DIGI111
Algoritmer og programmering

David Grellscheid

Dag 2

```
x = "Hello"
score = 10

if 'e' in x:
    score = score + 1

if len(x) == 5:
    score = -3 * score

print(score)
```

```
x = "Hello"
score = 10

if 'e' in x:
    score = score + 1

if len(x) == 5:
    score = -3 * score

print(score)
```

-33

```
a = 5
b = 7
while a < b:
    c = a + b
    print(c)
    b += 2
    a += 3
    print(a)
```

```
a = 5
b = 7
while a < b:
    c = a + b
    print(c)
    b += 2
    a += 3
    print(a)
```

```
12
8
17
11
```

Løkker / Loops — while

```
s = 2

while s < 1000:
    print(s)
    s = 2 * s
```

```
i = 0
tekst = 'Hei'

while i < 3:
    print(tekst)
    i = i+1
```

Løkker / Loops — while

```
s = 2

while s < 1000:
    print(s)
    s = 2 * s
```

```
2
4
8
16
32
64
128
256
512
```

```
i = 0
tekst = 'Hei'

while i < 3:
    print(tekst)
    i = i+1
```

Løkker / Loops — while

```
s = 2

while s < 1000:
    print(s)
    s = 2 * s
```

```
2
4
8
16
32
64
128
256
512
```

```
i = 0
tekst = 'Hei'

while i < 3:
    print(tekst)
    i = i+1
```

```
Hei
Hei
Hei
```

Løkker / Loops — while

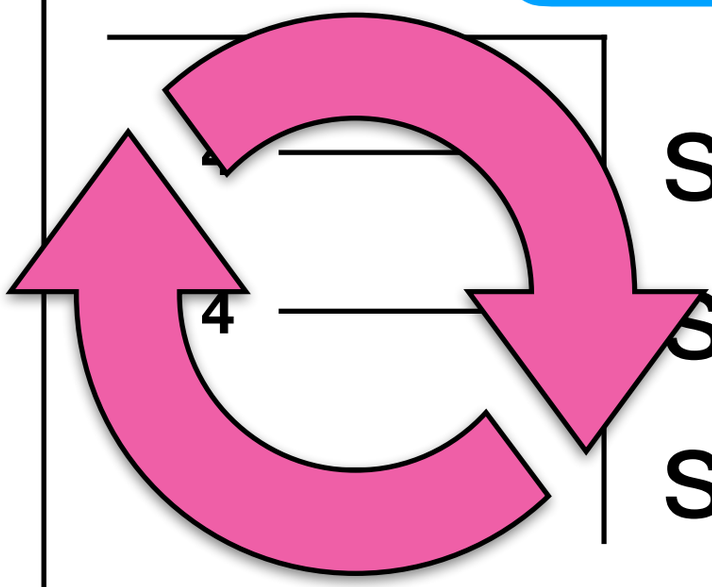
```
while True / False :
    4 → så lenge True
    4 → så lenge True
    4 → så lenge True
resten av programmet
resten av programmet
```

alle uttryk som har bool verdi (True/False) kan stå her

Løkker / Loops — while

alle uttryk som har
bool verdi (True/False)
kan stå her

```
while True / False :
    så lenge True
    så lenge True
    så lenge True
resten av programmet
resten av programmet
```



Løkker / Loops — for

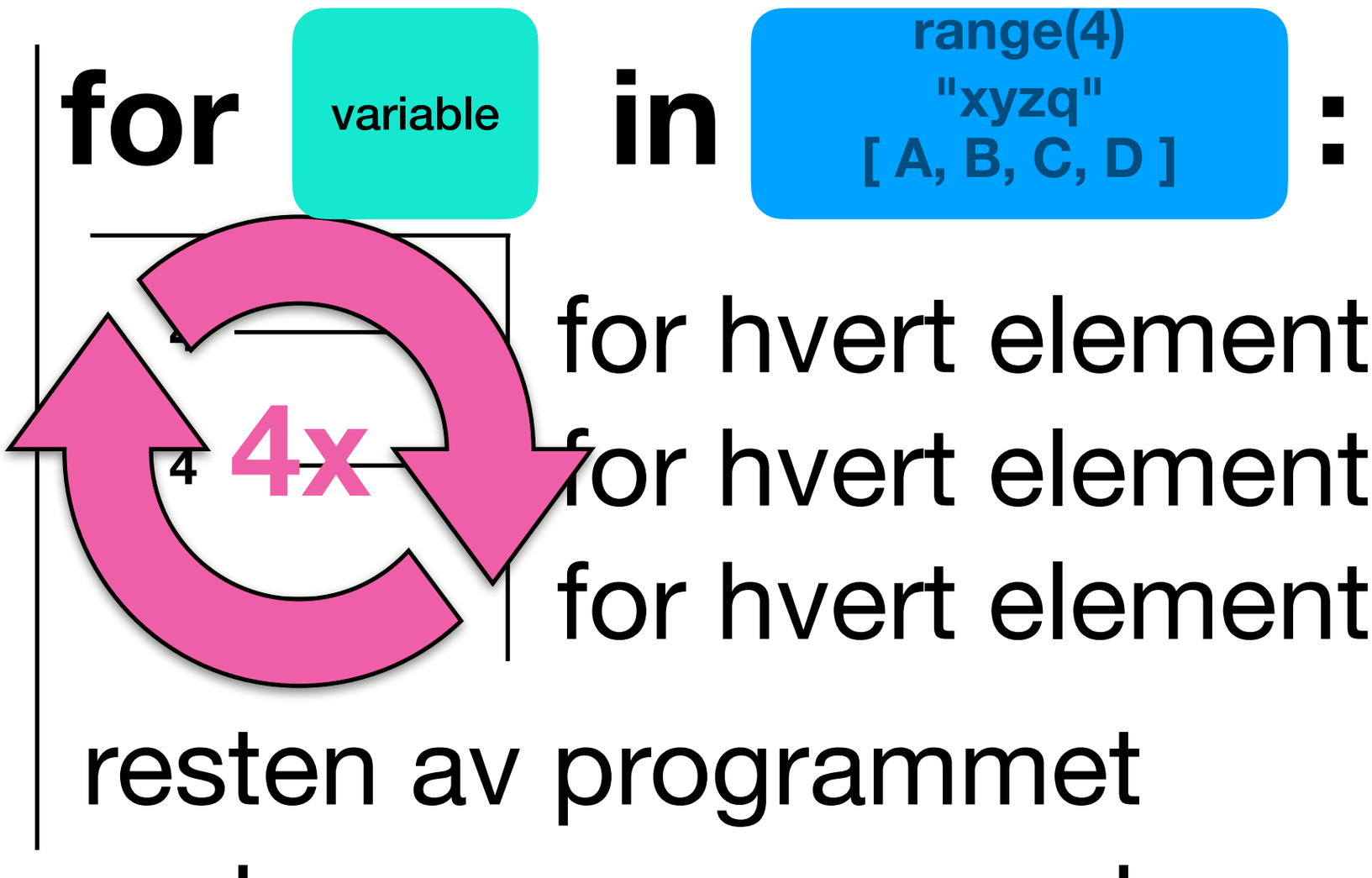
```
for x in "hei":
    print(x)
    print(f'*{x}*')
    print('---')
```

Løkker / Loops — for

```
for x in "hei":
    print(x)
    print(f'*{x}*')
    print('---')
```

```
h
*h*
---
e
*e*
---
i
*i*
---
```

Løkker / Loops — for



Funksjoner / functions

Funksjoner / functions

```
def doublesum(x,y,z):
    return 2*(x+y+z)
```

Funksjoner / functions

```
def doublesum(x,y,z):
    return 2*(x+y+z)
```

```
abc = doublesum(5,7,1)
print(abc)

xyz = doublesum(3,2,1)
print(xyz)
```

Funksjoner / functions

```
def doublesum(x,y,z):
    return 2*(x+y+z)
```

```
abc = doublesum(5,7,1)
print(abc)

xyz = doublesum(3,2,1)
print(xyz)
```

```
26
12
```

Funksjoner / functions

```
def doublesum(x,y,z):
    return 2*(x+y+z)
```

Funksjonsdefinisjon

```
abc = doublesum(5,7,1)
print(abc)

xyz = doublesum(3,2,1)
print(xyz)
```

Funksjonskall

Funksjonskall

```
26
12
```

def navn (input A, B, C, D):

4 → gjør noe

4 → gjør noe

4 → **return** resultat

resten av programmet

x = navn (3,4,5,6)

def navn (input A, B, C, D):

4 → gjør noe

4 → gjør noe

4 → **return** resultat

resten av programmet

x = navn (3,4,5,6)

def navn (input A, B, C, D):

4 → gjør noe

4 → gjør noe

4 → **return** resultat

esten av programmet

x = navn (3,4,5,6)

def navn (input A, B, C, D):

4 → gjør noe

4 → gjør noe

4 → **return** resultat

esten av programmet

x = navn (3,4,5,6)

def navn (input A, B, C, D):

4 → gjør noe

4 → gjør noe

4 → **return** resultat

esten av programmet

x = navn (3,4,5,6)

def navn (input A, B, C, D):

4 → gjør noe

4 → gjør noe

4 → **return** resultat

resten av programmet

x = navn (3,4,5,6)

def navn (input A, B, C, D):

4 → gjør noe

4 → gjør noe

4 → **return** resultat

resten av programmet

x = navn (3,4,5,6)

def navn (input A, B, C, D):

4 → gjør noe

4 → gjør noe

4 → **return** resultat

resten av programmet

x = resultat

def navn **(** input A, B, C, D **):**

4 → gjør noe

4 → gjør noe

4 → **return** resultat

resten av programmet

x = resultat

Funksjoner / functions

Expression / Uttryk	Value / Verdi
`len("Hei")`	`3`
`math.sqrt(2)`	`1.4142`
`abs(-3.2)`	`3.2`
`max(3,7)`	`7`
`float("55")`	`55.0`

Funksjoner / functions

Expression / Uttryk	Value / Verdi
`len("Hei")`	`3`
`math.sqrt(2)`	`1.4142`
`abs(-3.2)`	`3.2`
`max(3,7)`	`7`
`float("55")`	`55.0`

navn

argument / parameter

resultat / verdi

Funksjoner / functions

Funksjoner / functions

```
def doublesum(x,y,z):
    return 2*(x+y+z)
```

Funksjoner / functions

```
def doublesum(x,y,z):
    return 2*(x+y+z)
```

```
abc = doublesum(5,7,1)
print(abc)

xyz = doublesum(3,2,1)
print(xyz)
```

Funksjoner / functions

```
def doublesum(x,y,z):
    return 2*(x+y+z)
```

```
abc = doublesum(5,7,1)
print(abc)

xyz = doublesum(3,2,1)
print(xyz)
```

```
26
12
```

Funksjoner / functions

```
def doublesum(x,y,z):
    return 2*(x+y+z)
```

Funksjonsdefinisjon

```
abc = doublesum(5,7,1)
print(abc)

xyz = doublesum(3,2,1)
print(xyz)
```

Funksjonskall

Funksjonskall

```
26
12
```

programmflyt

def navn **(** input A, B, C, D **):**

4 → gjør noe

4 → gjør noe

4 → **return** resultat

resten av programmet

x = navn (3,4,5,6)

programmflyt

def navn (input A, B, C, D):

4 → gjør noe

4 → gjør noe

4 → **return** resultat

resten av programmet

x = navn (3,4,5,6)

programmflyt

def navn (input A, B, C, D):

4 → gjør noe

4 → gjør noe

4 → **return** resultat

esten av programmet

x = navn (3,4,5,6)

programmflyt

def navn (input A, B, C, D):

    4 → gjør noe
    4 → gjør noe
    4 → **return** resultat

esten av programmet

x = navn (3,4,5,6)

programmflyt

def navn (input A, B, C, D):

4 → gjør noe

4 → gjør noe

4 → **return** resultat

esten av programmet

x = navn (3,4,5,6)

def

def navn **(** input A, B, C, D **):**

4 → gjør noe

4 → gjør noe

4 → **return** resultat

resten av programmet

x = navn (3,4,5,6)

def

def navn **(** input A, B, C, D **):**

4 → gjør noe

4 → gjør noe

4 → **return** resultat

resten av programmet

x = navn (3,4,5,6)

def

def navn **(** input A, B, C, D **):**

4 → gjør noe

4 → gjør noe

4 → **return** resultat

resten av programmet

x = resultat

def

def navn **(** input A, B, C, D **):**

4 → gjør noe

4 → gjør noe

4 → **return** resultat

resten av programmet

x = resultat

```
def doublesum(x,y,z):
    sum = x+y+z
    return 2*sum
```

```
abc = doublesum(5,7,1)
print(abc)
```

26

```
foo = doublesum('ab','xz','d')
print(foo)
```

abxzdabxzd

(VScode demo: program flow, nested calls)

Funksjoner / functions

Expression / Uttryk	Value / Verdi
`len("Hei")`	`3`
`math.sqrt(2)`	`1.4142`
`abs(-3.2)`	`3.2`
`max(3,7)`	`7`
`float("55")`	`55.0`
`print("hei")`	`None` (men skriver noe)
`turtle.left(20)`	`None` (men snur turtle)

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
`[ ]`	

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
`[ ]`	`[]`

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
`[ ]`	`[]`
`[1]`	

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
`[ ]`	`[]`
`[1]`	`[1]`

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
`[ ]`	`[]`
`[1]`	`[1]`
x	

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
`[ ]`	`[]`
`[1]`	`[1]`
`x`	`[5,7,3,8]`

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
`[ ]`	`[]`
`[1]`	`[1]`
`x`	`[5,7,3,8]`
`x.append(99)`	

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
`[ ]`	`[]`
`[1]`	`[1]`
`x`	`[5,7,3,8]`
`x.append(99)`	

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
`[ ]`	`[]`
`[1]`	`[1]`
`x`	`[5,7,3,8]`
`x.append(99)`	
`x`	

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
`[ ]`	`[]`
`[1]`	`[1]`
`x`	`[5,7,3,8]`
`x.append(99)`	
`x`	`[5,7,3,8,99]`

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
`[ ]`	`[]`
`[1]`	`[1]`
`x`	`[5,7,3,8]`
`x.append(99)`	
`x`	`[5,7,3,8,99]`
`[1,2] + [5,6]`	

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
`[ ]`	`[]`
`[1]`	`[1]`
`x`	`[5,7,3,8]`
`x.append(99)`	
`x`	`[5,7,3,8,99]`
`[1,2] + [5,6]`	`[1,2,5,6]`

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
`[ ]`	`[]`
`[1]`	`[1]`
`x`	`[5,7,3,8]`
`x.append(99)`	
`x`	`[5,7,3,8,99]`
`[1,2] + [5,6]`	`[1,2,5,6]`
`len(y)`	

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
`[ ]`	`[]`
`[1]`	`[1]`
`x`	`[5,7,3,8]`
`x.append(99)`	
`x`	`[5,7,3,8,99]`
`[1,2] + [5,6]`	`[1,2,5,6]`
`len(y)`	`4`

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
`[ ]`	`[]`
`[1]`	`[1]`
`x`	`[5,7,3,8]`
`x.append(99)`	
`x`	`[5,7,3,8,99]`
`[1,2] + [5,6]`	`[1,2,5,6]`
`len(y)`	`4`
`y[2]`	

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
[ ]	[]
[1]	[1]
x	[5,7,3,8]
x.append(99)	
x	[5,7,3,8,99]
[1,2] + [5,6]	[1,2,5,6]
len(y)	4
y[2]	'hei'

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
[ ]	[]
[1]	[1]
x	[5,7,3,8]
x.append(99)	
x	[5,7,3,8,99]
[1,2] + [5,6]	[1,2,5,6]
len(y)	4
y[2]	'hei'
y[1:3]	

list / [... , ... , ... , ...]

```
x = [ 5, 7, 3, 8 ]
```

```
y = [ 1, 3.141, 'hei', True ]
```

Expression / Uttryk	Value / Verdi
`[ ]`	`[]`
`[1]`	`[1]`
`x`	`[5,7,3,8]`
`x.append(99)`	
`x`	`[5,7,3,8,99]`
`[1,2] + [5,6]`	`[1,2,5,6]`
`len(y)`	`4`
`y[2]`	`'hei'`
`y[1:3]`	`[3.141,'hei']`

list-funksjoner

- https://docs.python.org/3/tutorial/datastructures.html#more-on-lists

```
list.append(x)
list.extend([3,4,5])
list.insert(i,x)
list.remove(x)
list.pop()
list.clear()
```

```
list.index(x)
list.count(x)
list.sort()
list.reverse()
list.copy()
```

nyttige string-funksjoner

- str**.split(***sep***)** -> liste av ord, kan angi *sep, f.eks.* **'\n' ','**

```
a = """Many
different
lines
"""
b = a.split()
```

- '*tekst*'**.join(**liste**)** -> bruker *tekst* som forbindelse

nyttige string-funksjoner

- str**.split(***sep***)** -> liste av ord, kan angi *sep, f.eks.* **'\n' ','**

```
a = """Many
different
lines
"""
b = a.split()
```

```
'Many\ndifferent\nlines'
```

- '*tekst*'**.join(**liste**)** -> bruker *tekst* som forbindelse

nyttige string-funksjoner

- str**.split(***sep***)** -> liste av ord, kan angi *sep, f.eks.* **'*n*' ','**

```
a = """Many
different
lines
"""
b = a.split()
```

```
'Many\ndifferent\nlines'
['Many', 'different', 'lines']
```

- '*tekst*'**.join(**liste**)** -> bruker *tekst* som forbindelse

nyttige string-funksjoner

- str**.split(***sep***)** -> liste av ord, kan angi *sep, f.eks.* **'\n' ','**

```
a = """Many
different
lines
"""
b = a.split()
```

```
'Many\ndifferent\nlines'
['Many', 'different', 'lines']
```

- '*tekst*'**.join(**liste**)** -> bruker *tekst* som forbindelse

```
a = ['en', 'to', 'tre']
b = '=='.join(a)
c = ''.join(a)
d = '\n'.join(a)
```

nyttige string-funksjoner

- str**.split(***sep***)** -> liste av ord, kan angi *sep, f.eks.* **'*n*' ','**

```
a = """Many
different
lines
"""
b = a.split()
```

```
'Many\ndifferent\nlines'
['Many', 'different', 'lines']
```

- '*tekst*'**.join(**liste**)** -> bruker *tekst* som forbindelse

```
a = ['en', 'to', 'tre']
b = '=='.join(a)
c = ''.join(a)
d = '\n'.join(a)
```

```
'en==to==tre'
```

nyttige string-funksjoner

- str**.split(***sep***)** -> liste av ord, kan angi *sep, f.eks.* **'\n' ','**

```
a = """Many
different
lines
"""
b = a.split()
```

```
'Many\ndifferent\nlines'
['Many', 'different', 'lines']
```

- '*tekst*'**.join(**liste**)** -> bruker *tekst* som forbindelse

```
a = ['en', 'to', 'tre']
b = '=='.join(a)
c = ''.join(a)
d = '\n'.join(a)
```

```
'en==to==tre'
'entotre'
```

nyttige string-funksjoner

- str**.split(***sep***)** -> liste av ord, kan angi *sep, f.eks.* **'*n*' ','**

```
a = """Many
different
lines
"""
b = a.split()
```

```
'Many\ndifferent\nlines'
['Many', 'different', 'lines']
```

- '*tekst*'**.join(**liste**)** -> bruker *tekst* som forbindelse

```
a = ['en', 'to', 'tre']
b = '=='.join(a)
c = ''.join(a)
d = '\n'.join(a)
```

```
'en==to==tre'
'entotre'
'en\nto\ntre'
```

nyttige string-funksjoner

- str**.split(***sep***)** -> liste av ord, kan angi *sep, f.eks.* **'*n*' ','**

```
a = """Many
different
lines
"""
b = a.split()
```

```
'Many\ndifferent\nlines'
```

```
['Many', 'different', 'lines']
```

- '*tekst*'**.join(**liste**)** -> bruker *tekst* som forbindelse

```
a = ['en', 'to', 'tre']
b = '=='.join(a)
c = ''.join(a)
d = '\n'.join(a)
```

```
'en==to==tre'
```

```
'entotre'
```

```
'en\nto\ntre'
```

```
"""en
to
tre"""
```

dict: dictionary { a:x, b:y, c:z }

- https://docs.python.org/3/tutorial/datastructures.html#dictionaries

- Key - Value pairs

```
tlf = {
  'Armin'  : 3542,
  'Mikasa' : 1234,
  'Eren'   : 5125,
}

tlf['Eren']

tlf['Levi'] = 4545

'Eren' in tlf

tlf.keys()
tlf.values()

tlf.items()
```

dict: dictionary { a:x, b:y, c:z }

- https://docs.python.org/3/tutorial/datastructures.html#dictionaries

- Key - Value pairs

```
tlf = {
  'Armin'  : 3542,
  'Mikasa' : 1234,
  'Eren'   : 5125,
}

tlf['Eren']

tlf['Levi'] = 4545

'Eren' in tlf

tlf.keys()
tlf.values()

tlf.items()
```

```
5125



True

dict_keys(['Armin', 'Mikasa', 'Eren', 'Levi'])
dict_values([3542, 1234, 5125, 4545])
```

dict: dictionary { a:x, b:y, c:z }

- https://docs.python.org/3/tutorial/datastructures.html#dictionaries

- Key - Value pairs

```
tlf = {
  'Armin'  : 3542,
  'Mikasa' : 1234,
  'Eren'   : 5125,
}

tlf['Eren']

tlf['Levi'] = 4545

'Eren' in tlf

tlf.keys()
tlf.values()

tlf.items()
```

```
5125



True

dict_keys(['Armin', 'Mikasa', 'Eren', 'Levi'])
dict_values([3542, 1234, 5125, 4545])
```

```
dict_items([('Armin', 3542), ('Mikasa', 1234), ('Eren',
5125), ('Levi', 4545)])
```

dict: dictionary { a:x, b:y, c:z }

- bruk i løkker:

```
dict_items([('Armin', 3542), ('Mikasa', 1234),
('Eren', 5125), ('Levi', 4545)])
```

```
for navn, num in tlf.items():
    print(navn,'--',num)
```

```
Armin -- 3542
Mikasa -- 1234
Eren -- 5125
Levi -- 4545
```

File I/O

- open(filename, mode) mode: **r**ead, **w**rite, **a**ppend

File I/O

- open(filename, mode) mode: **r**ead, **w**rite, **a**ppend

```
with open("some.txt","r") as f:
    all_in_one = f.read()
```

File I/O

- open(filename, mode) mode: **r**ead, **w**rite, **a**ppend

```
with open("some.txt","r") as f:
    all_in_one = f.read()
```

```
with open("another.txt","r") as f:
    list_of_lines = f.readlines()
```

File I/O

- open(filename, mode) mode: **r**ead, **w**rite, **a**ppend

```
with open("some.txt","r") as f:
    all_in_one = f.read()
```

```
with open("another.txt","r") as f:
    list_of_lines = f.readlines()
```

```
with open("third.txt","r") as f:
    for line in f:
        one_line = line
```

File I/O

- open(filename, mode) mode: **r**ead, **w**rite, **a**ppend

File I/O

- open(filename, mode) mode: **r**ead, **w**rite, **a**ppend

```
out = "Hello, how are you?"
with open(“hello.txt”,”w”) as f:
    f.write(out)
    f.write('\n')
```

File I/O

- open(filename, mode) mode: **r**ead, **w**rite, **a**ppend

```
out = "Hello, how are you?"
with open(“hello.txt”,”w”) as f:
    f.write(out)
    f.write('\n')
```

```
in = ‘inputdata.txt’
out = ‘reverse.txt’

with open(in,’r’) as fi, open(out,’w’) as fo:
    for line in fi:
        outline = line[::-1] # reverse
        fo.write(outline)
```

Løsningseksempel
ord i *alice.txt*

- Les inn *alice.txt* og
  tell hvor mange ganger hvert ord finnes i teksten
- Begynn med et tomt *dict*: `antall={ }`

```
with open('alice.txt') as f:
    for line in f:
        line = line.split()
        …
```

Løsningseksempel bokstaver i *alice.txt*

- Les inn *alice.txt* og
  tell hvor mange ganger hver bokstav finnes i teksten
- Bruk et dict: f.eks. `antall={ 'a':0, 'b':0, …}`

```
with open('alice.txt') as f:
    for line in f:
        for bokstav in line:
            …
```