

MUTASJON OG ALIAS

INF100

HØST 2026

Torstein Strømme

'dette er en streng'

egentlig bare en samling med tegn

```
['d', 'e', 't', 't', 'e', ' ', 'e', 'r', ' ', 'e', 'n', ' ', 'l', 'i', 's', 't', 'e']
```

egentlig bare en samling med ting

list versus str: likheter

`x = ['b', 'a', 'r']` `x = 'bar'`

- Indeksering
- Beskjæring
- Løkker
- Operasjoner
 - in, not in, +, *
- Funksjoner og metoder
 - len, min, max
 - .count, .index

`x[0]`

`x[1:]`

`for el in x:`
 `...`

`'a' in x` `x * 2`

`'foo' + x`

`['f', 'o', 'o'] + x`

`len(x)`

`min(x)`

`max(x)`

`x.count('a')`

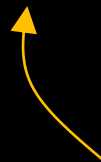
`x.index('b')`

list versus str: forskjeller

`x = ['b', 'a', 'r']` `x = 'bar'`

- Lister kan ha andre ting enn bare skrifttegn
- Lister kan muteres

`x[0] = 'f'`



Tilordning krasjer hvis x er en streng

Før: vi har sett tilordning med *variabel* på venstre side

`x = 42`

Nå: vi kan også tilorde med *posisjon i en liste* på venstre side

`a[0] = 42`

ENDRE FØRSTE BOKSTAV

Streng

```
s = 'abc'  
s = 'x' + s[1:]
```

```
print(s) # xbc
```

Mutasjon av strenger
er ikke mulig

Liste

```
a = ['a', 'b', 'c']  
a = ['x'] + a[1:]
```

```
print(a) # ['x', 'b', 'c']
```

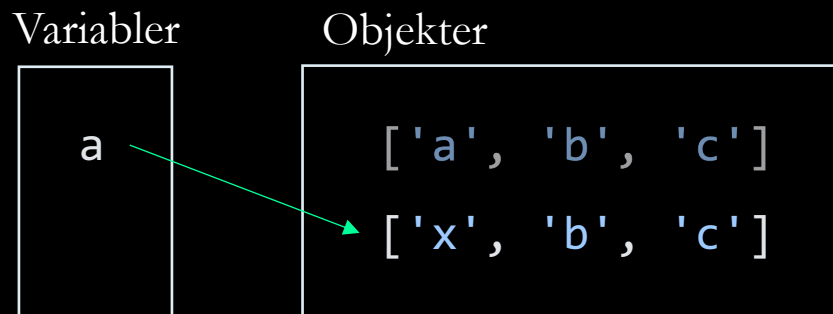
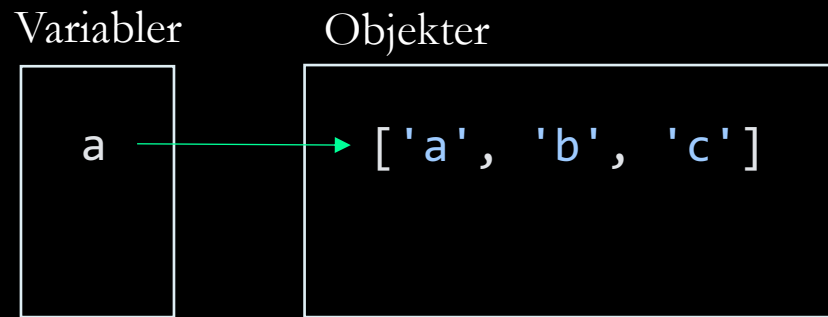
Ikke-mutasjon: variabelen
peker på nytt objekt

```
a = ['a', 'b', 'c']  
a[0] = 'x'
```

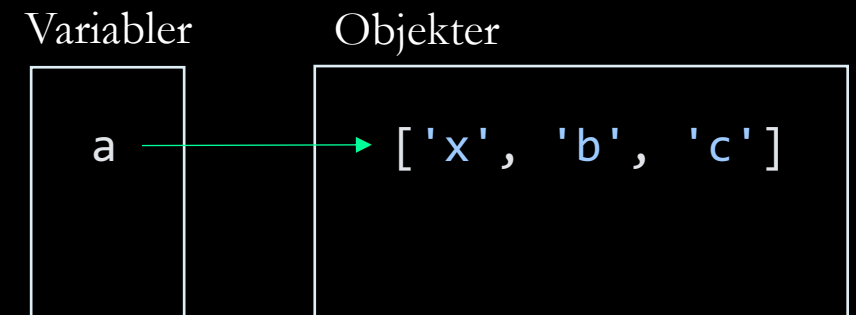
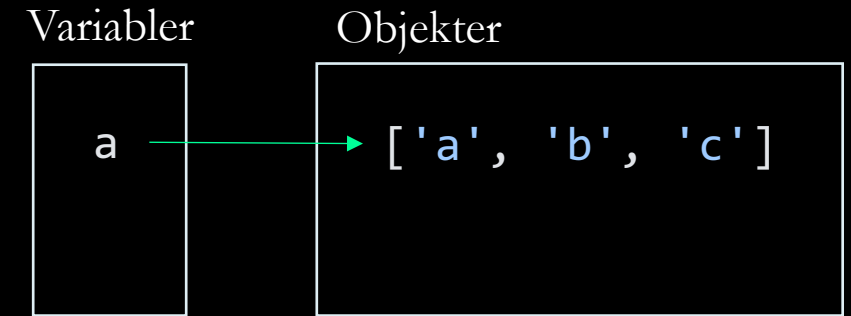
```
print(a) # ['x', 'b', 'c']
```

Mutasjon: selve listen
har blitt endret

```
→ a = ['a', 'b', 'c']  
a = ['x'] + a[1:]  
  
print(a) # ['x', 'b', 'c']
```



```
→ a = ['a', 'b', 'c']  
a[0] = 'x'  
  
print(a) # ['x', 'b', 'c']
```

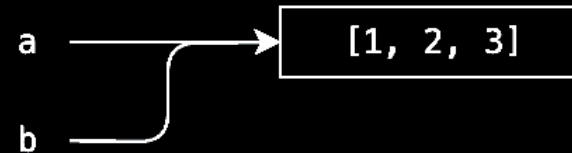


ALIAS

To variabler er *aliaser* dersom de peker på det samme objektet.

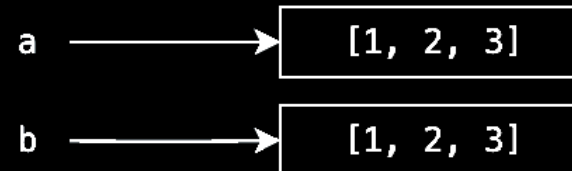
For eksempel,

```
a = [1, 2, 3]  
b = a
```



vil innebære at **a** og **b** er aliaser, mens

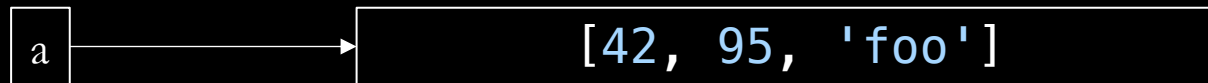
```
a = [1, 2, 3]  
b = [1, 2, 3]
```



vil innebære at **a** og **b** ikke er aliaser. Selv om objektene i det sistnevnt tilfelle er *like*, er det likevel hver sine objekter med ulik plassering i minnet.

→ `a = [42, 95, 'foo']`
`b = a`
`b[1] = 'bar'`

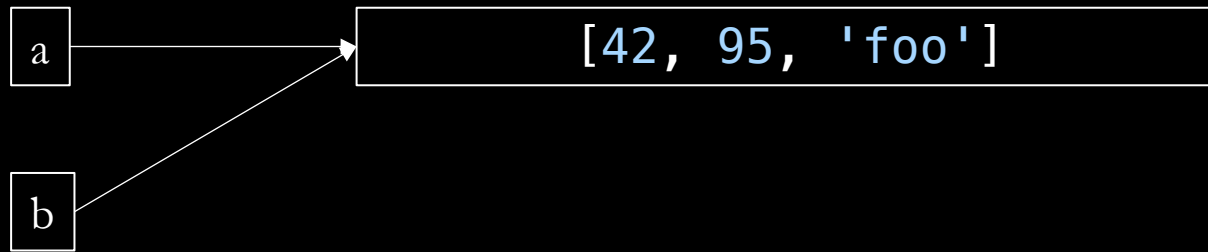
→ a = [42, 95, 'foo']
b = a
b[1] = 'bar'



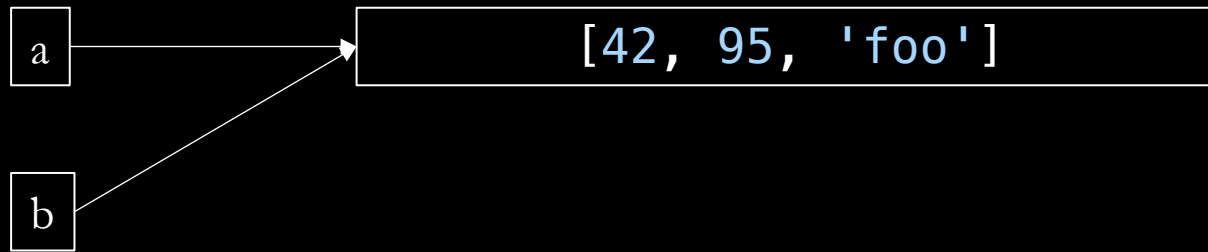
```
→ a = [42, 95, 'foo']  
   b = a  
   b[1] = 'bar'
```



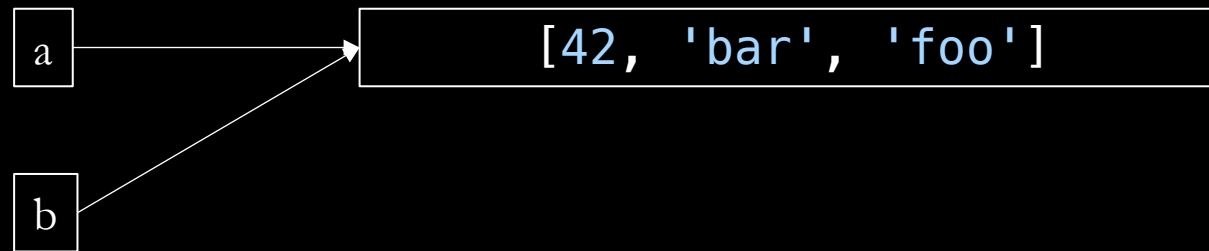
→
a = [42, 95, 'foo']
b = a
b[1] = 'bar'



```
a = [42, 95, 'foo']  
b = a  
→ b[1] = 'bar'
```



```
a = [42, 95, 'foo']  
b = a  
→ b[1] = 'bar'
```



Hvis et objekt med alias blir mutert: pass på!!

HVORDAN OPPSTÅR ALIAS?

```
foo = [1, 2, 3]
```

```
bar = foo
```

HVORDAN OPPSTÅR ALIAS?

```
foo = [1, 2, 3]
```

```
bar = foo
```

foo → [1, 2, 3]

bar ↗

HVORDAN OPPSTÅR ALIAS?

```
foo = [1, 2, 3]  
bar = foo
```

```
foo = {  
    "wha": [1, 2, 3],  
    "chi": [4, 5, 6]  
}  
bar = foo["wha"]
```

HVORDAN OPPSTÅR ALIAS?

```
foo = [1, 2, 3]
bar = foo
```

```
foo = {
    "wha": [1, 2, 3],
    "chi": [4, 5, 6]
}
bar = foo["wha"]
```

```
def fun(par):
    ...

foo = [1, 2, 3]
fun(foo)
```

HVORDAN OPPSTÅR ALIAS?

```
foo = [1, 2, 3]
bar = foo
```

```
foo = {
    "wha": [1, 2, 3],
    "chi": [4, 5, 6]
}
bar = foo["wha"]
```

```
def fun(par):
    ...

foo = [1, 2, 3]
fun(foo)
```

```
foo = [[1, 2, 3], [4, 5, 6]]

for el in foo:
    ...
```

HVORDAN OPPSTÅR ALIAS?

```
foo = [1, 2, 3]
bar = foo
```

```
def fun(par):
    ...

foo = [1, 2, 3]
fun(foo)
```

```
foo = {
    "wha": [1, 2, 3],
    "chi": [4, 5, 6]
}
bar = foo["wha"]
```

```
foo = [[1, 2, 3], [4, 5, 6]]

for el in foo:
    ...
```

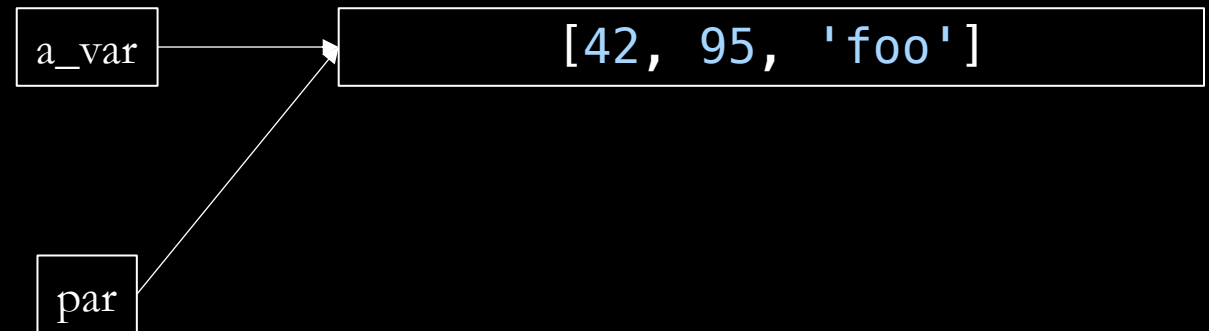
```
def fun(par):
    return par["chi"]

foo = {
    "wha": [1, 2, 3],
    "chi": [4, 5, 6]
}
bar = fun(foo)
```


ALIAS MED FUNKSJONSPARAMETRE

```
def my_function(par):  
    par[1] = 'bar'  
  
a_var = [42, 95, 'foo']  
my_function(a_var)  
print(a_var)
```

par = a_var



DESTRUKTIVITET

- En funksjon er *destruktiv* dersom den muterer en av sine argumenter

Husk: Et **argument** er et objekt som pekes på av en parameter når funksjonen begynner

EKSEMPEL

En funksjon som runder av flyttall i en liste til heltall

[1.9, 3.2, 9.9]



[2, 3, 10]

EKSEMPEL

En funksjon som runder av flyttall i en liste til heltall

[1.9, 3.2, 9.9]

[2, 3, 10]

rounded_to_int

Returverdi



EKSEMPEL

En funksjon som runder av flyttall i en liste til heltall

[1.9, 3.2, 9.9]

round_to_int

[2, 3, 10]

Effekt
(mutasjon)





rounded_to_int

```
def rounded_to_int(a):  
    result = []  
    for val in a:  
        rounded_val = round(val)  
        result.append(rounded_val)  
    return result
```

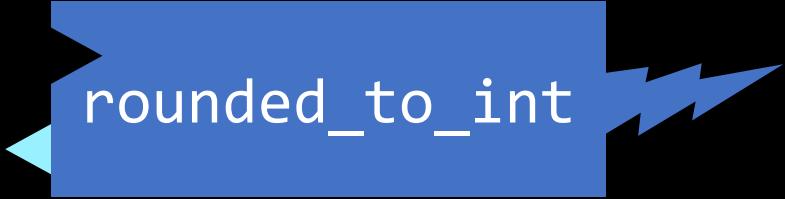
ikke-destruktiv



round_to_int

```
def round_to_int(a):  
    for i in range(len(a)):  
        val = a[i]  
        rounded_val = round(val)  
        a[i] = rounded_val
```

destruktiv



rounded_to_int

Variabler

Objekter

```
def rounded_to_int(a):  
    result = []  
    for val in a:  
        rounded_val = round(val)  
        result.append(rounded_val)  
    return result
```

```
→ nums = [1.9, 3.2, 9.9]  
org_nums = nums  
nums = rounded_to_int(nums)
```



rounded_to_int

```
def rounded_to_int(a):  
    result = []  
    for val in a:  
        rounded_val = round(val)  
        result.append(rounded_val)  
    return result
```

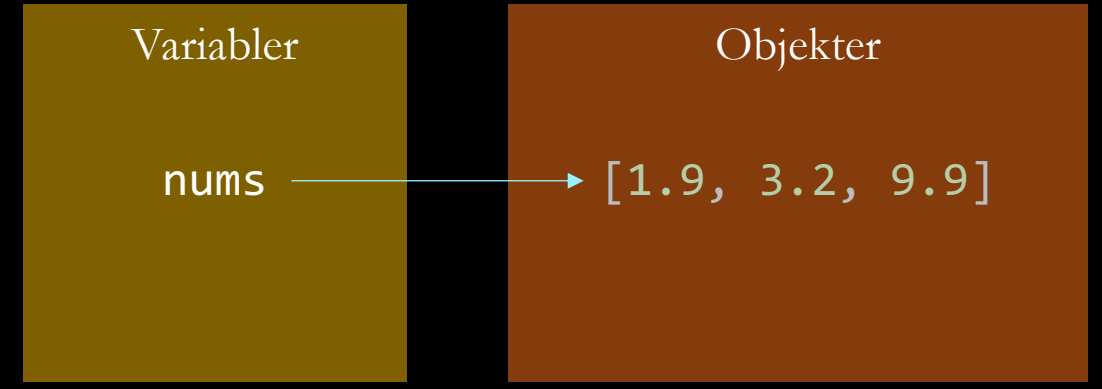
```
nums = [1.9, 3.2, 9.9]  
→ org_nums = nums  
nums = rounded_to_int(nums)
```

Variabler

nums

Objekter

[1.9, 3.2, 9.9]



```
graph LR; A[Variabler] --> B[Objekter]; A --- C[nums]; B --- D["[1.9, 3.2, 9.9]"]
```

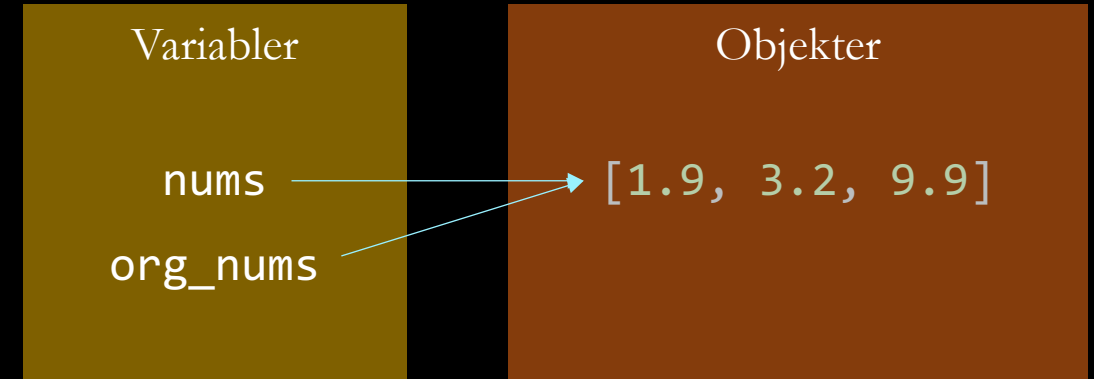
rounded_to_int

```
def rounded_to_int(a):  
    result = []  
    for val in a:  
        rounded_val = round(val)  
        result.append(rounded_val)  
    return result
```

```
nums = [1.9, 3.2, 9.9]
```

```
org_nums = nums
```

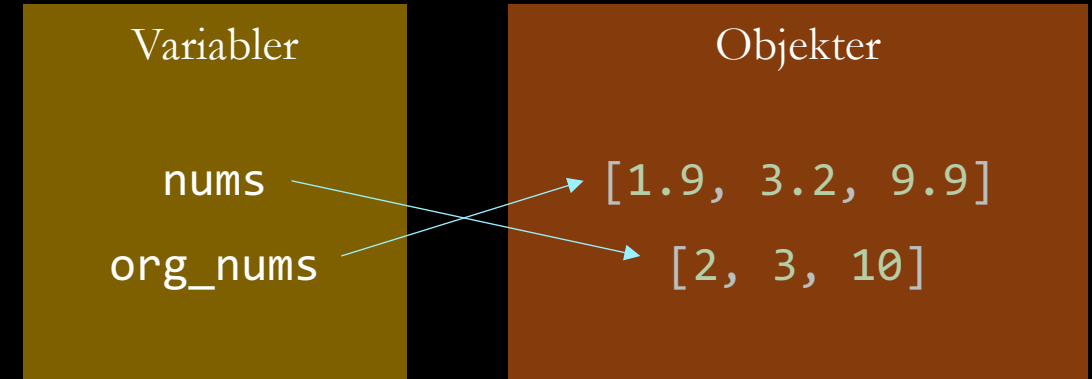
```
→ nums = rounded_to_int(nums)
```



rounded_to_int

```
def rounded_to_int(a):  
    result = []  
    for val in a:  
        rounded_val = round(val)  
        result.append(rounded_val)  
    return result
```

```
nums = [1.9, 3.2, 9.9]  
org_nums = nums  
nums = rounded_to_int(nums)
```



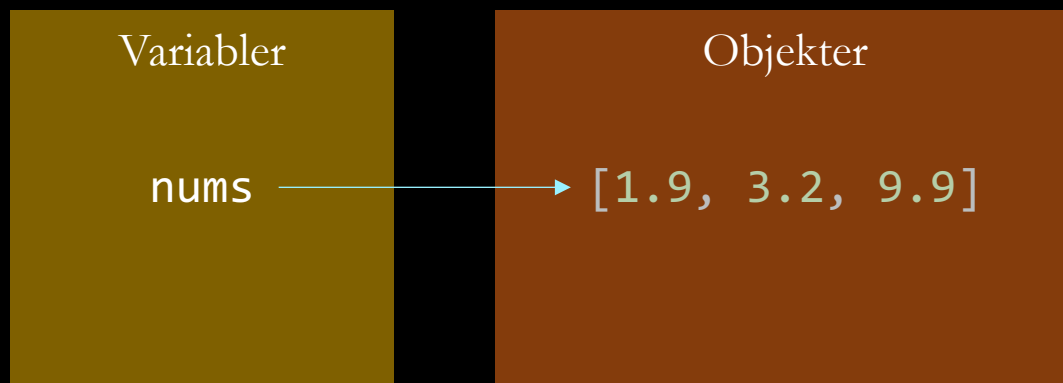
Variabler

Objekter

round_to_int

```
def round_to_int(a):  
    for i in range(len(a)):  
        val = a[i]  
        rounded_val = round(val)  
        a[i] = rounded_val
```

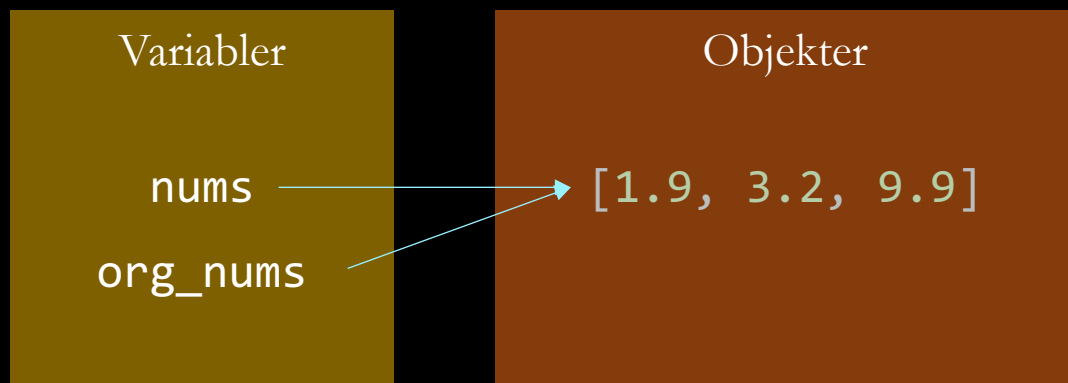
```
→ nums = [1.9, 3.2, 9.9]  
org_nums = nums  
round_to_int(nums)
```



`round_to_int` 

```
def round_to_int(a):  
    for i in range(len(a)):  
        val = a[i]  
        rounded_val = round(val)  
        a[i] = rounded_val
```

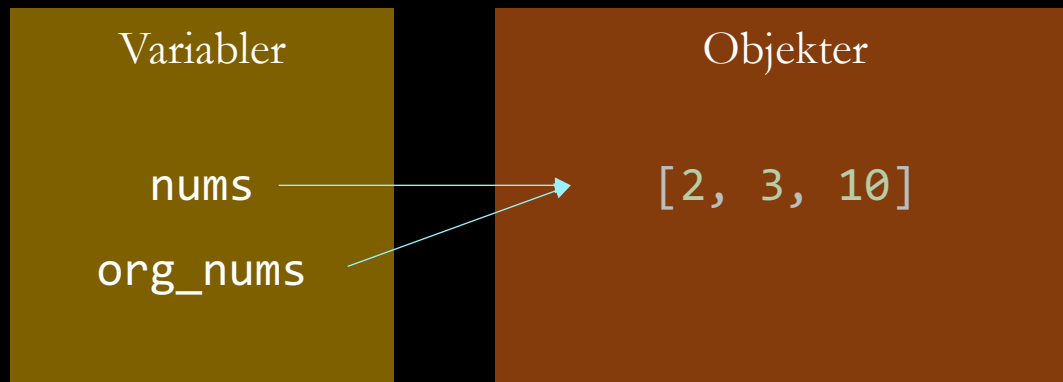
```
nums = [1.9, 3.2, 9.9]  
→ org_nums = nums  
   round_to_int(nums)
```



round_to_int ⚡

```
def round_to_int(a):  
    for i in range(len(a)):  
        val = a[i]  
        rounded_val = round(val)  
        a[i] = rounded_val
```

```
nums = [1.9, 3.2, 9.9]  
org_nums = nums  
→ round_to_int(nums)
```

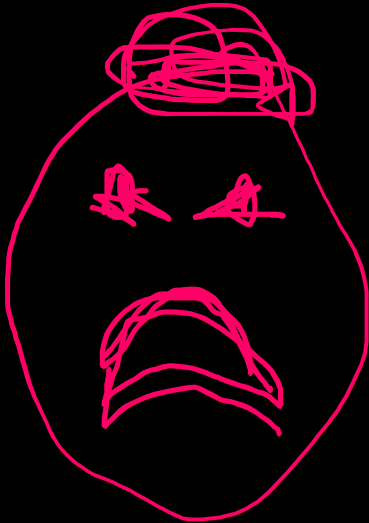
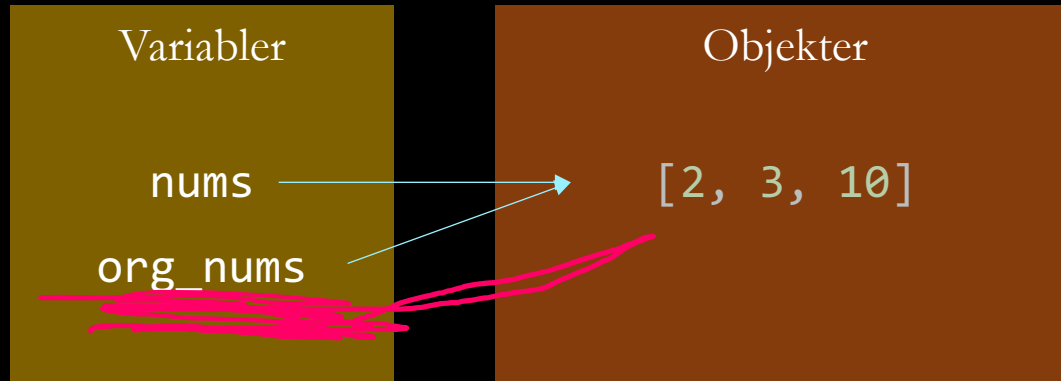


`round_to_int` 

```
def round_to_int(a):  
    for i in range(len(a)):  
        val = a[i]  
        rounded_val = round(val)  
        a[i] = rounded_val
```

```
nums = [1.9, 3.2, 9.9]  
org_nums = nums  
round_to_int(nums)
```





`round_to_int` ⚡

```
def round_to_int(a):  
    for i in range(len(a)):  
        val = a[i]  
        rounded_val = round(val)  
        a[i] = rounded_val
```

```
nums = [1.9, 3.2, 9.9]  
org_nums = nums  
round_to_int(nums)
```



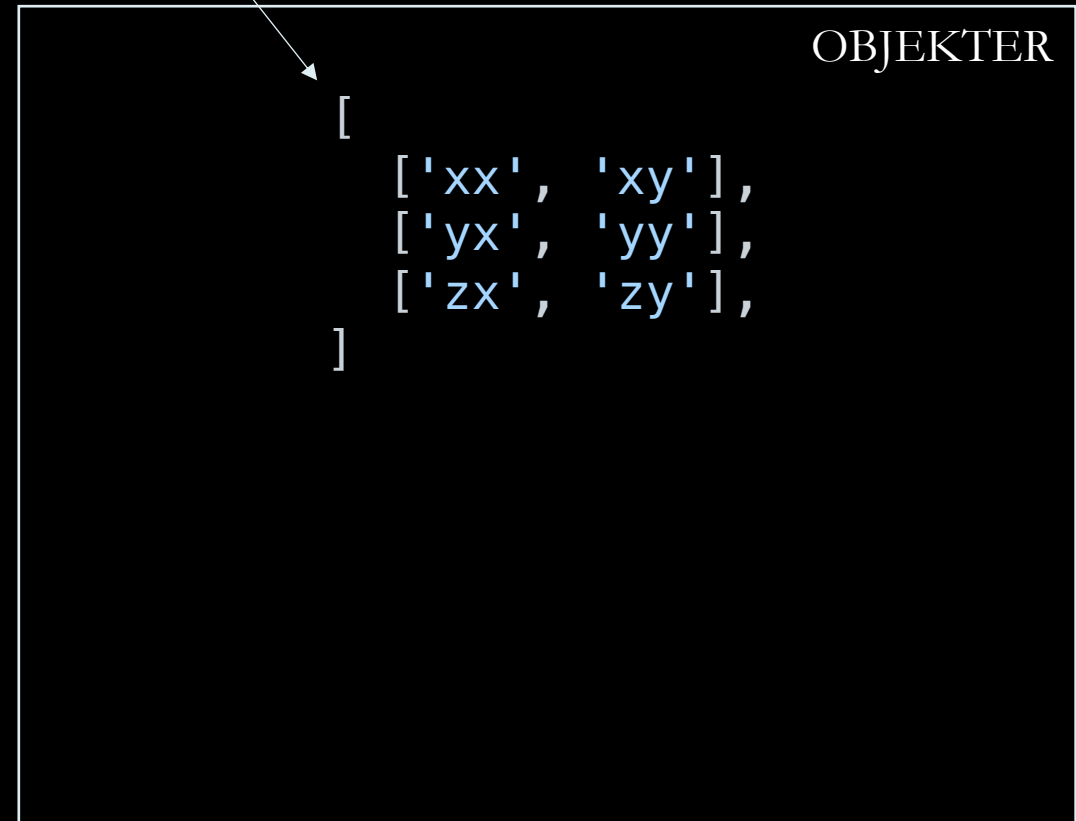
MUTASJON AV 2D-LISTER

2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```

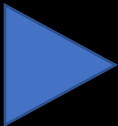


```
b = a  
b[1][0] = 'foo'  
print(a[1][0])
```

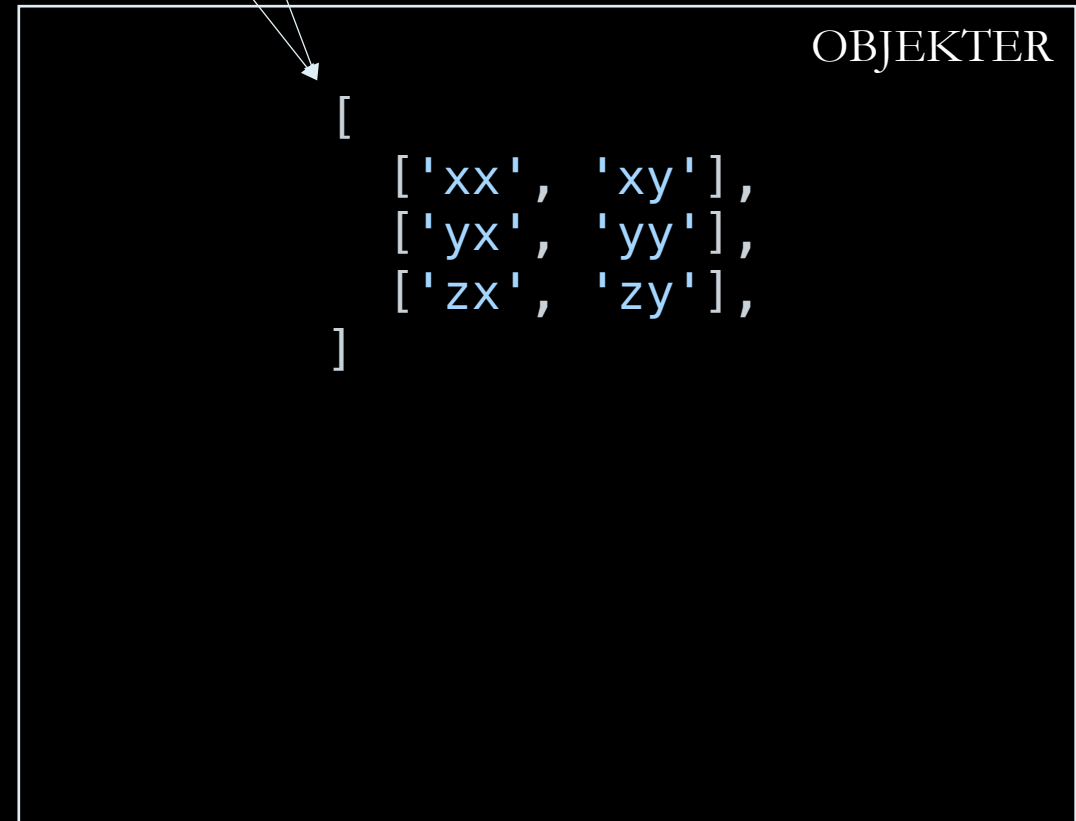
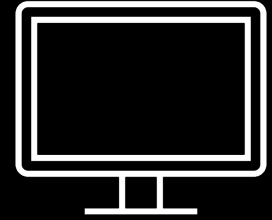
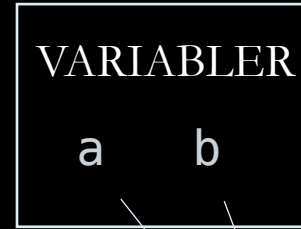


2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```



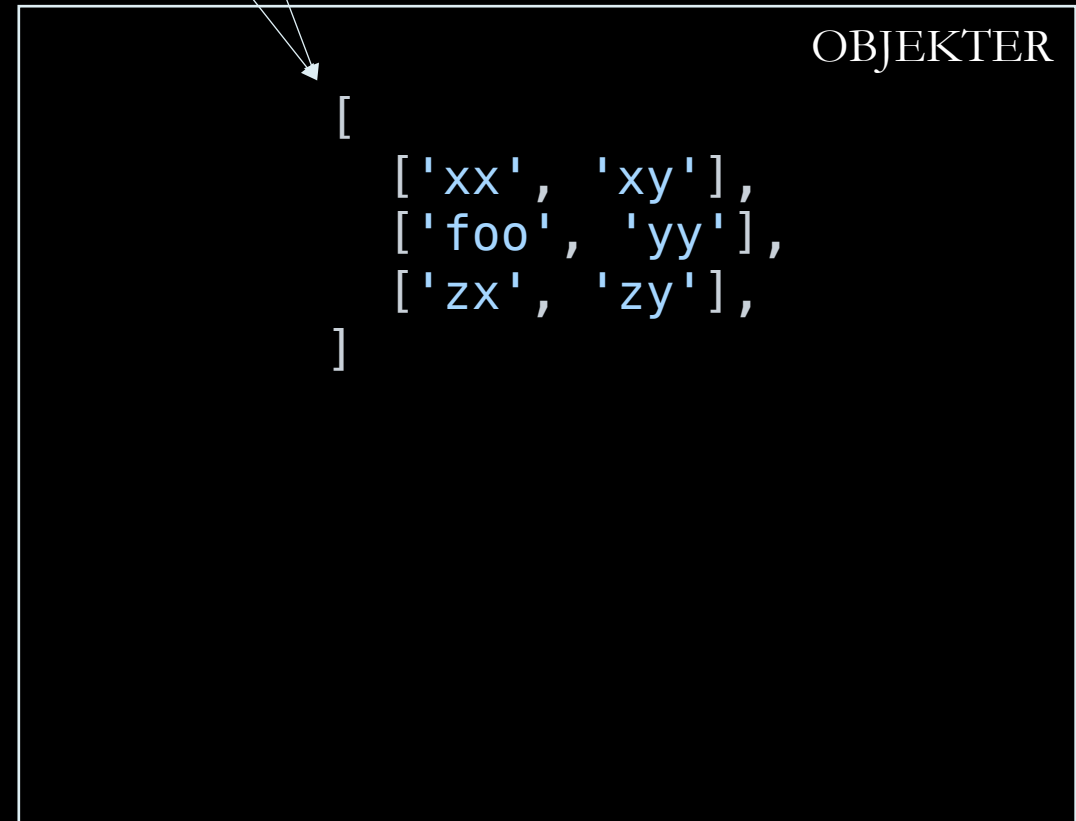
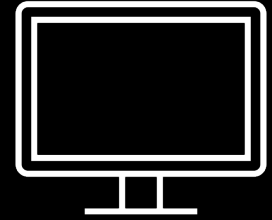
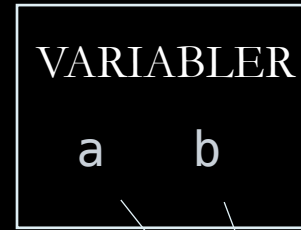
```
b = a  
b[1][0] = 'foo'  
print(a[1][0])
```



2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```

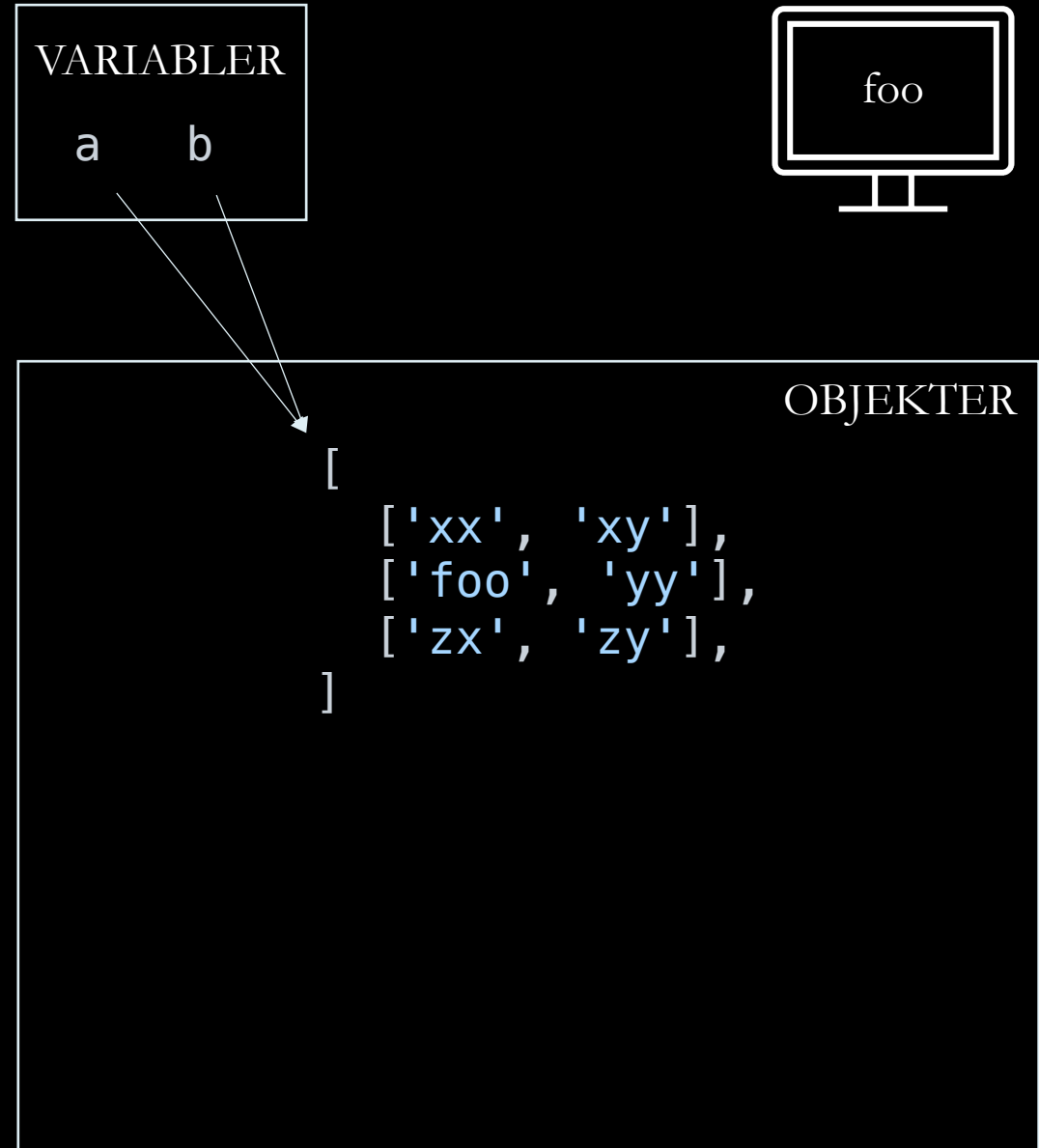
```
b = a  
b[1][0] = 'foo'  
print(a[1][0])
```



2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```

```
b = a  
b[1][0] = 'foo'  
print(a[1][0])
```



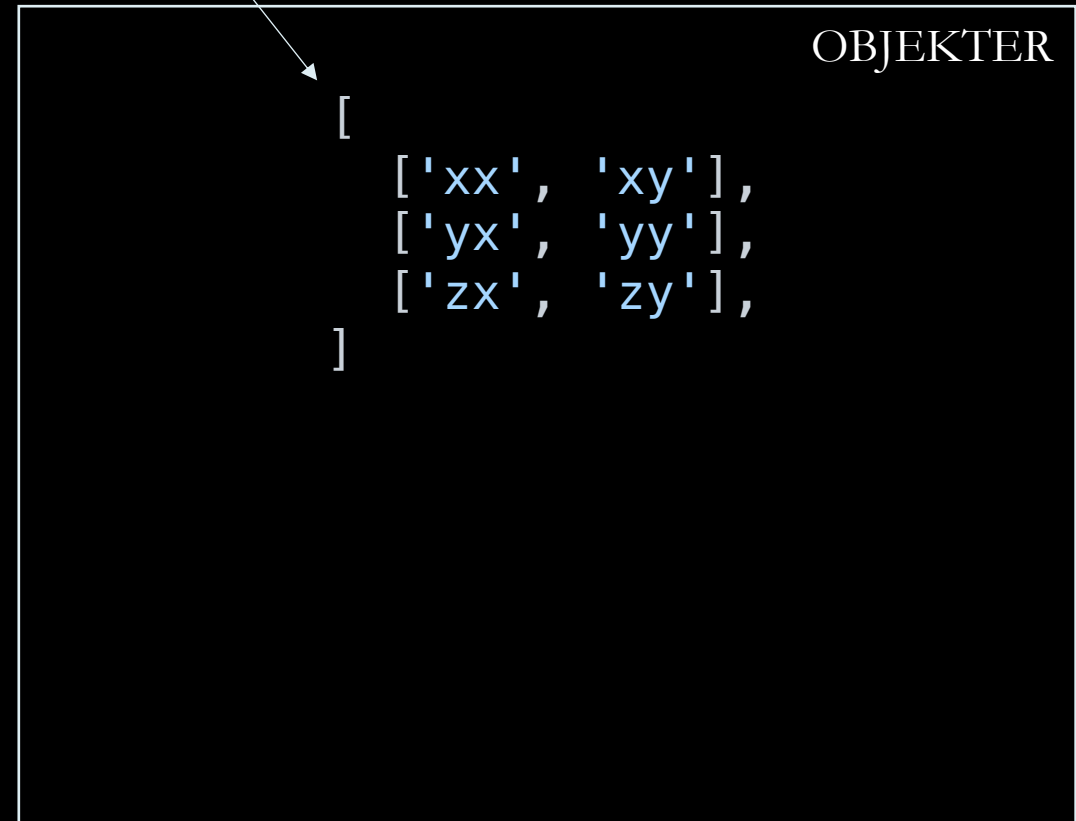
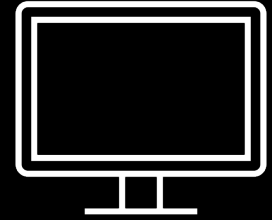
NYTT EKSEMPEL

2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```

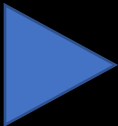


```
b = a[1]  
b[0] = 'foo'  
print(a[1][0])
```

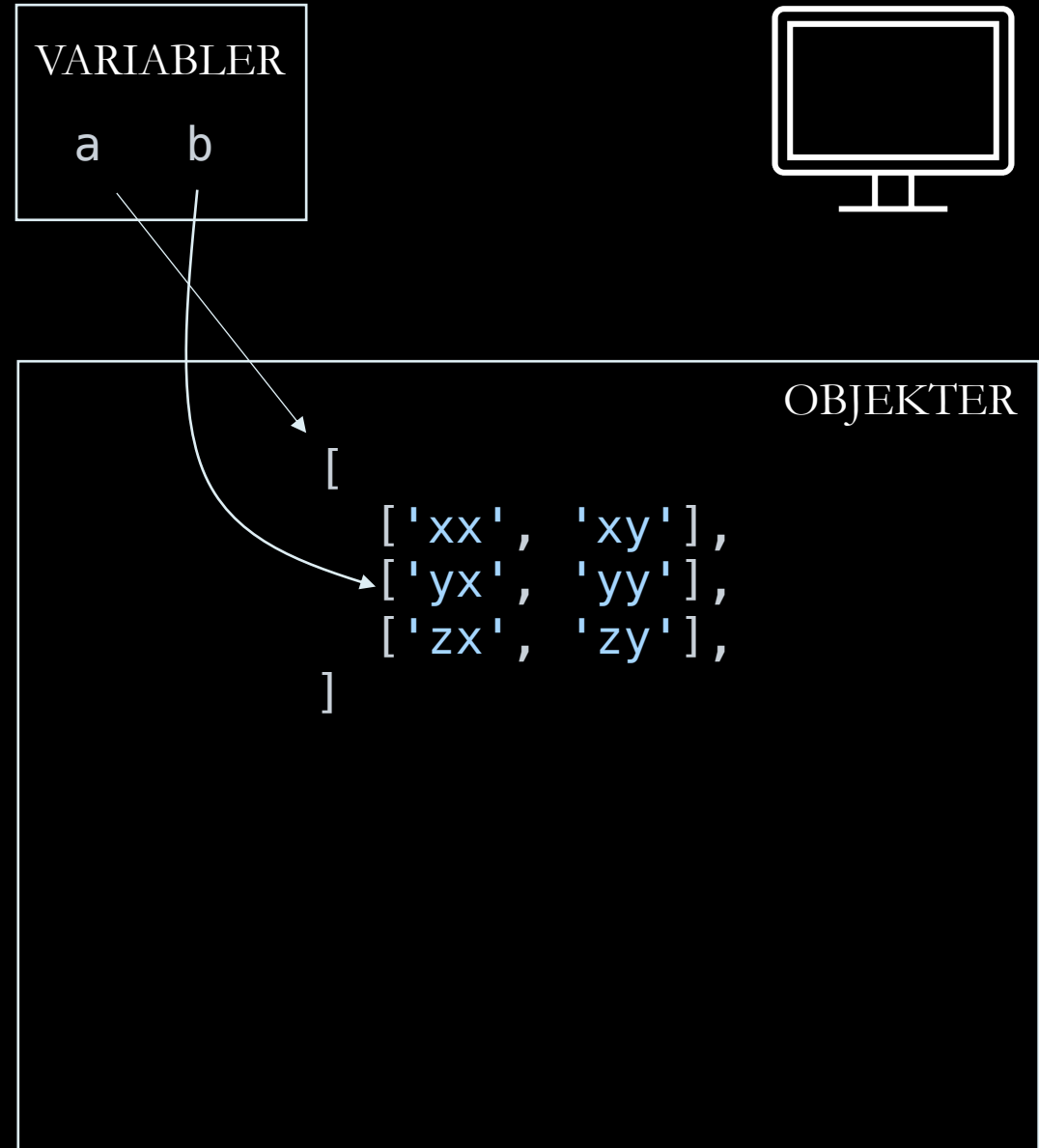


2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```



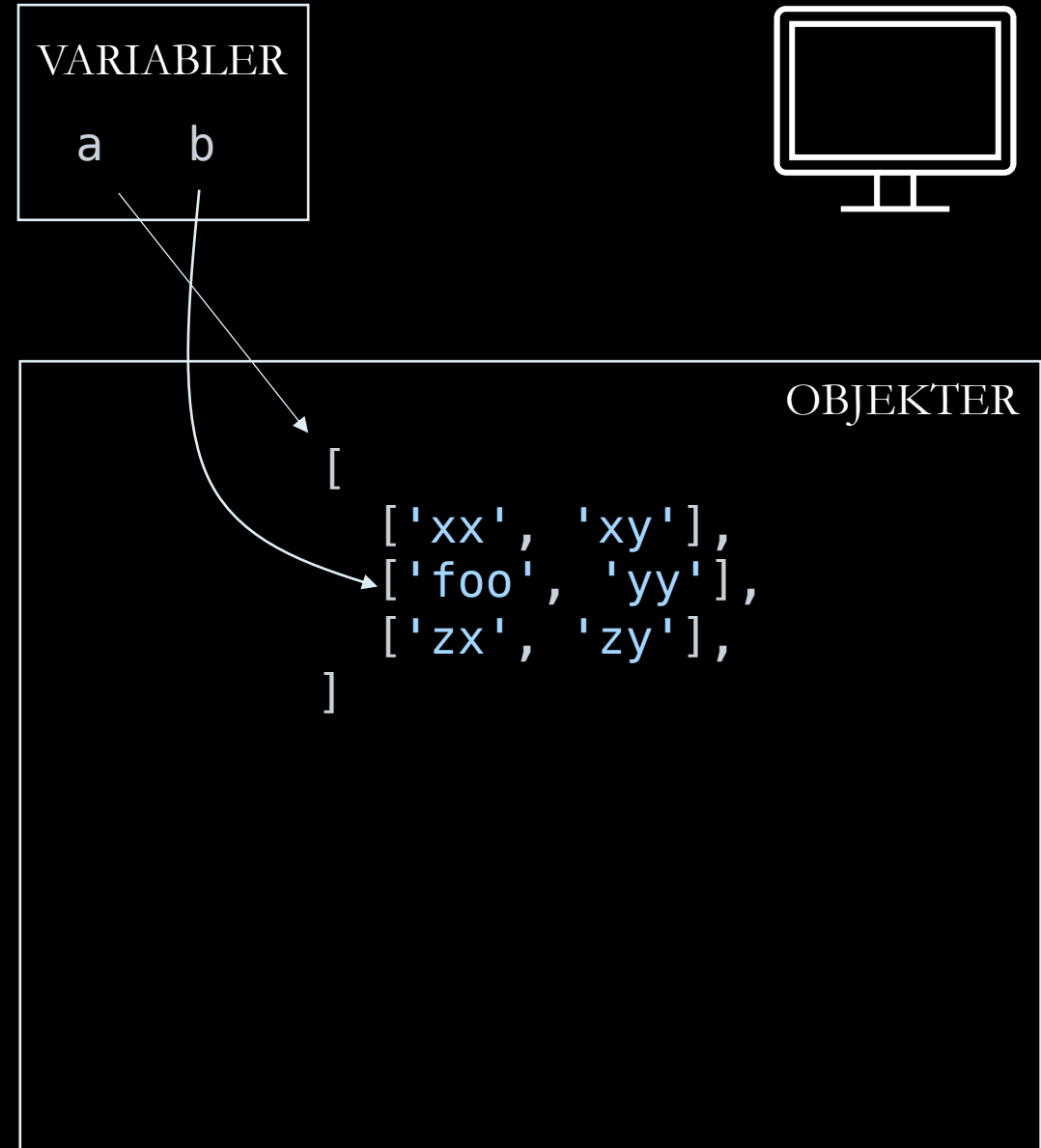
```
b = a[1]  
b[0] = 'foo'  
print(a[1][0])
```



2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```

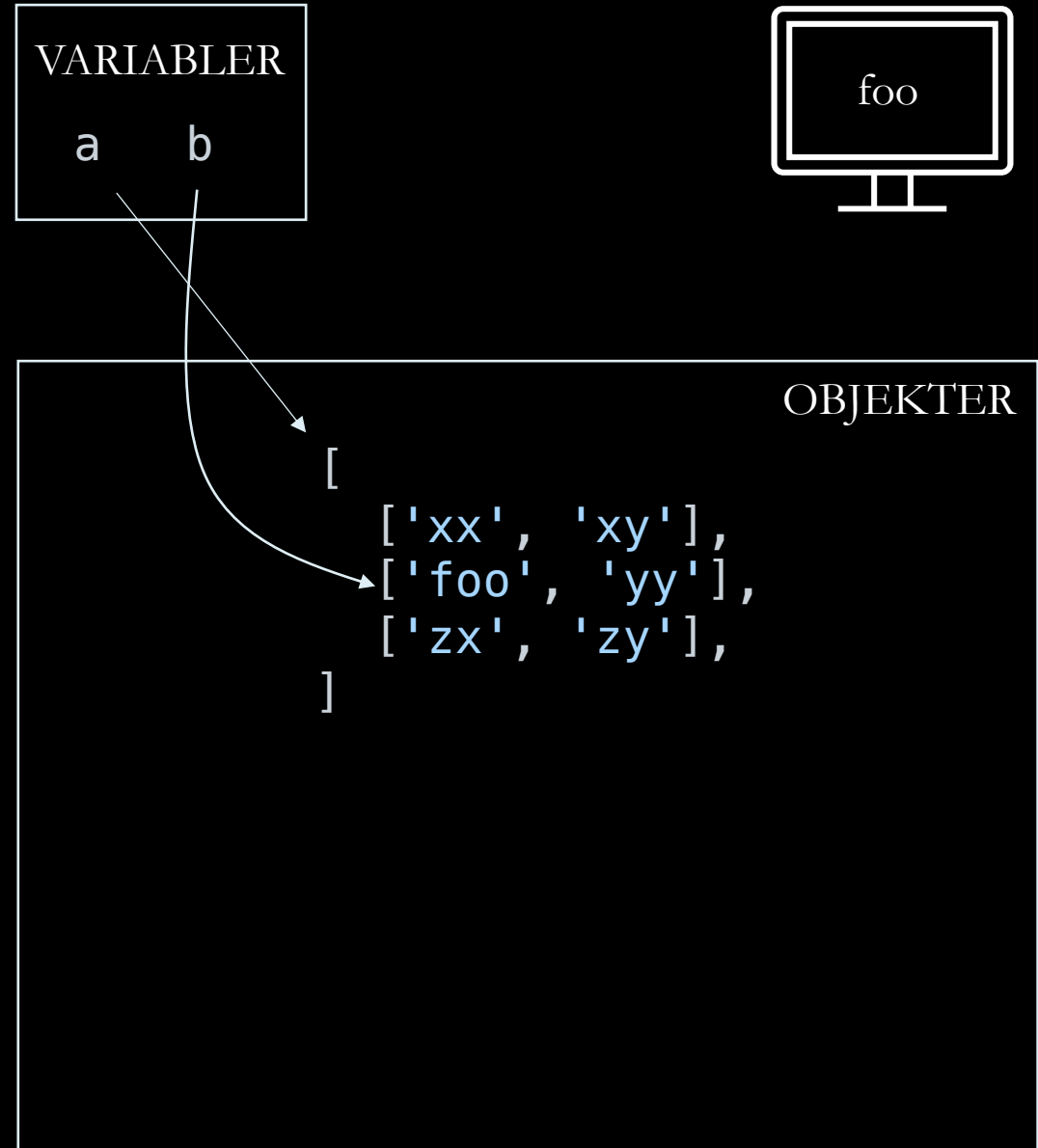
```
b = a[1]  
b[0] = 'foo'  
print(a[1][0])
```



2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```

```
b = a[1]  
b[0] = 'foo'  
print(a[1][0])
```



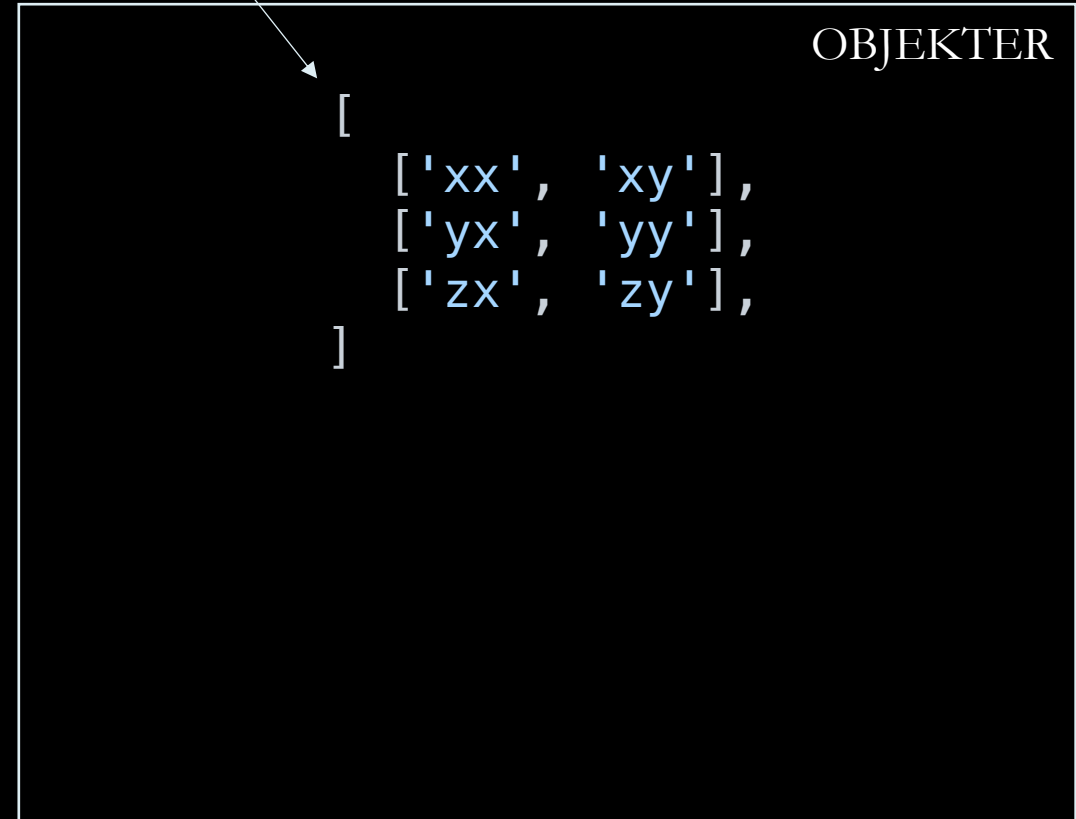
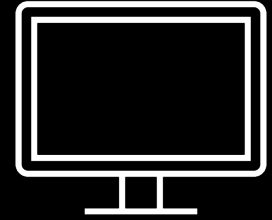
NYTT EKSEMPEL

2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```

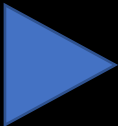


```
b = a[1]  
a[1] = ['hi', 'ha']  
print(a[1][0])  
print(b[0])
```

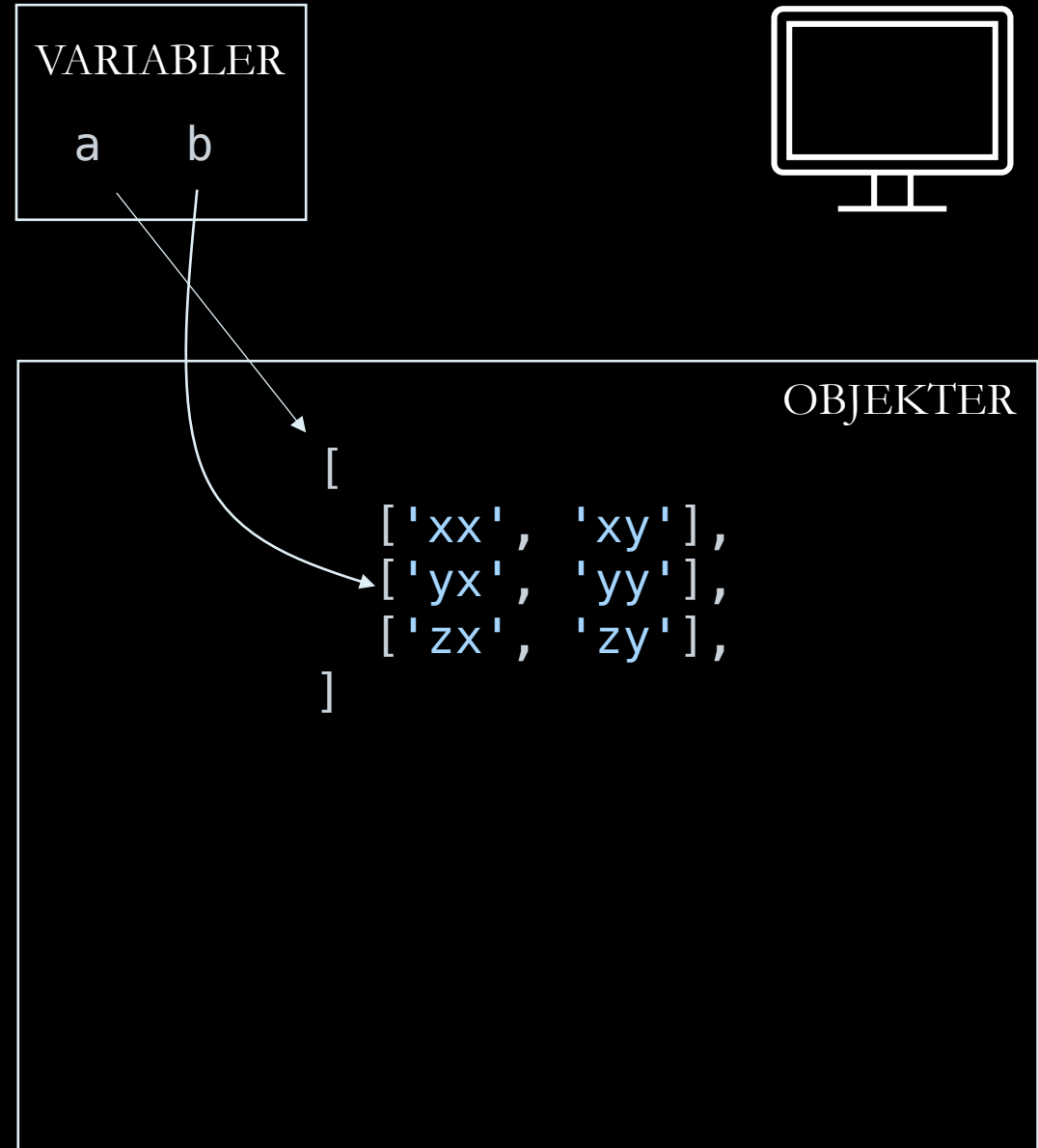


2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```



```
b = a[1]  
a[1] = ['hi', 'ha']  
print(a[1][0])  
print(b[0])
```



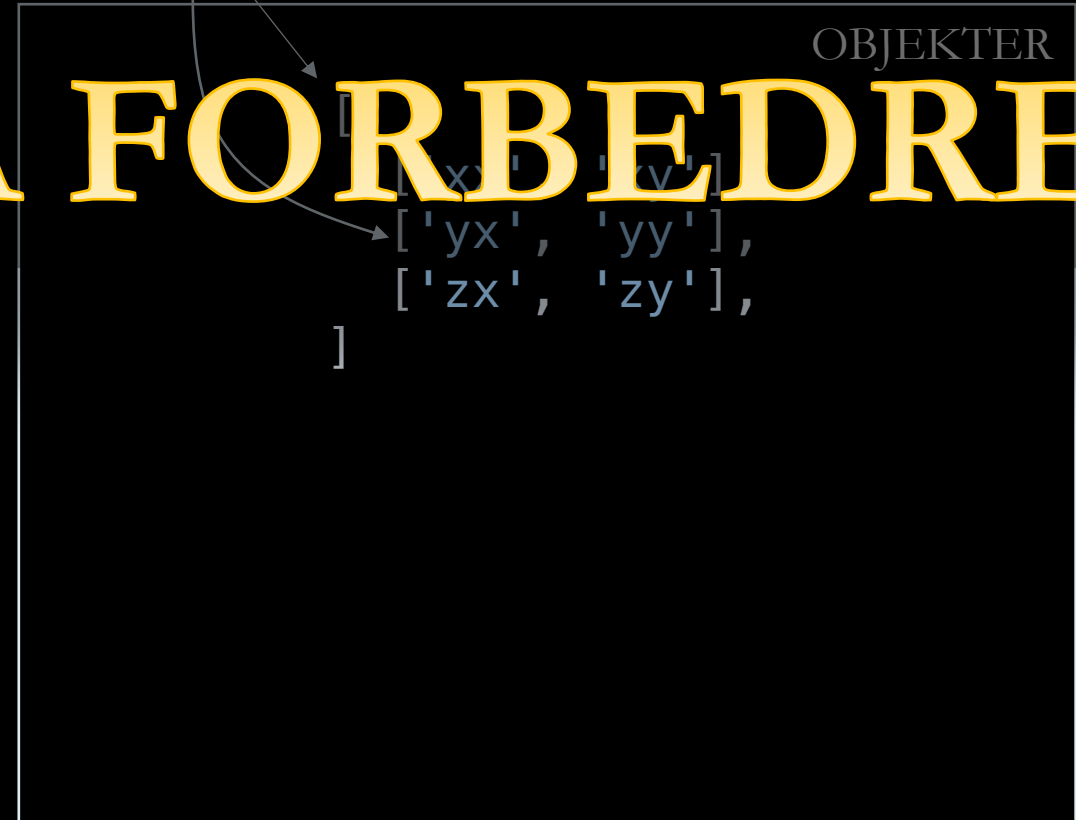
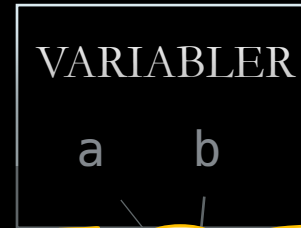
2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```

```
b = a[1]  
a[1] = ['hi', 'ha']  
print(a[1][0])  
print(b[0])
```



MODELLEN MÅ FORBEDRES!



2D-LISTE

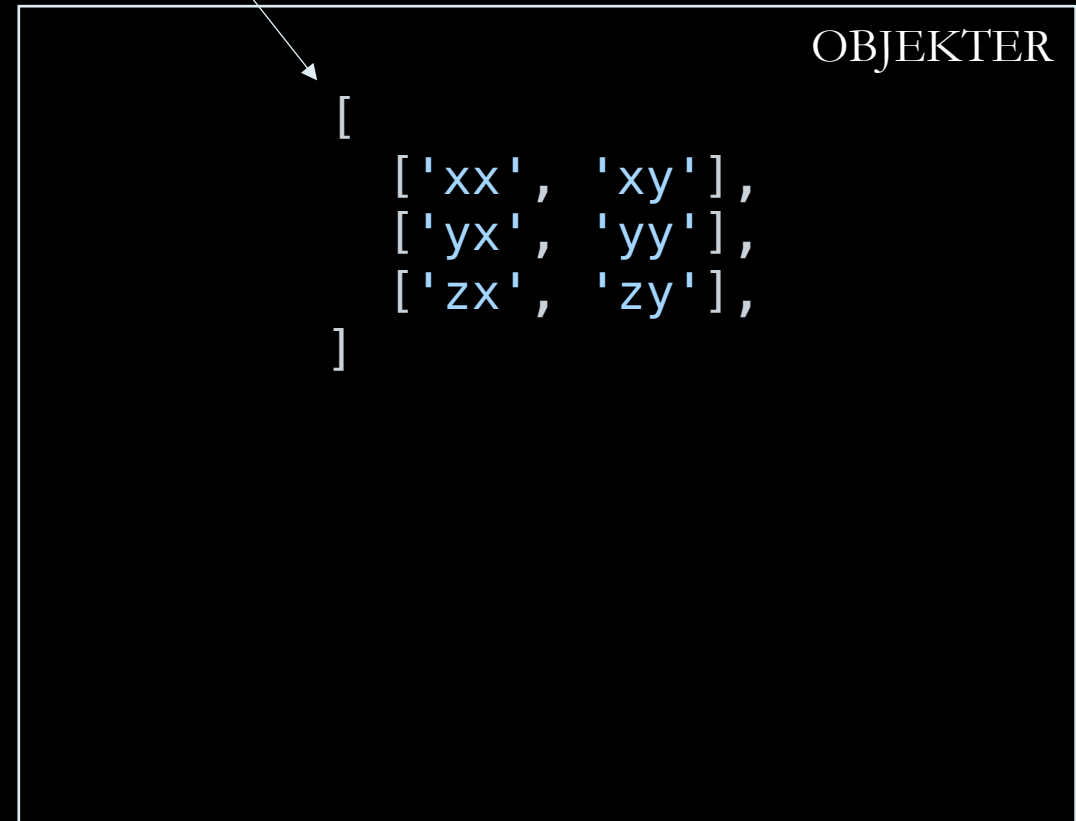
```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```



```
b = a[1]  
a[1] = ['hi', 'ha']  
print(a[1][0])  
print(b[0])
```



FØR

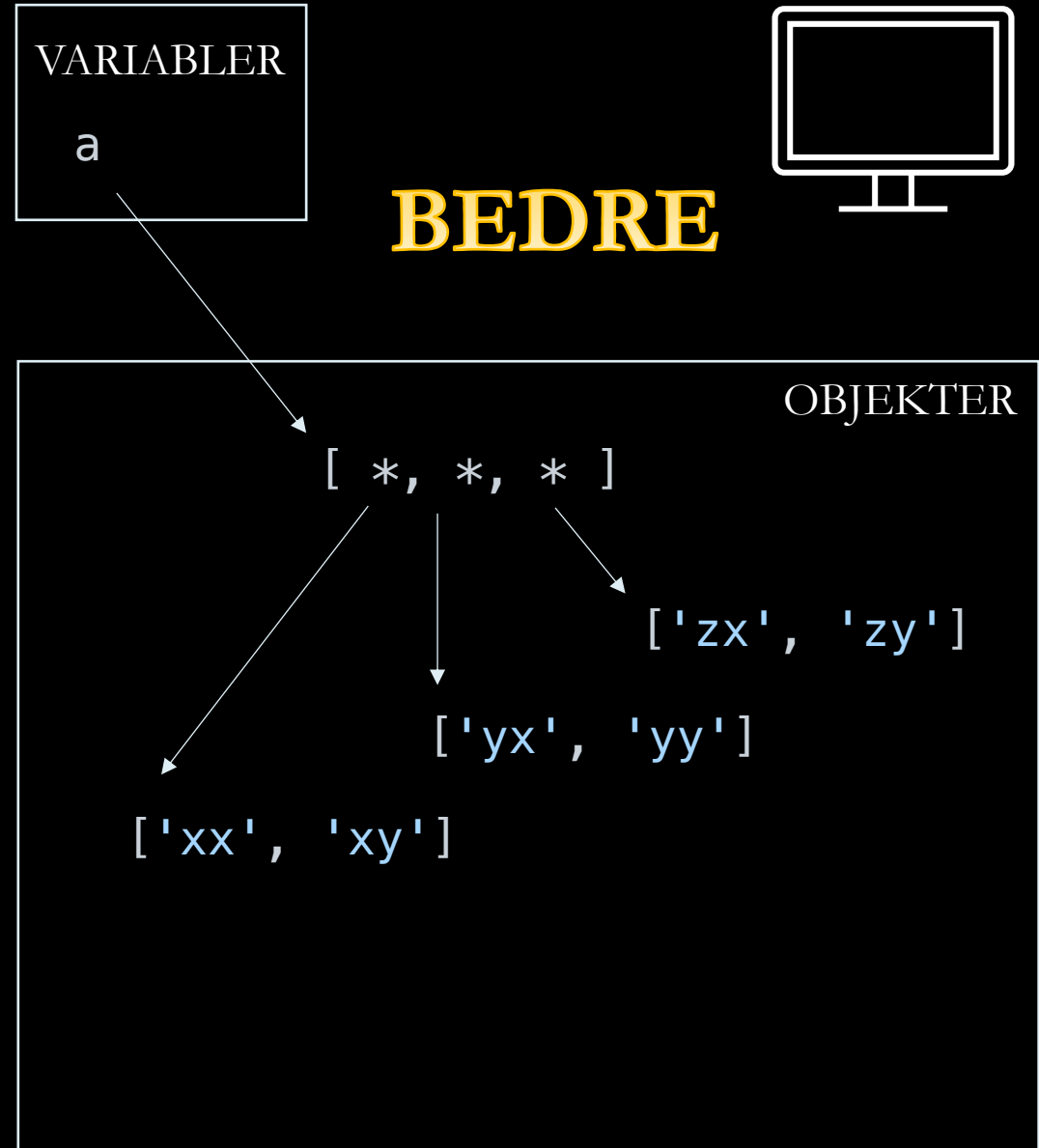


2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```

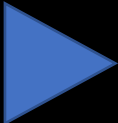


```
b = a[1]  
a[1] = ['hi', 'ha']  
print(a[1][0])  
print(b[0])
```



2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```



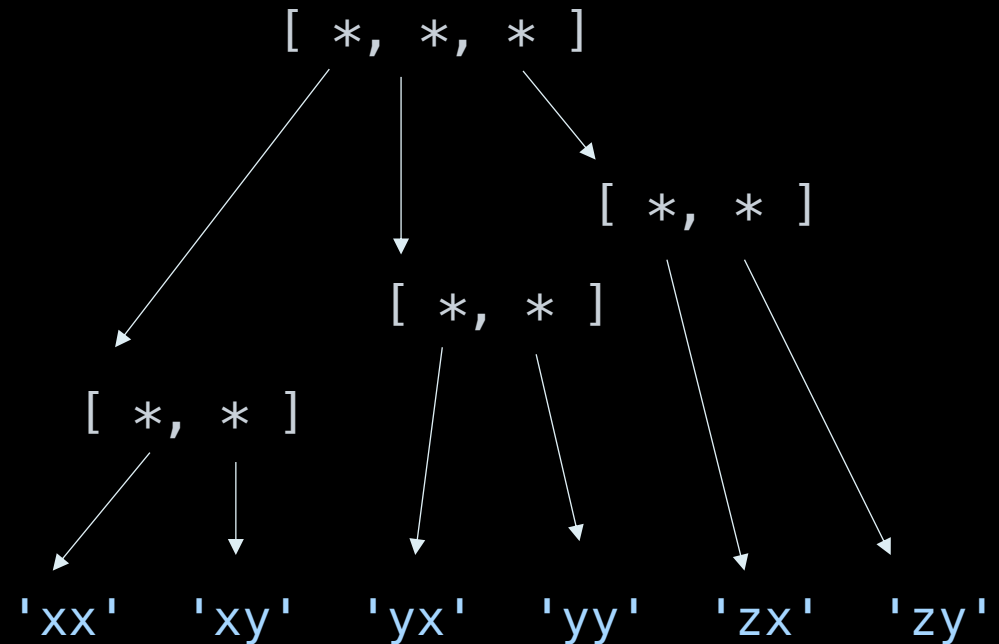
```
b = a[1]  
a[1] = ['hi', 'ha']  
print(a[1][0])  
print(b[0])
```

VARIABLER

a

**NÆRMERE
VIRKELIGHETEN**

OBJEKTER



2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```

**ENDA NÆRMERE
VIRKELIGHETEN**

VARIABLER

Variabelnavn	Adresse
a	10240

OBJEKTER

Adresse	Klasse	Verdi
1024	str	'xx'
2048	str	'xy'
3072	str	'yx'
4096	str	'yy'
5120	str	'zx'
6144	str	'zy'
7168	list	[1024, 2048]
8192	list	[3072, 4096]
9216	list	[5120, 6144]
10240	list	[7168, 8192, 9216]

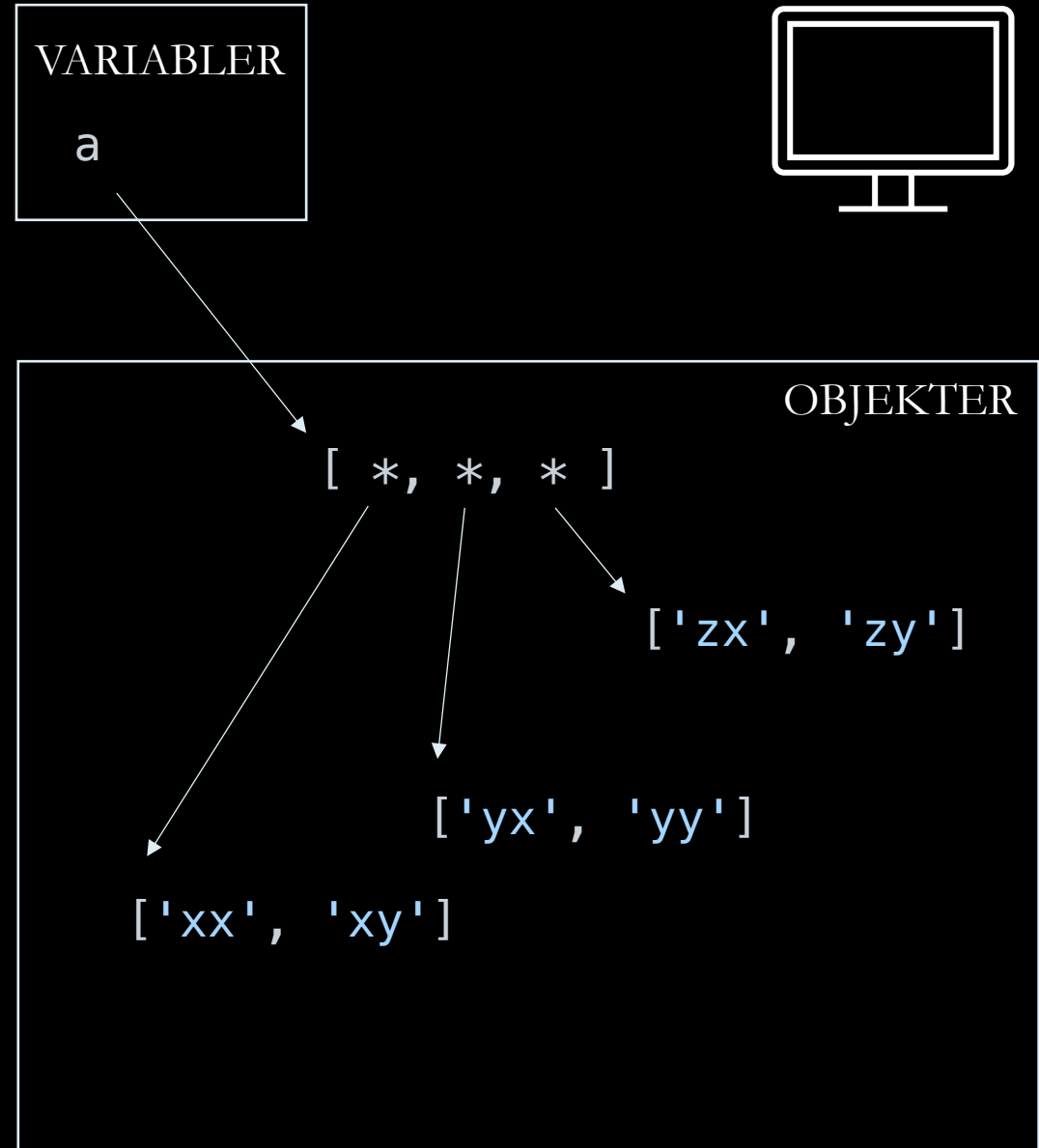
NYTT FORSØK

2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```

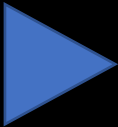


```
b = a[1]  
a[1] = ['hi', 'ha']  
print(a[1][0])  
print(b[0])
```

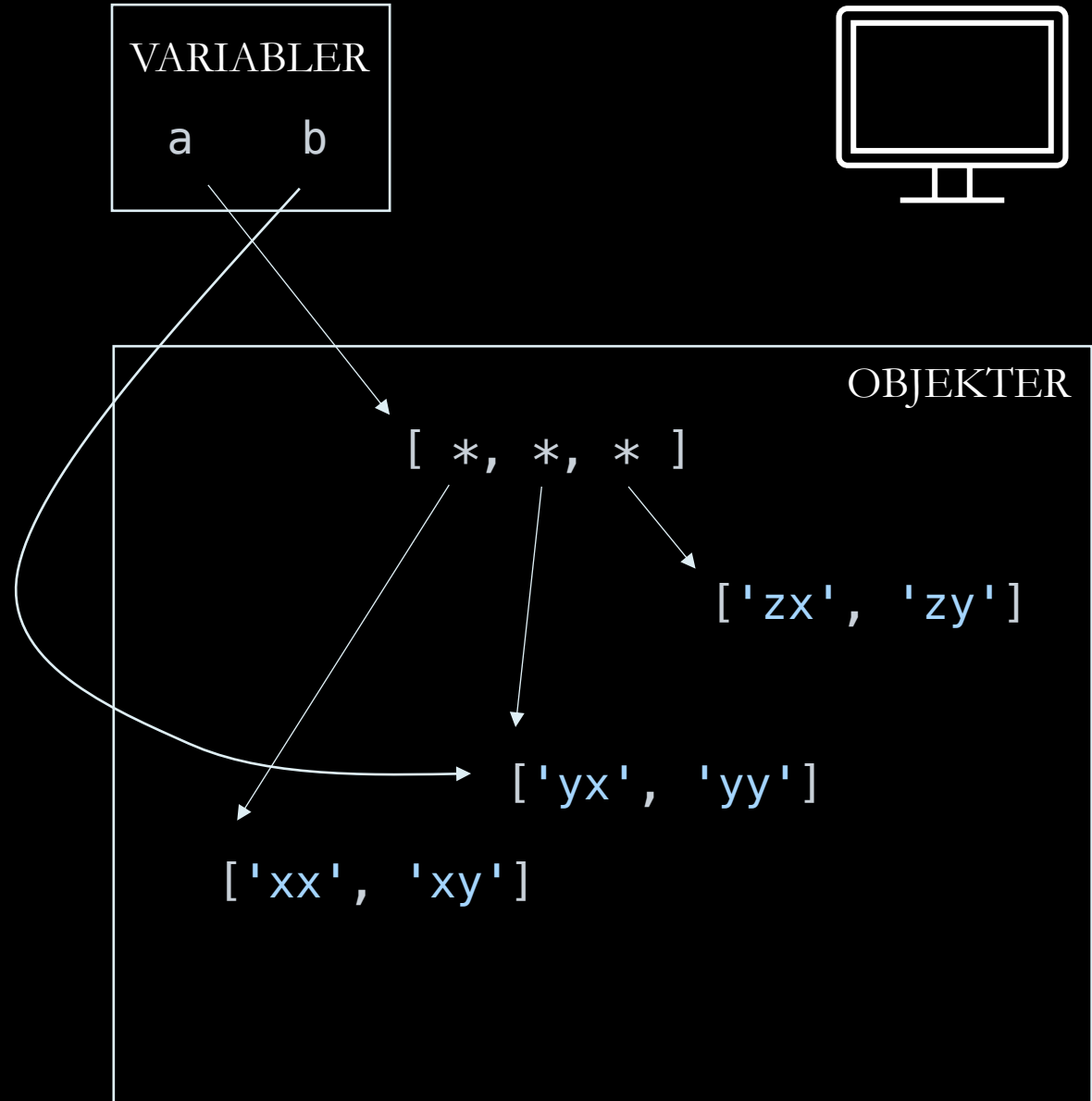


2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```



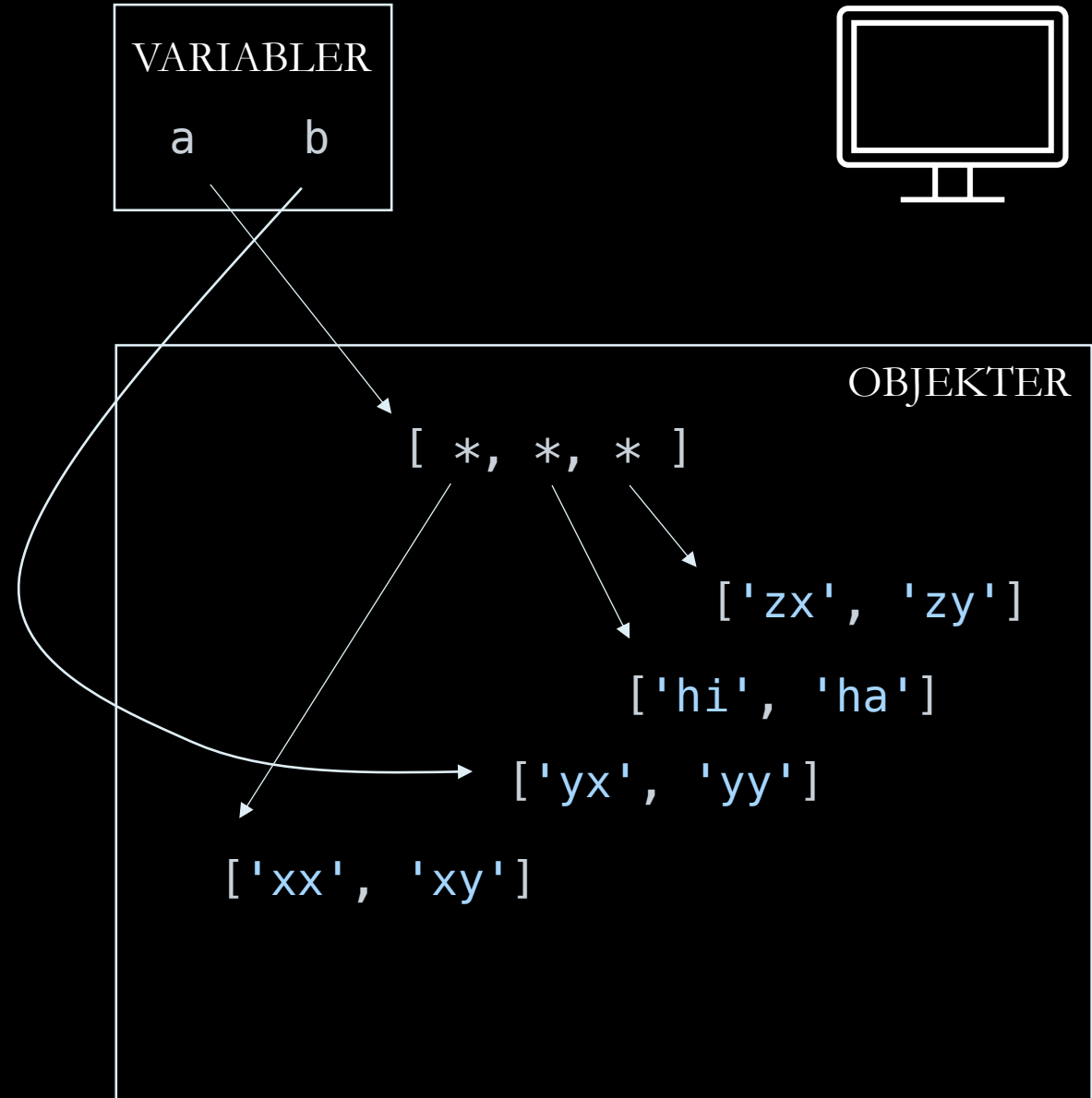
```
b = a[1]  
a[1] = ['hi', 'ha']  
print(a[1][0])  
print(b[0])
```



2D-LISTE

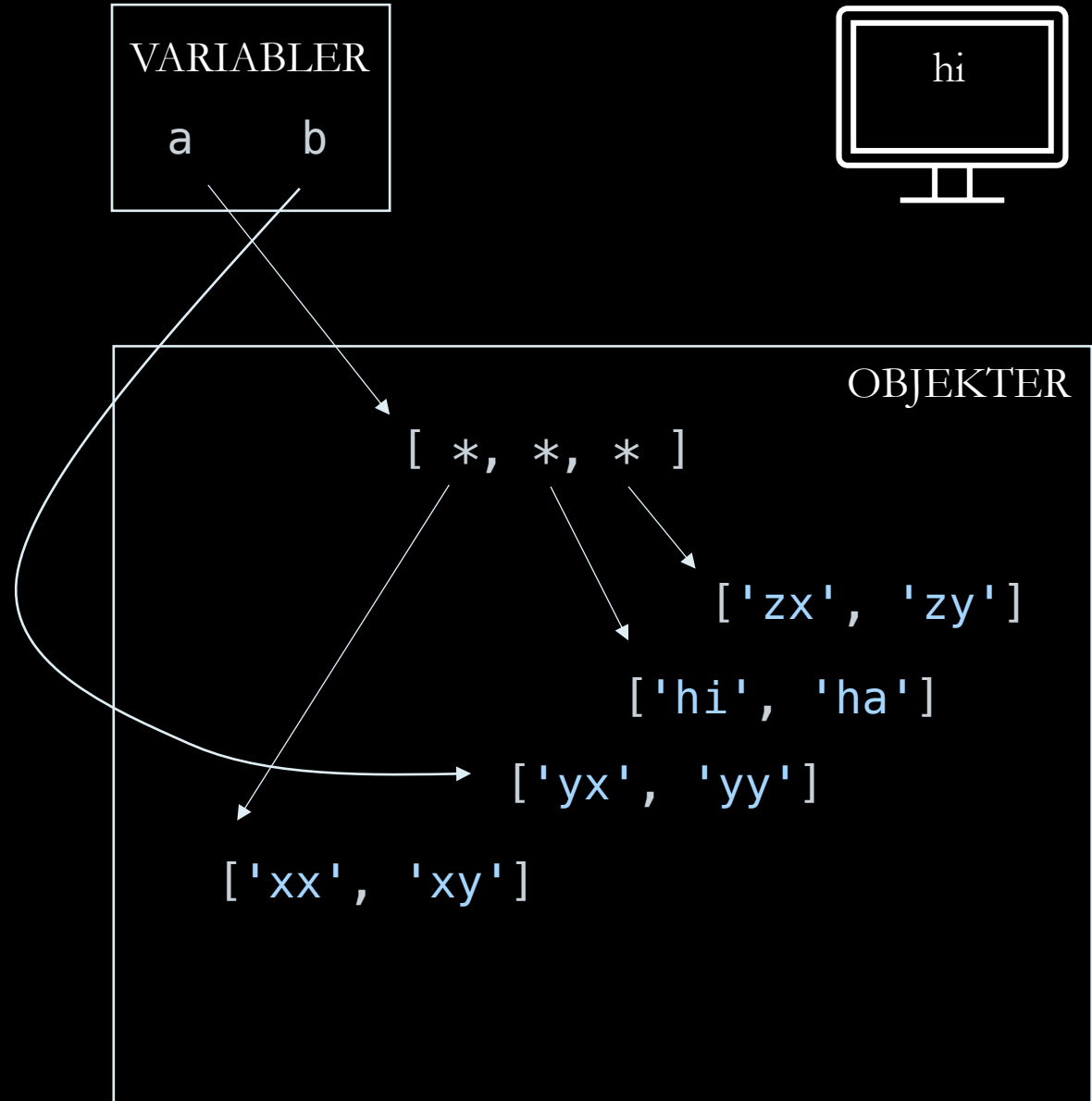
```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]
```

```
b = a[1]  
a[1] = ['hi', 'ha']  
print(a[1][0])  
print(b[0])
```



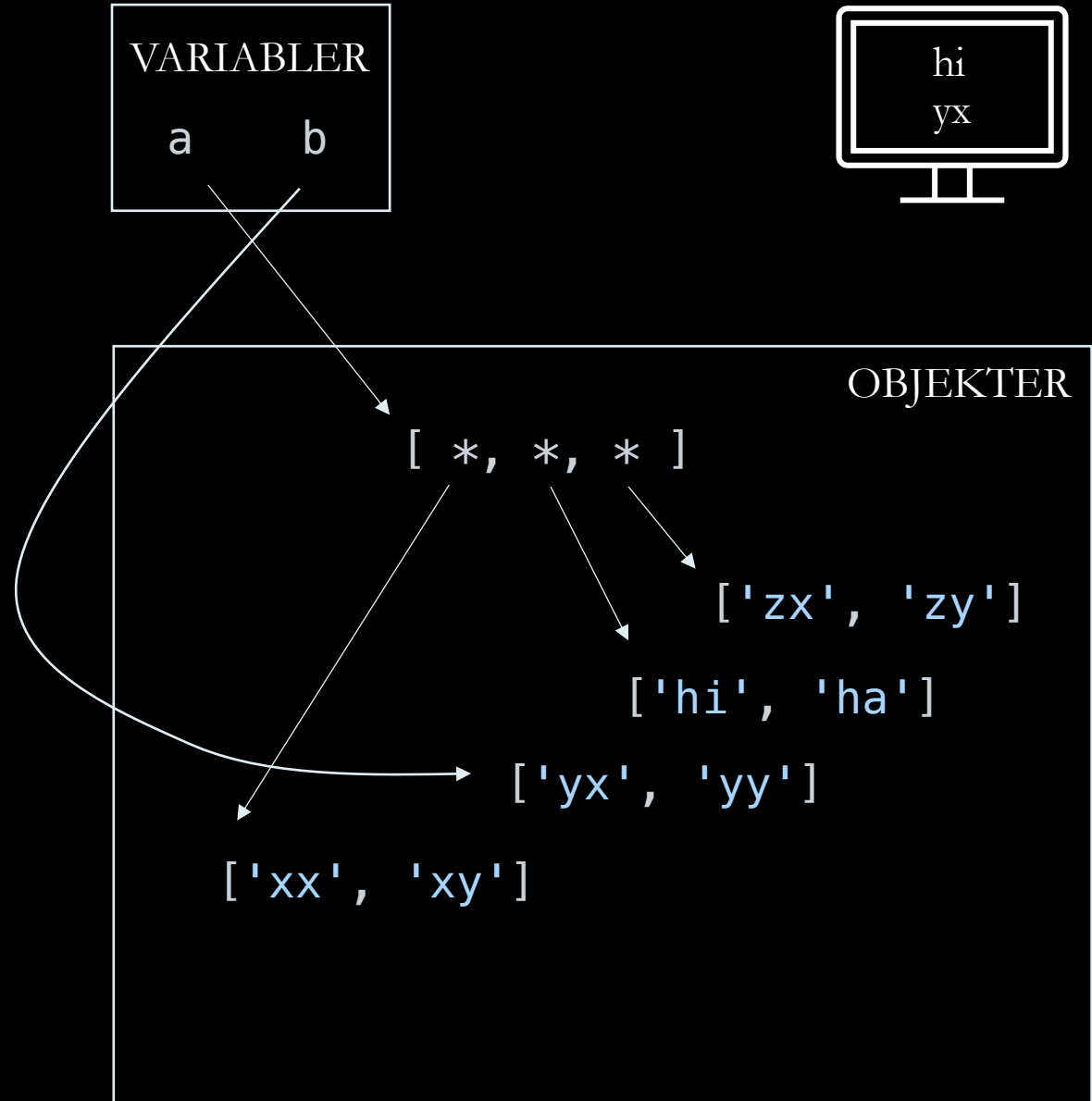
2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]  
  
b = a[1]  
a[1] = ['hi', 'ha']  
print(a[1][0])  
print(b[0])
```



2D-LISTE

```
a = [  
    ['xx', 'xy'],  
    ['yx', 'yy'],  
    ['zx', 'zy'],  
]  
  
b = a[1]  
a[1] = ['hi', 'ha']  
print(a[1][0])  
print(b[0])
```



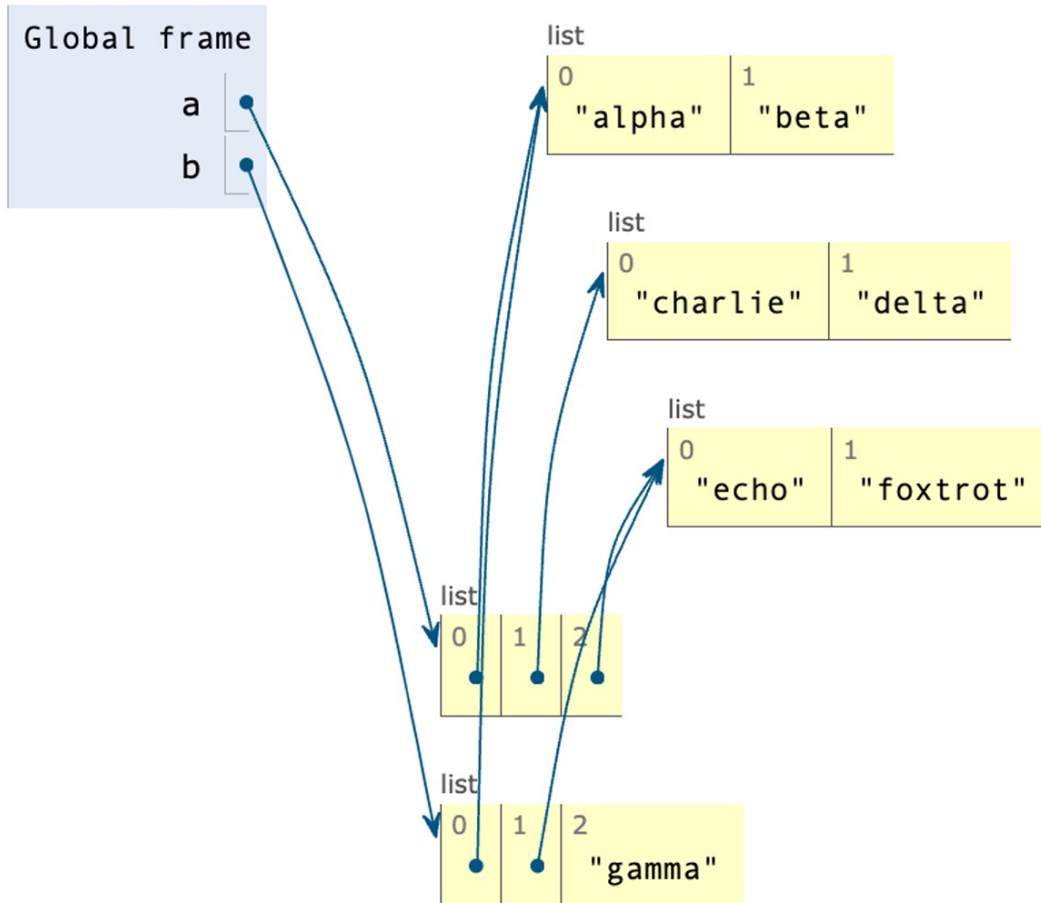
PYTHON TUTOR

- Gratis, reklamefinansiert visualiseringsverktøy

<https://pythontutor.com/>

EKSAMENSOPPGAVE HØST 2023

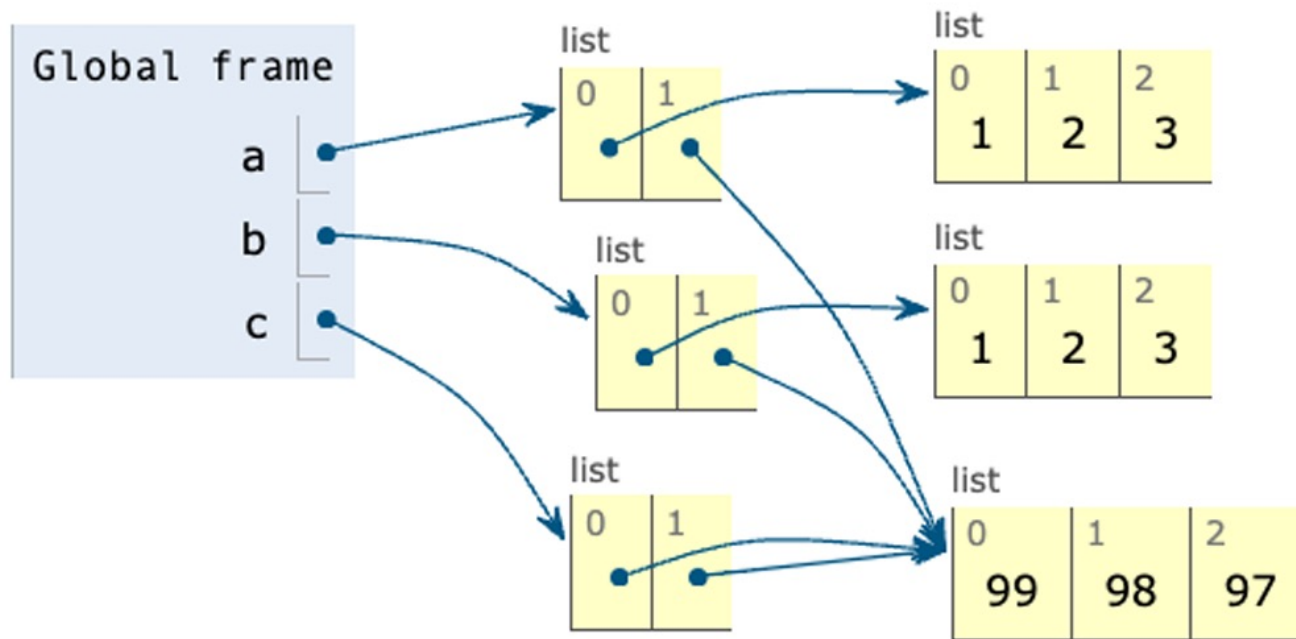
1(l)



Gitt at minnet har tilstanden vist over, hva blir skrevet ut etter setningen `print(a[1][1] + b[1][1])`?
(hvis programmet krasjer, skriv kun 'Error')

EKSAMENSOPPGAVE HØST 2023

3(a)



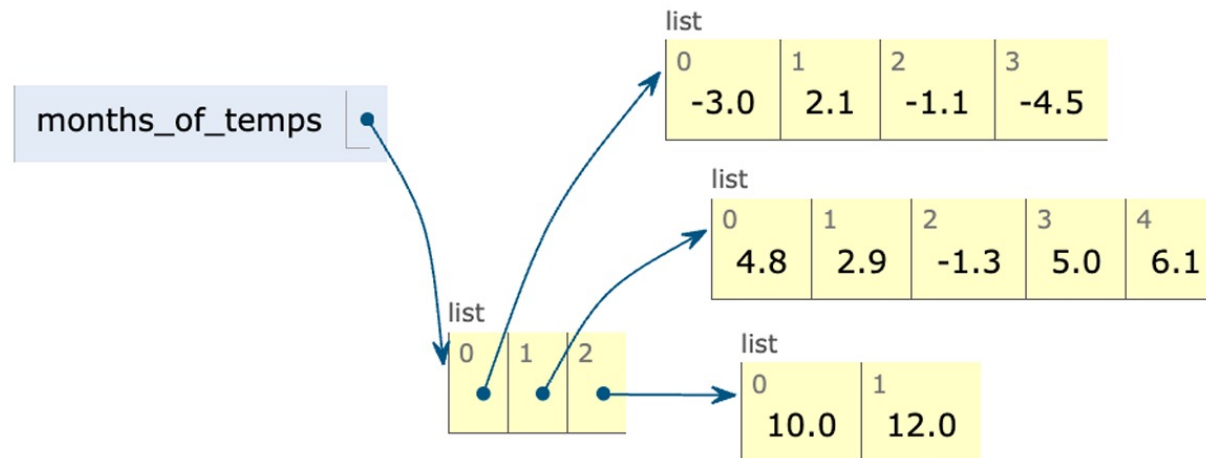
Skriv en kodesnutt slik at variabler og minnets tilstand for variablene a, b og c blir som vist over.

EKSAMENSOPPGAVE HØST 2023

3(b) Anta at **months_of_temps** er en variabel som peker på en to-dimensjonal liste av flyttall. De indre listene inneholder flyttall som representerer gjennomsnittstemperaturen for hver dag i en måned; mens den ytterste listen inneholder flere ulike måneder. Forskjellige måneder kan ha ulikt antall dager.

Skriv en funksjon **average_temp** med en parameter **months_of_temps** som beskrevet over. La funksjonen returnere gjennomsnittstemperaturen for alle dagene, uansett måned.

Eksempel på **months_of_temps**:



Hvis funksjonen du skriver kalles med eksempelet vist over som argument, skal returverdien bli 3.0: i eksempelet er det 11 ulike måleverdier (dager med temperaturmålinger), og summen av alle måleverdiene er 33.0.

2D-LISTER

- Opprette en 2D-liste med 100 rader og 10 kolonner med kun 0'ere
- Sett elementet på posisjon [3][5] til 1000
- Sett elementet på posisjon [99][2] til 1
- Bruk en nøstet løkke for å telle sammen summen av tallene i 2D-listen

2D-LISTER

- Opprette en 2D-liste med 100 rader og 10 kolonner med kun 0'ere

Buggy!

```
one_row = [0] * 10  
table = [one_row] * 100
```

Riktig!

```
table = []  
for _ in range(100):  
    one_row = [0] * 10  
    table.append(one_row)
```

2D-LISTER

- Opprette en 2D-liste med 100 rader og 10 kolonner med kun 0'ere

Buggy!

```
table = [[0] * 10] * 100
```

Riktig!

```
table = [[0] * 10 for _ in range(100)]
```

Oppgave:

Hva er faktorene i tallet 209414732?

Oppgave:
Er tallet x et primtall?

Oppgave:

Hva er det n 'te primtallet?