UNIVERSITETET I BERGEN

MINNET

INF100

VÅR 2024

Torstein Strømme

I DAG

- Recap
  - funksjoner

- Feil
- Minnet
- hvis tid: tyvstart på løkker

Oppgave: skriv en funksjon som regner ut volumet av en boks

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

```
IndentionError
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
```

RECAP: FUNKSJONER

Oppgave: skriv en funksjon som sjekker om to tall er nesten like

- Hvordan er input representert?
- Hva er det matematiske grunnlaget?
- Skal funksjonen ha en effekt eller en returverdi?
  - Effekt: f. eks. noe som vises på skjermen
  - Returverdi: hvis vi vil bruke verdien til noe mer enn å se på den
- Hvordan tester vi om funksjonen fungerer?

bestemmer parametrene til funksjonen

ellers får vi logiske feil

print vs return

assert

RECAP: FUNKSJONER

Oppgave: skriv en funksjon som gir oss avstanden mellom to punkter

- Hvordan er input representert?
- Hva er det matematiske grunnlaget?
- Skal funksjonen ha en effekt eller en returverdi?
  - Effekt: f. eks. noe som vises på skjermen
  - Returverdi: hvis vi vil bruke verdien til noe mer enn å se på den
- Hvordan tester vi om funksjonen fungerer?

bestemmer parametrene til funksjonen

ellers får vi logiske feil

print vs return

assert

KODESPORING: FUNKSJONER

```
def charlie(p, q):
    r = p - q
    s = delta(r, p)
    s = delta(s, q)
    return s
def delta(t, u):
    v = t + u
    return v - 1
print(charlie(5, 2))
```

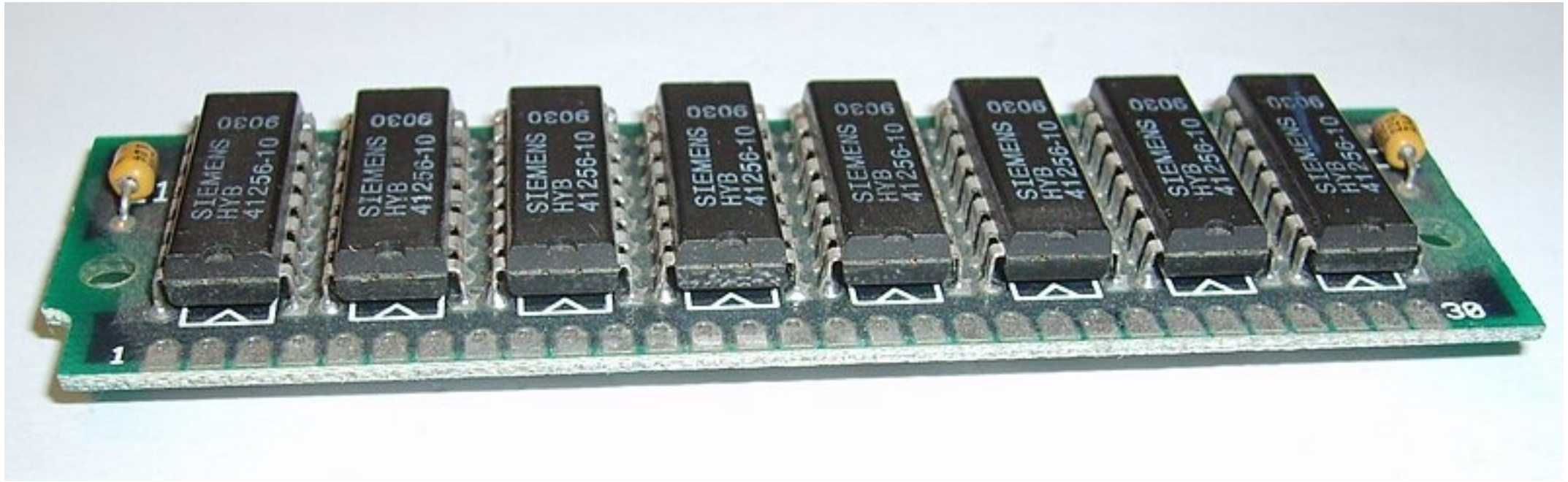
Hva skriver dette programmet ut? (hvis programmet krasjer, skriv kun 'Error')

RECAP: BETINGELSER

```
def foxtrot(x):
    if x >= 10:
        return 10
    else:
        x += 10
    if x > 10:
        if x % 2 == 0:
            x += 1
        elif x >= 15:
            x -= 1
    else:
        return 42

    return x - 10
```

print(foxtrot(0))	
print(foxtrot(2))	
print(foxtrot(3))	
print(foxtrot(5))	
print(foxtrot(foxtrot(2))	

MINNET



Random Access Memory (RAM)

MINNET

- En lagringsplass for 0'er og 1'ere

én **bit** med informasjon

8 bit = 1 Byte

4 GB RAM tilsvarer 32 000 000 000 bits

MINNET

- En lagringsplass for 0'er og 1'ere

én **bit** med informasjon

0 1 2 3 4 …

hver bit har en **adresse**

MINNET

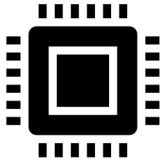
- En del av minnet er forbehold **objekter**
  - Et **objekt** er et område i minnet som «hører sammen»
  - Objekter har en minneadresse (id), en type (klasse) og en verdi

MINNET

- En variabel er en navngitt referanse til et objekt
  - «referanse til et objekt» = egentlig bare en minneadresse*

*en hvit løgn du kan leve fint med helt frem til du skal konstruere dine egne programmeringsspråk eller operativsystemer

NÅR PYTHON KJØRER



Prosessor

Programpeker
(peker på neste setning)

Kildekoden

VARIABLER

VERDIER

```
x
course_code
```

```
5
42
False
"INF100"
```

VARIABLER

```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```

VARIABLER

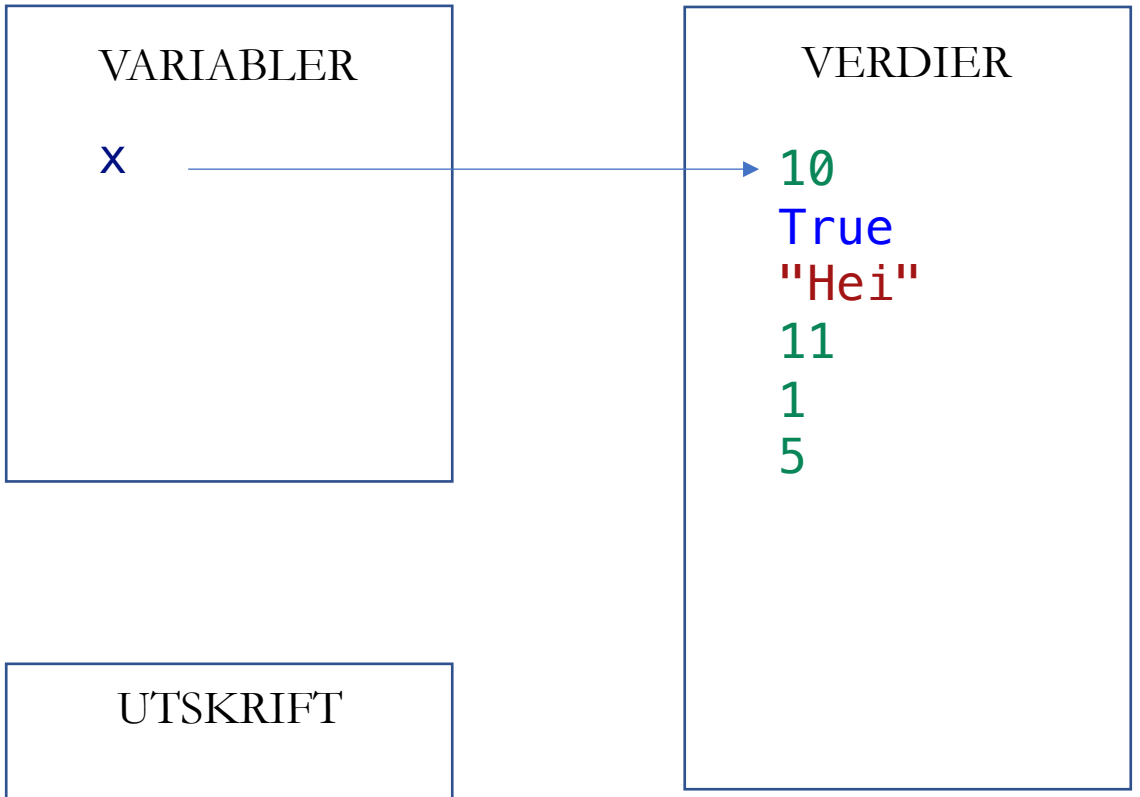
UTSKRIFT

VERDIER

```
10
True
"Hei"
11
1
5
```

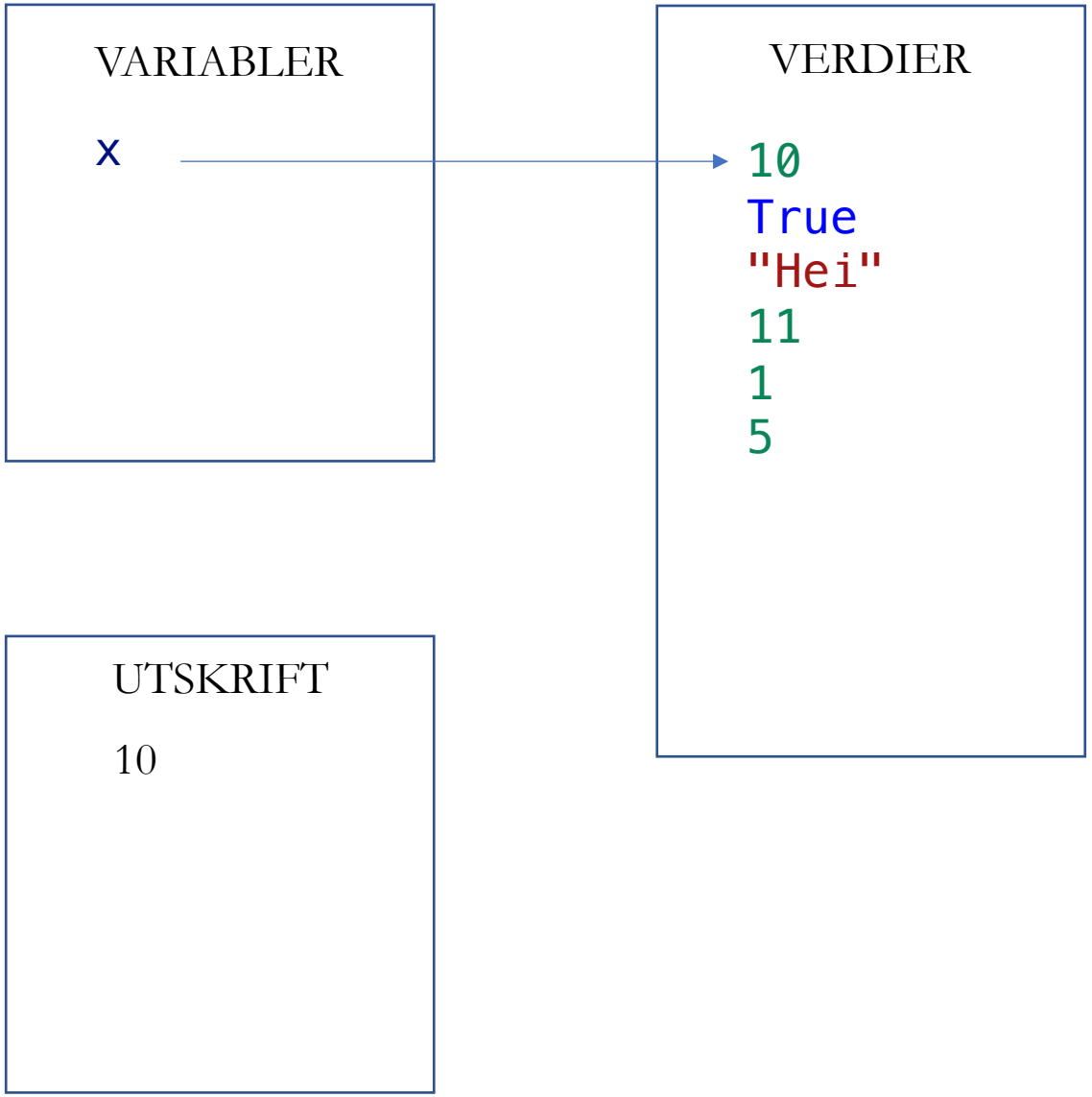
VARIABLER

```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```



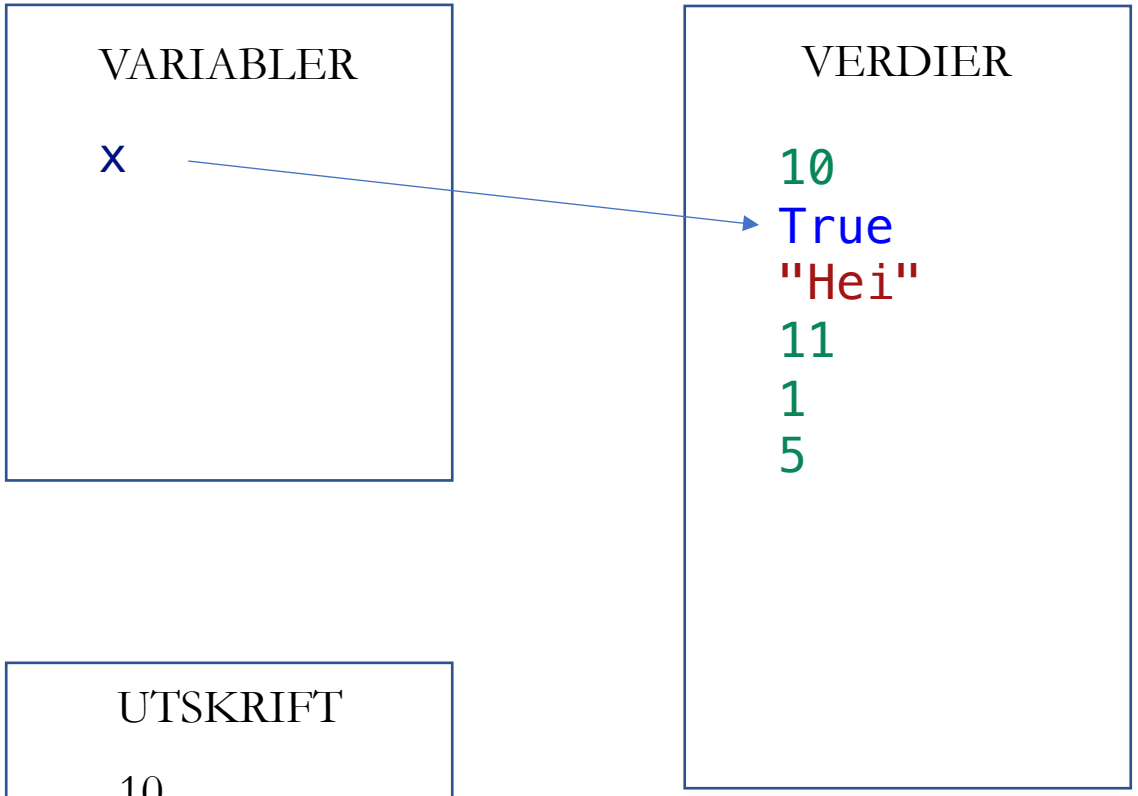
VARIABLER

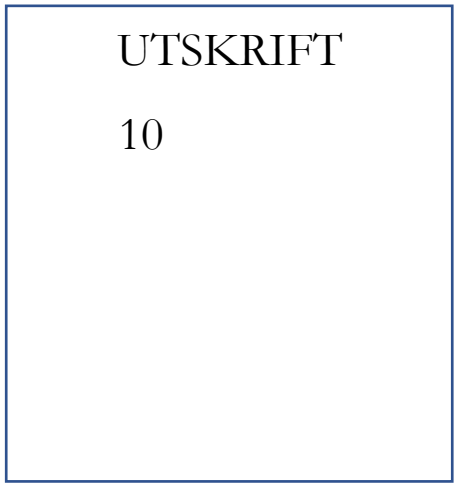
```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```



VARIABLER

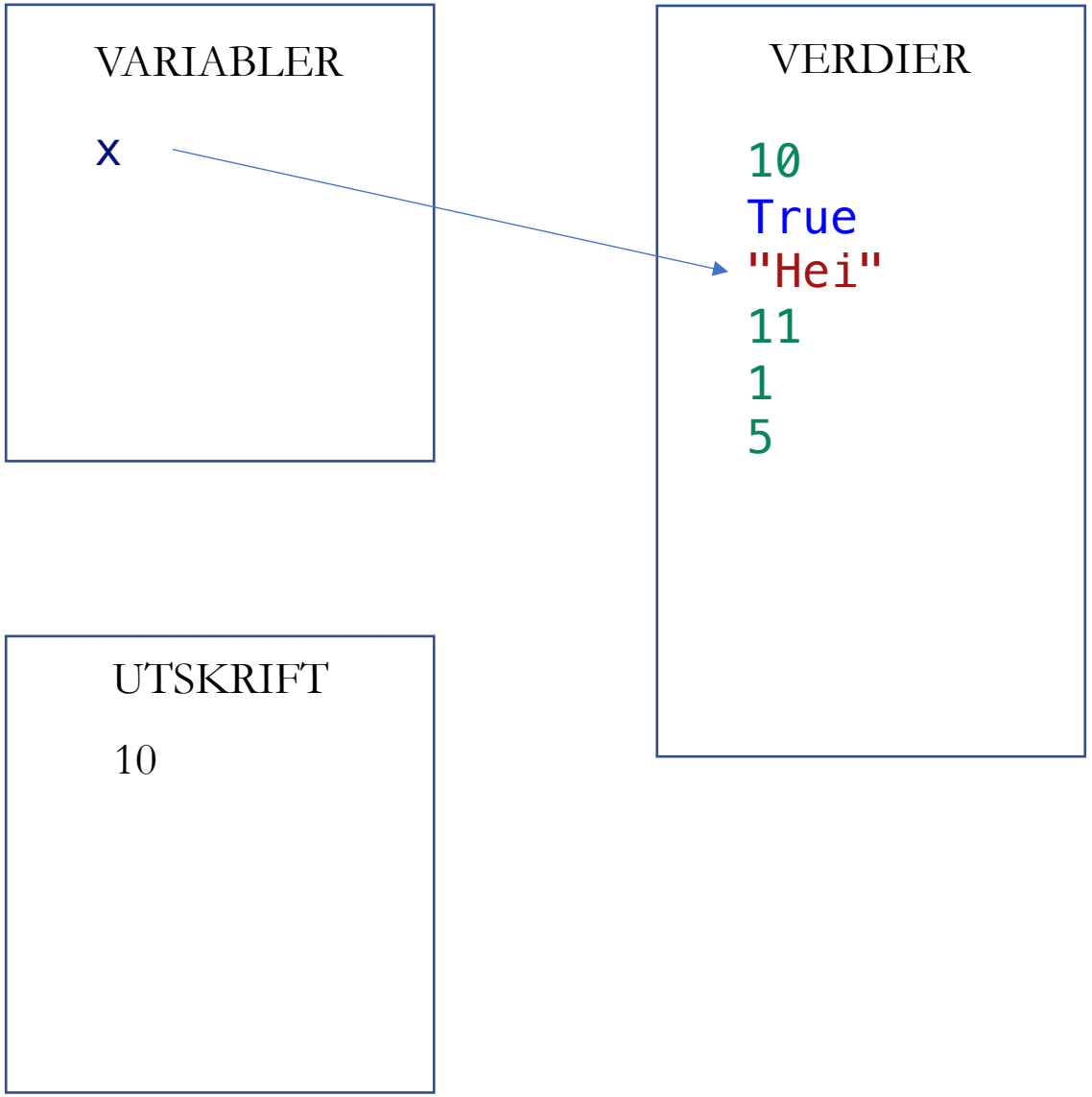
```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```





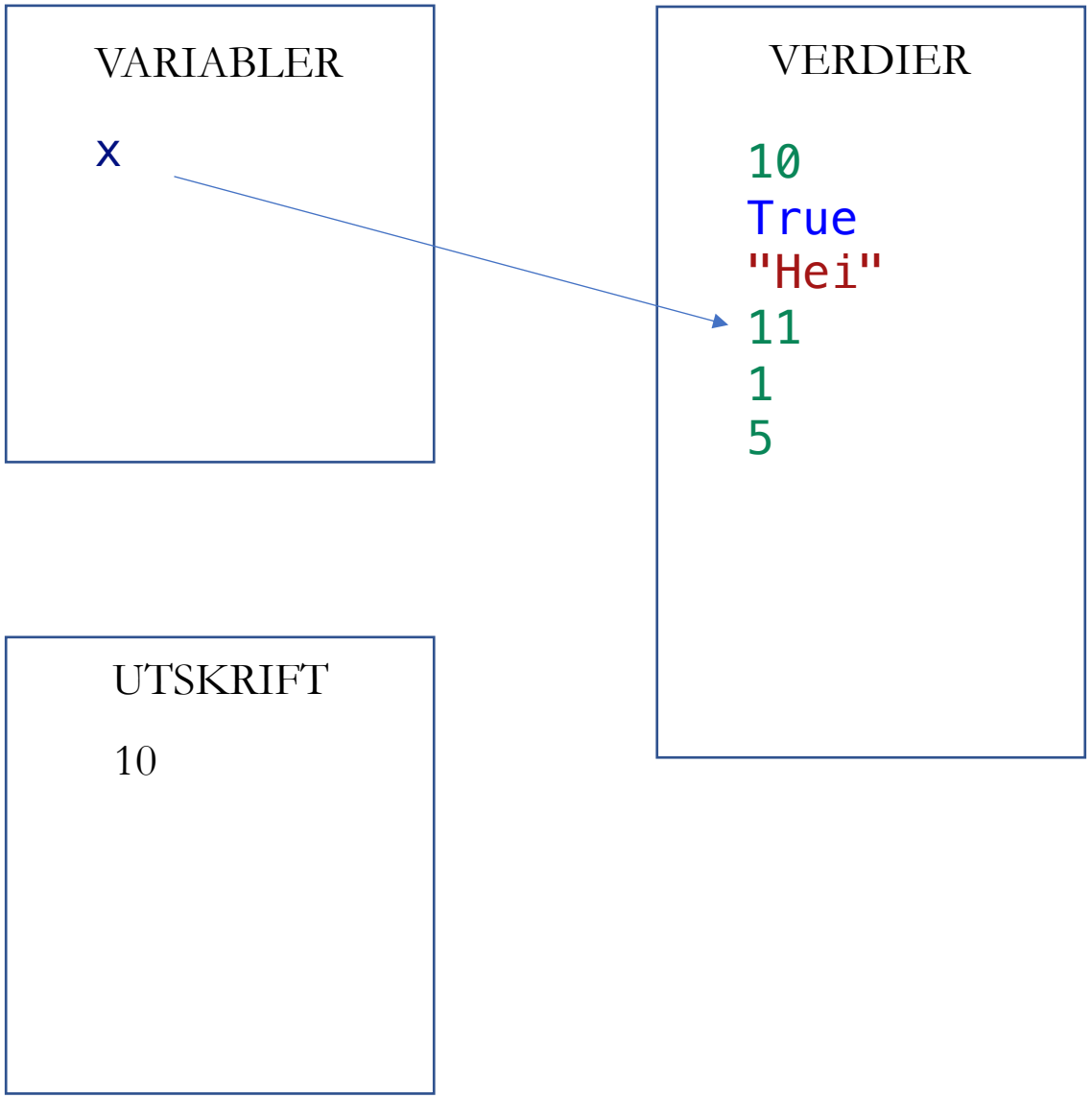
VARIABLER

```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```



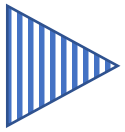
VARIABLER

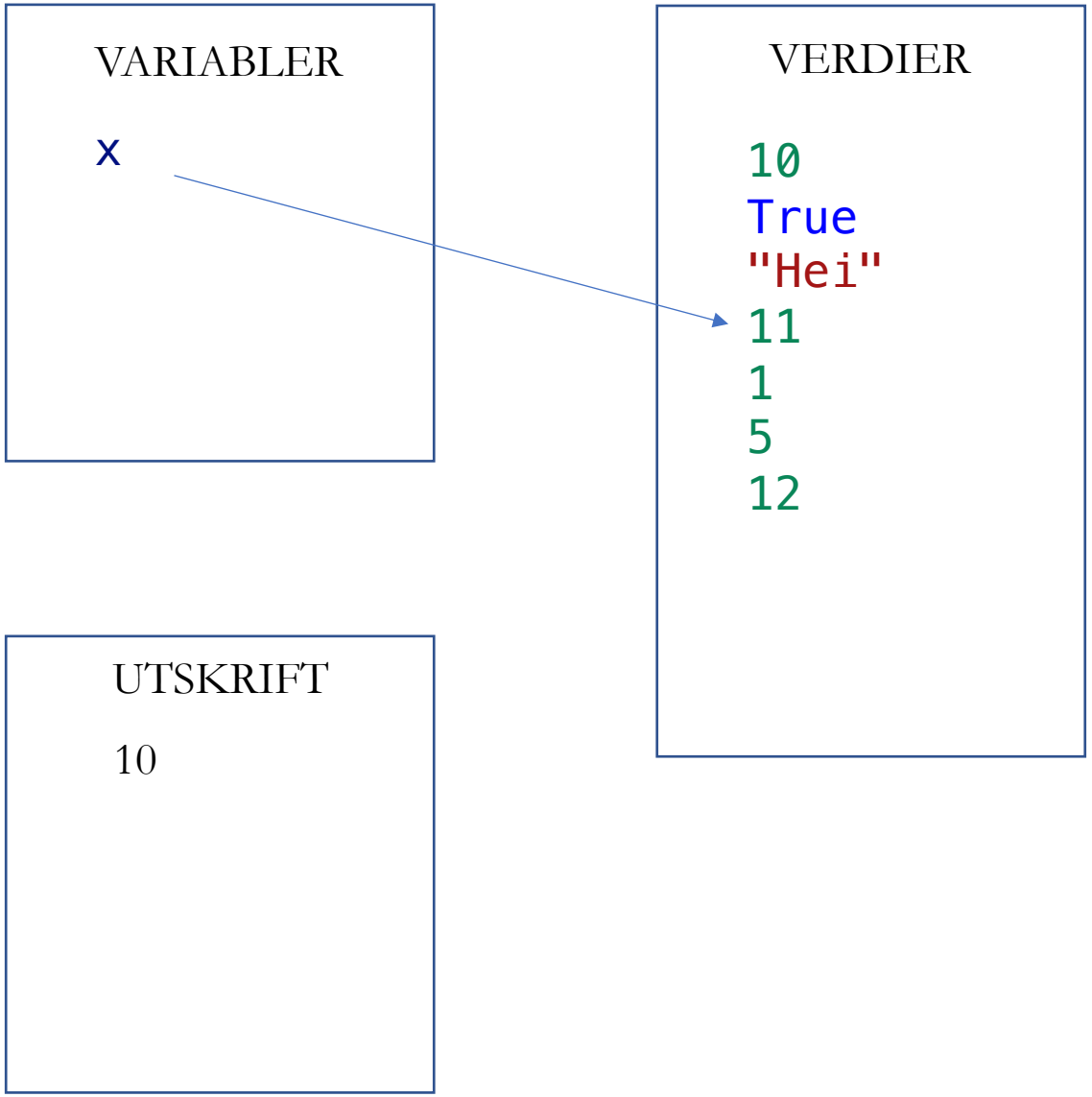
```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```



VARIABLER

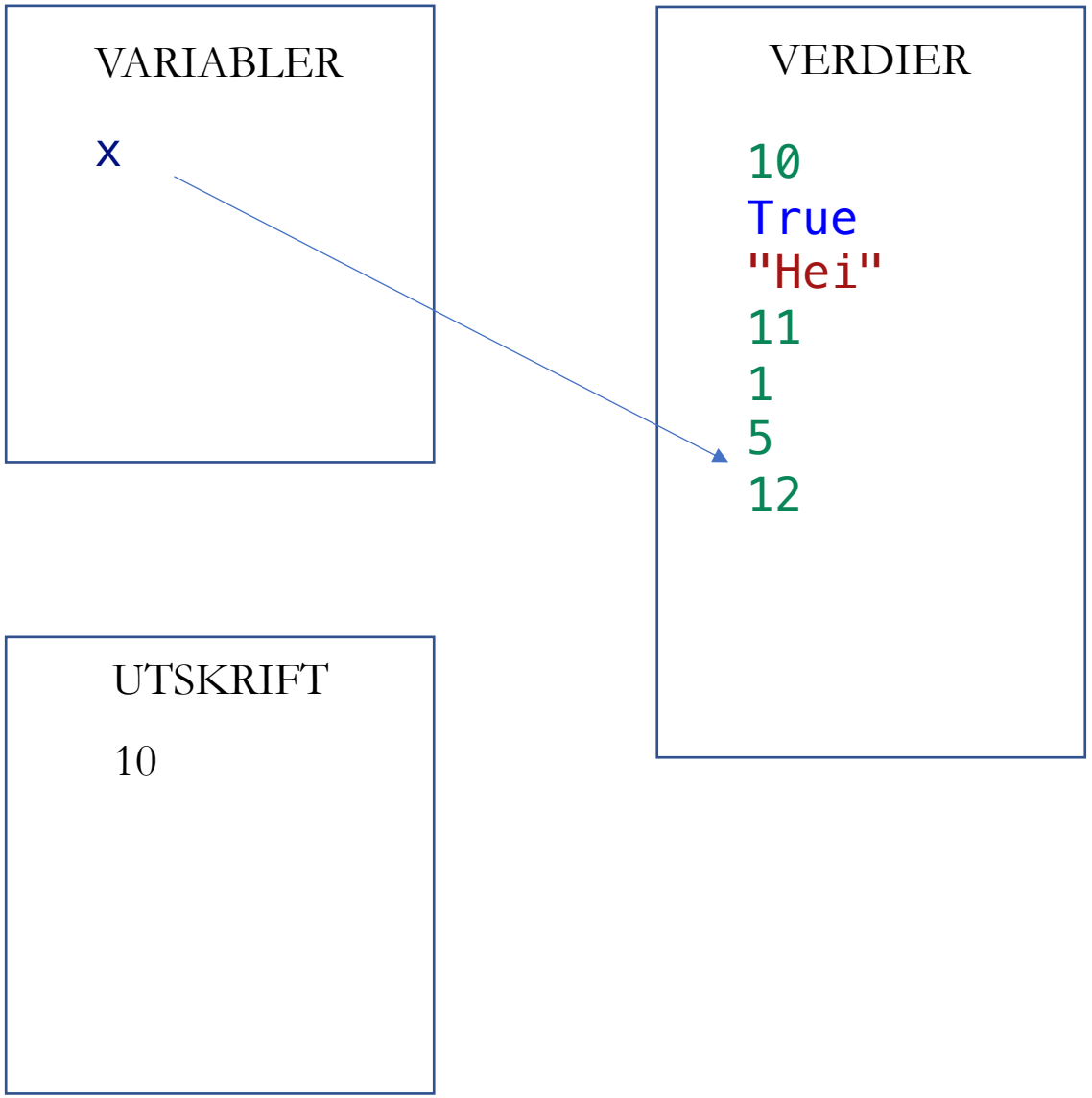
```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```





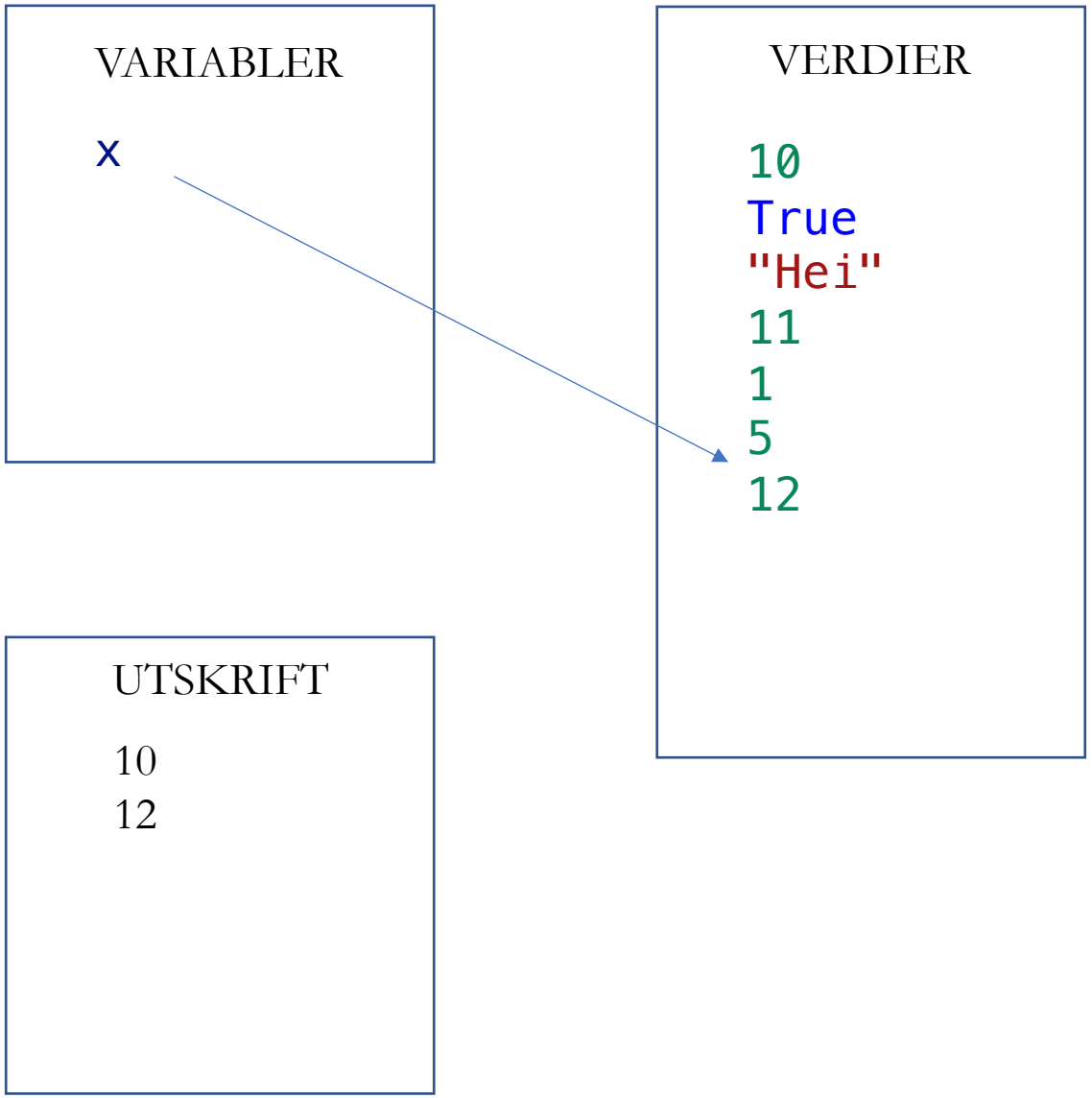
VARIABLER

```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```



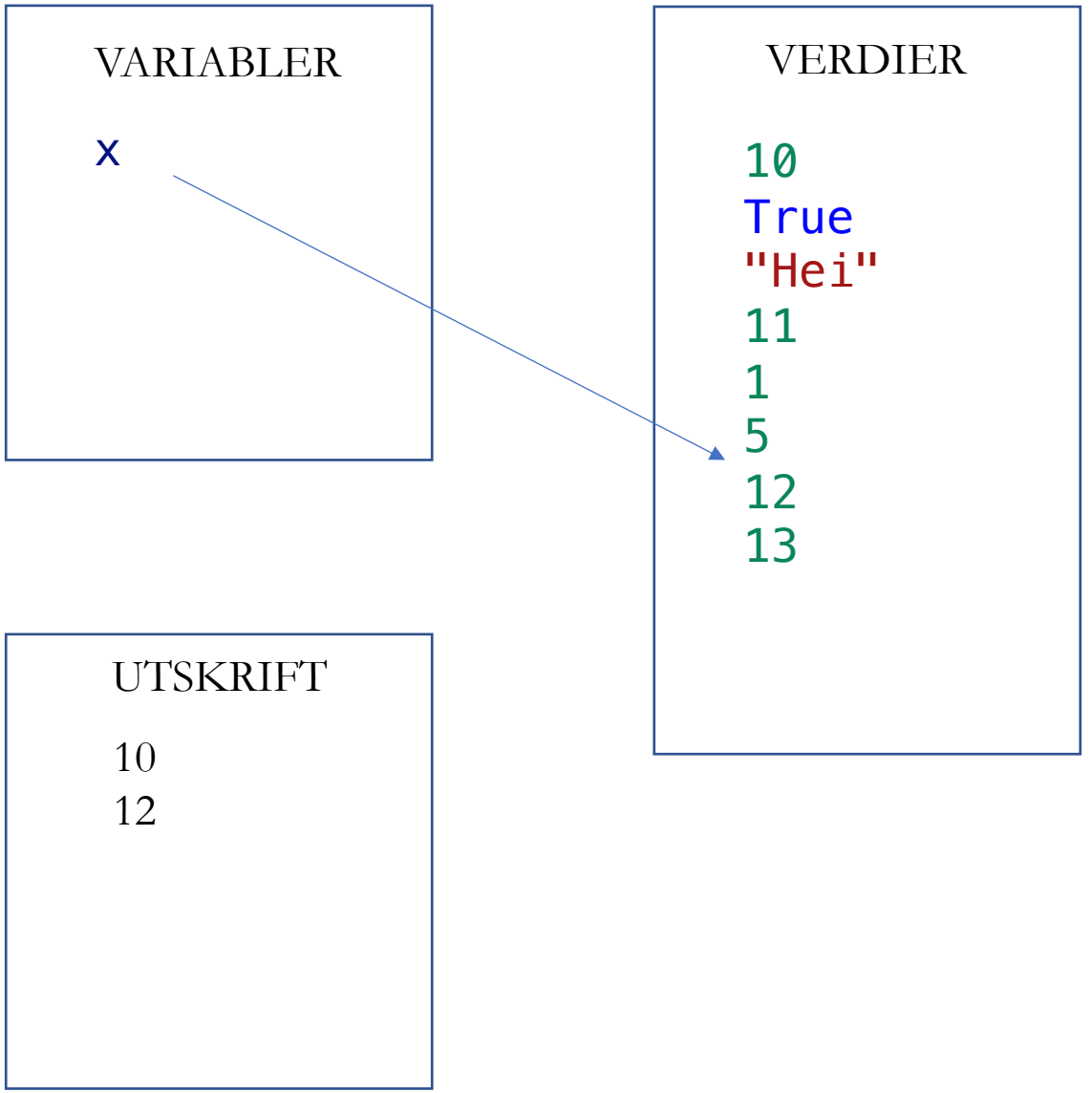
VARIABLER

```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```



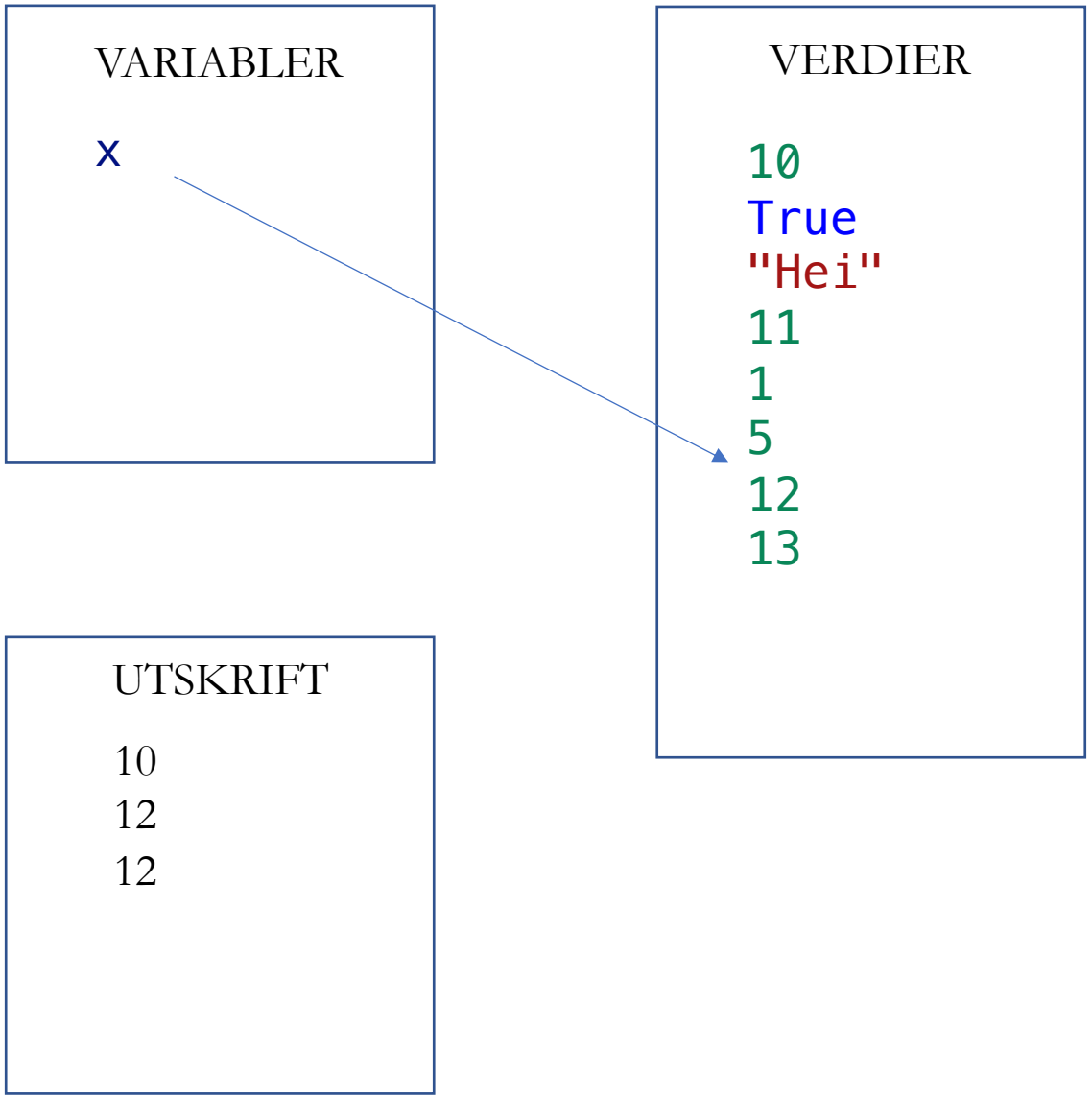
VARIABLER

```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```



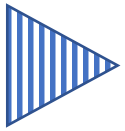
VARIABLER

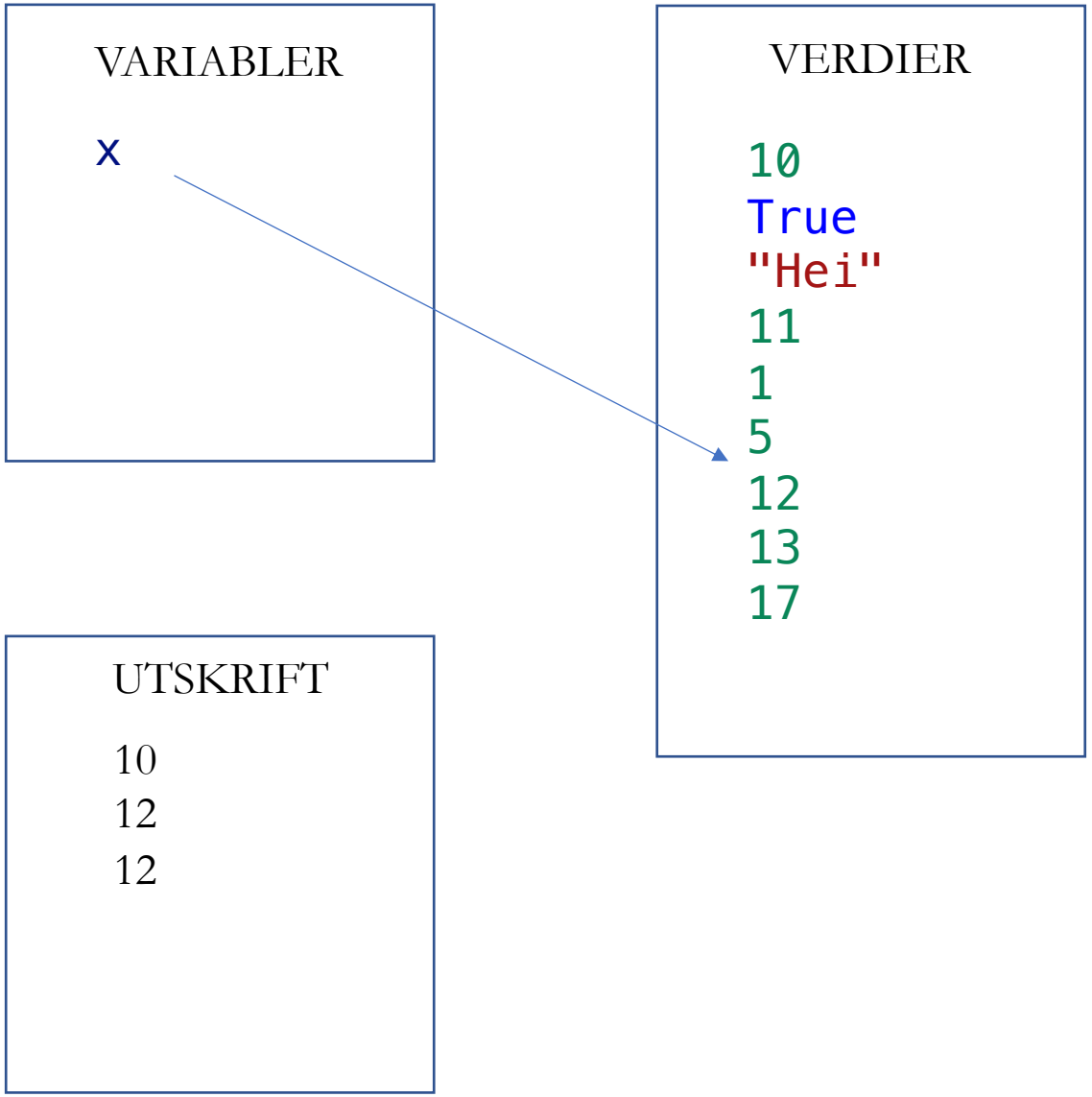
```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```



VARIABLER

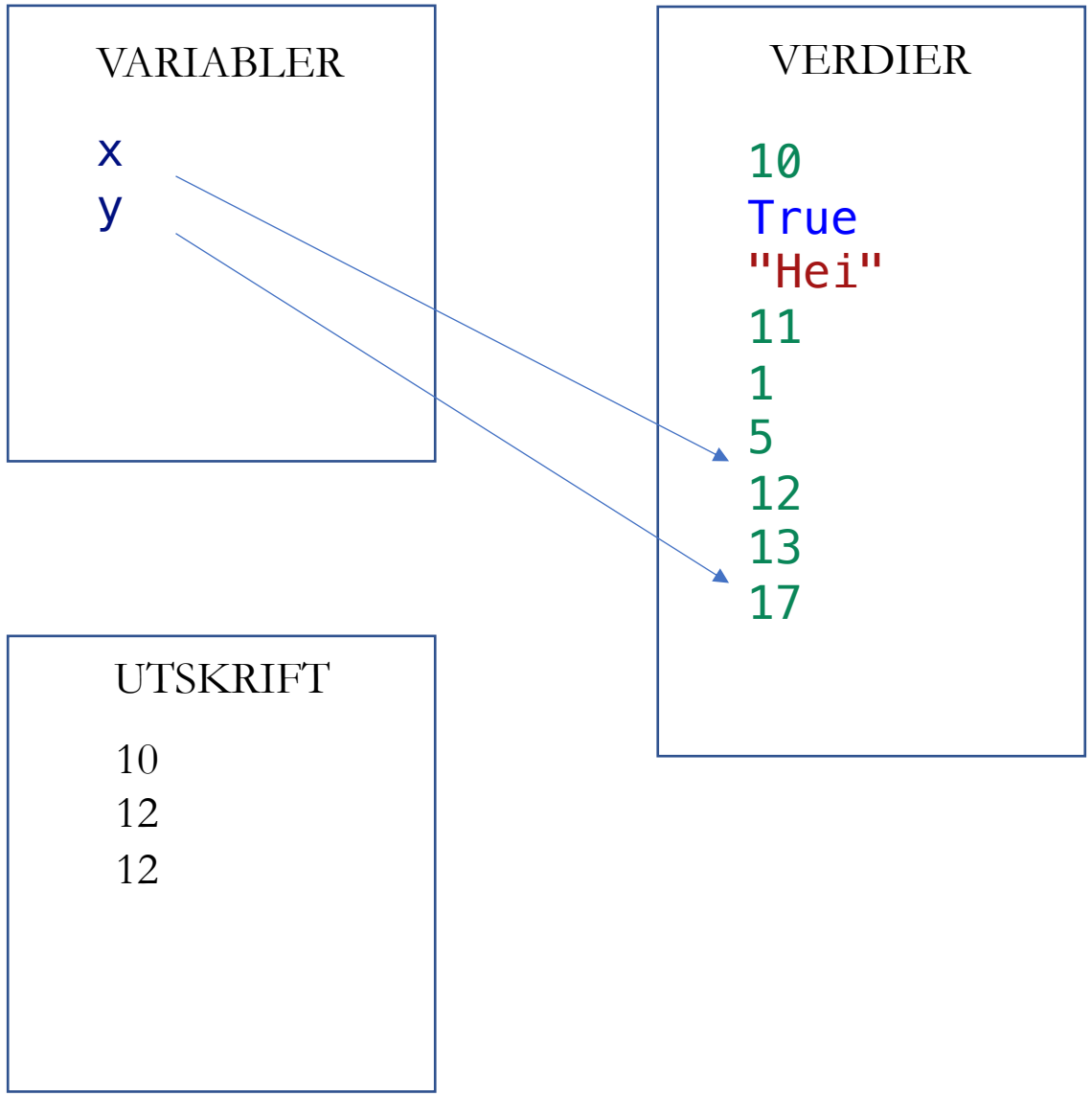
```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```





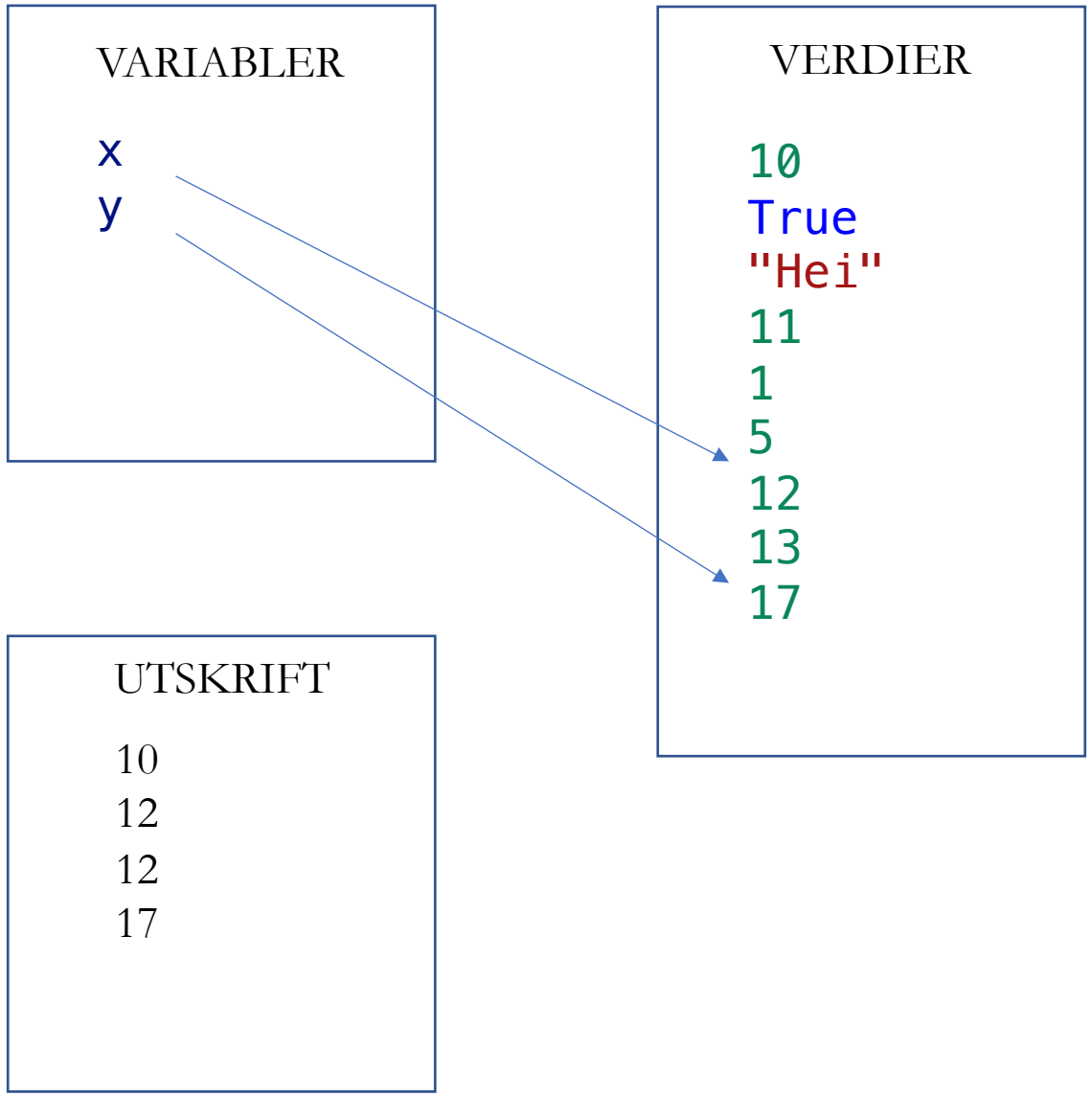
VARIABLER

```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```



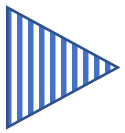
VARIABLER

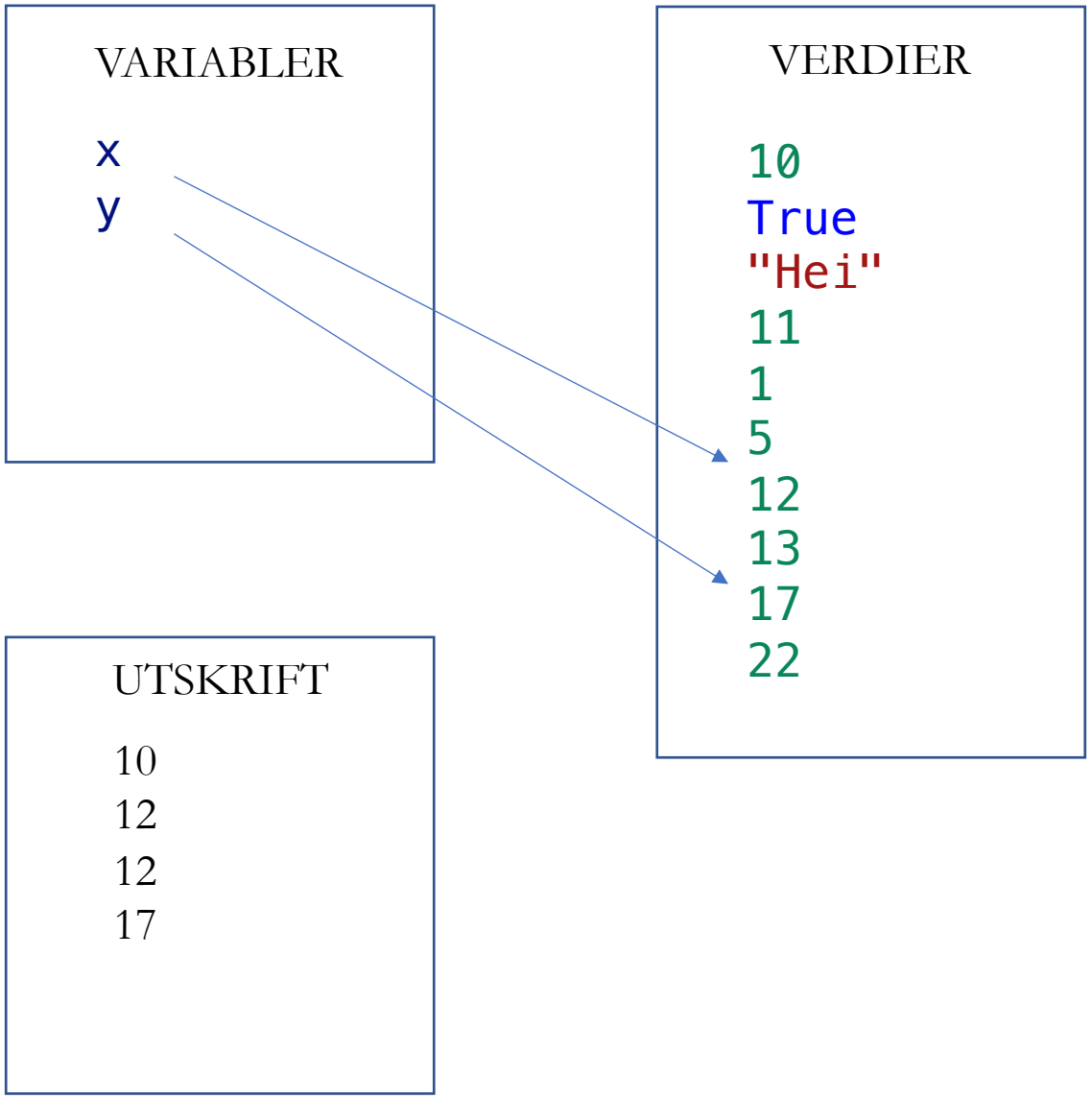
```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```



VARIABLER

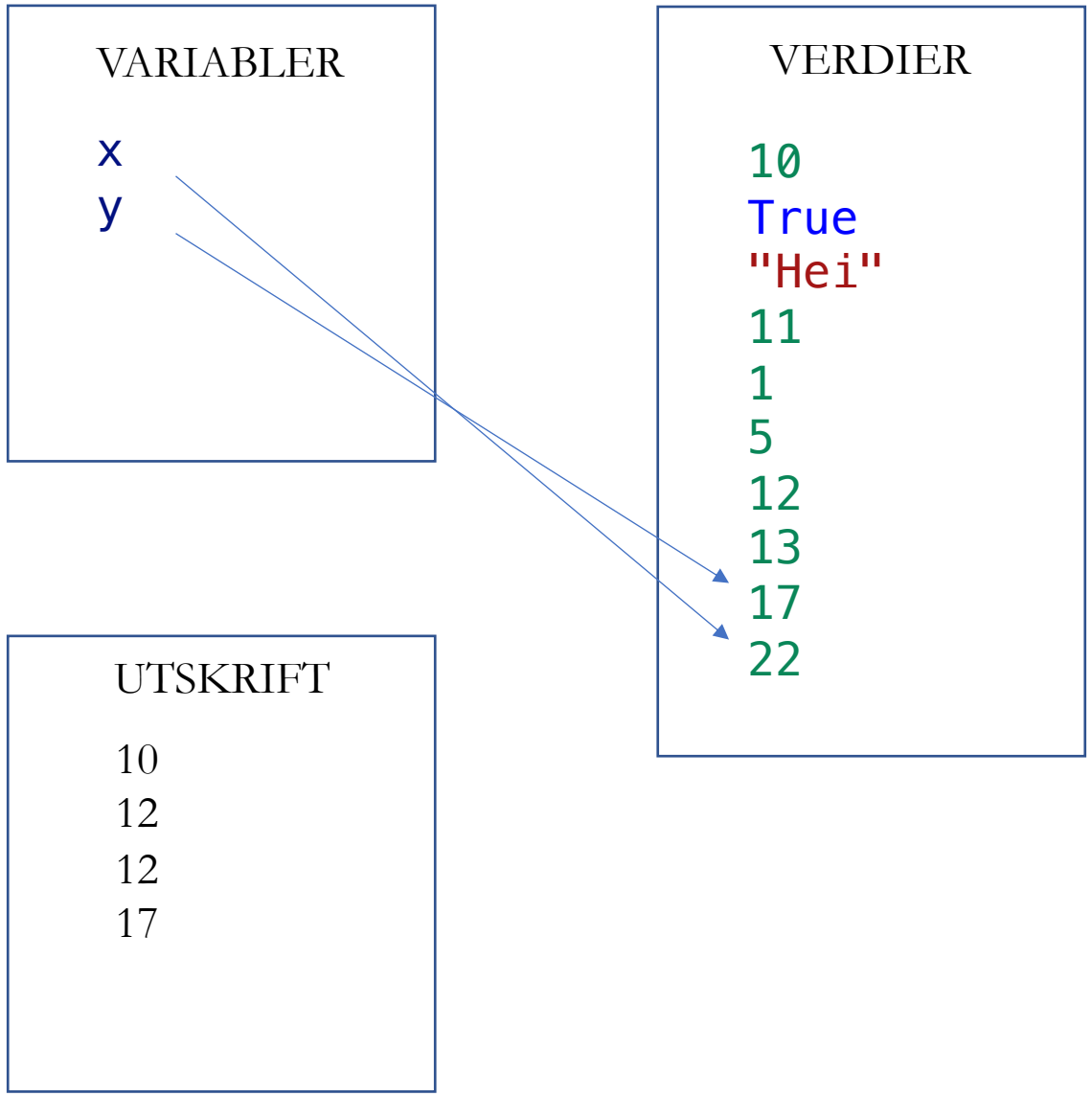
```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```





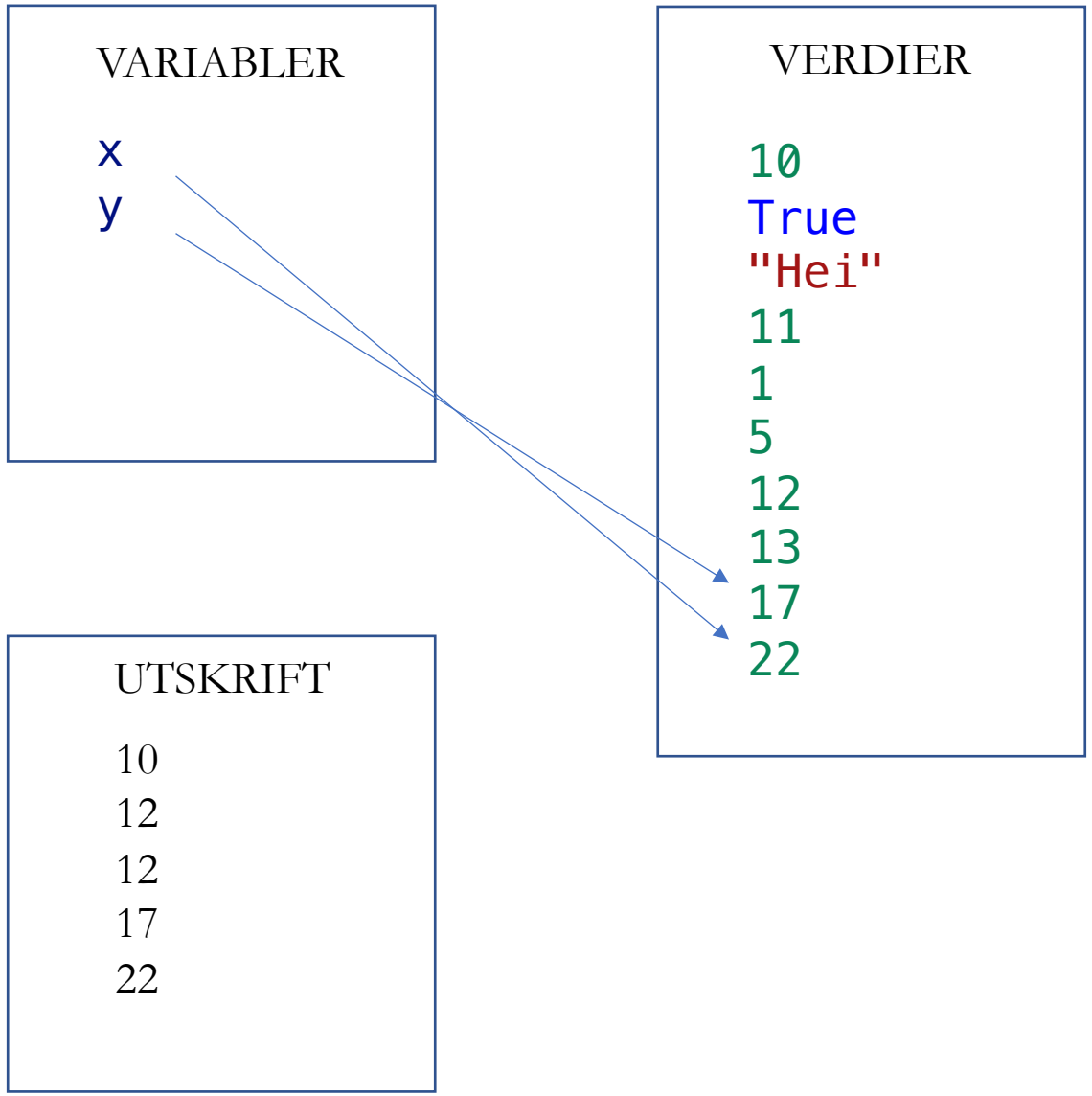
VARIABLER

```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```



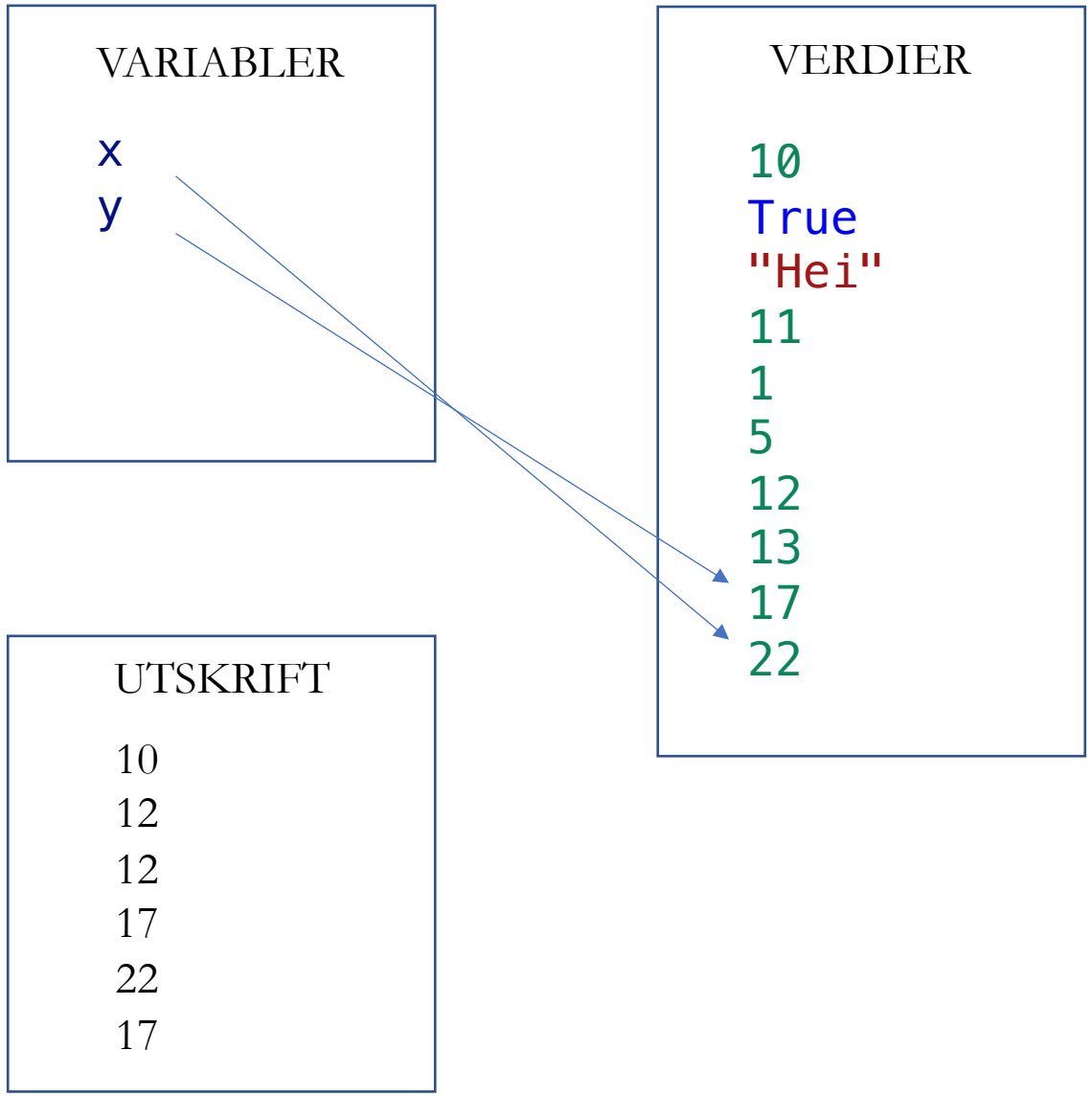
VARIABLER

```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```



VARIABLER

```
x = 10
print(x)

x = True
x = "Hei"
x = 11
x = x + 1
print(x)
x + 1
print(x)

y = x + 5
print(y)

x = x + 10
print(x)
print(y)
```



LEVENDE KODESPORING

```
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

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```

VARIABLER

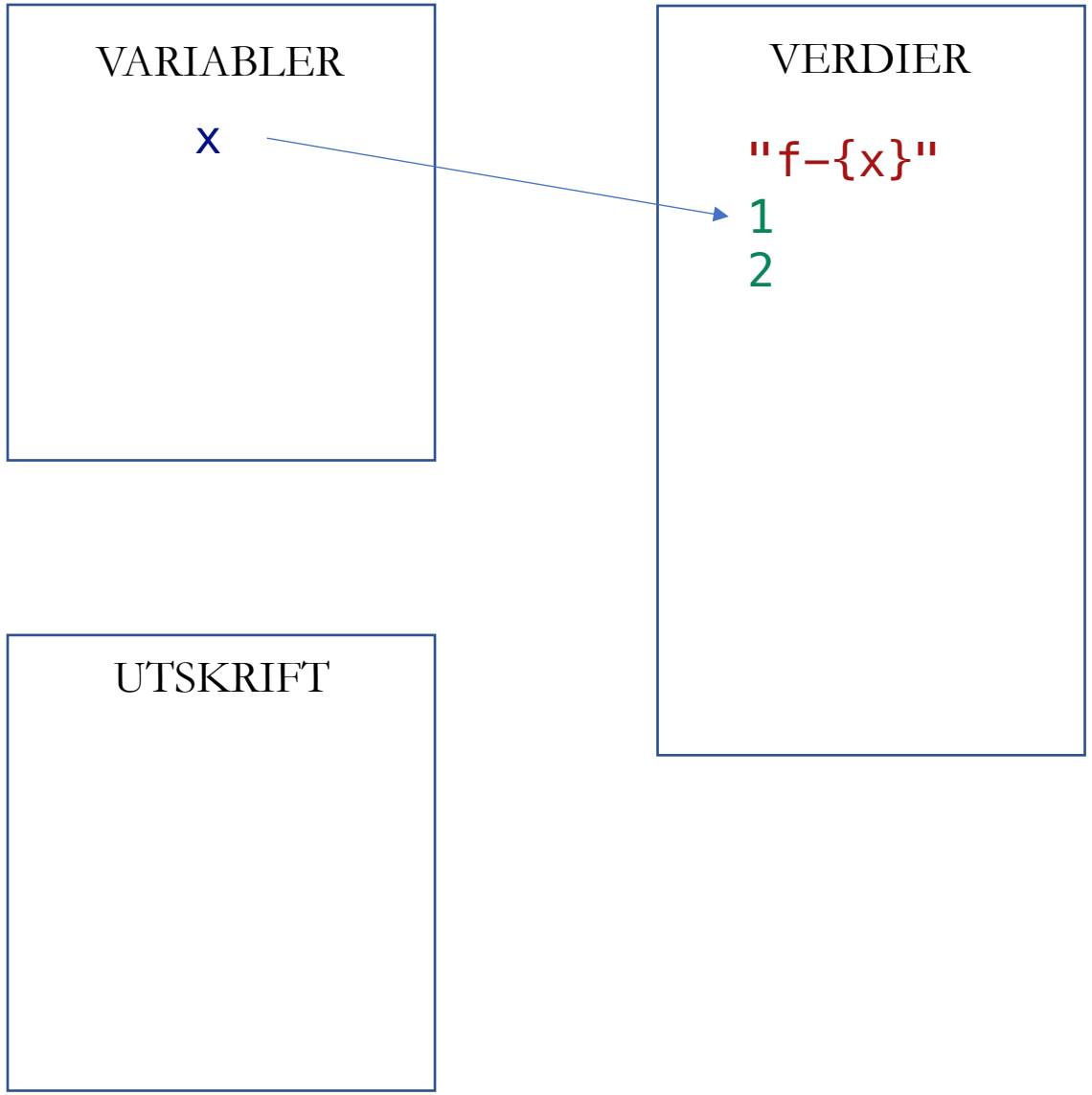
VERDIER

"f-{x}"
1
2

UTSKRIFT

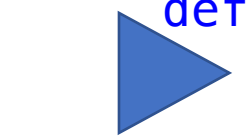
KODESPORING

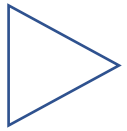
```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```

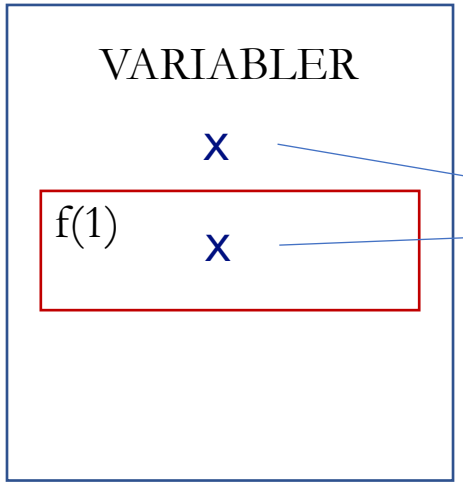


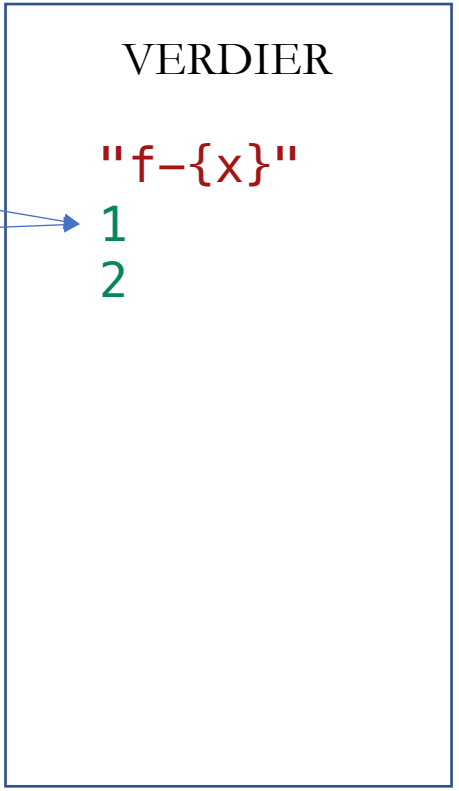
KODESPORING

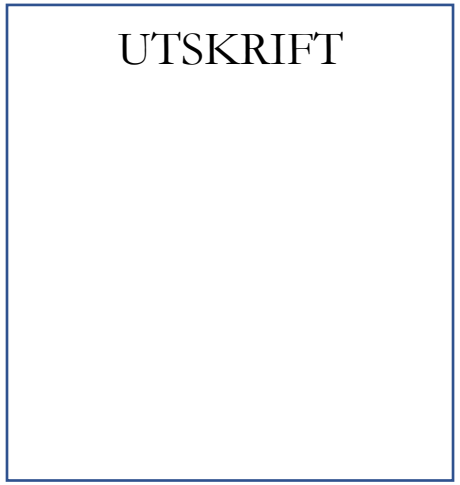
```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```





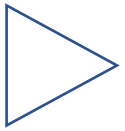


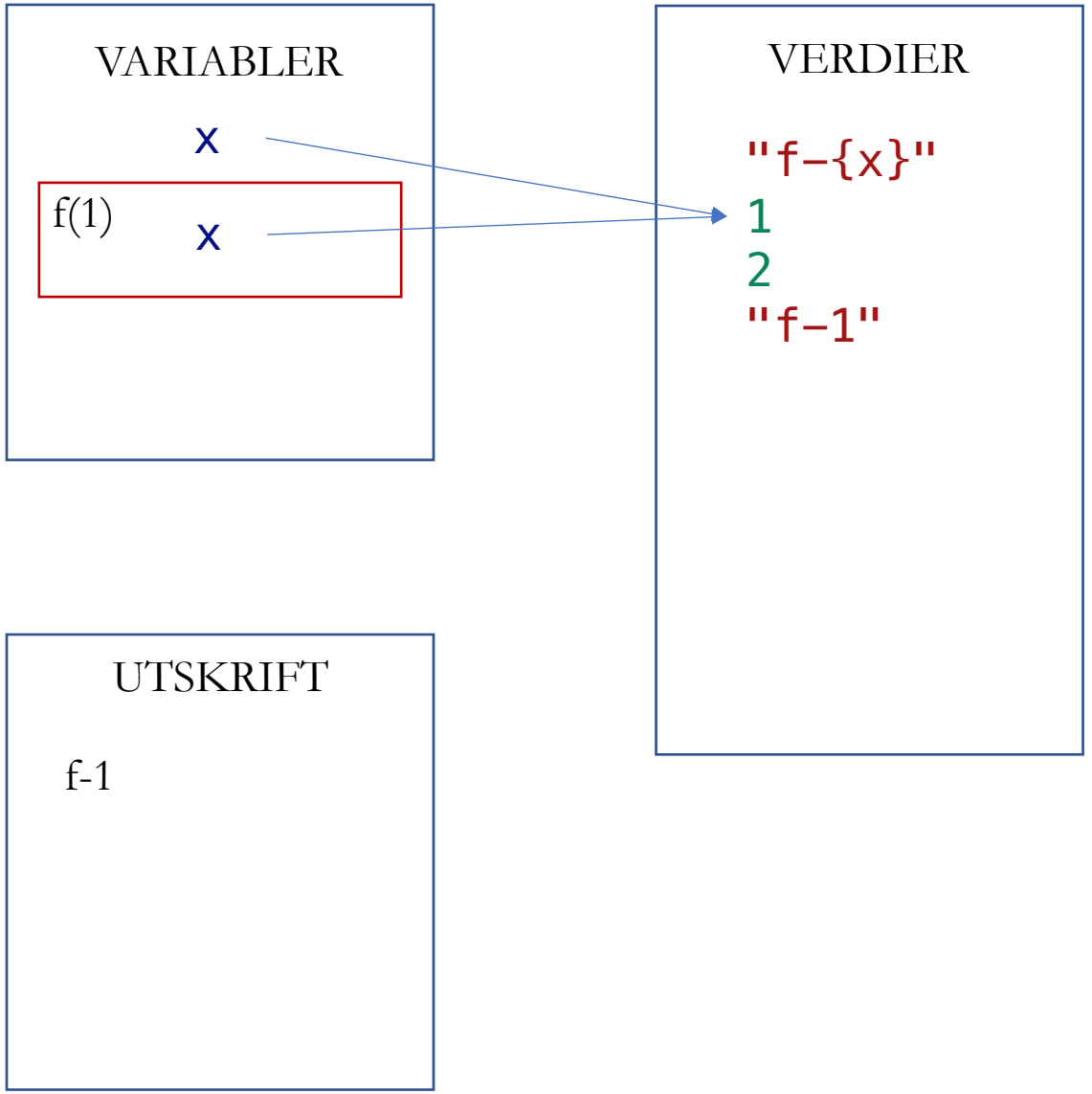




KODESPORING

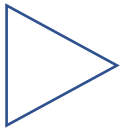
```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```

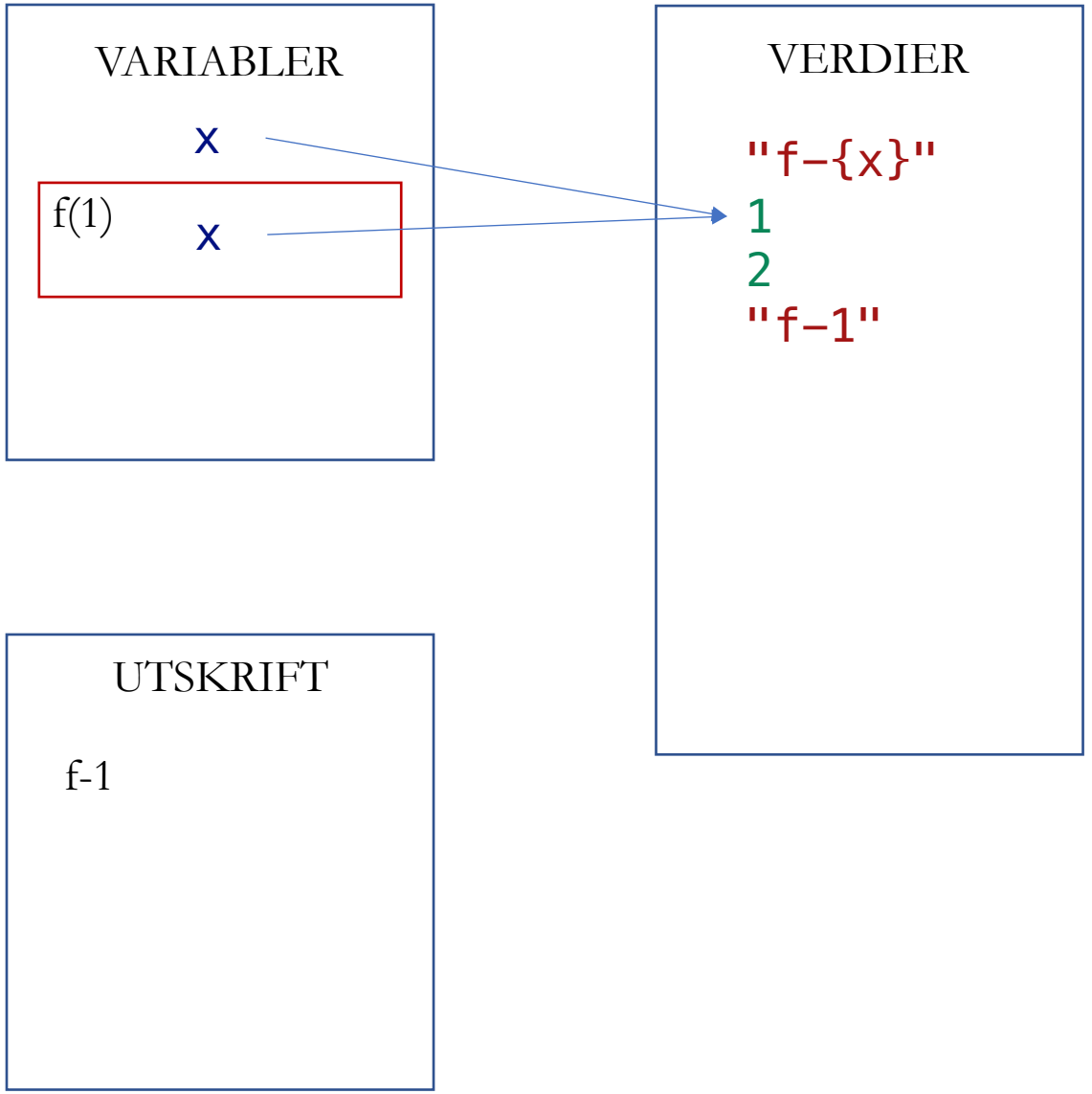




KODESPORING

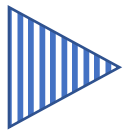
```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```

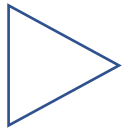


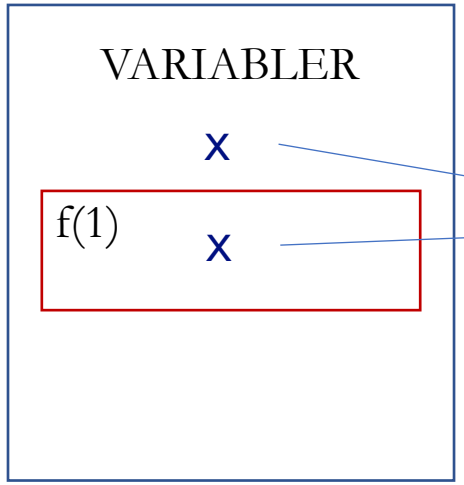


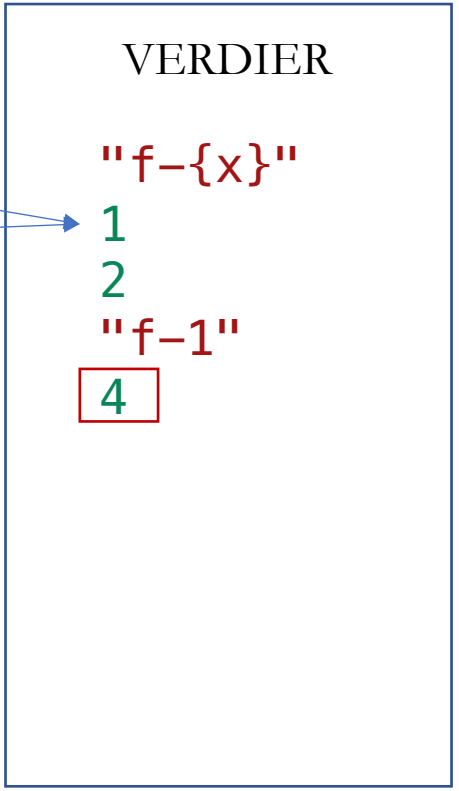
KODESPORING

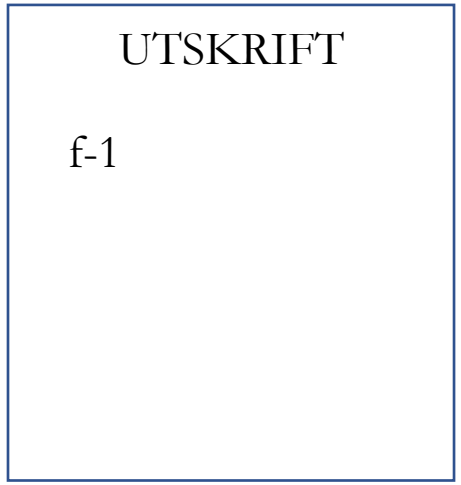
```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```





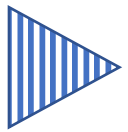


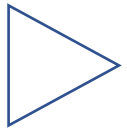


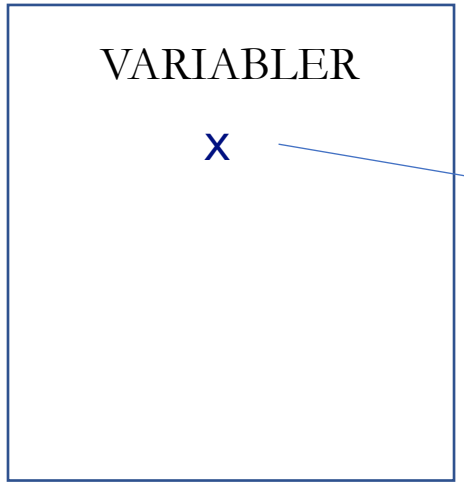


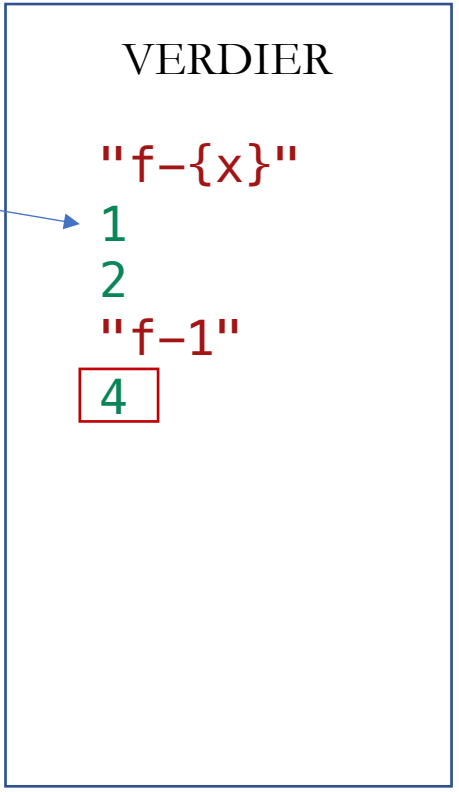
KODESPORING

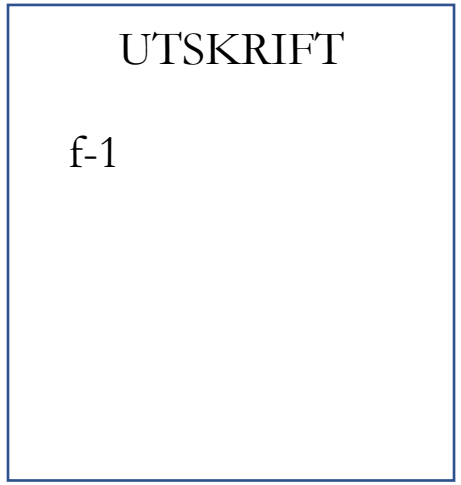
```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```





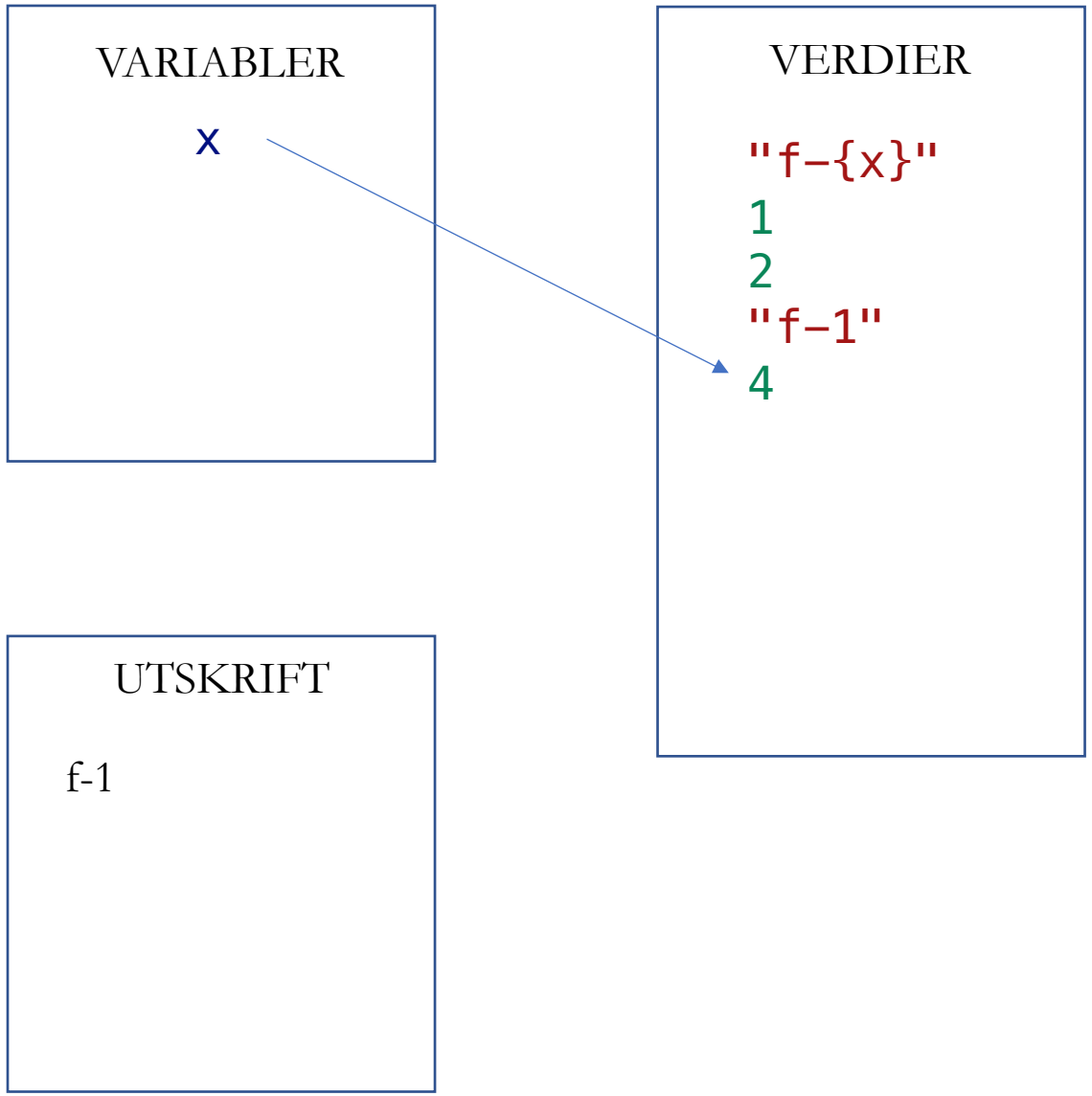






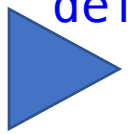
KODESPORING

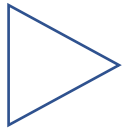
```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```

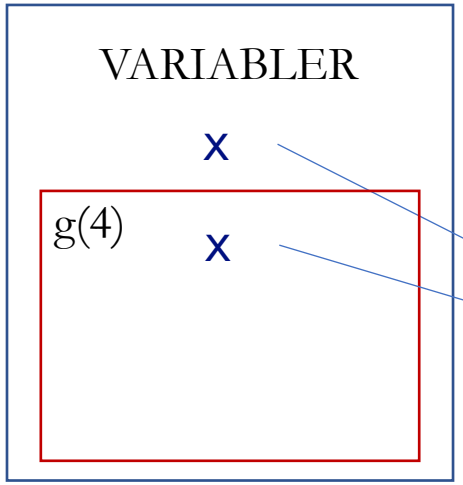


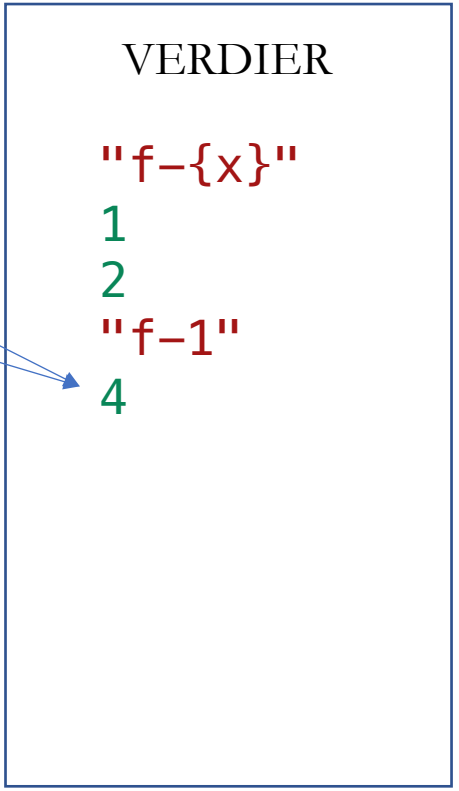
KODESPORING

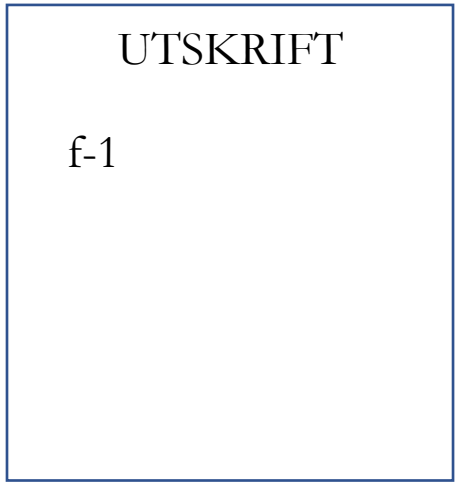
```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```





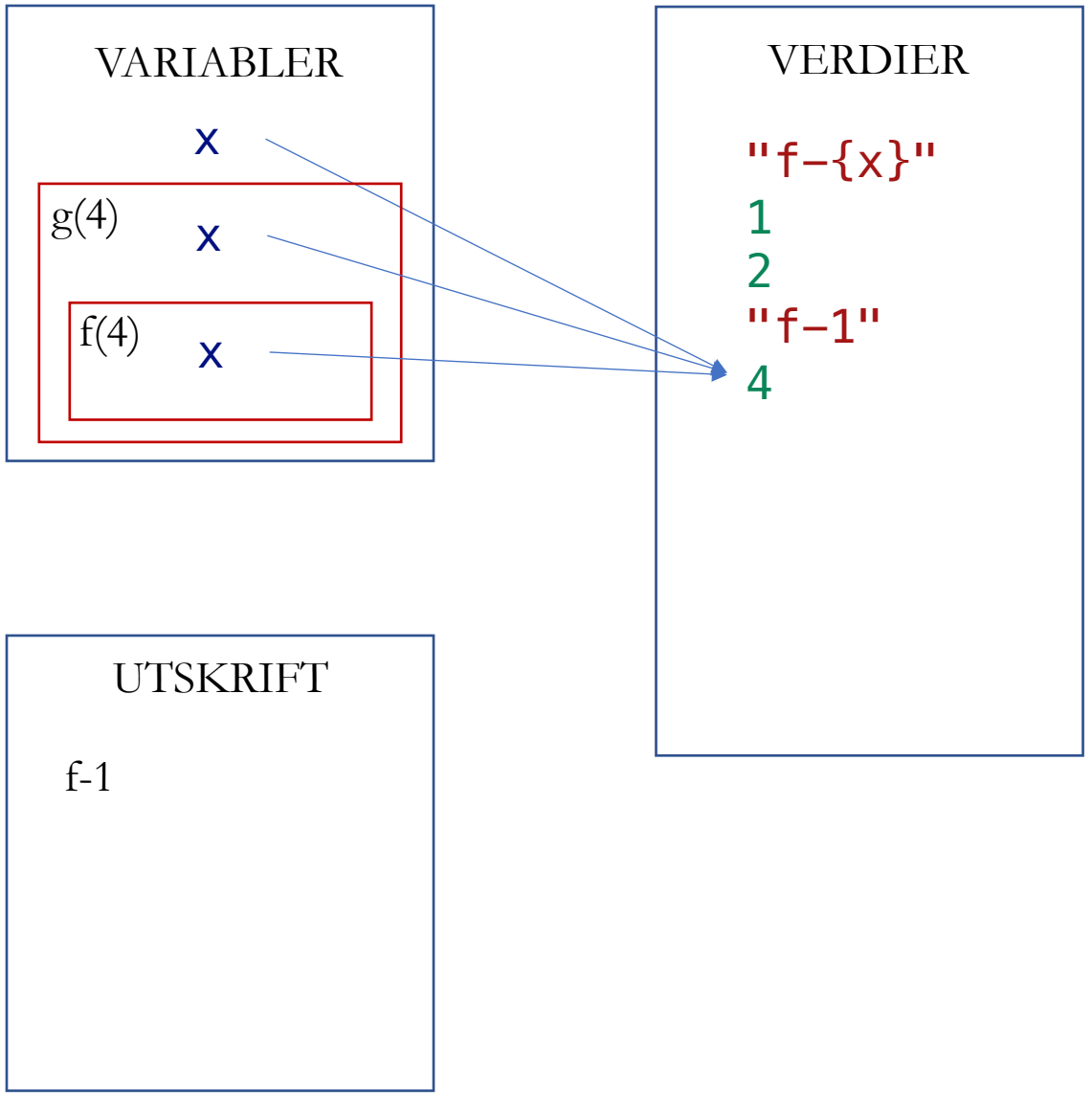






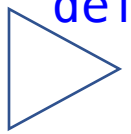
KODESPORING

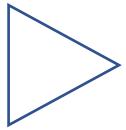
```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```

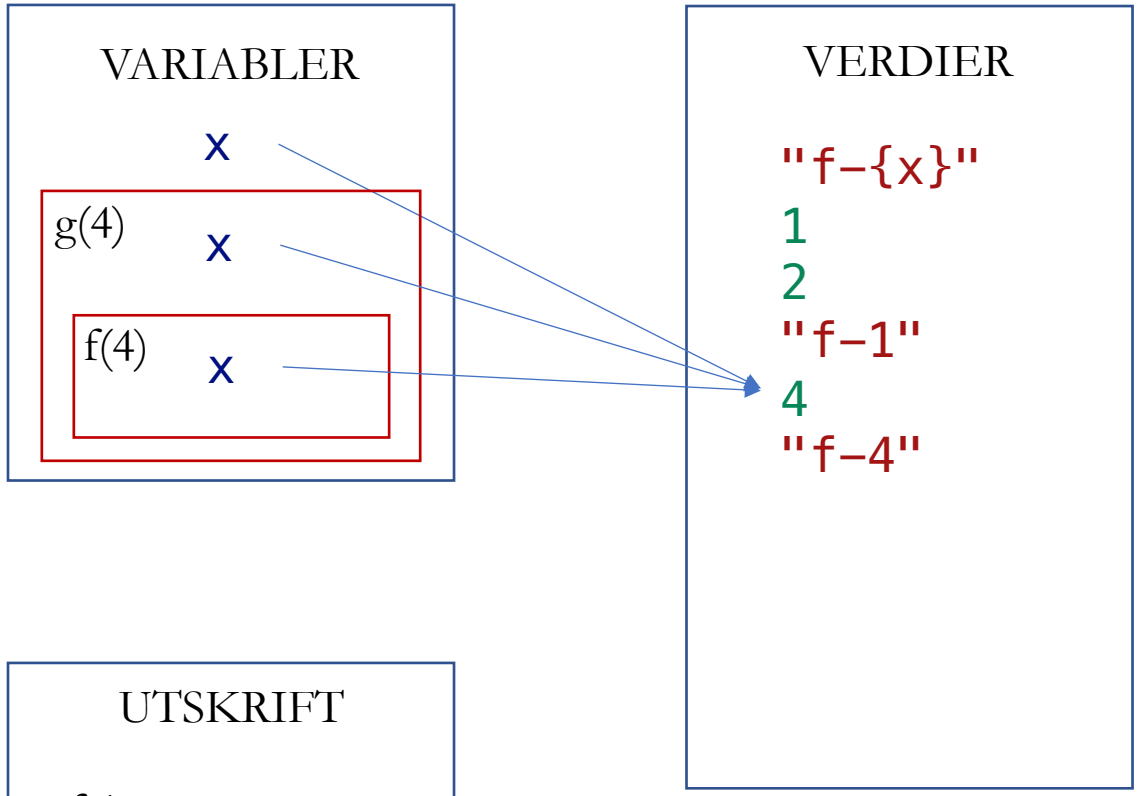


KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```





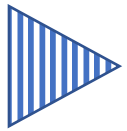


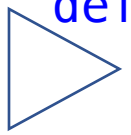
UTSKRIFT

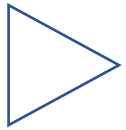
f-1
f-4

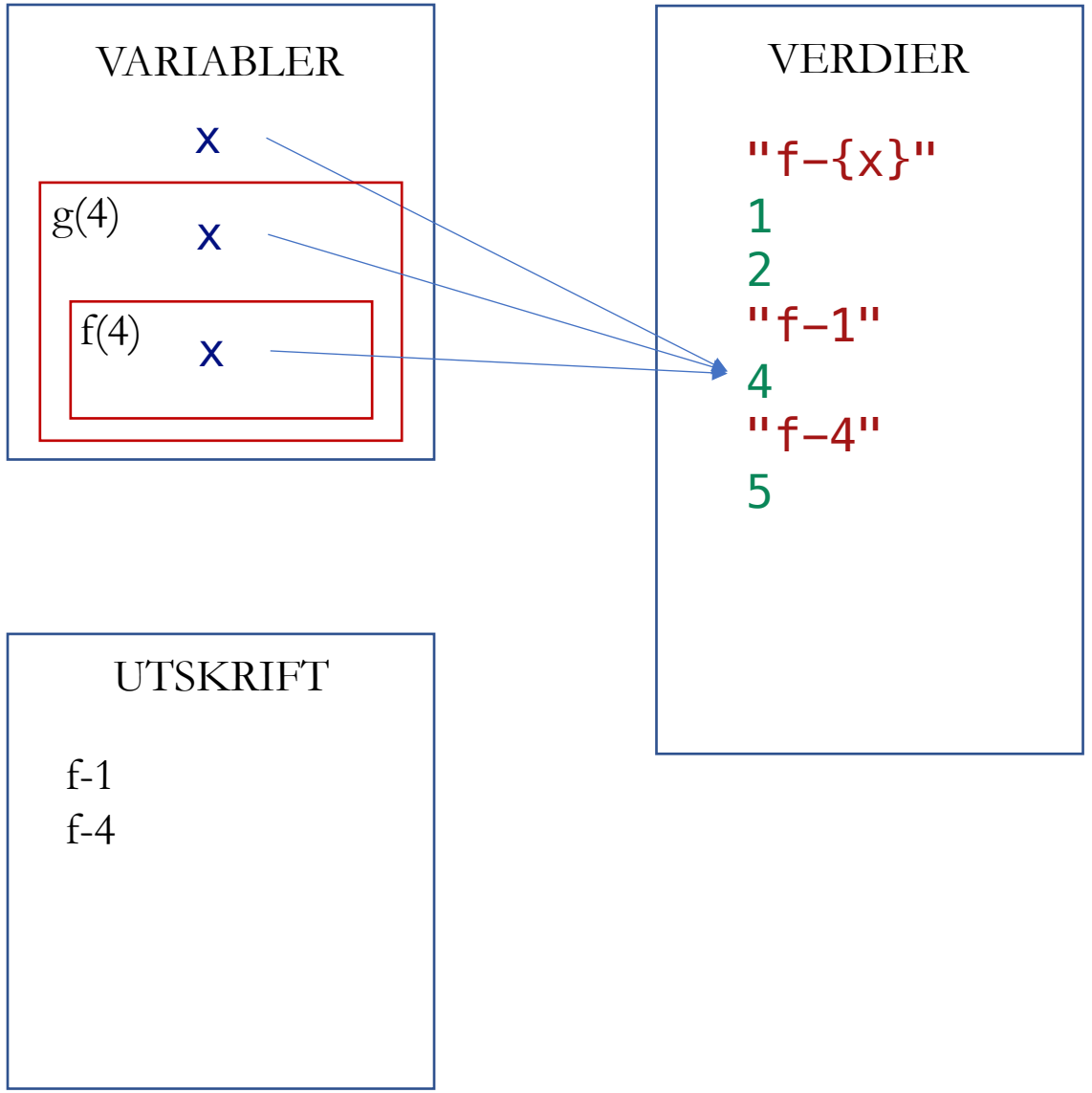
KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```



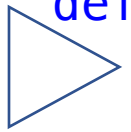


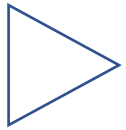


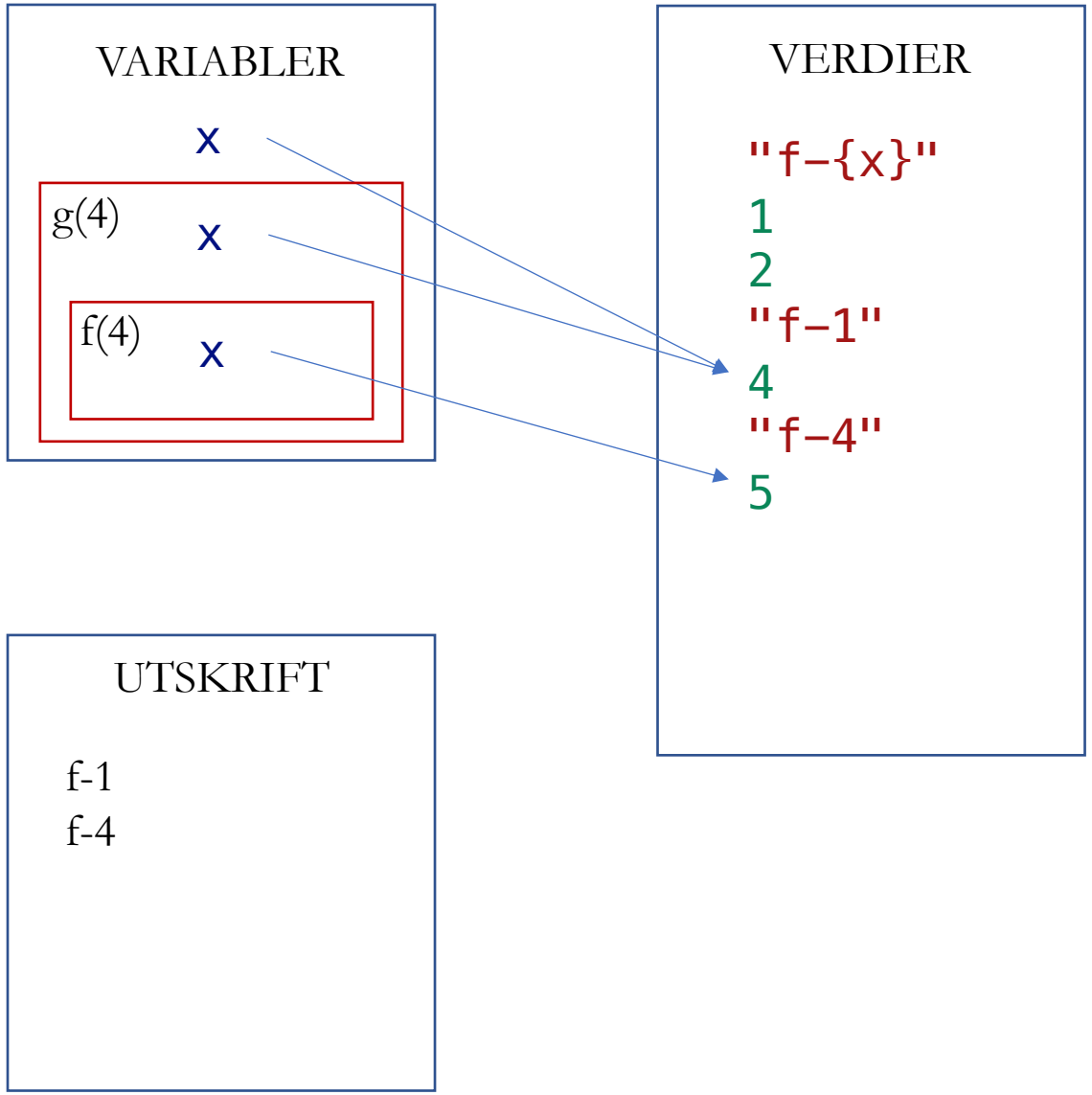


KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```

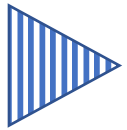


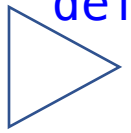


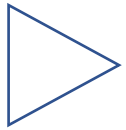


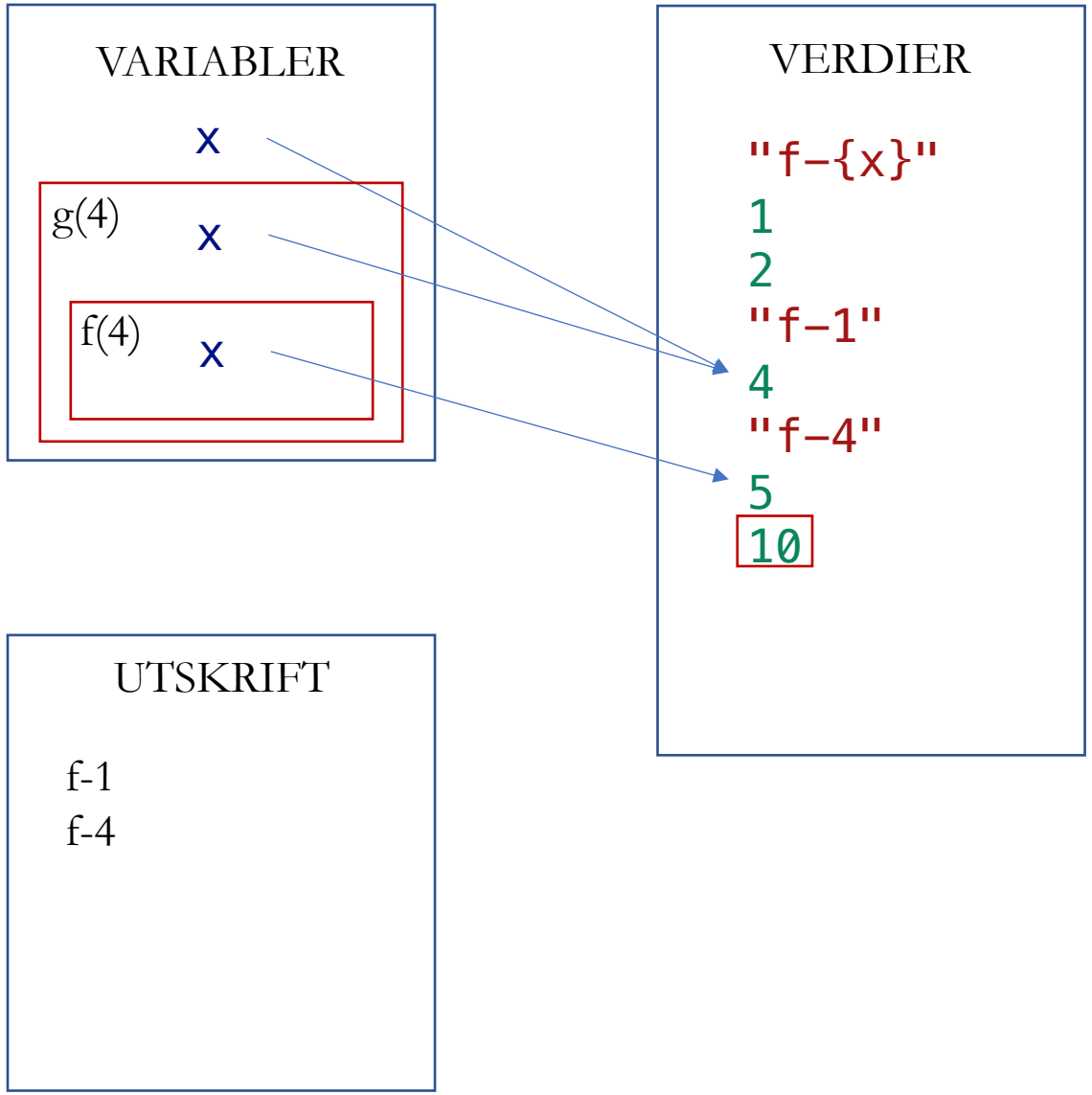
KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```



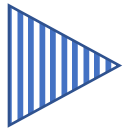


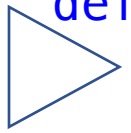


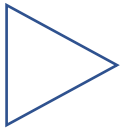


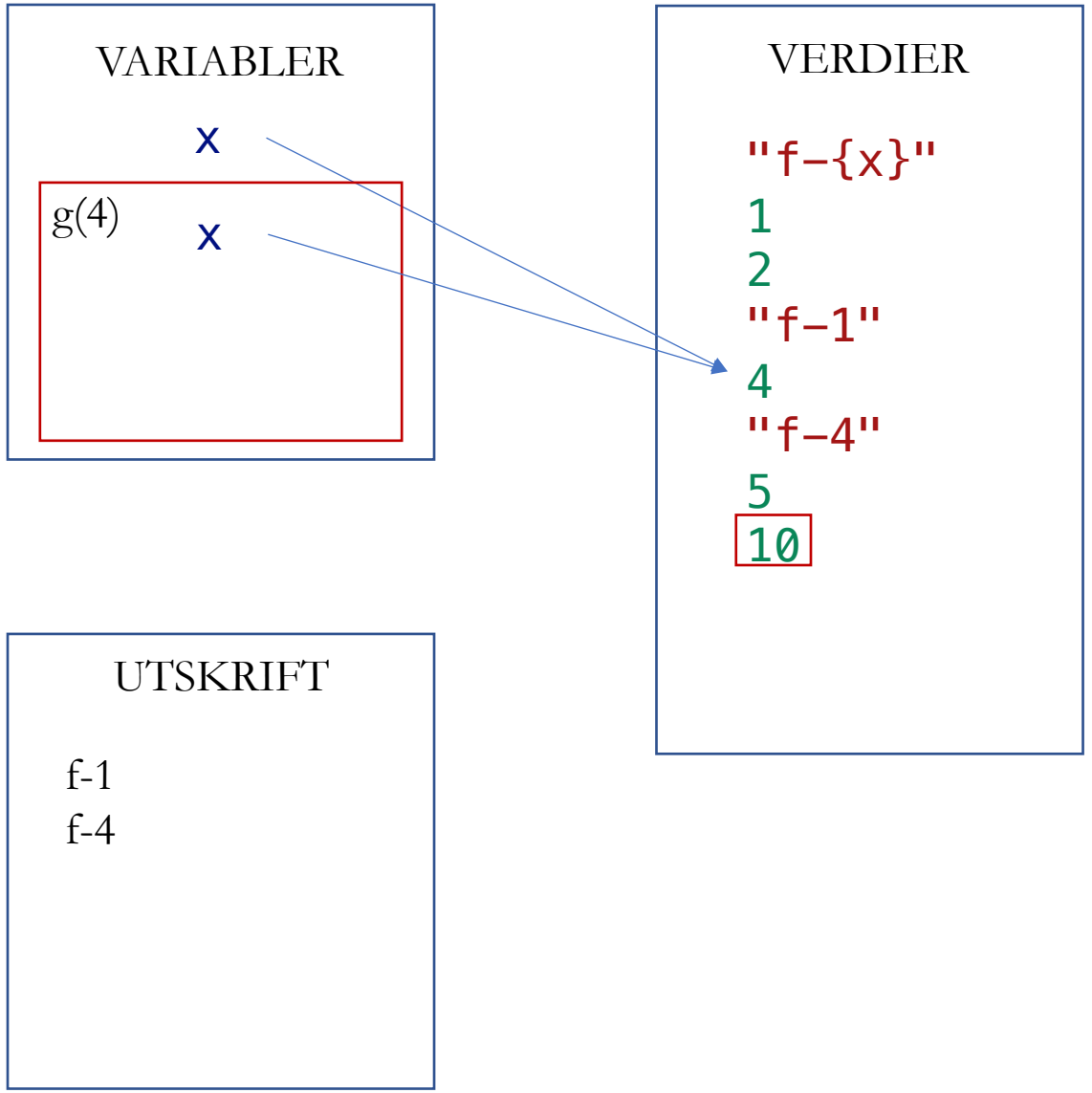
KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```



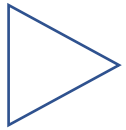


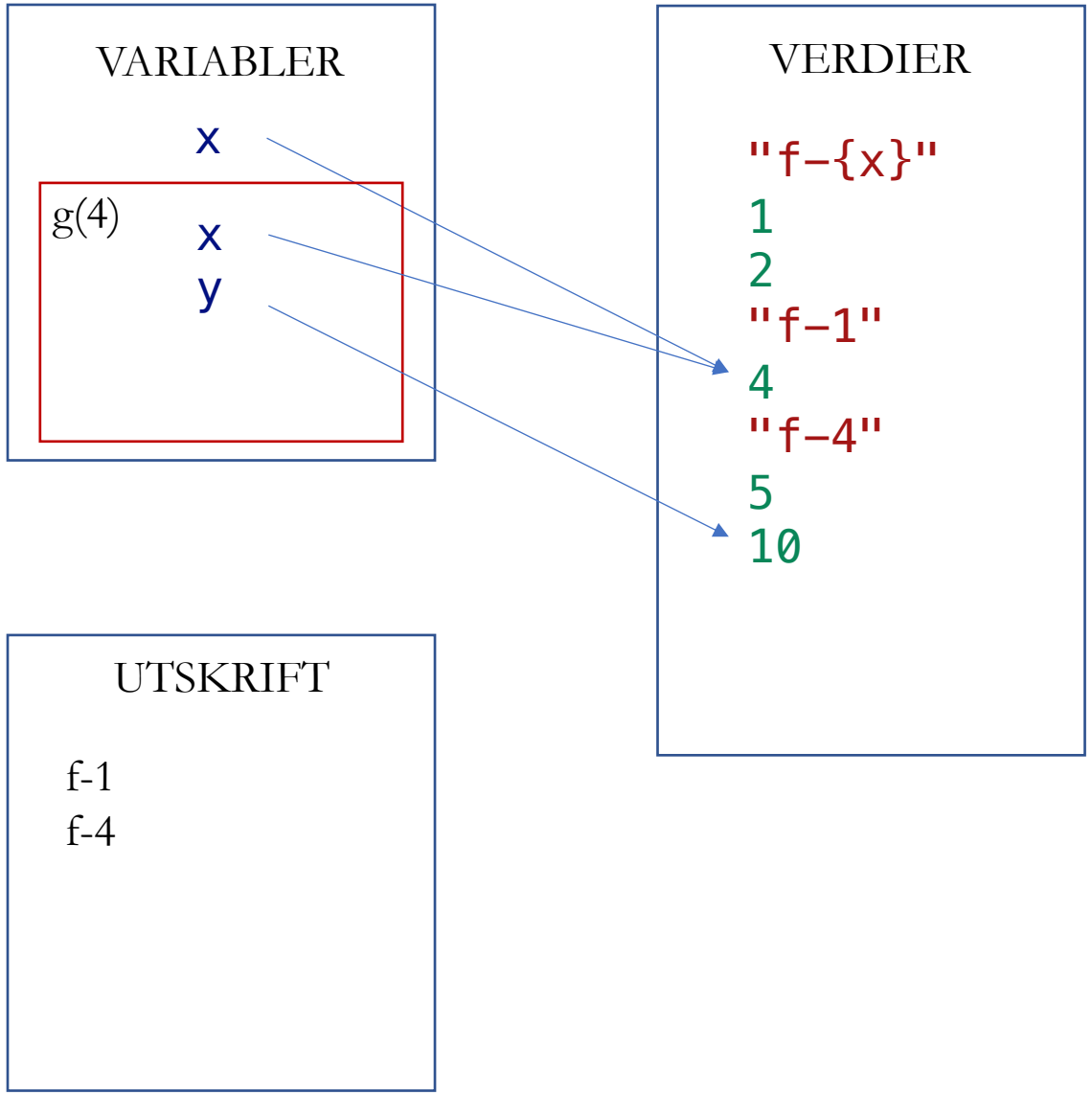




KODESPORING

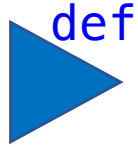
```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```

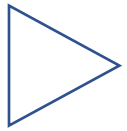


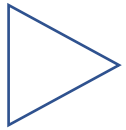


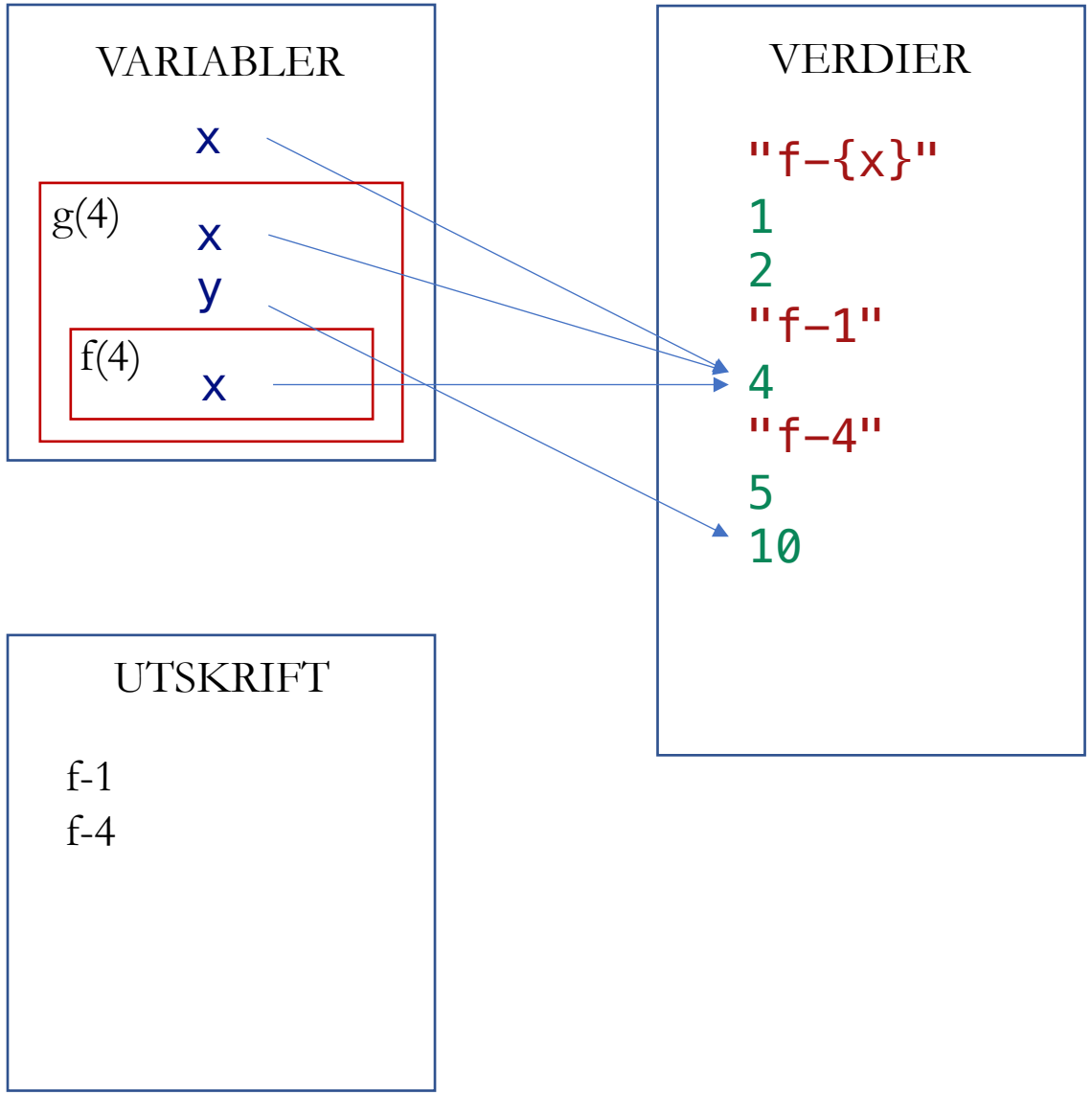
KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```



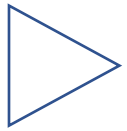


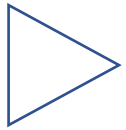


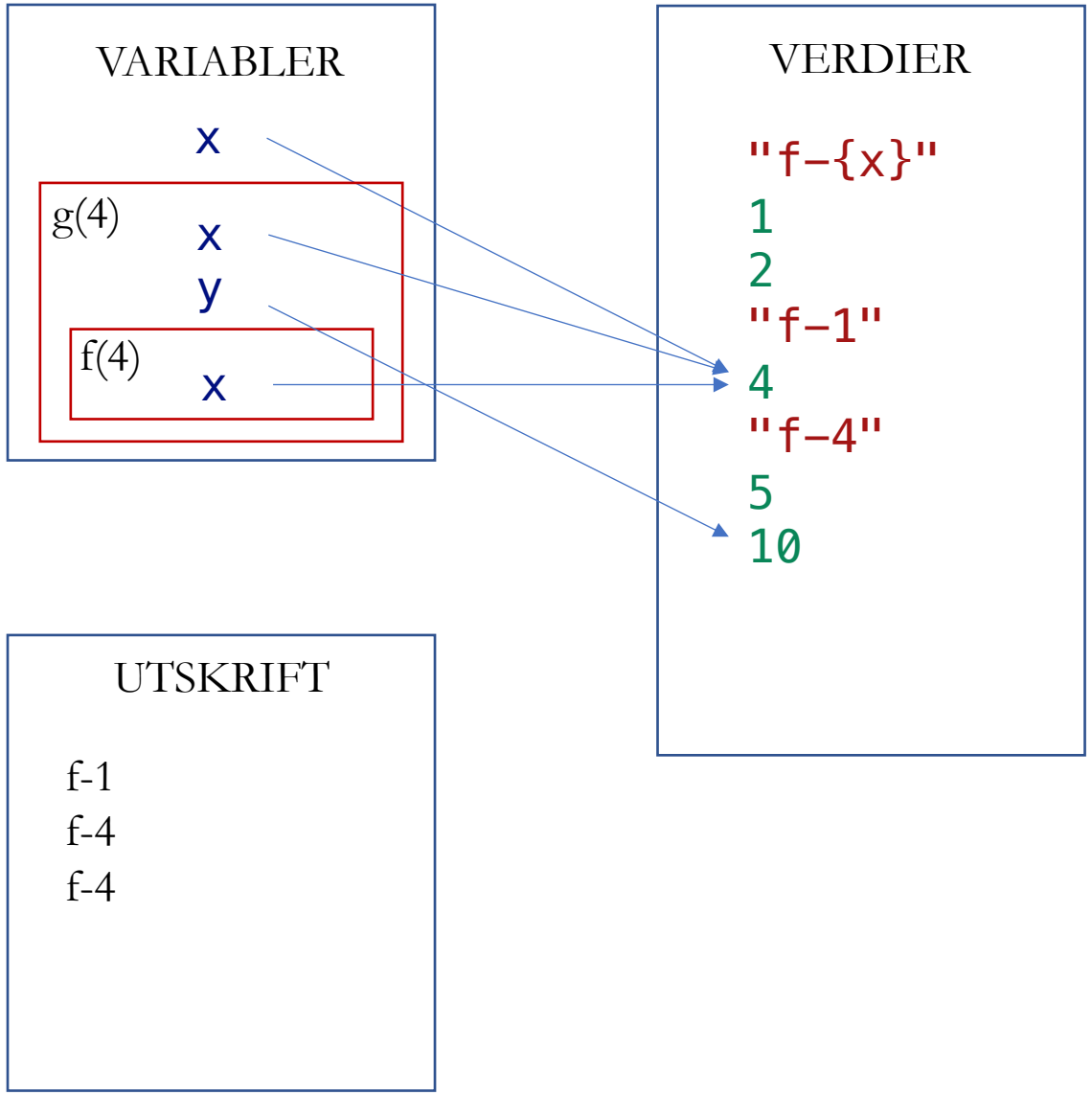


KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```

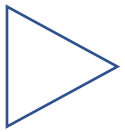


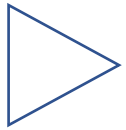


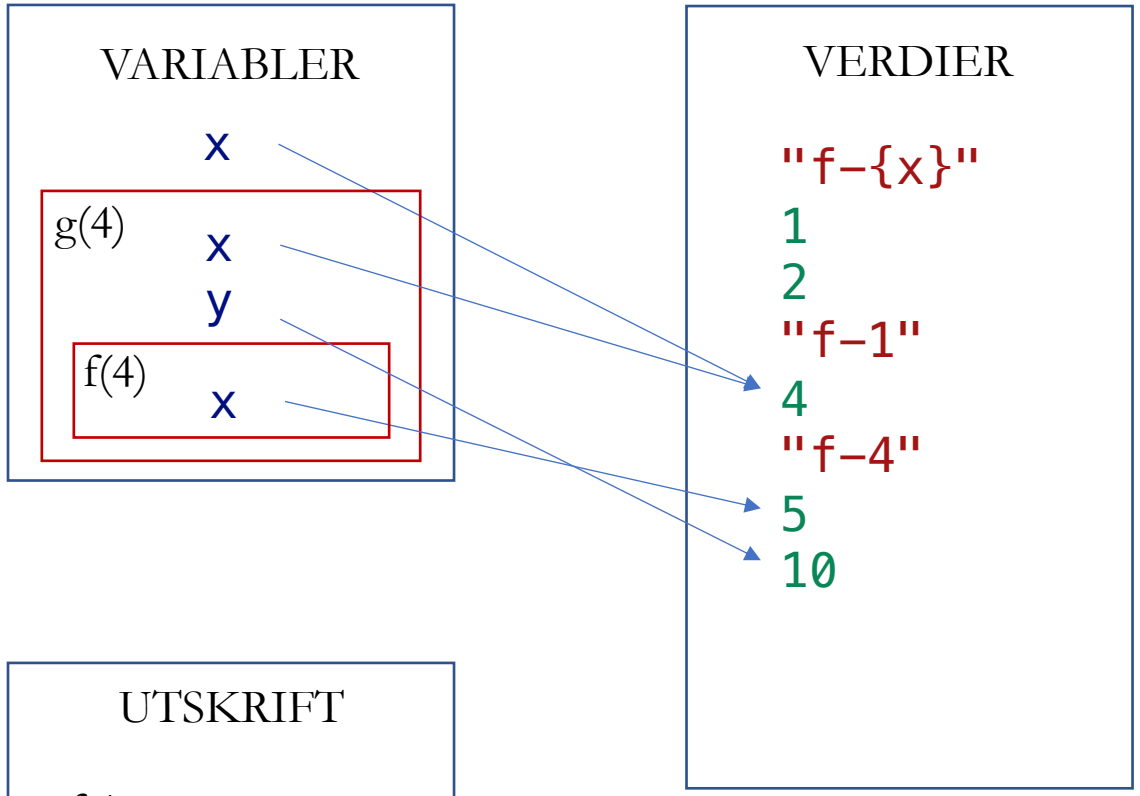


KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```





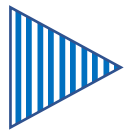


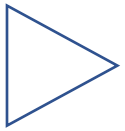
UTSKRIFT

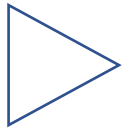
f-1
f-4
f-4

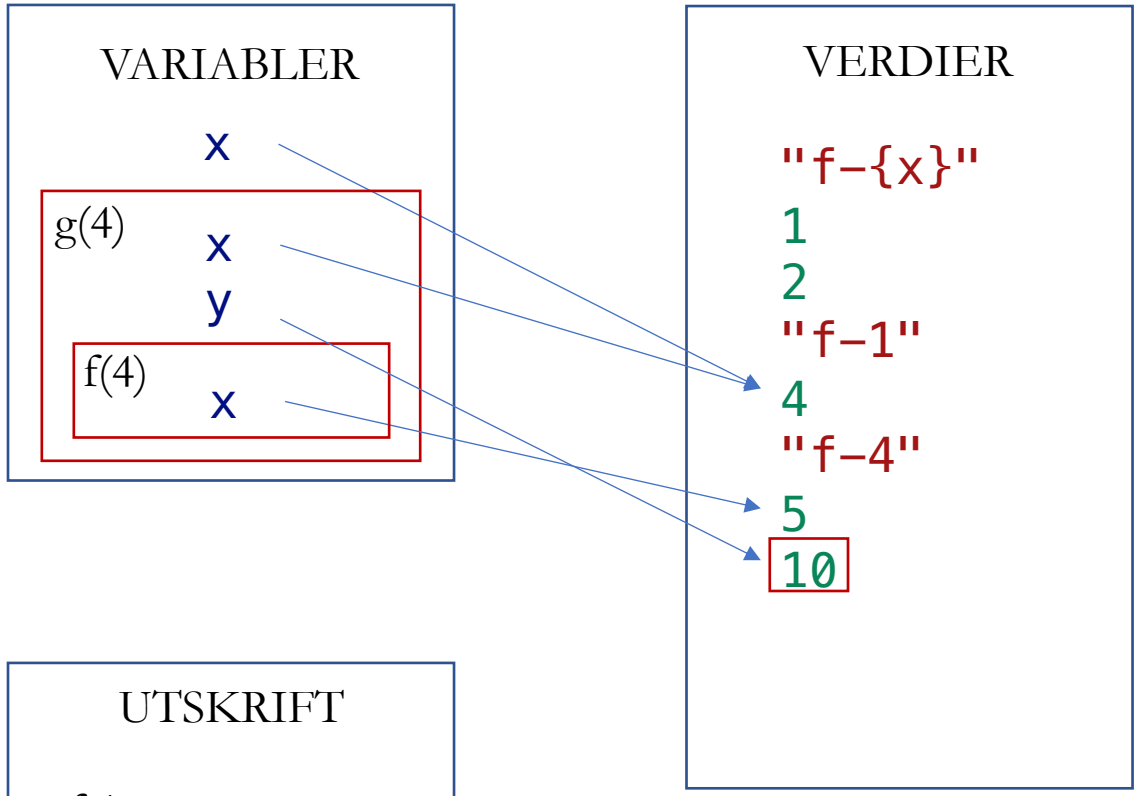
KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```







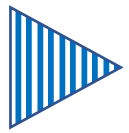


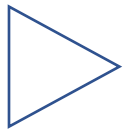
UTSKRIFT

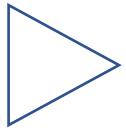
f-1
f-4
f-4

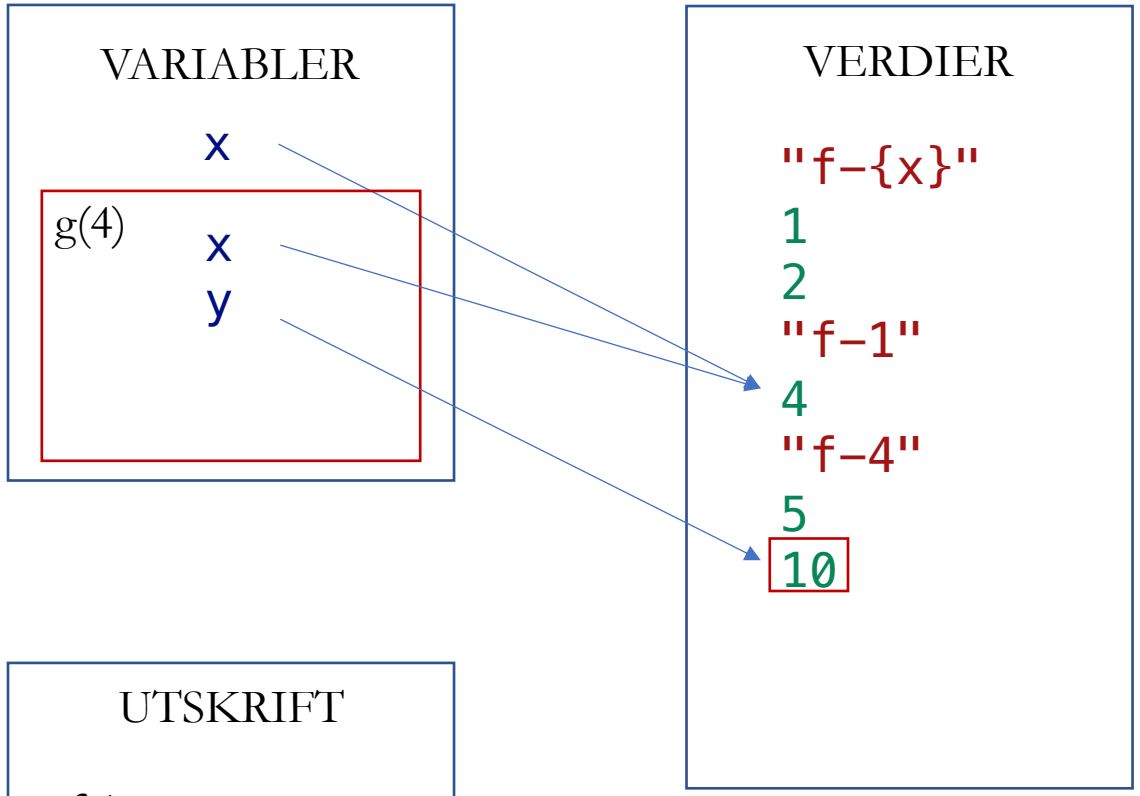
KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```







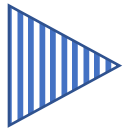


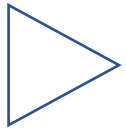
UTSKRIFT

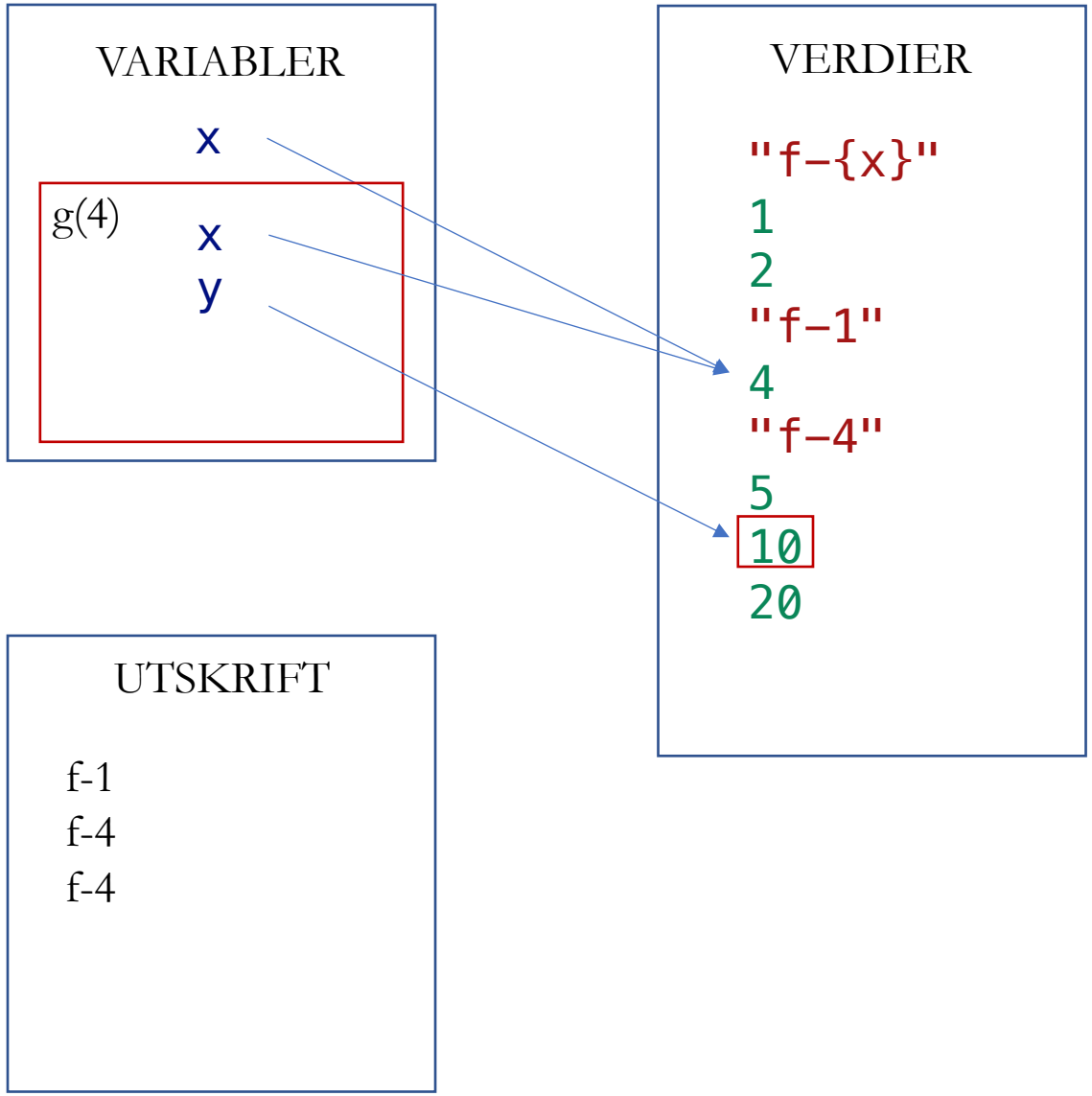
f-1
f-4
f-4

KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```

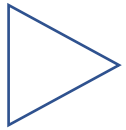


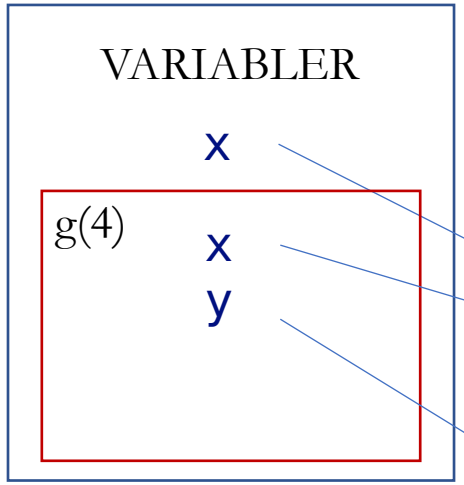


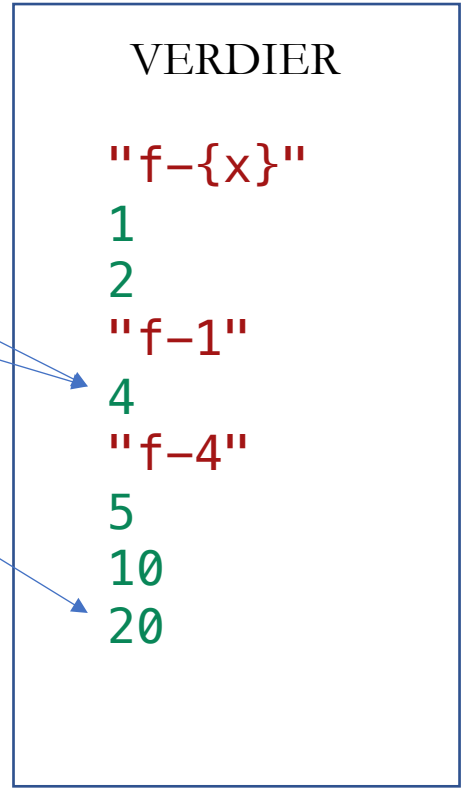


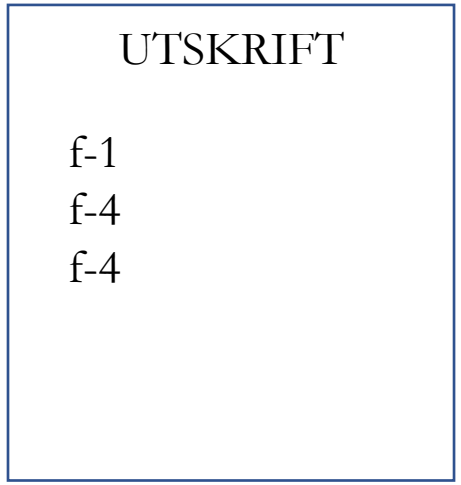
KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```



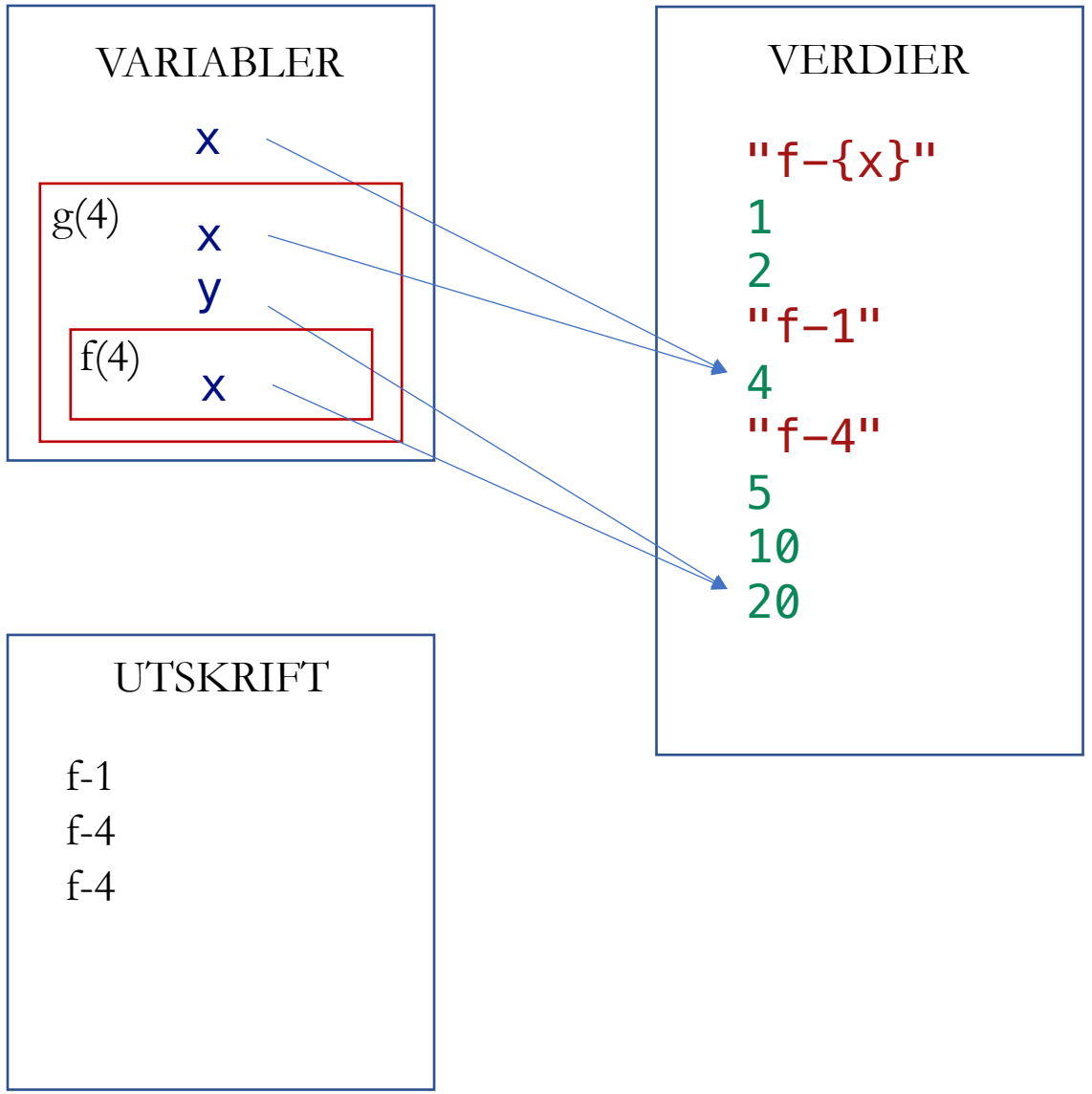






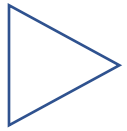
KODESPORING

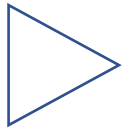
```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```

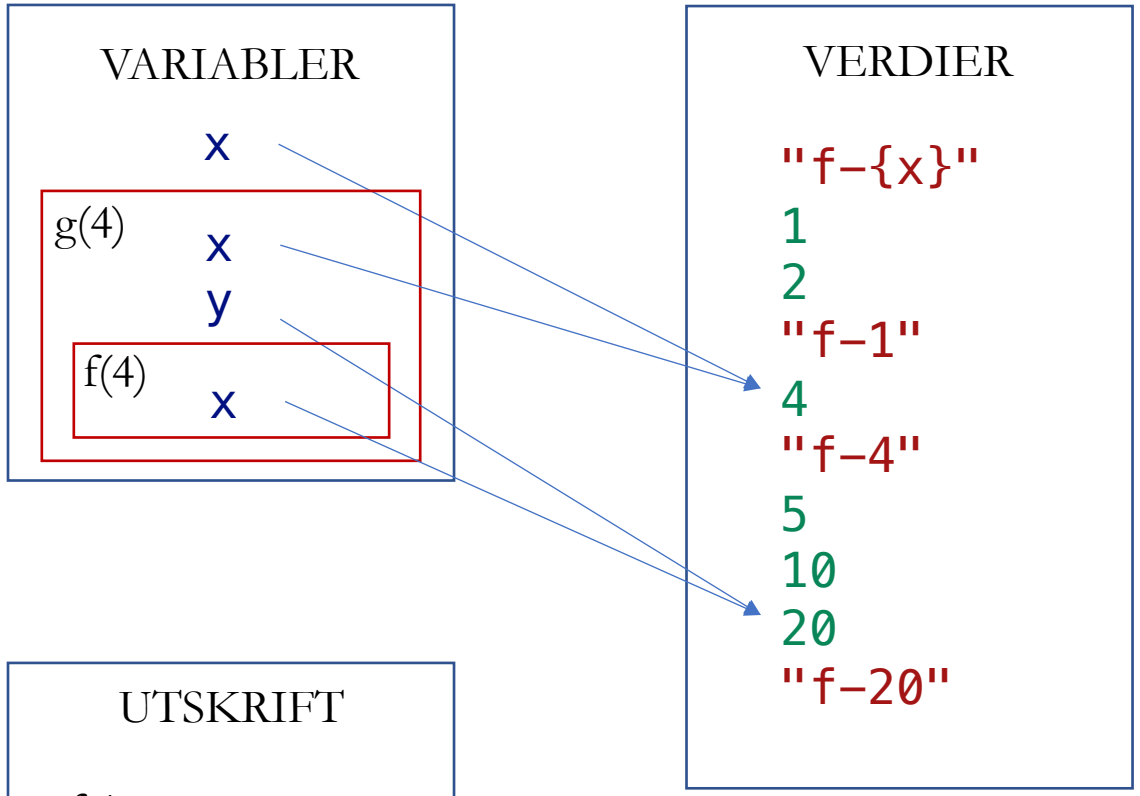


KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```





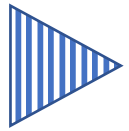


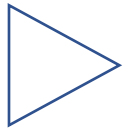
UTSKRIFT

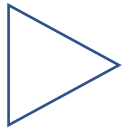
f-1
f-4
f-4
f-20

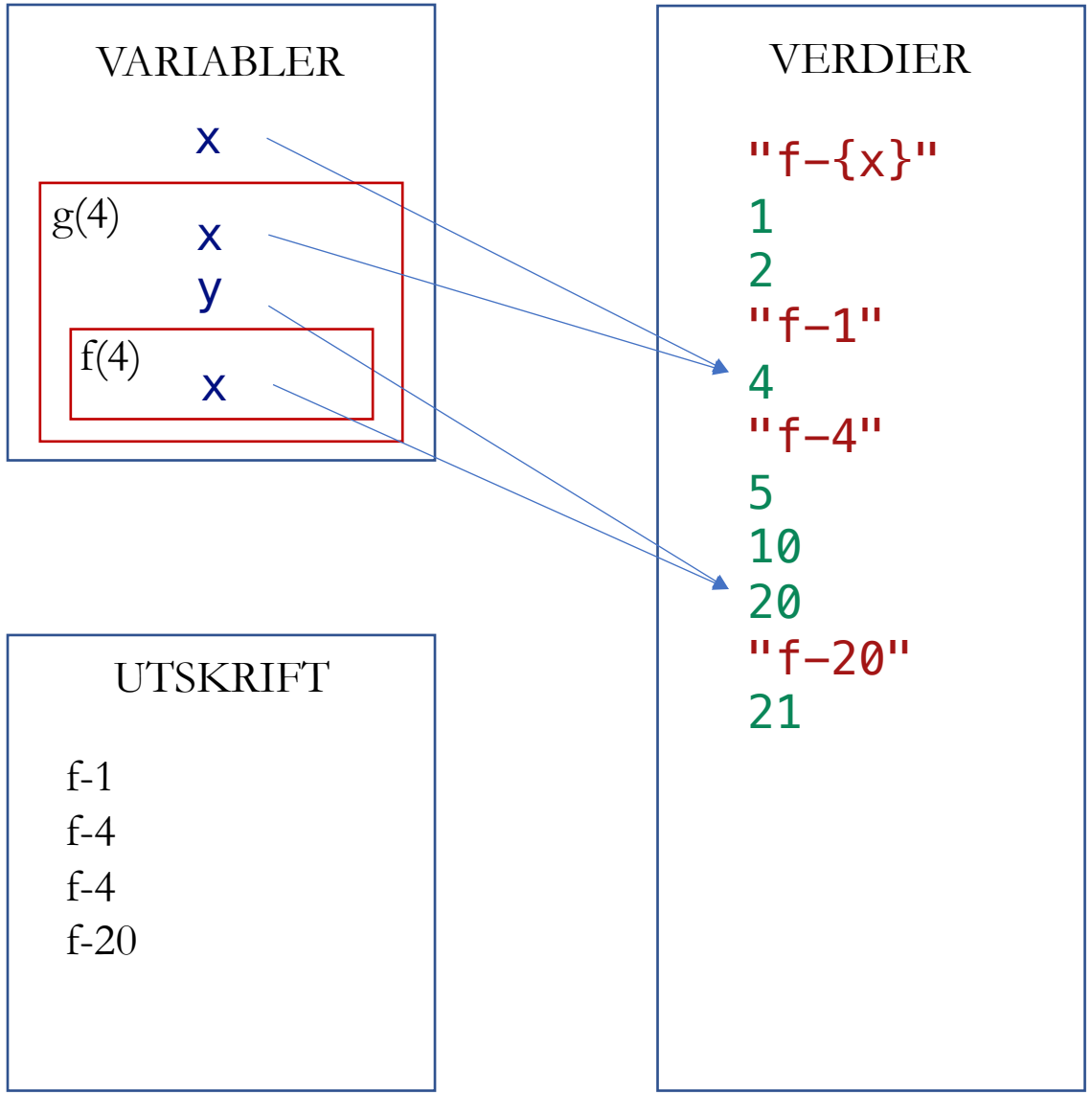
KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```



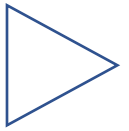


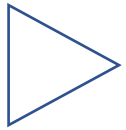


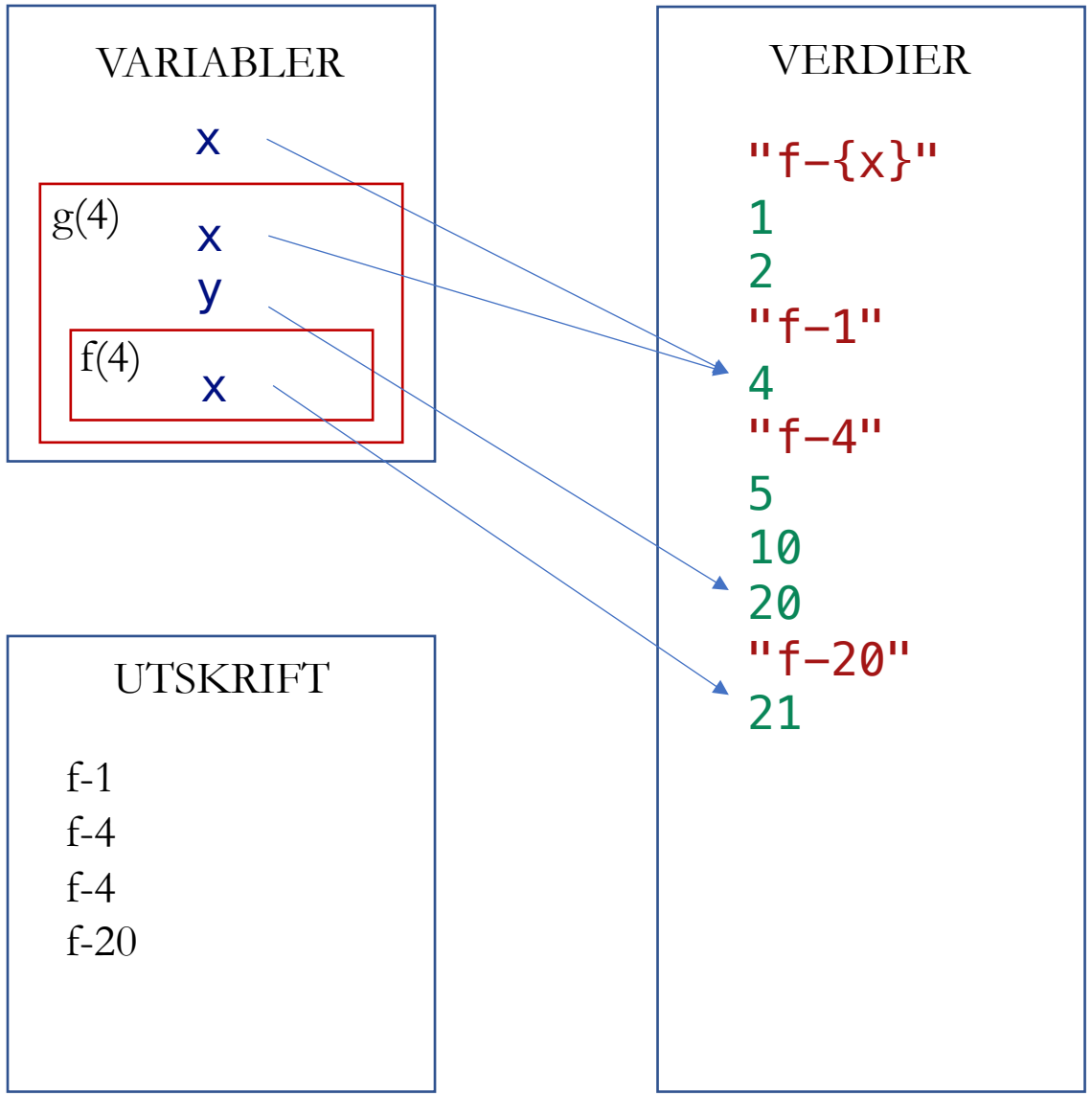


KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```

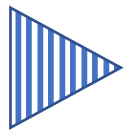


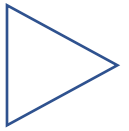


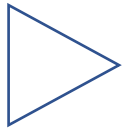


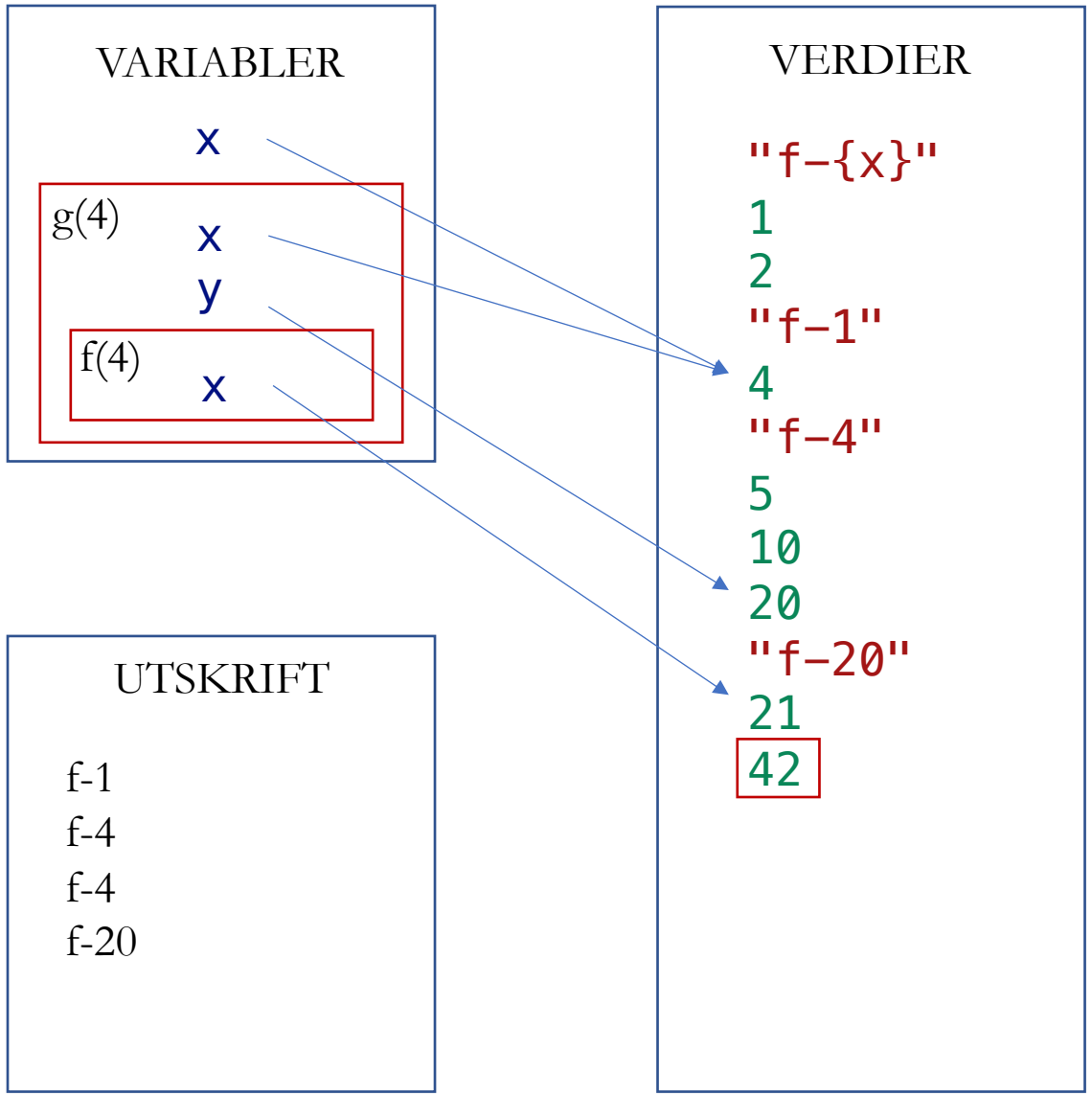
KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```



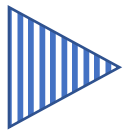


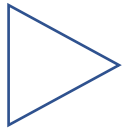


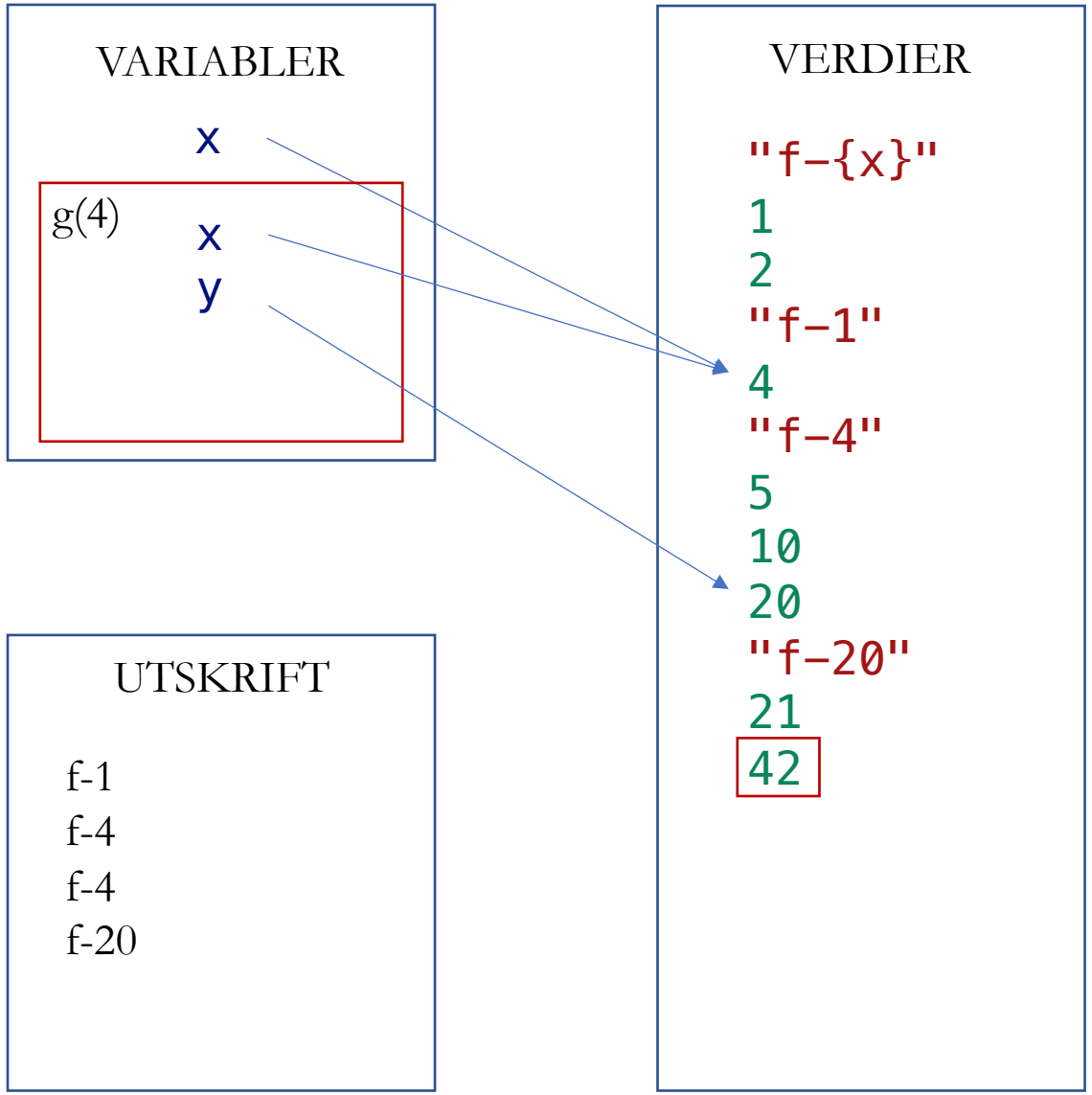


KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```

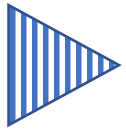






KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```



VARIABLER

x

UTSKRIFT

f-1
f-4
f-4
f-20

VERDIER

"f-{x}"
1
2
"f-1"
4
"f-4"
5
10
20
"f-20"
21
42

KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```

VARIABLER

x

UTSKRIFT

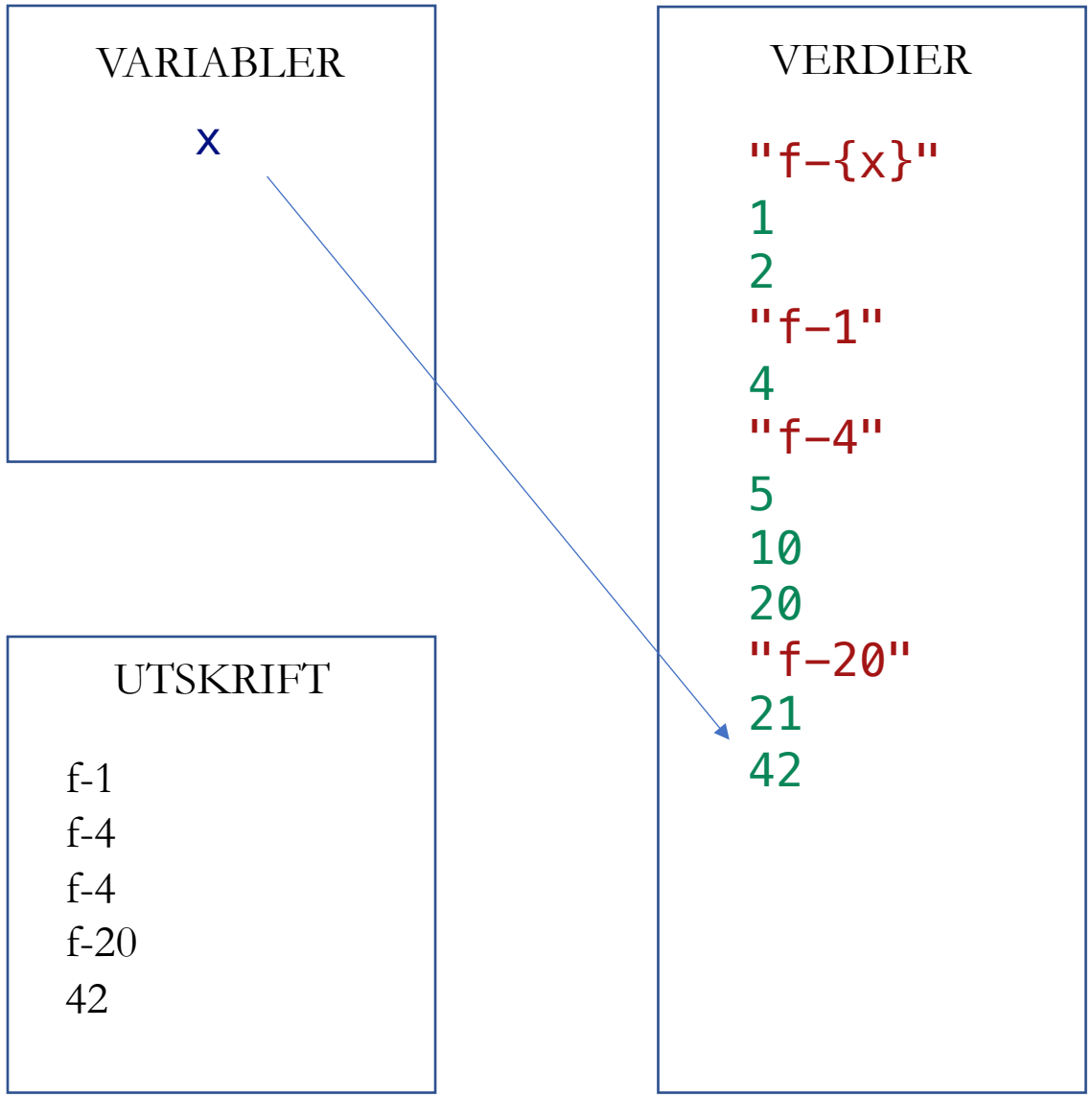
f-1
f-4
f-4
f-20

VERDIER

"f-{x}"
1
2
"f-1"
4
"f-4"
5
10
20
"f-20"
21
42

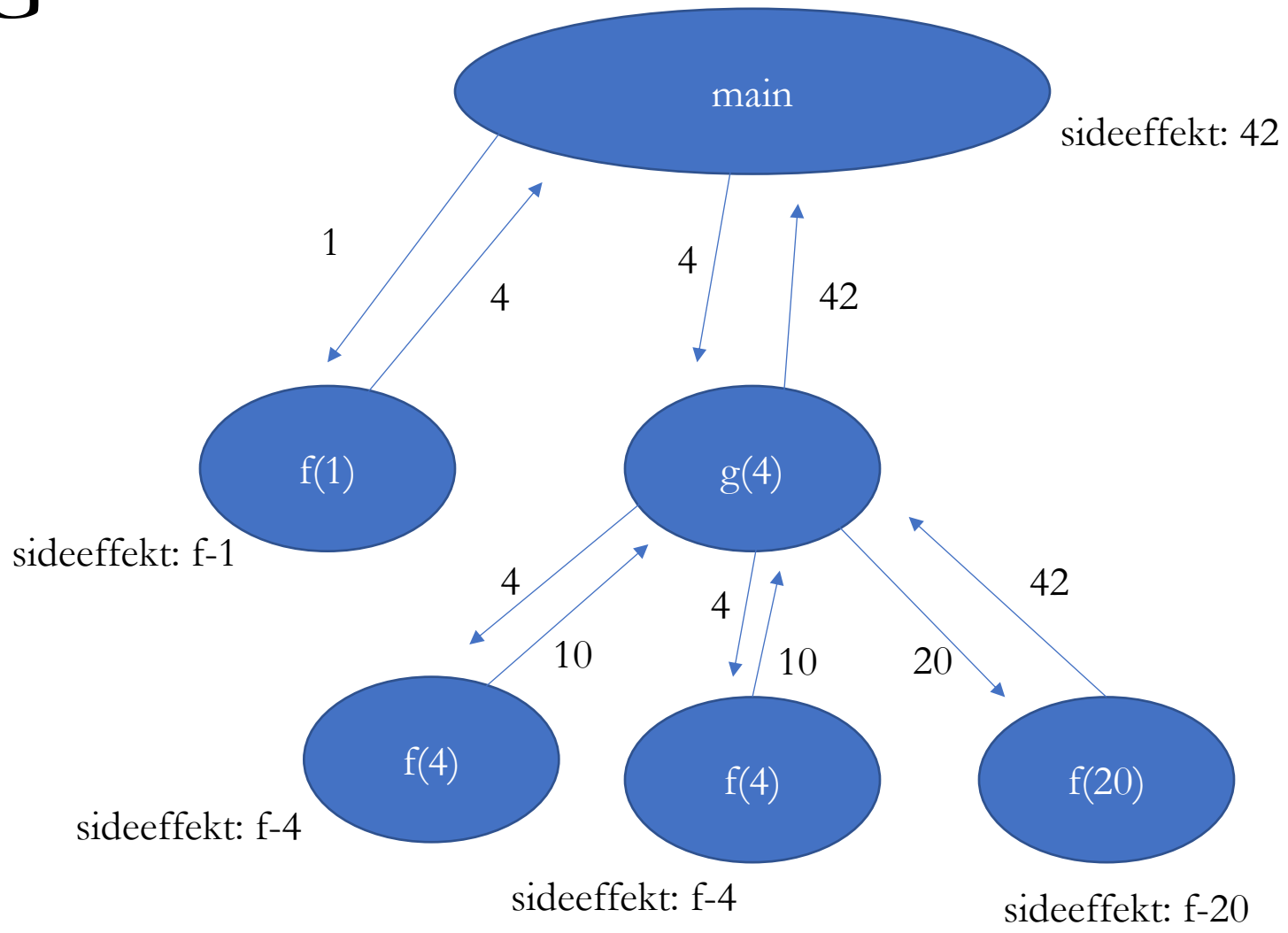
KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```



KODESPORING

```
def f(x):
    print(f"f-{x}")
    x += 1
    return 2*x

def g(x):
    y = f(x)
    y += f(x)
    return f(y)

x = 1
x = f(x)
x = g(x)
print(x)
```



SENTRALE BEGREPER

- Verdier
  - Data lagret i minnet
- Type
  - Alle verdier har en type (int, str, float, bool)
- Variabel
  - En navngitt referanse til en verdi
  - Tilordnes verdi med =
- Uttrykk og operasjoner
  - Et regnestykke som evaluerer til en verdi
  - Presedens og evaluering
- Betinget oppførsel
  - if/elif/else
- Funksjoner
- Løkker
  - while
  - for
    - range
- Lister
- Oppslagsverk og mengder

LØKKER

WHILE

```
bla()
bla()

while <betingelse>:
        bla()
        bla()
        bla()

bla()
bla()
```

boolsk uttrykk

så lenge betingelsen er True

WHILE

```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```

VARIABLER

UTSKRIFT

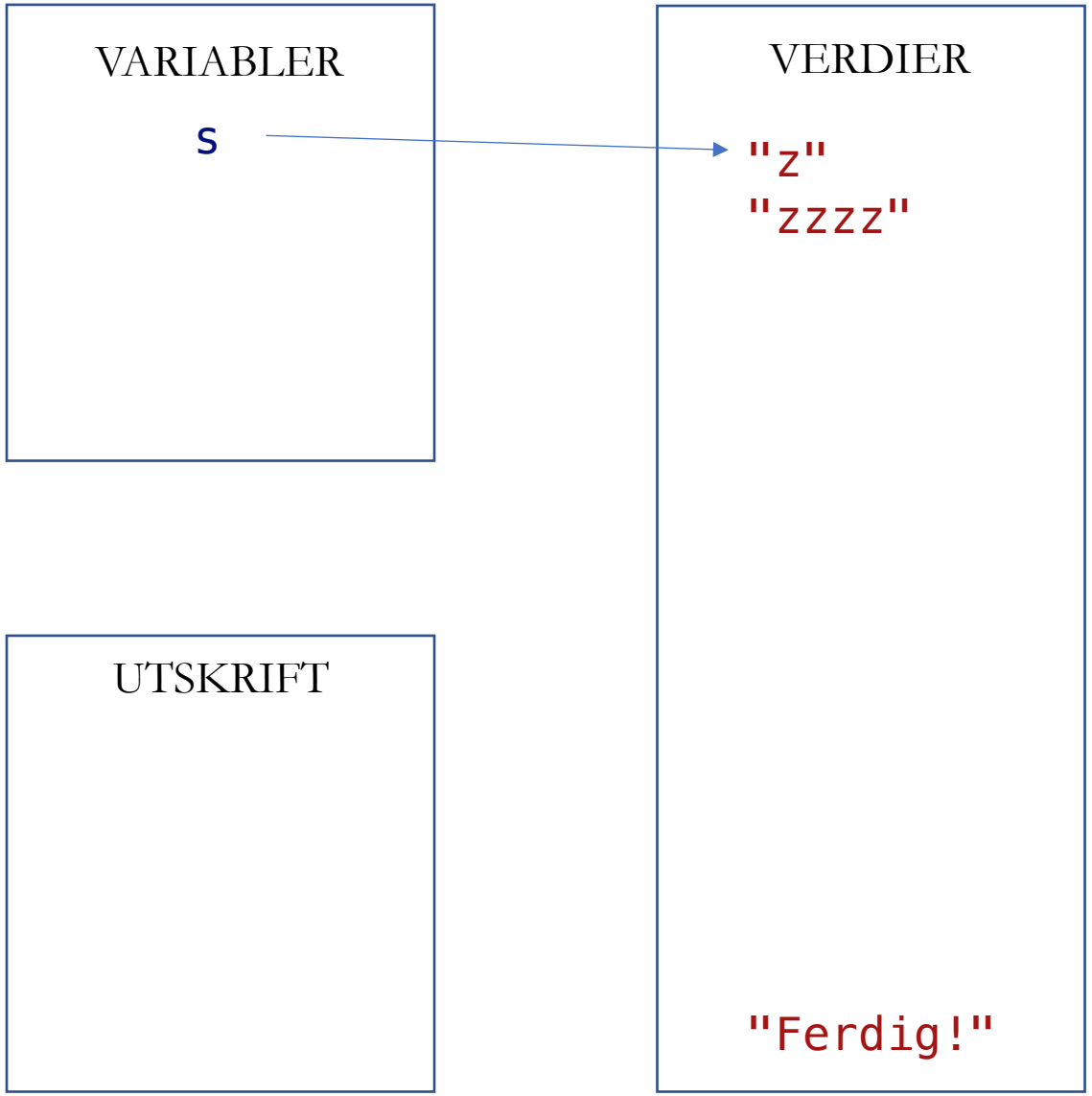
VERDIER

"z"
"zzzz"

"Ferdig!"

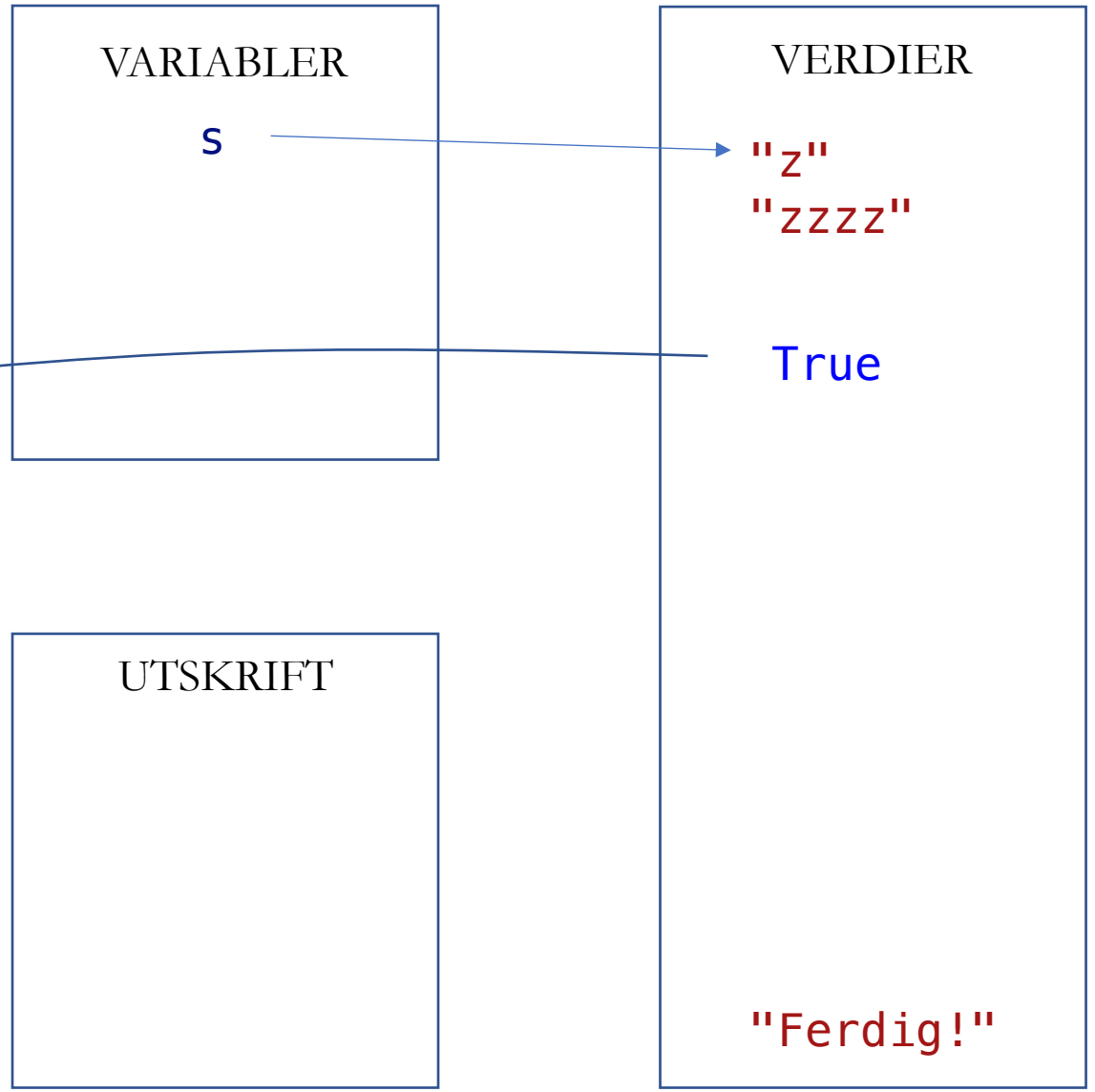
WHILE

```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```



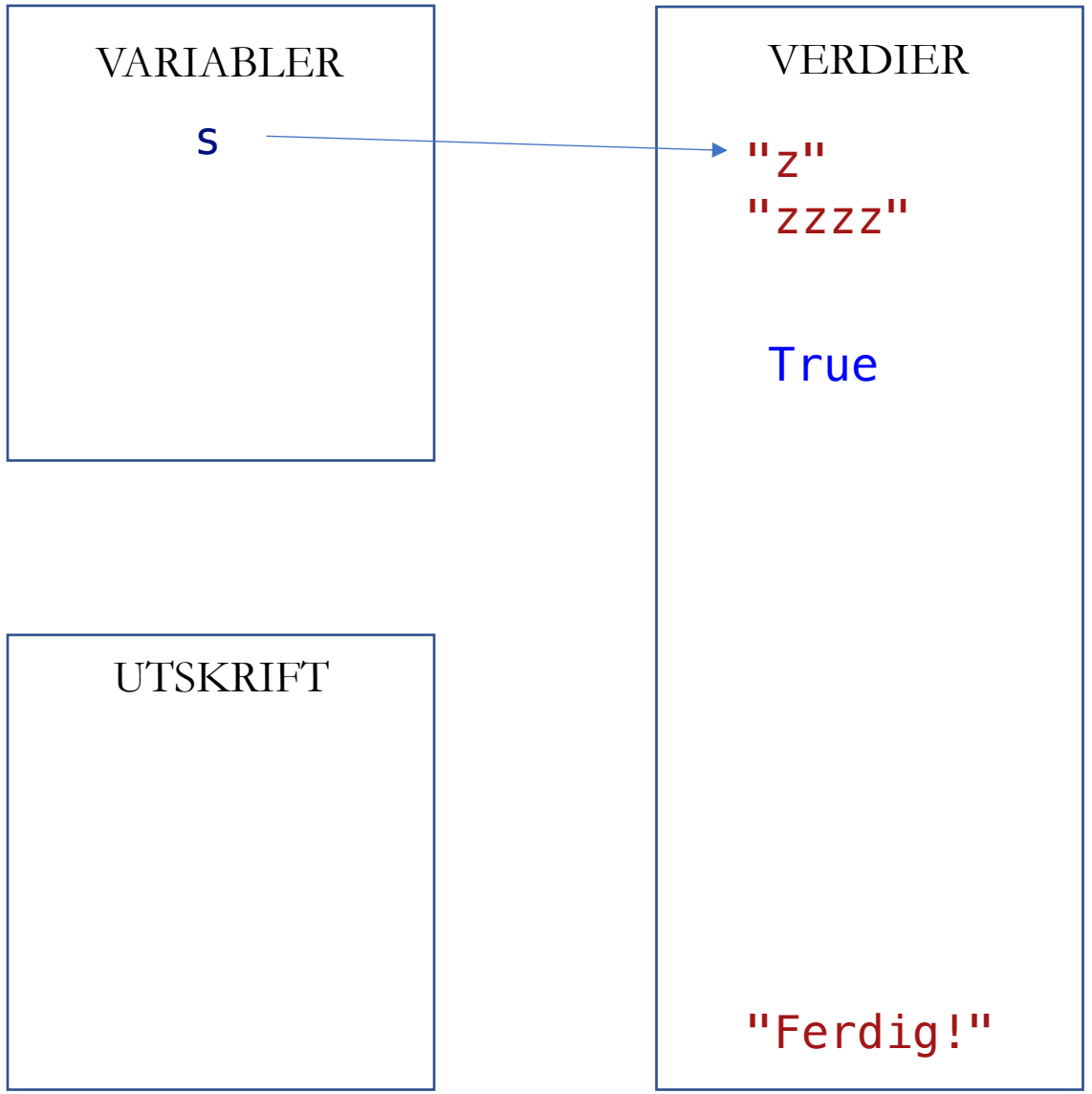
WHILE

```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```



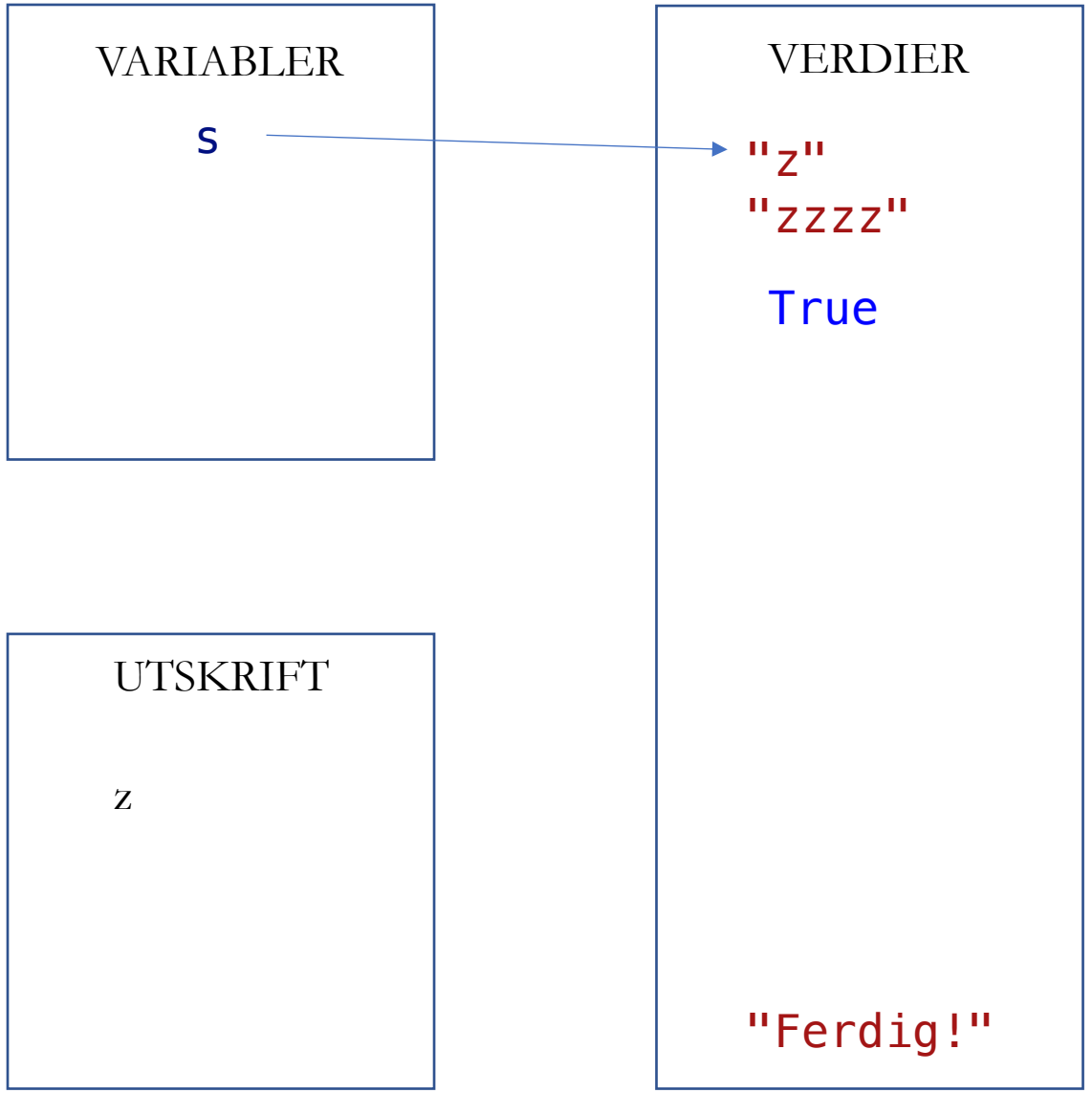
WHILE

```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```



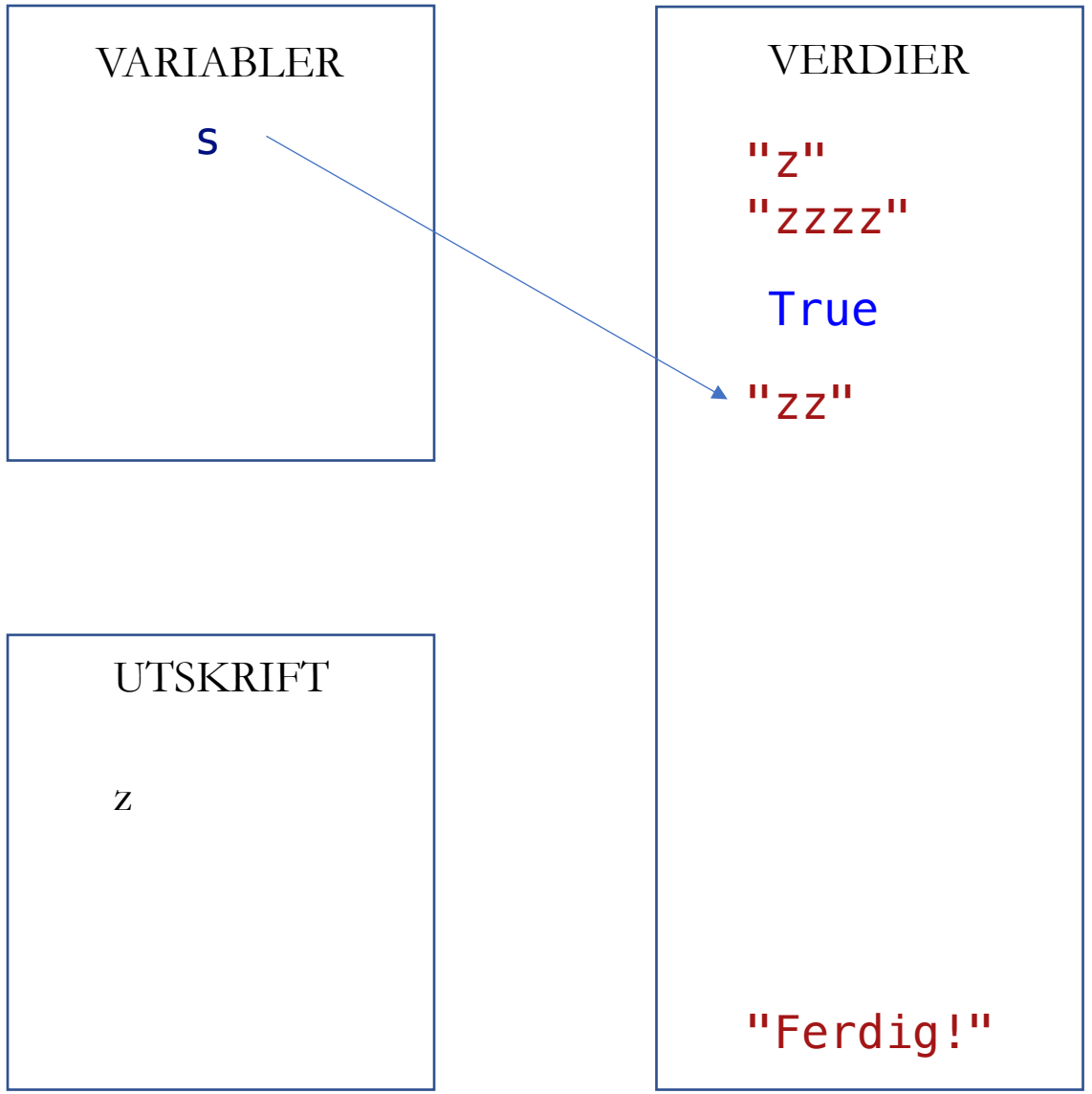
WHILE

```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```



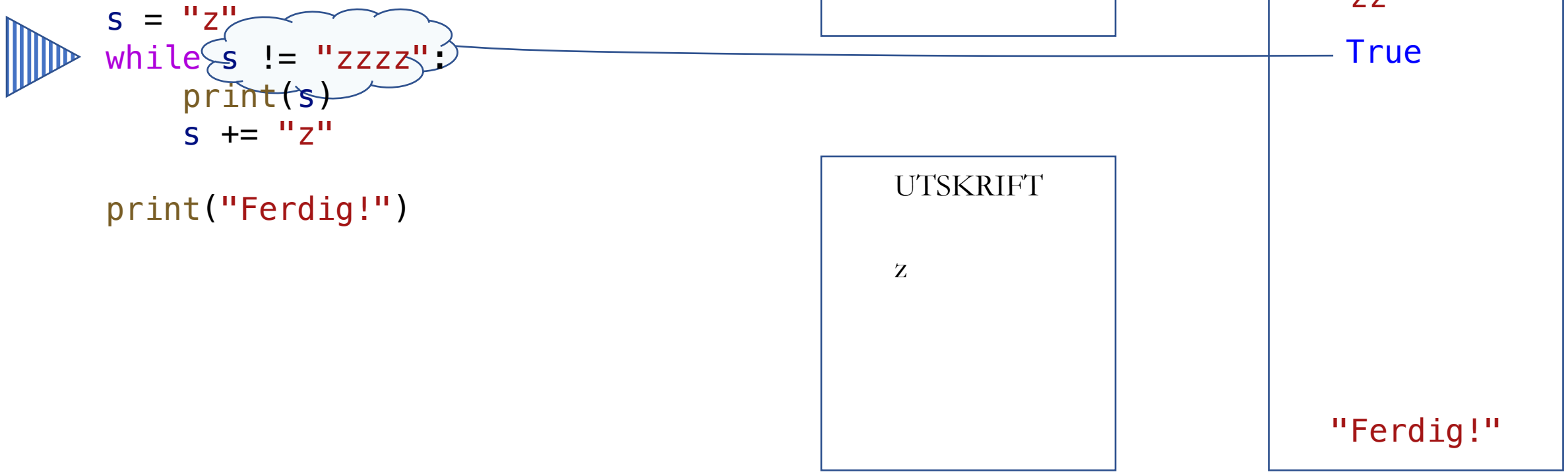
WHILE

```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```



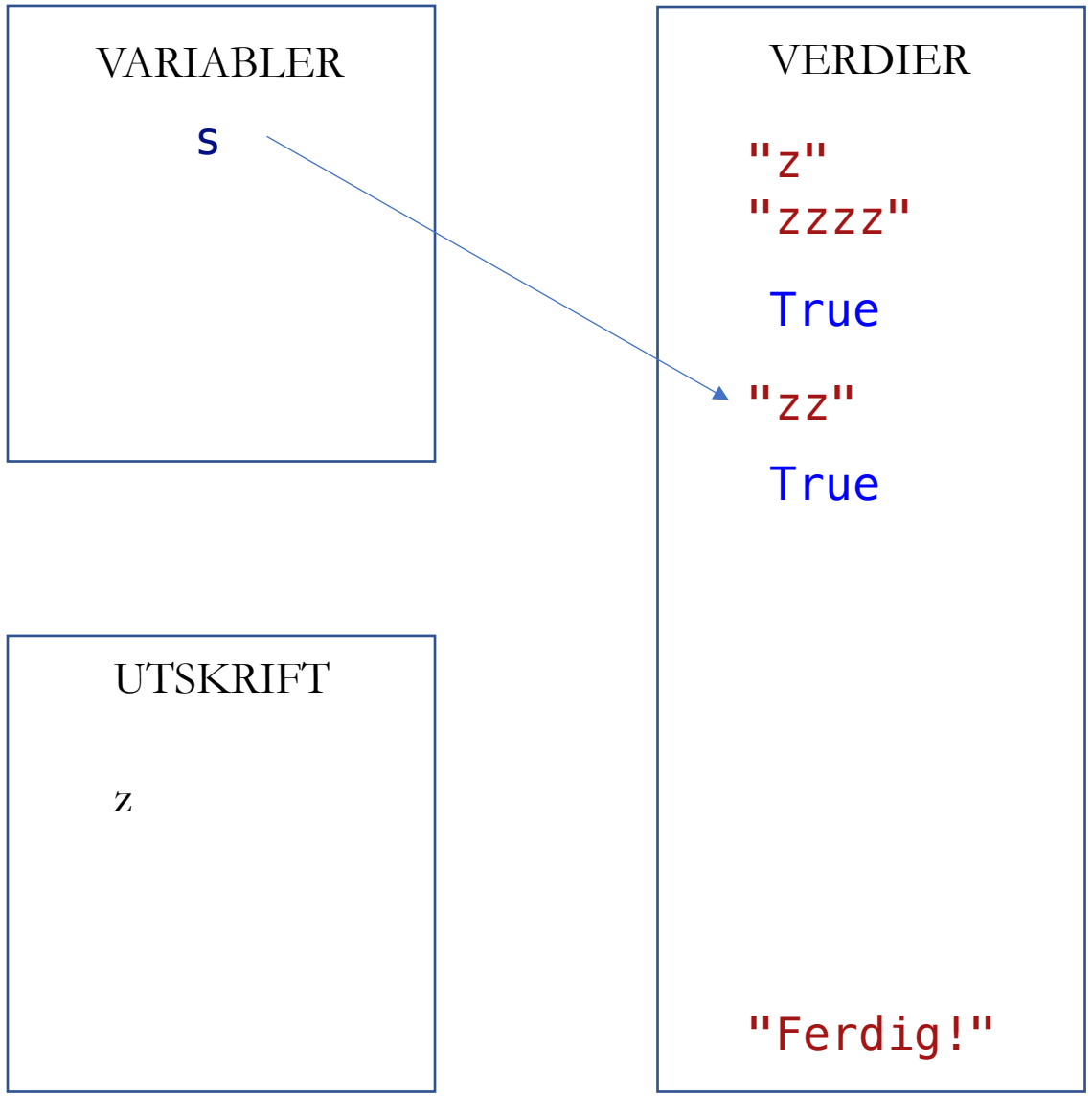
WHILE

```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```



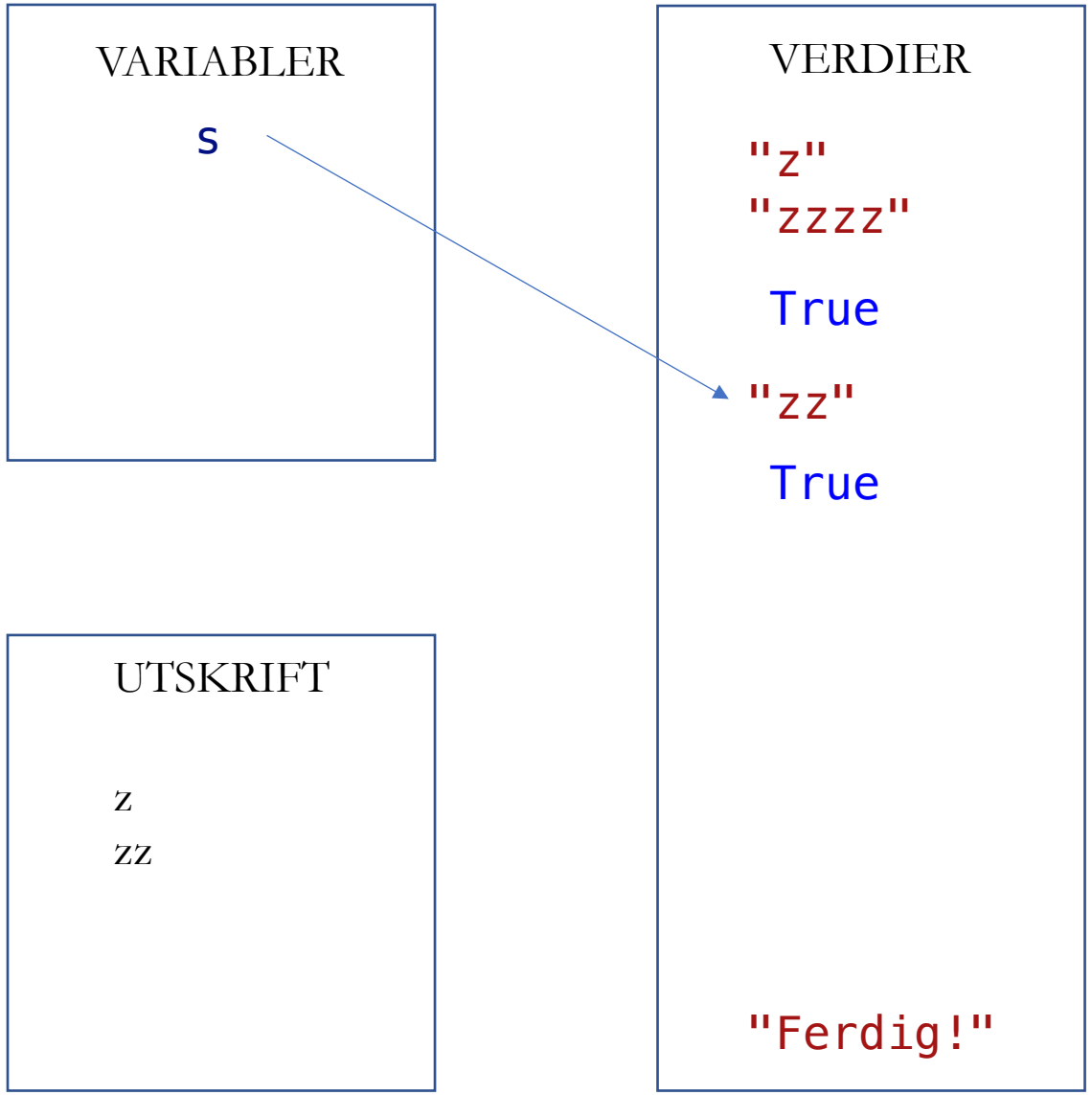
WHILE

```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```



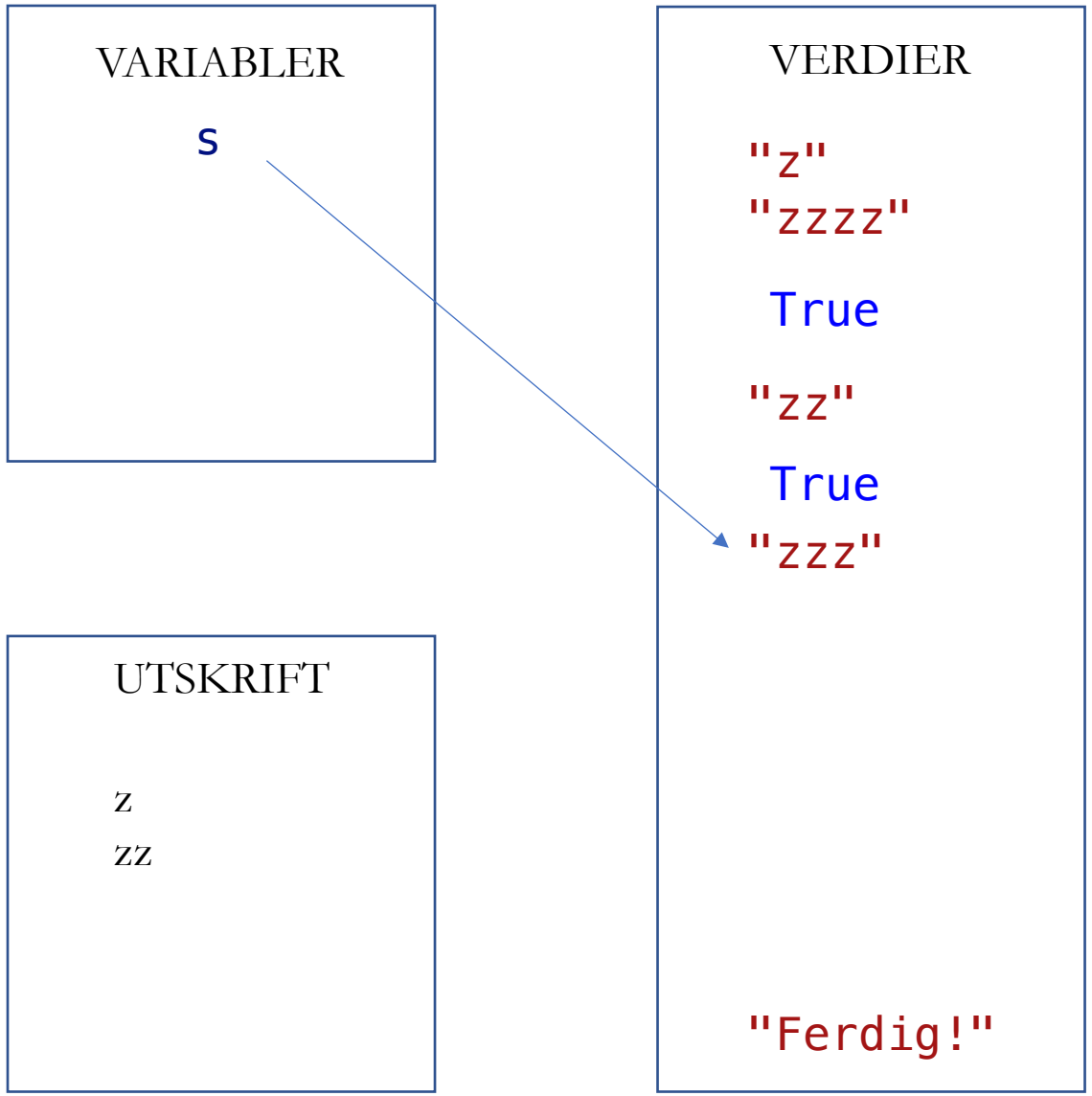
WHILE

```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```



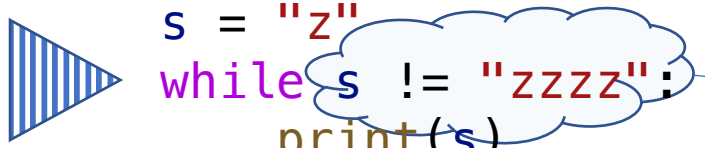
WHILE

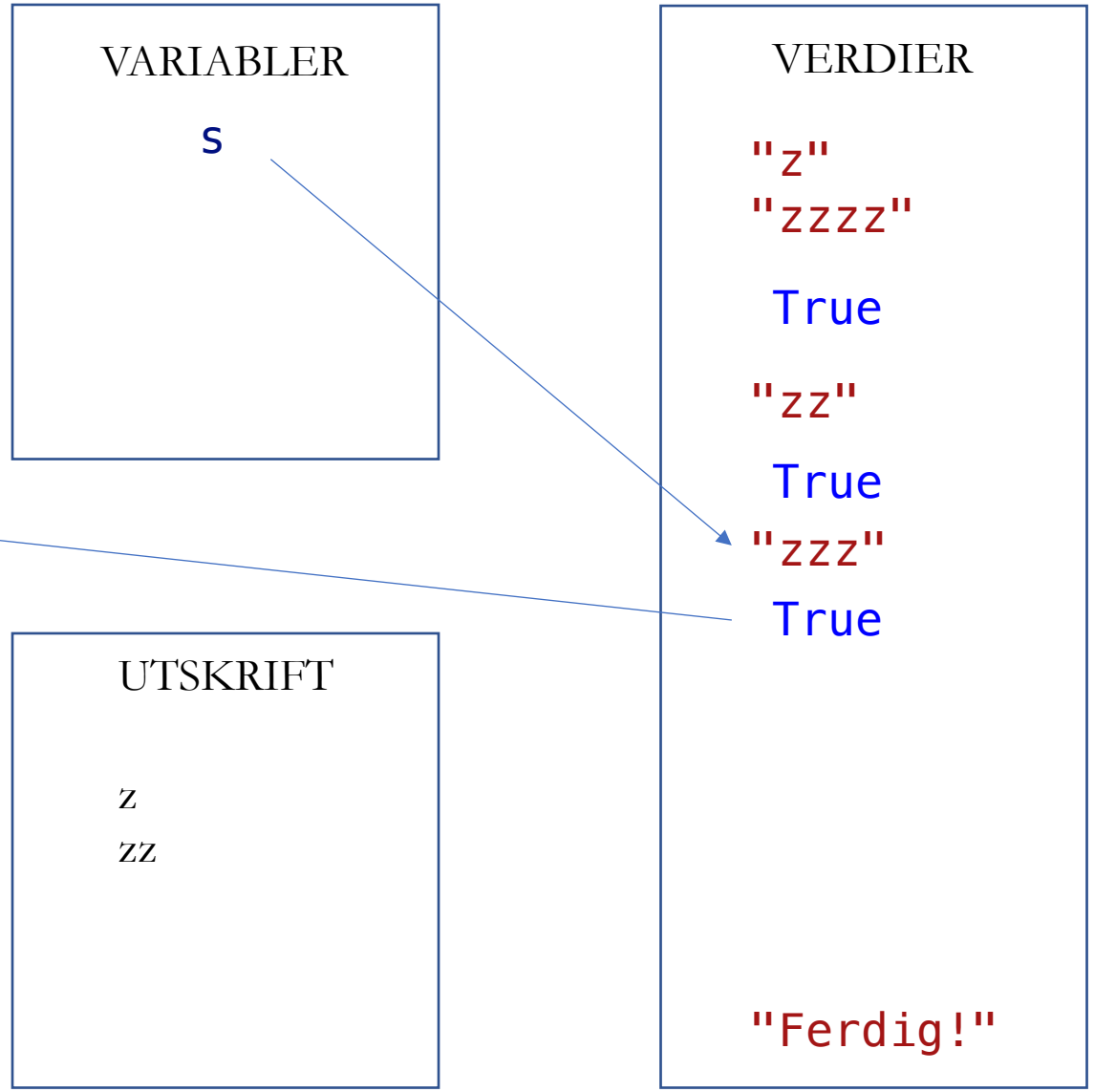
```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```



WHILE

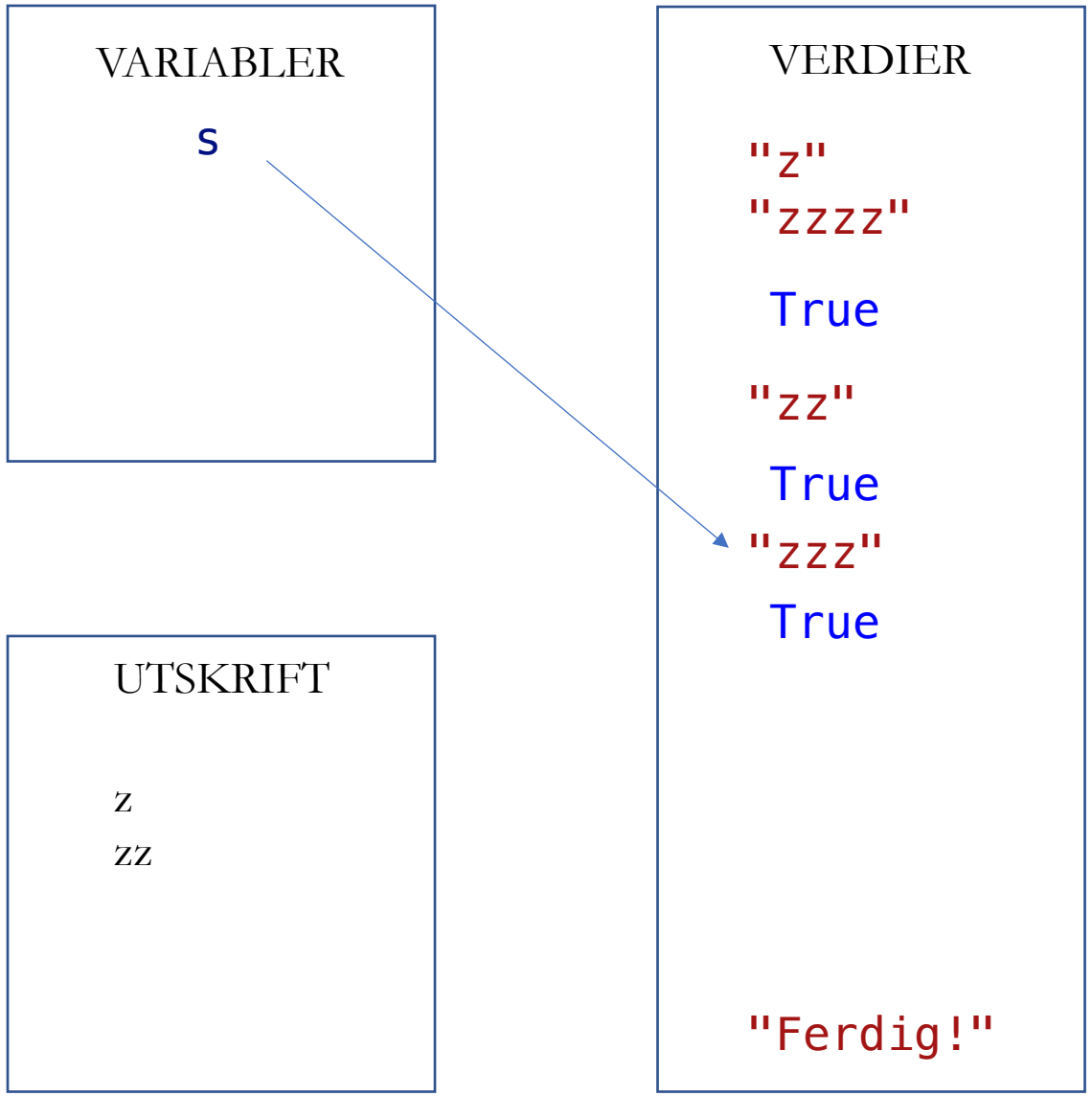
```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```





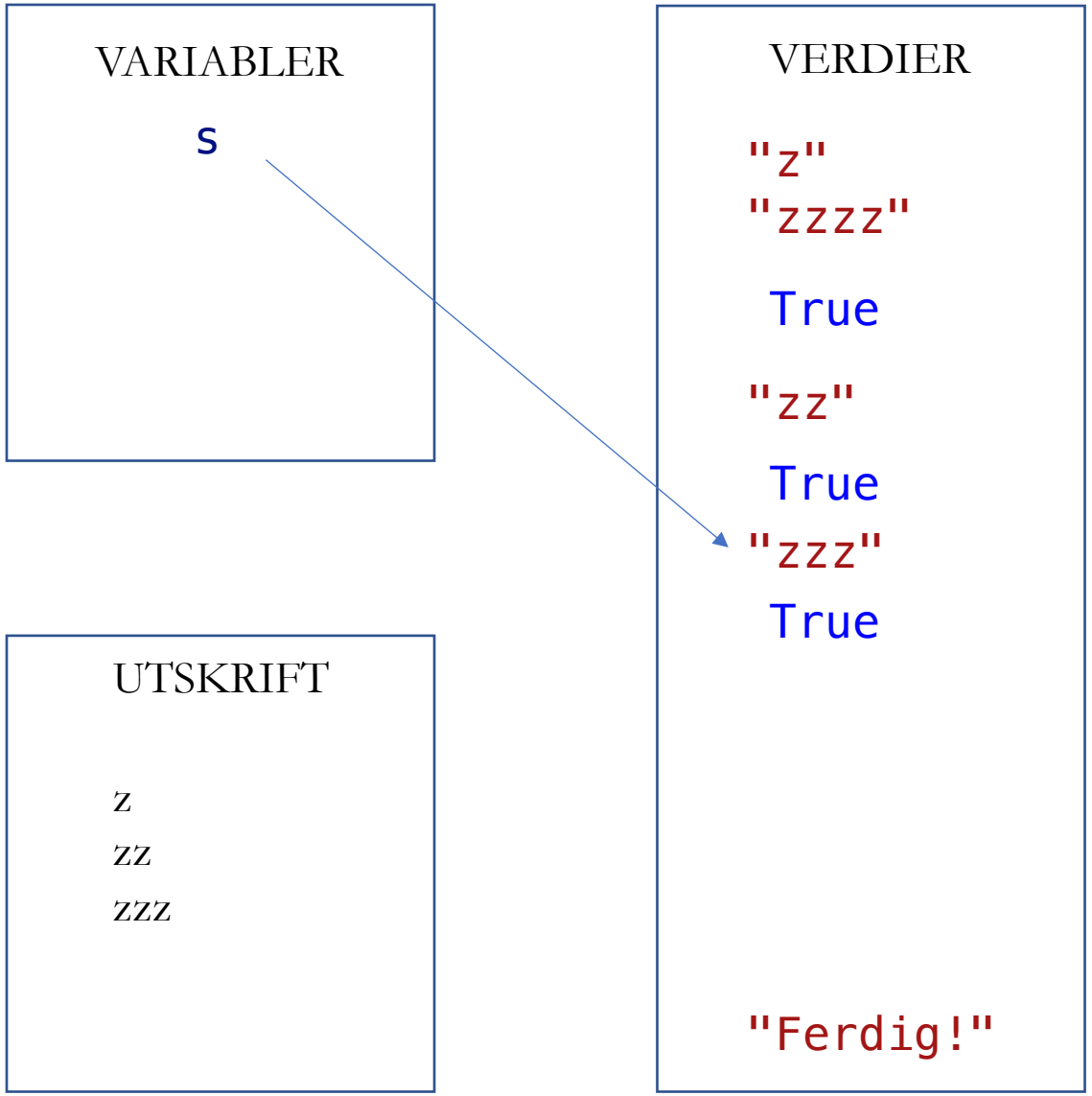
WHILE

```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```



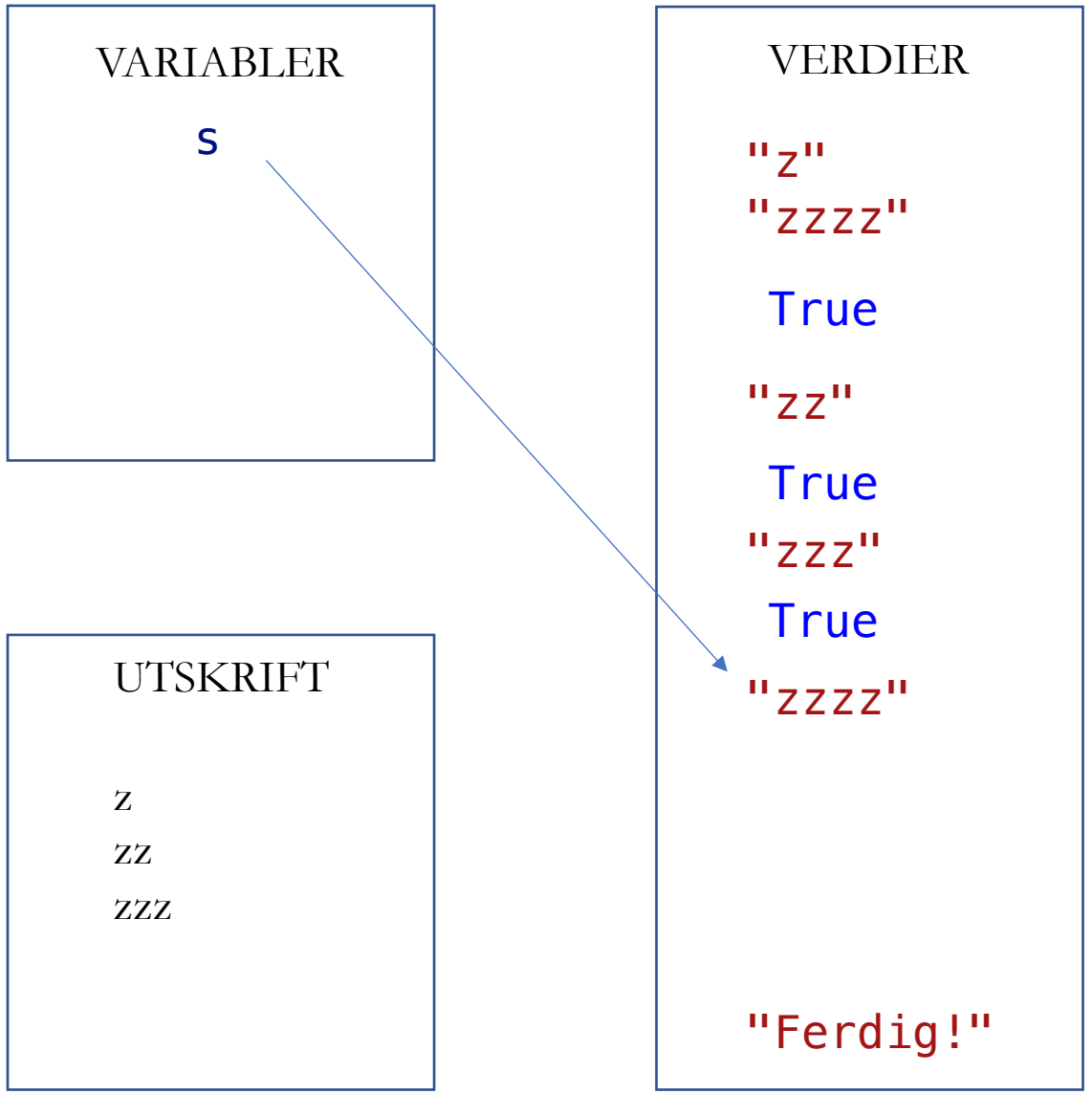
WHILE

```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```



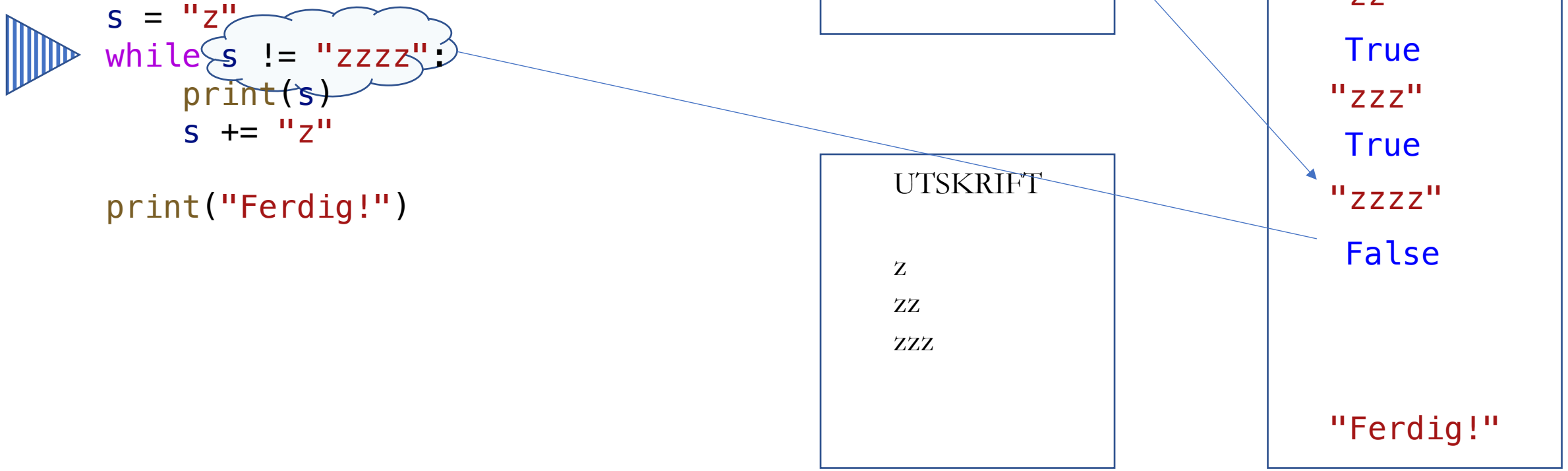
WHILE

```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```



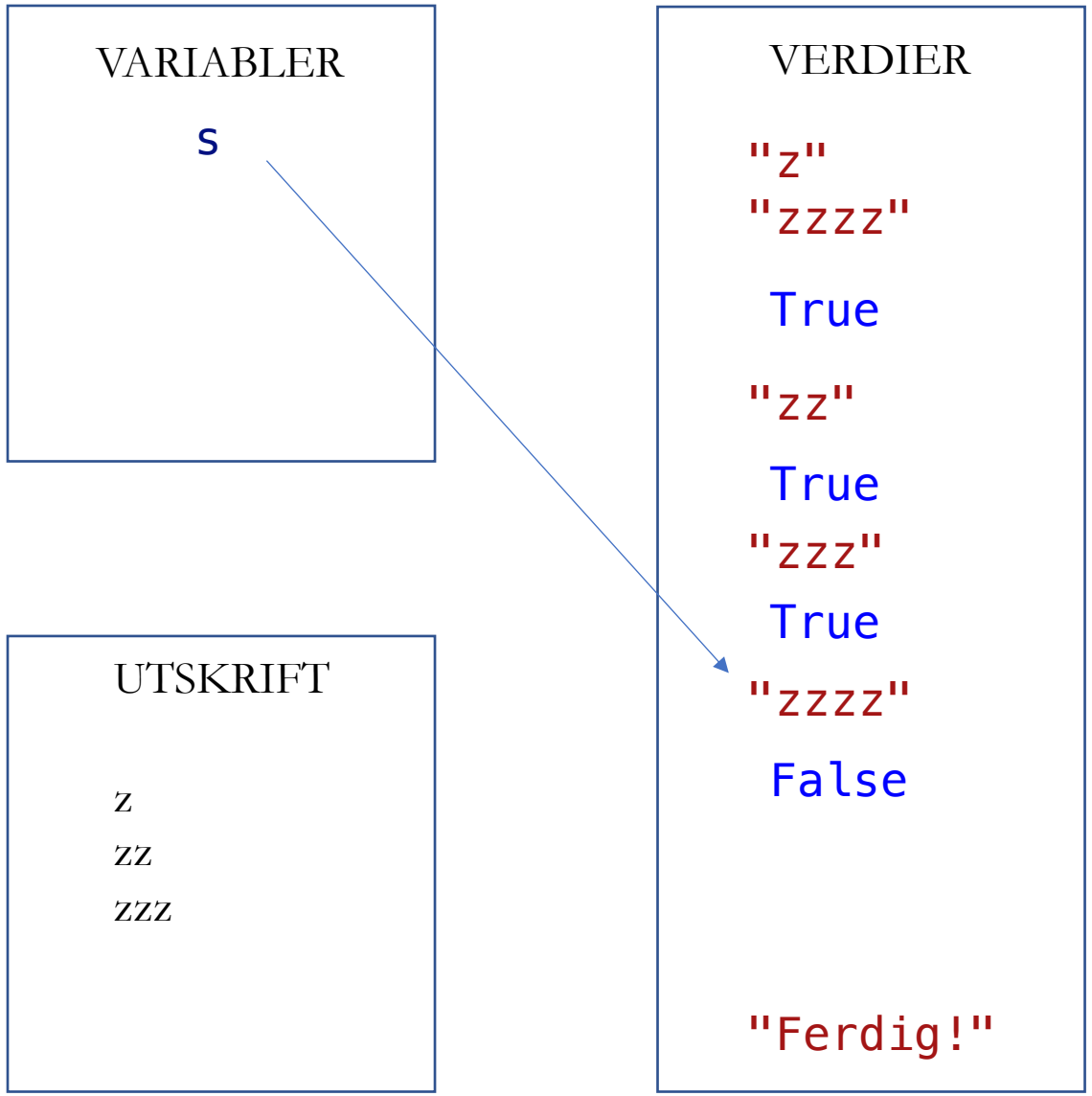
WHILE

```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```



WHILE

```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```



WHILE

```
s = "z"
while s != "zzzz":
    print(s)
    s += "z"

print("Ferdig!")
```