UNIVERSITETET I BERGEN

FUNKSJONER

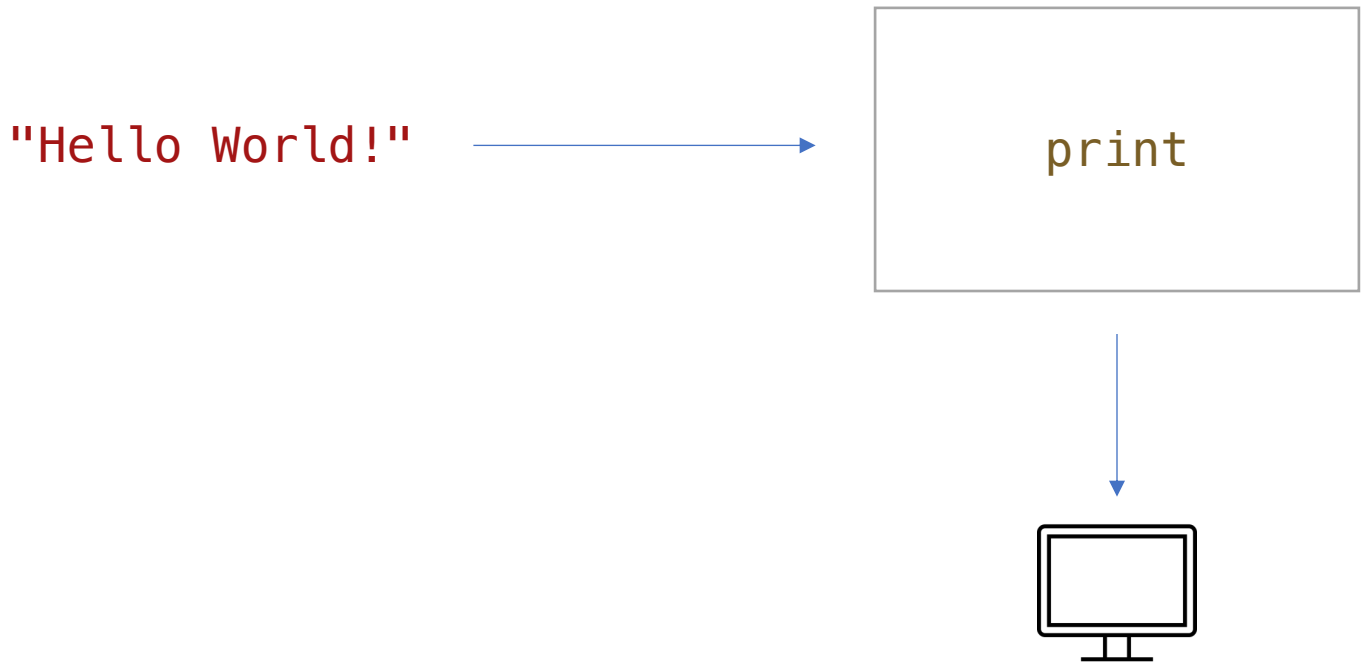
INF100

HØST 2023

Torstein Strømme og David Grellscheid

FUNKSJONER

```
print("Hello World!")
```



FUNKSJONER MED RETURVERDI

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

FUNKSJONER MED RETURVERDI

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

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

Å DEFINERE EN FUNKSJON

Vi *definerer* en funksjon som heter print_twice

Mellom parentesene er *parametre* (input til funksjonen)

```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)
```

Kolon

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

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

```
print_twice("Hello World!")
```

EKSEMPELKJØRING

```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)

print_twice("foo")

x = "bar"
print_twice(x + "!")
```

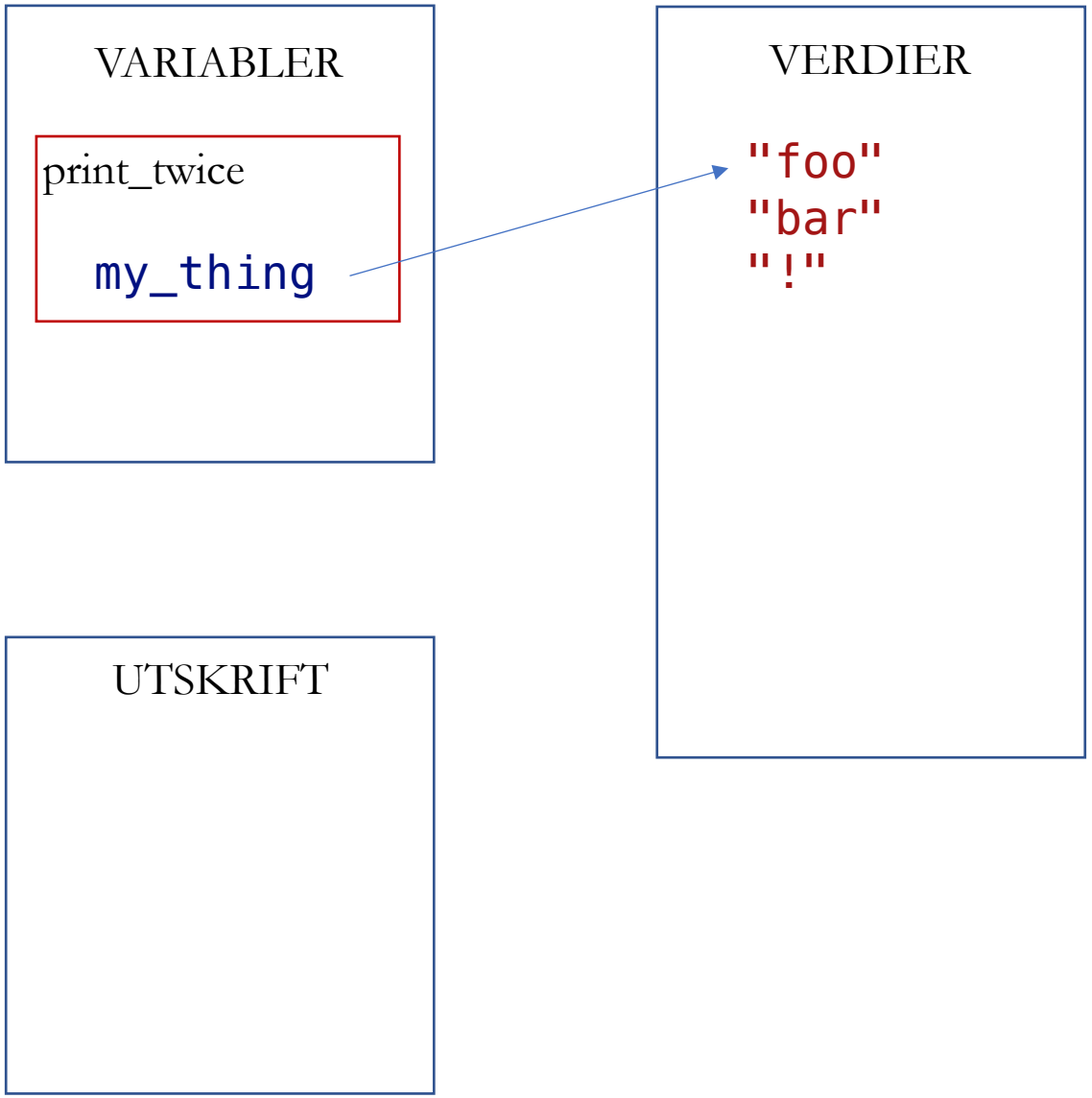
VARIABLER

UTSKRIFT

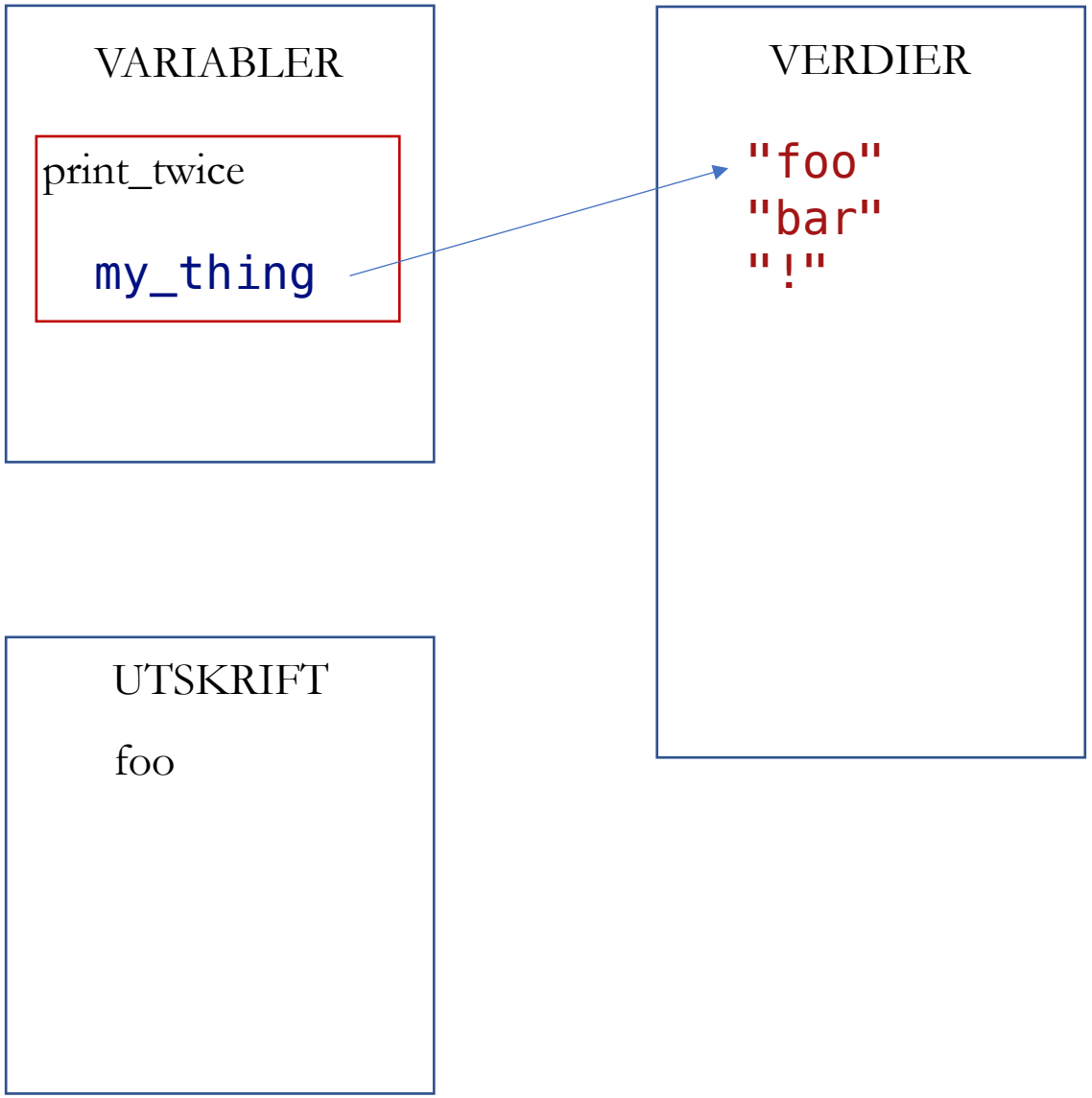
VERDIER

"foo"
"bar"
"!"

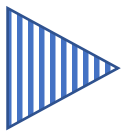
```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)

print_twice("foo")

x = "bar"
print_twice(x + "!")
```

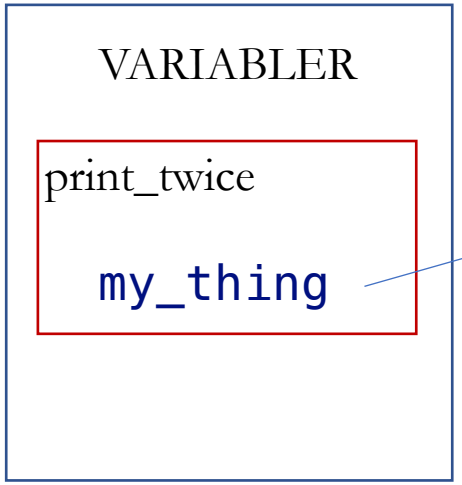


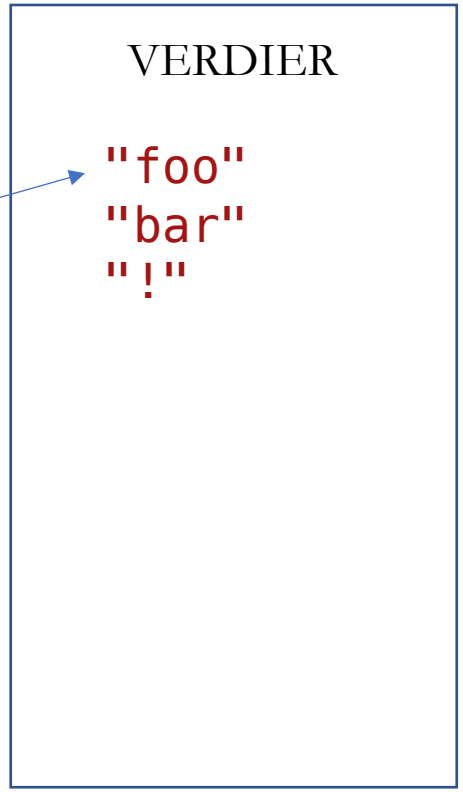
```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)

print_twice("foo")

x = "bar"
print_twice(x + "!")
```

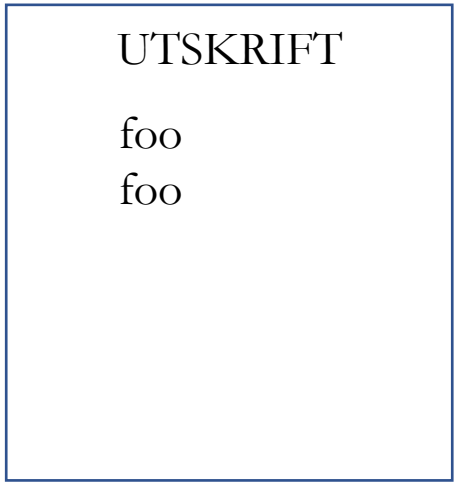


```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)

print_twice("foo")

x = "bar"
print_twice(x + "!")
```









```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)

print_twice("foo")

x = "bar"
print_twice(x + "!")
```

VARIABLER

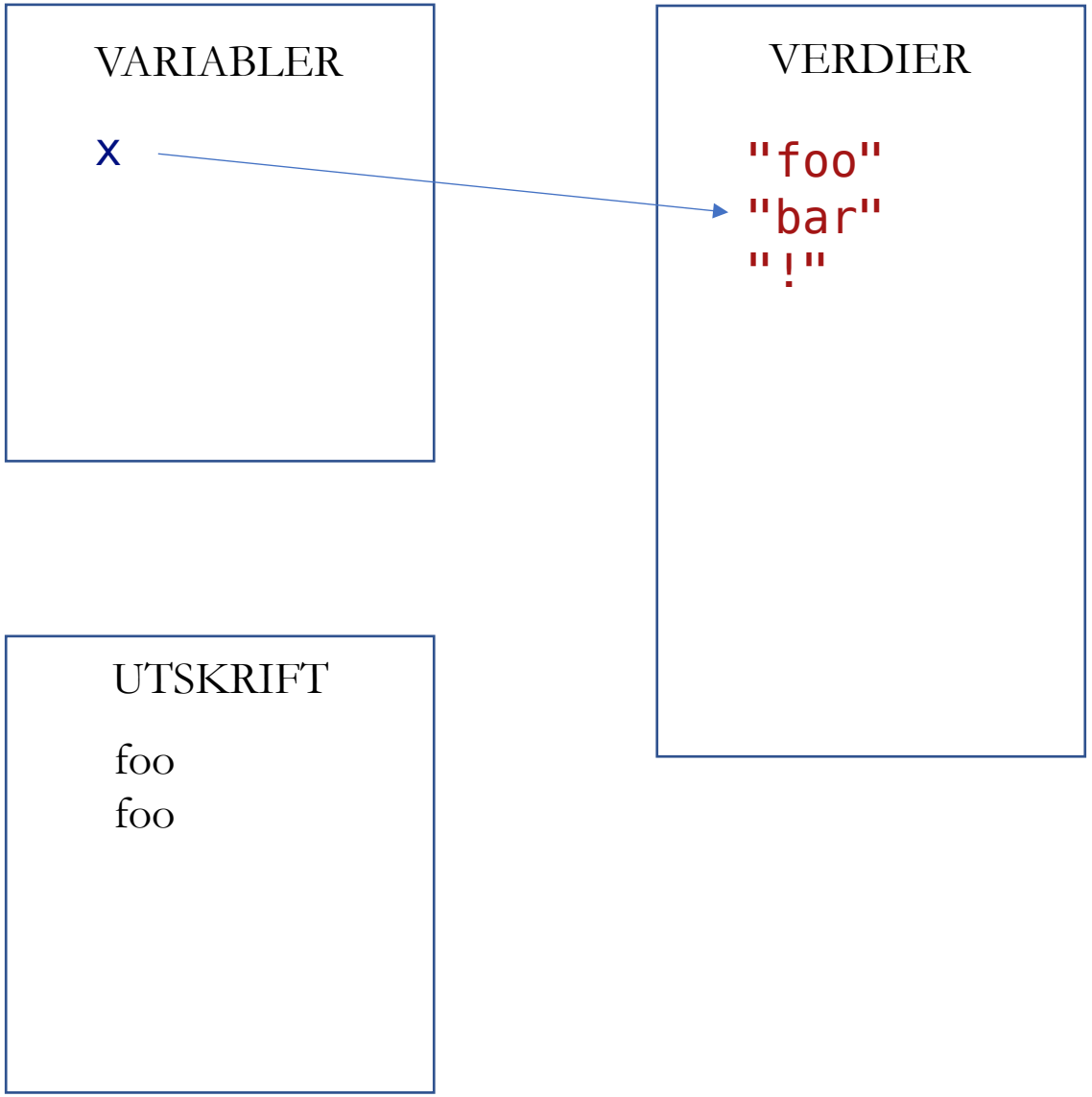
VERDIER

"foo"
"bar"
"!"

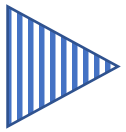
UTSKRIFT

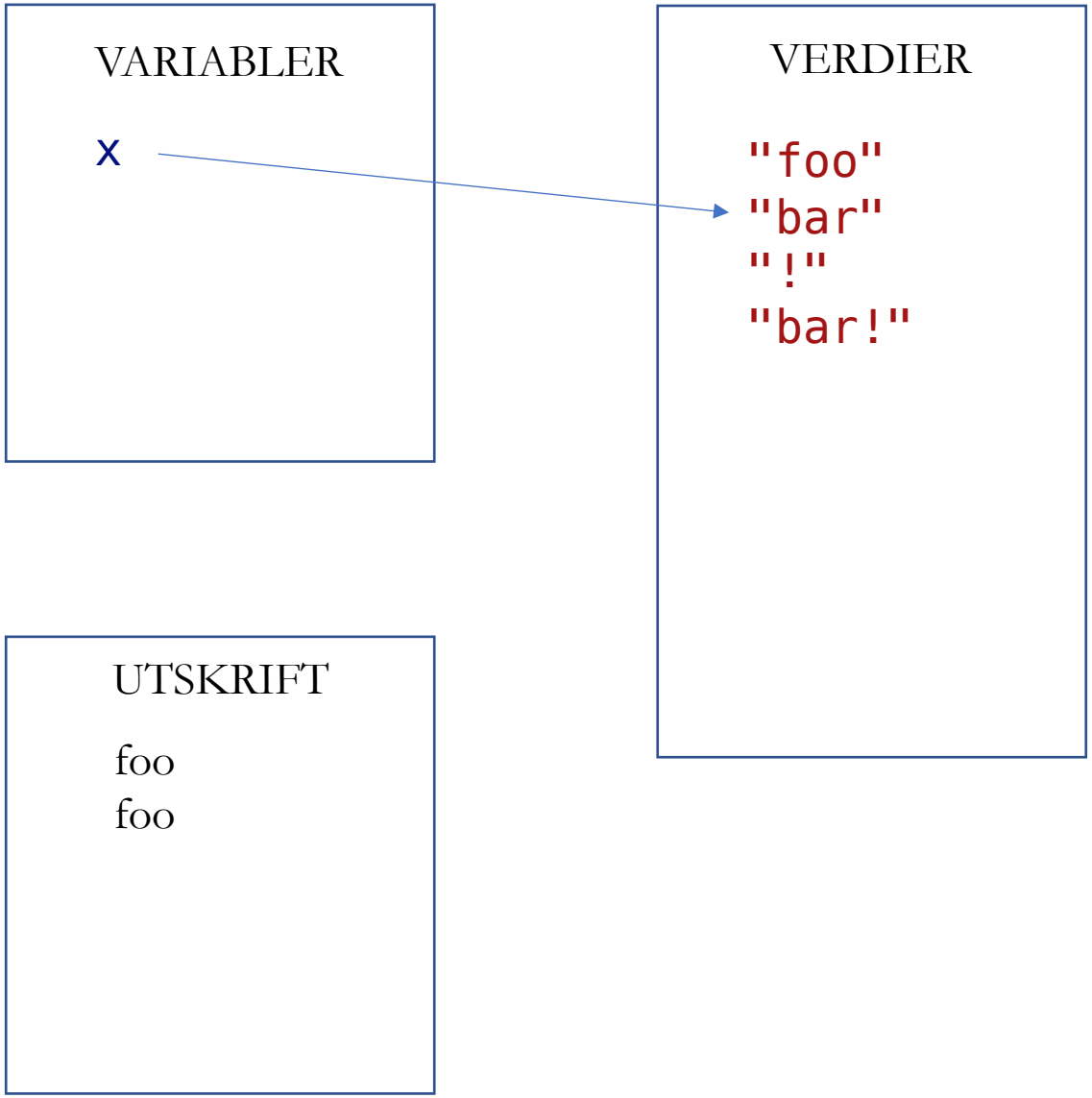
foo
foo

```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)

print_twice("foo")

x = "bar"
print_twice(x + "!")
```

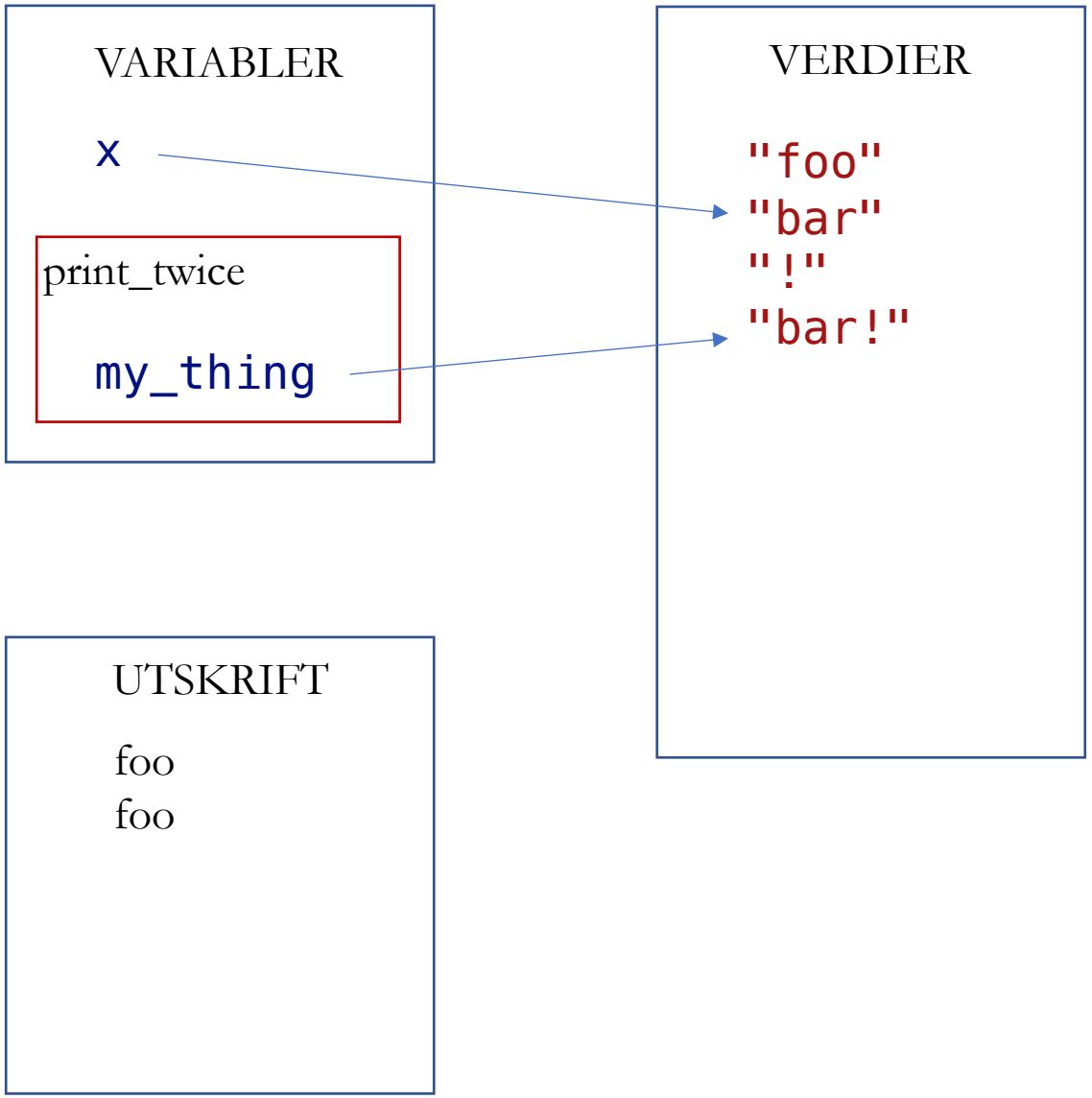


```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)

print_twice("foo")

x = "bar"
print_twice(x + "!")
```

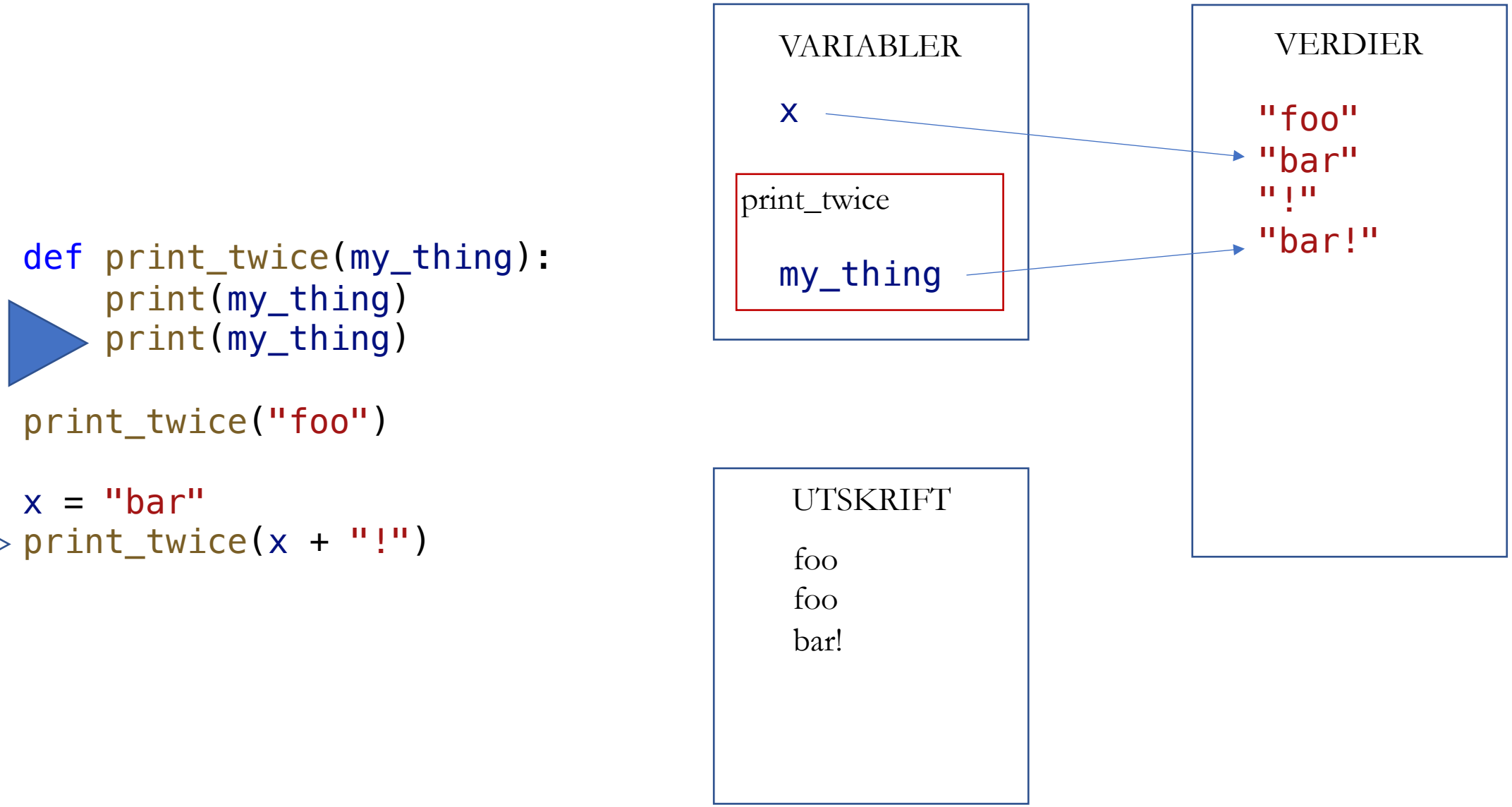




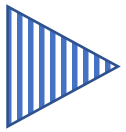
```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)

print_twice("foo")

x = "bar"
print_twice(x + "!")
```

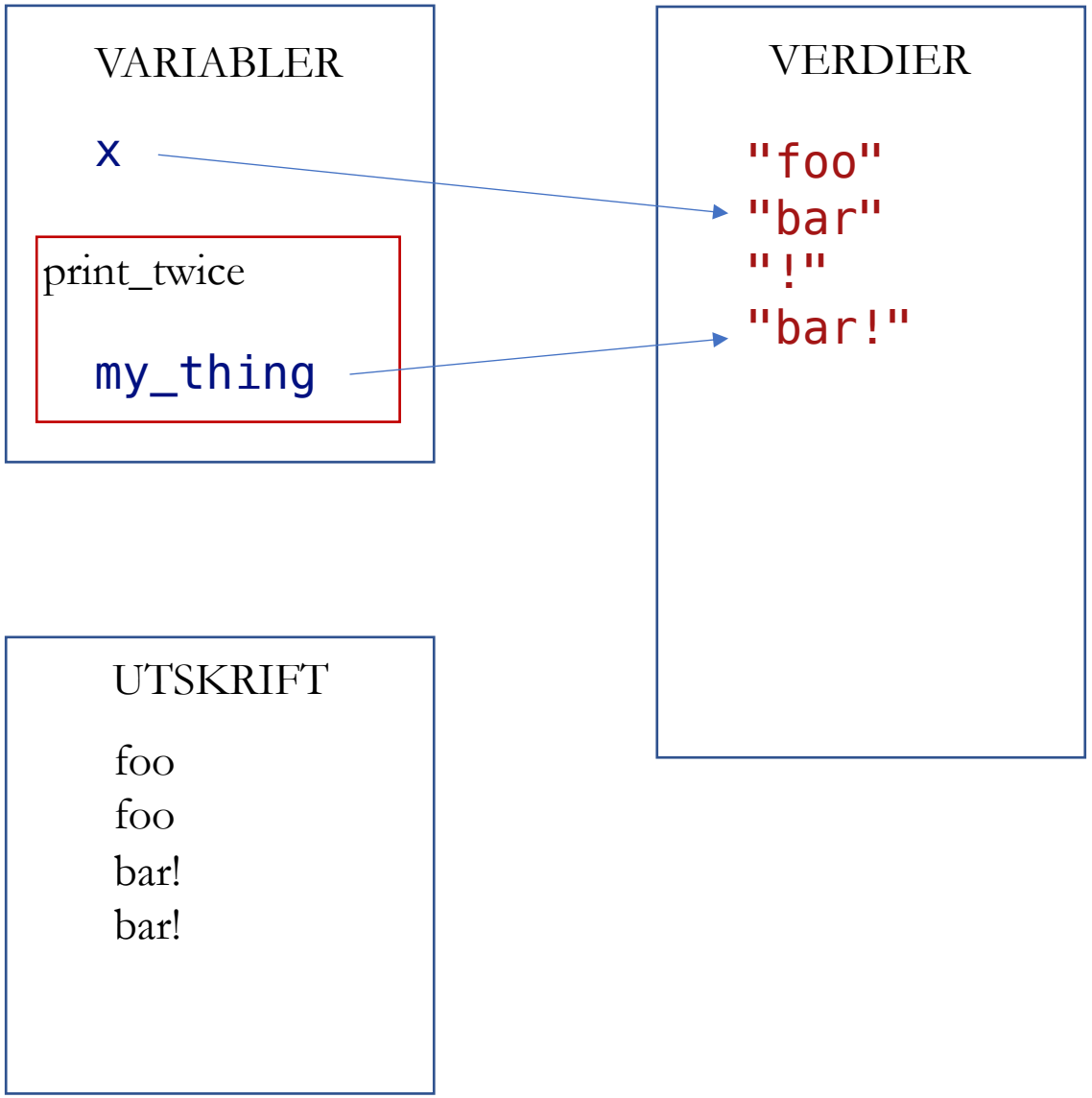


```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)

print_twice("foo")

x = "bar"
print_twice(x + "!")
```

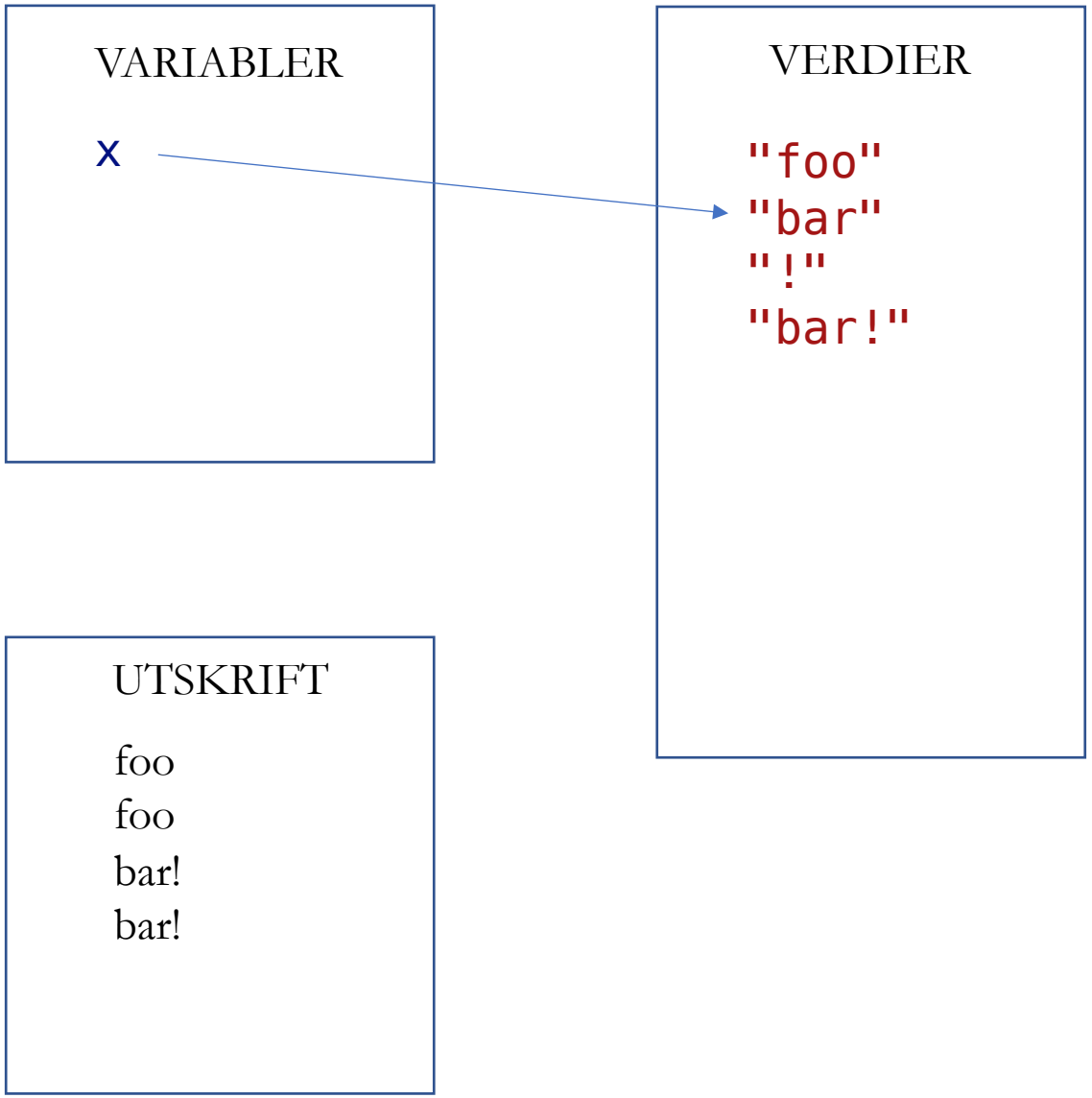


```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)

print_twice("foo")

x = "bar"
print_twice(x + "!")
```



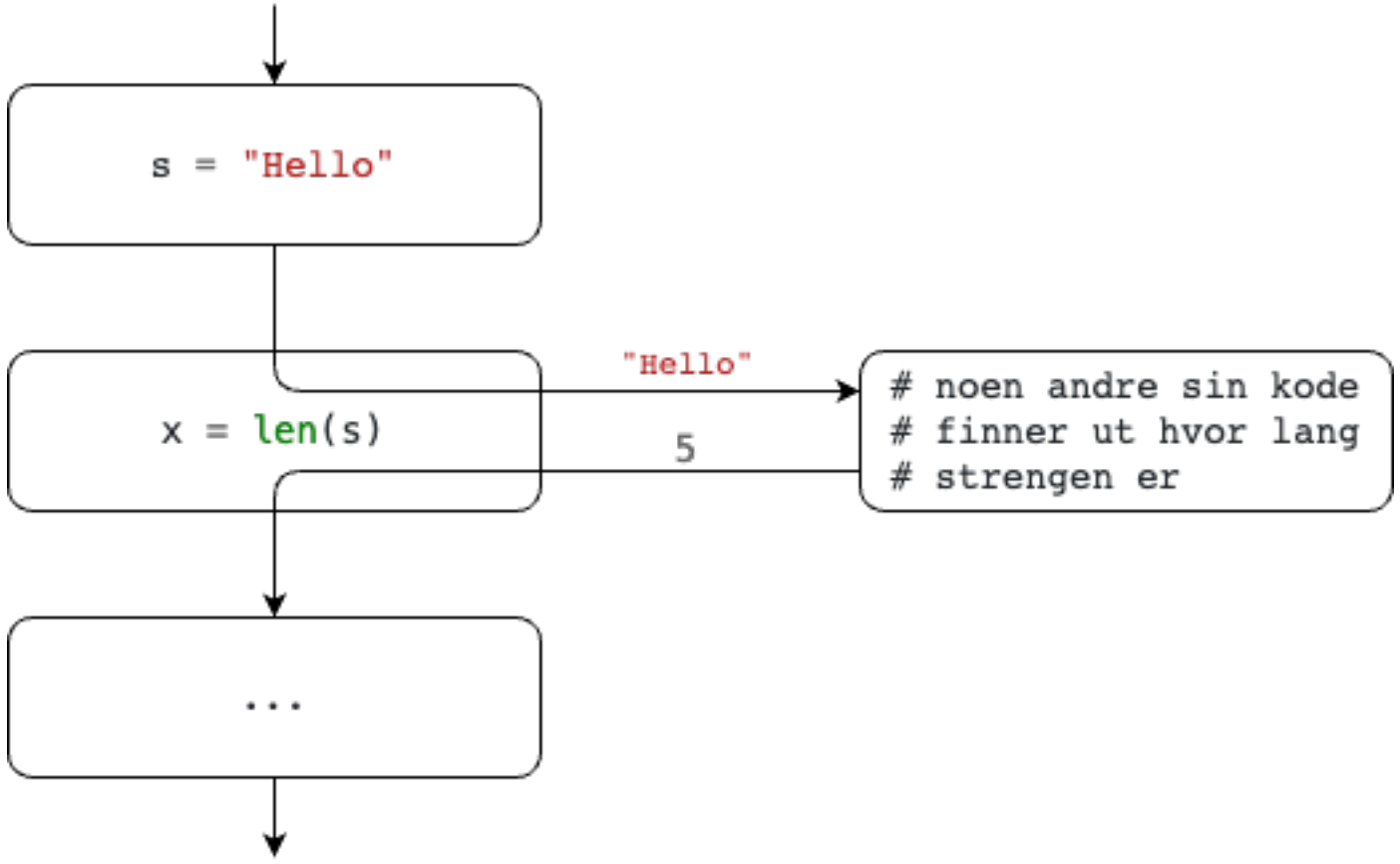


```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)

print_twice("foo")

x = "bar"
print_twice(x + "!")
```



RETURVERDI

```
s = "Hello"
x = len(s)

print(2 * x)
```



RETURVERDI

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

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

print(c)
```

PRINT ≠ RETURVERDI

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

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

print(c)
```

LIVEKODING

- Les in lengde, bredde, høyde (cm)
- Skriv ut volumen av boksen (cm^3)
- Massetetthetene er:

gull	19.32 g/cm^3
jern	7.87 g/cm^3
vann	0.997 g/cm^3

- Skriv ut massen til boksen (g)

ORDBOK

```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)


print_twice("foo")
```

signatur. Navnet på funksjonen og hvilke parametre den har.

ORDBOK

parameter. En variabel som tilordnes verdi når funksjonen kalles.

signatur. Navnet på funksjonen og hvilke parametre den har.

```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)


print_twice("foo")
```

ORDBOK

parameter. En variabel som tilordnes verdi når funksjonen kalles.

signatur. Navnet på funksjonen og hvilke parametre den har.

```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)

print_twice("foo")
```

funksjonskropp. Setningene som utføres når funksjonen kalles. Må ha innrykk.

ORDBOK

parameter. En variabel som tilordnes verdi når funksjonen kalles.

signatur. Navnet på funksjonen og hvilke parametre den har.

funksjonskropp. Setningene som utføres når funksjonen kalles. Må ha innrykk.

```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)

print_twice("foo")
```

kall. Et funksjonskall instruerer at funksjonskroppen skal utføres og angir hvilke argumenter som fyller parametrene.

ORDBOK

parameter. En variabel som tilordnes verdi når funksjonen kalles.

signatur. Navnet på funksjonen og hvilke parametre den har.

funksjonskropp. Setningene som utføres når funksjonen kalles. Må ha innrykk.

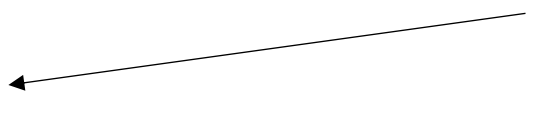
```
def print_twice(my_thing):
    print(my_thing)
    print(my_thing)

print_twice("foo")
```

argument. En verdi som fyller en parameter når funksjonen kalles.

kall. Et funksjonskall instruerer at funksjonskroppen skal utføres og angir hvilke argumenter som fyller parametrene.

ORDBOK

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

a = 5
b = incremented_thrice(a)
```



retursetning. Avslutter funksjonen. Returverdien bestemmes av hva uttrykket evaluerer til.

ORDBOK

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

a = 5
b = incremented_thrice(a)
```

8

retursetning. Avslutter funksjonen. Returverdien bestemmes av hva uttrykket evaluerer til.

returverdi. Verdien et funksjonskall evaluerer til.

UNIVERSITETET I BERGEN

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

```
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
```