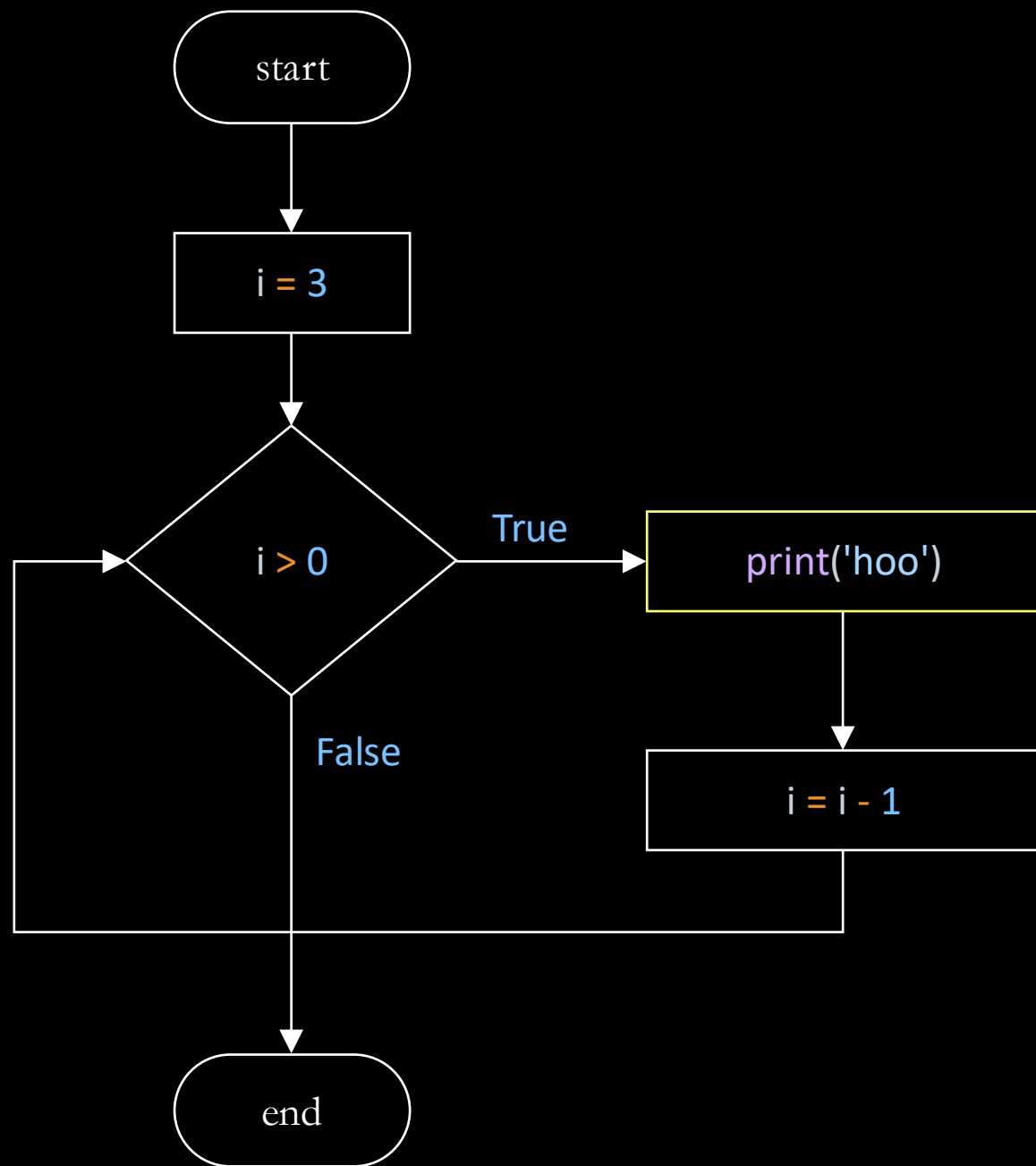
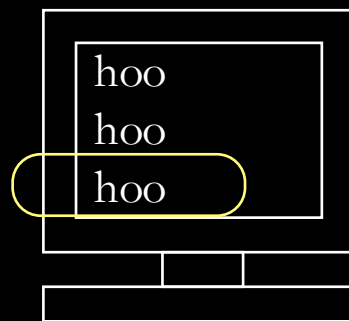
→
i = 3
while i > 0:
 print('hoo')
 i = i - 1

Variabler	Objekter
i	3
	2
	1



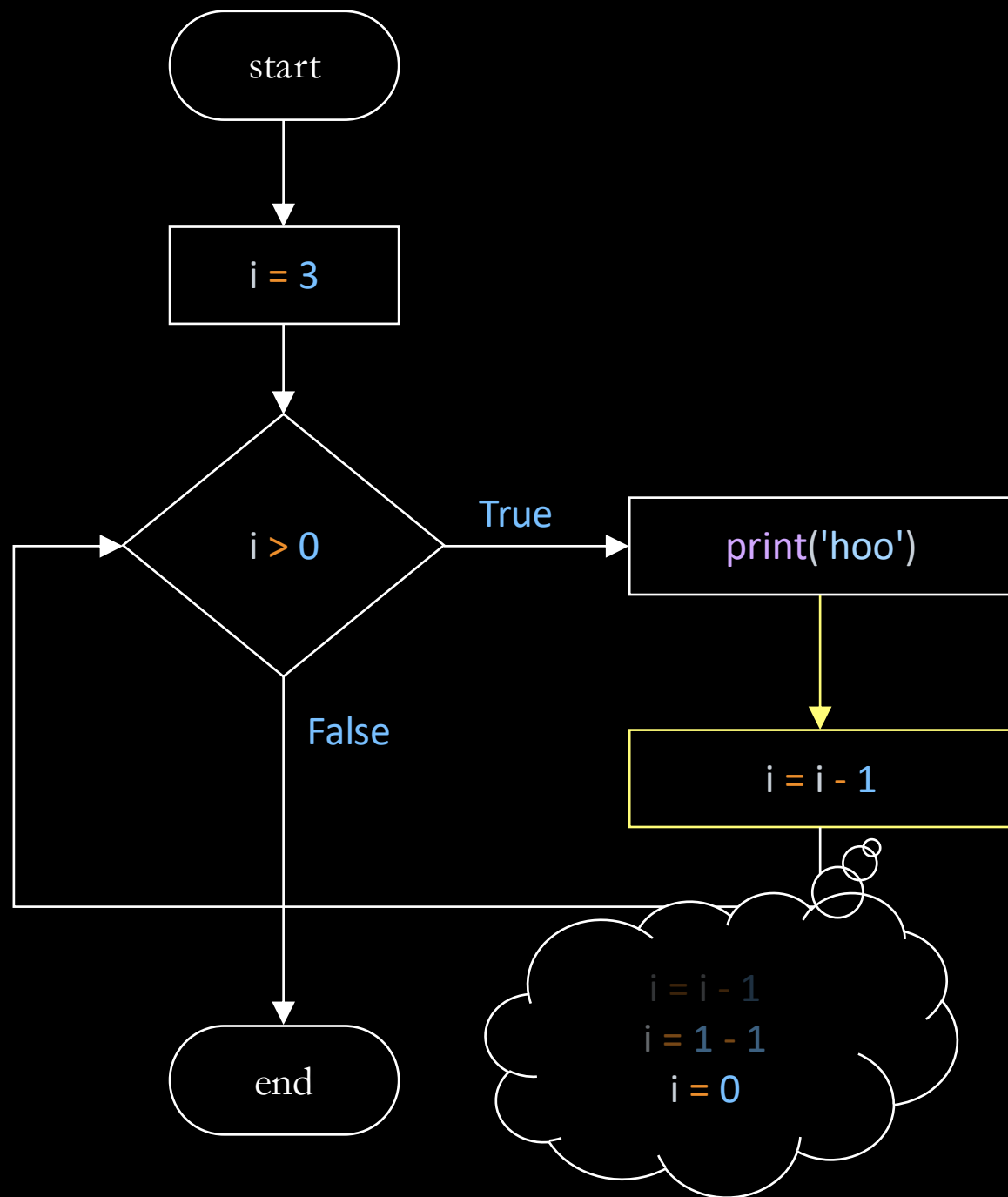
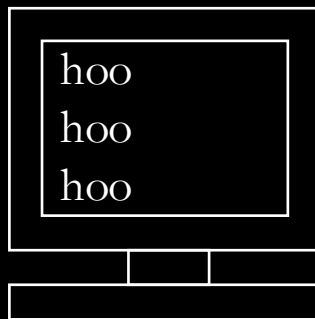
→
i = 3
while i > 0:
 print('hoo')
 i = i - 1

Variabler	Objekter
i	3
	2
	1



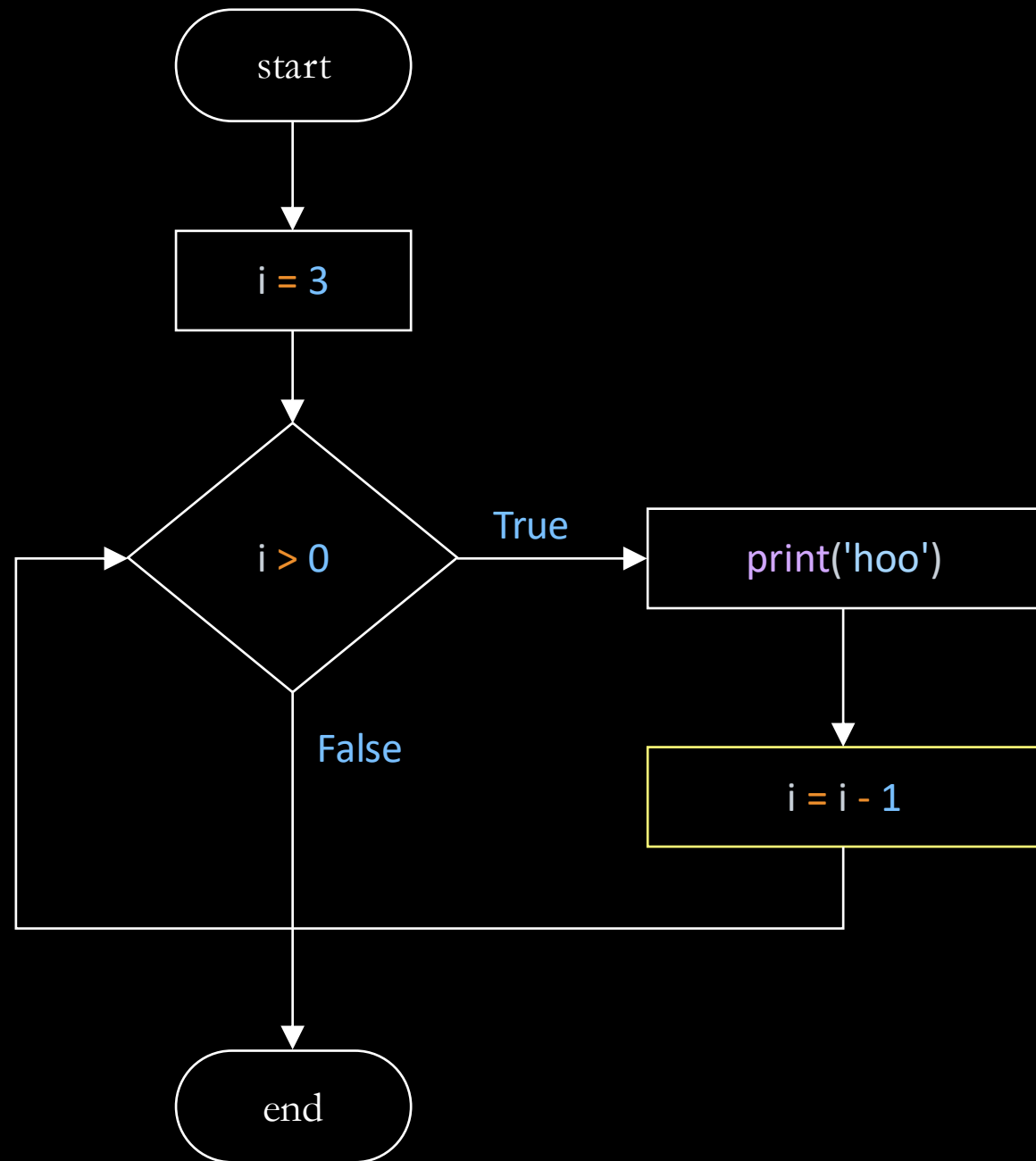
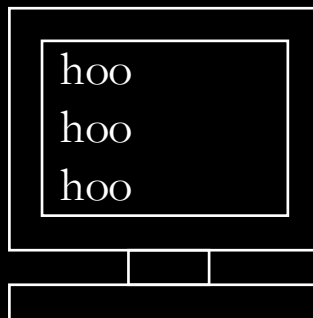
→
i = 3
while i > 0:
 print('hoo')
 i = i - 1

Variabler	Objekter
i	3
	2
	1



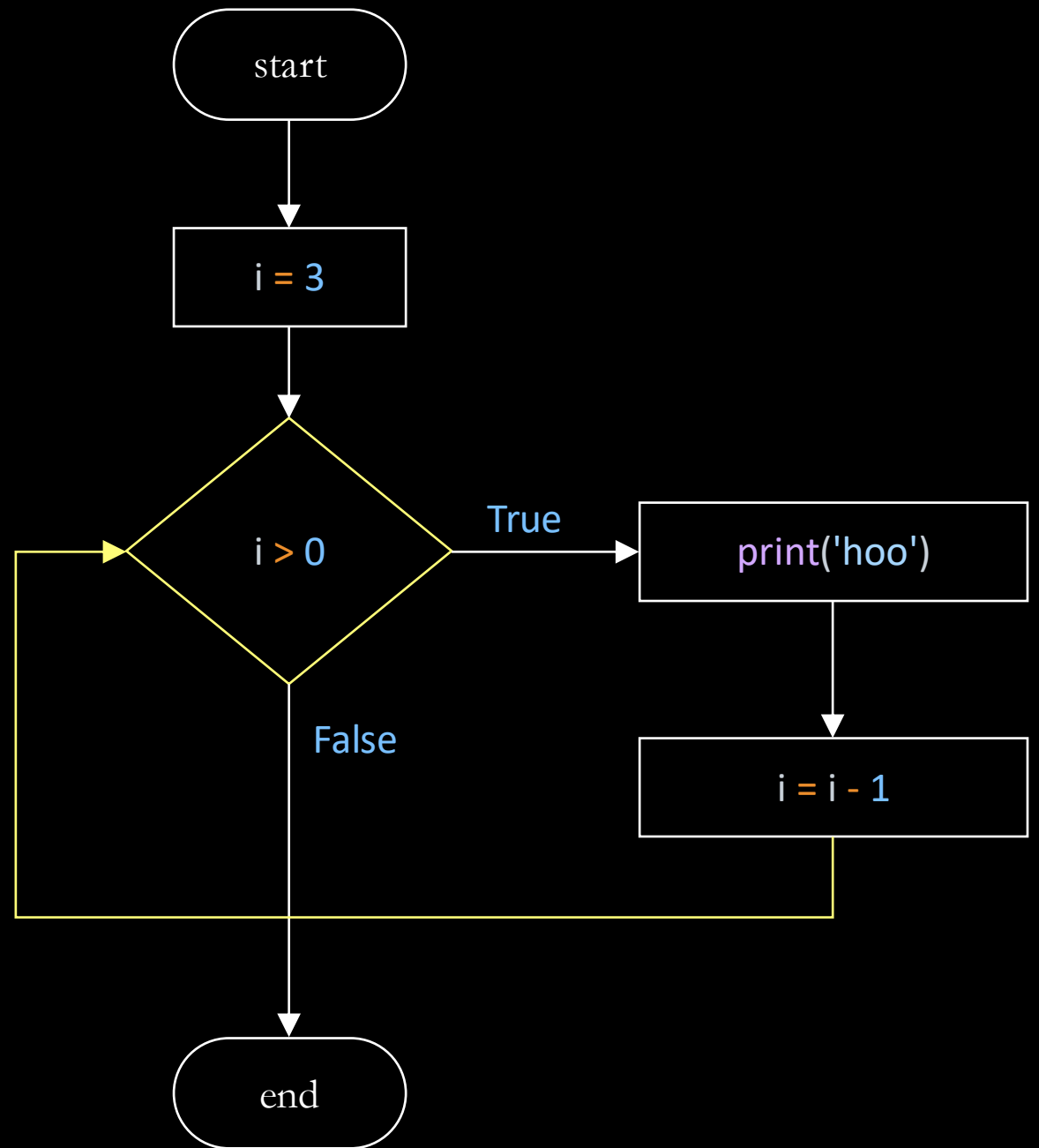
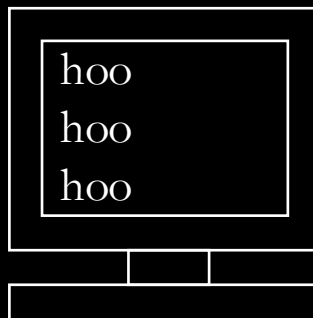
→
i = 3
while i > 0:
 print('hoo')
 i = i - 1

Variabler	Objekter
i	0
	2
	1



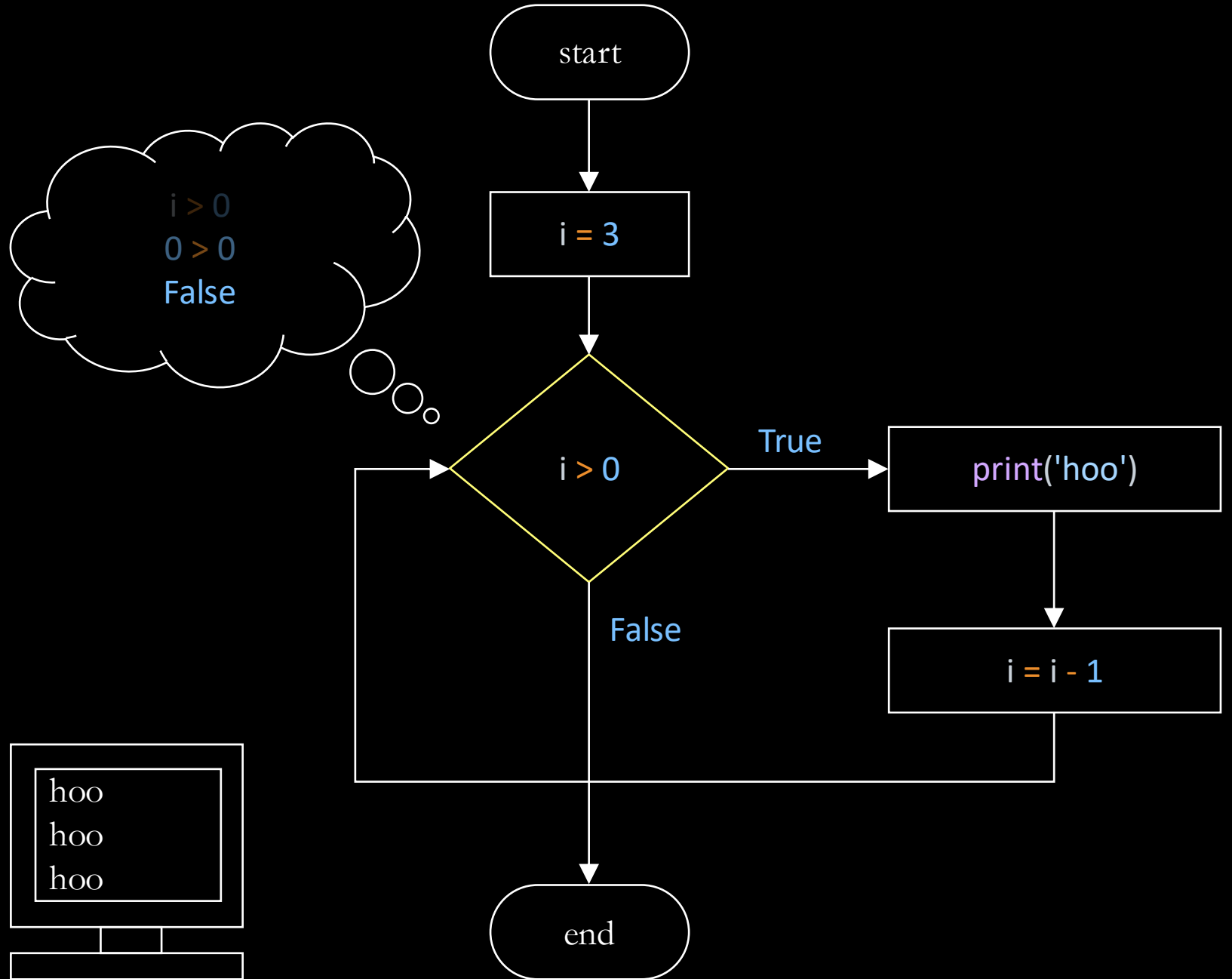
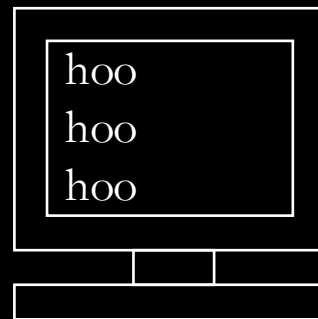
→ `i = 3`
`while i > 0:`
 `print('hoo')`
 `i = i - 1`

Variabler	Objekter
i	0
	2
	1



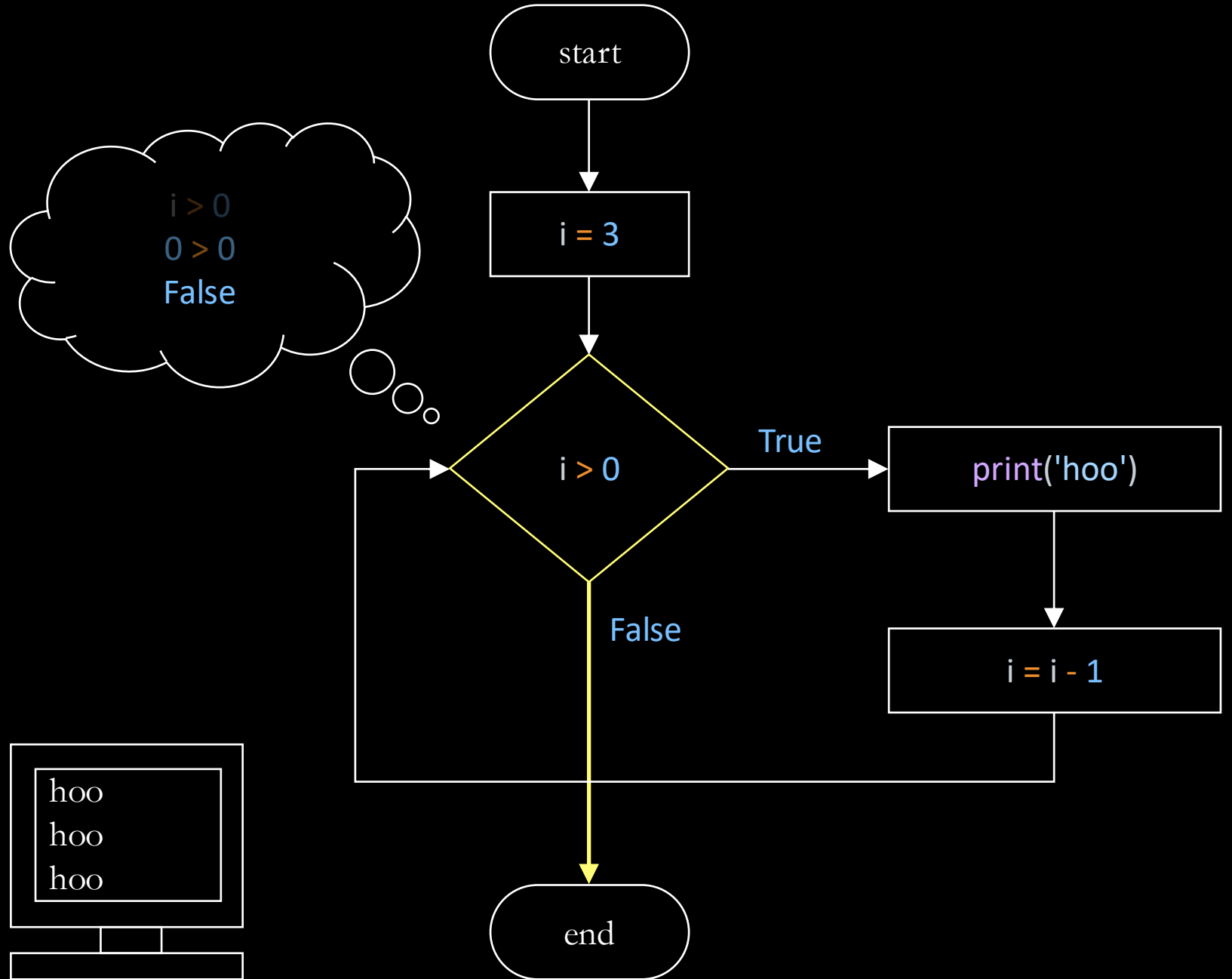
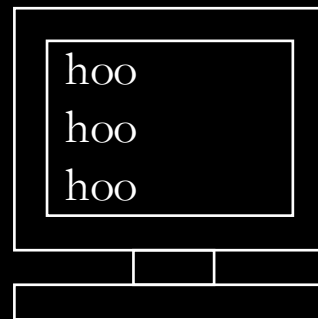
→ `i = 3`
`while i > 0:`
 `print('hoo')`
 `i = i - 1`

Variabler	Objekter
i	0
	2
	1



→ `i = 3`
`while i > 0:`
 `print('hoo')`
 `i = i - 1`

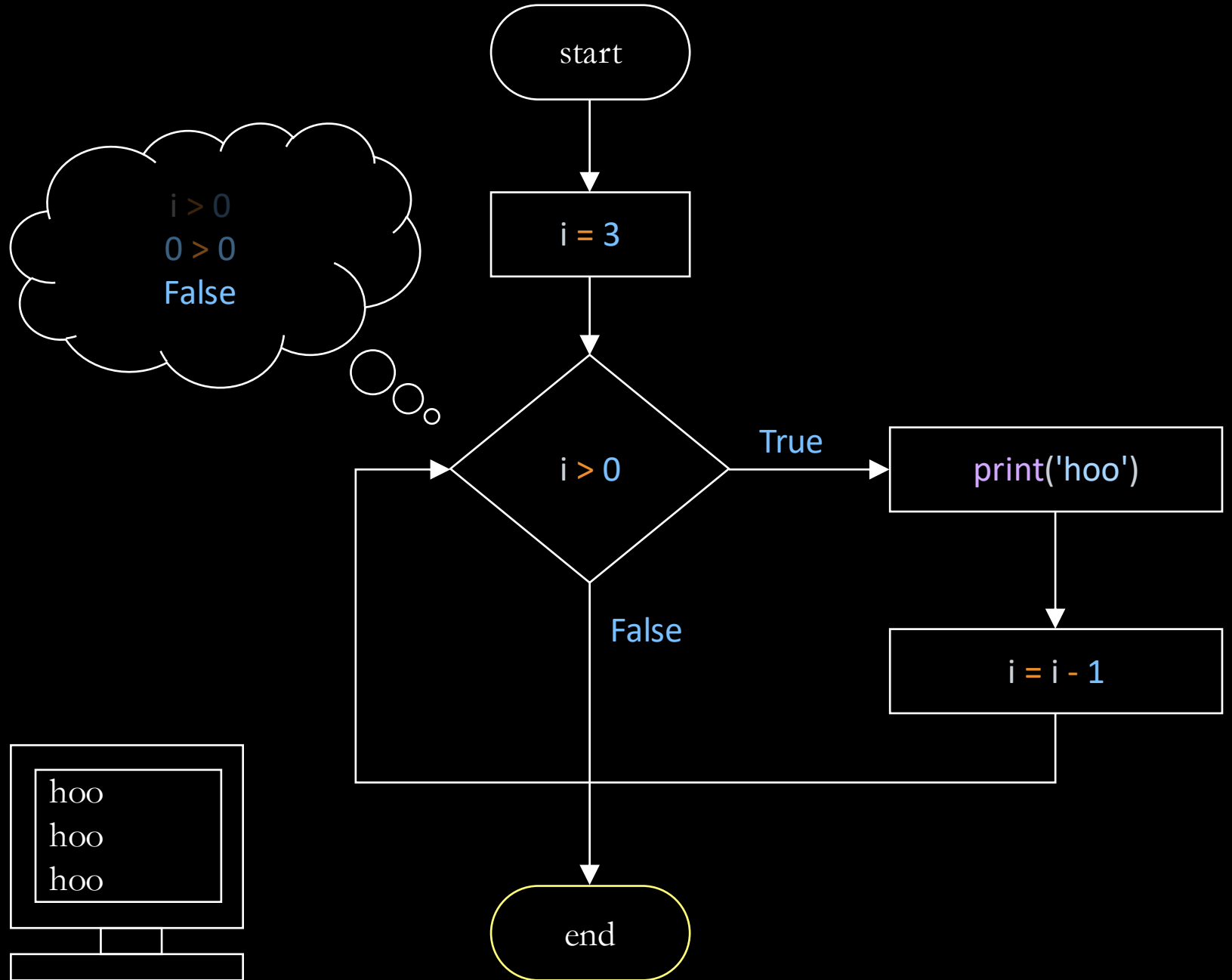
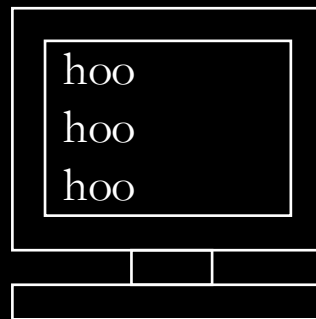
Variabler	Objekter
i	0
	2
	1



i = 3
while i > 0:
 print('hoo')
 i = i - 1



Variabler		Objekter
i	→	0
		2
		1



```
i = 3  
while i > 0:  
    print('hoo')  
    i = i - 1
```



```
i = 0  
while i < 3:  
    print('hoo')  
    i += 1
```

```
i = 0  
while i < 3:  
    print('hoo')  
    i += 1
```



```
for i in [0, 1, 2]:  
    print('hoo')
```

```
for i in [0, 1, 2]:  
    print('hoo')
```



```
for i in range(3):  
    print('hoo')
```



```
range(5)
```

```
0, 1, 2, 3, 4
```

```
range(3, 7)
```

```
3, 4, 5, 6
```

```
range(10, 20, 2)
```

```
10, 12, 14, 16, 18
```

```
range(20, 10, -2)
```

```
20, 18, 16, 14, 12
```

```
for i in range(3):  
    print('hoo')
```



```
for _ in range(3):  
    print('hoo')
```

for

while

kan vare evig



kortere å skrive



mindre bugs

break

avbryt løkken umiddelbart

continue

avbryt denne iterasjon av løkken

JOBBE I FLERE FILER

foo.py

```
from bar import add

num1 = int(input('num1 = '))
num2 = int(input('num2 = '))

ans = add(num1, num2)
print(f'Added nums: {ans}')
```

```
$ python foo.py
Testing add... OK
Testing multiply... OK
num1 = 2
num2 = 2
Added nums: 4
```

bar.py

```
def add(a, b):
    return a + b

print('Testing add... ', end='')
assert 4 == add(2, 2)
print('OK')

def multiply(a, b):
    return a * b

print('Testing multiply... ', end='')
assert 6 == multiply(2, 3)
print('OK')
```

```
IF __NAME__ == "__MAIN__":
```

foo.py

```
from bar import add

num1 = int(input('num1 = '))
num2 = int(input('num2 = '))

ans = add(num1, num2)
print(f'Added nums: {ans}')
```

```
$ python foo.py
num1 = 2
num2 = 2
Added nums: 4
```

bar.py

```
def add(a, b):
    return a + b

def test_add():
    print('Testing add... ', end='')
    assert 4 == add(2, 2)
    print('OK')

def multiply(a, b):
    return a * b

def test_multiply():
    print('Testing multiply... ', end='')
    assert 6 == multiply(2, 3)
    print('OK')

if __name__ == '__main__':
    test_add()
    test_multiply()
```