Velkommen!

•••

johannph på mattermost
johannph@uio.no på mail!

DU SKAL TIL ENHVER TID…

Ha god håndhygiene, vasking og antibac

Holde avstand = 1 m

… Og føler du deg syk skal du gå hjem/holde deg hjemme

HVIS JEG FØLER MEG SYK

GÅ HJEM UMIDDEL-BART / HOLD DEG HJEMME

Kontakt helsevesenet gjennom fastlege / koronatelefon – de avklarer om testing er nødvendig

UiO er i kontakt med helsevesenet og omvendt

Om meg - Johanna

- Har en bachelor i statsvitenskap
- Gått to år informatikk prosa
- Går nå master prosa
- Kunne ikke noe om programmering før jeg startet
- Hadde P-matte på vgs (jeg ville heller spille candy crush) og ingen andre realfag.
- Har veldig høydeskrekk
- Elsker reality (selling sunset, anyone?)
- Send meg en melding på mattermost når som helst hvis du tenger hjelp med noe :)

Hvem skal jeg kontakte?

Spørsmål om faget:

Gruppelæreren din eller foreleser.

Studieinfo: spørsmål, klager, utsettelse av frister

https://www.mn.uio.no/ifi/studier/kontakt/

UiO forvei: bekymringer, noen å snakke med

https://www.mn.uio.no/studier/forvei/

SiO: fysisk og psykisk helsehjelp

https://www.sio.no/helse
https://www.sio.no/helse/noen-%C3%A5-snakke-med

Tips og trix

Gjør oppgaver helt til dere synes det er lett

In1000 er kjempeviktig for å forstå in1010 (som dere kanskje skal ta neste semester)

Gjør trix-oppgavene!

Ting bygger på hverandre, programmering krever innsats over tid og mengdetrening.

In1000 skal være 13t jobb i uka, ikke så rart at faget er vanskelig hvis man jobber mindre enn det.

Dere må gjøre oppgaver for å få til eksamen, oppgaver er alfa og omega.

Jobbe i grupper

Vi skal jobbe mye i grupper i dag, håper dere blir godt kjent! :)

Snakk sammen to og to

1) Navn, hvilken linje dere går, hvor dere er fra, funfact
2) Hva er dere stolte av å ha fått til i in1000?
3) Hvilke temaer henger dere bak på? Hva er vanskelig å forstå? Konkrete spørsmål?
4) Hver gruppe formulerer fire spørsmål som dere legger inn på mentimeter:
   https://www.menti.com/g9975zmhxv
   menti.com, kode: 89 33 87 6
5) Stem på de tingene dere også lurer på!

Boolske uttrykk

...

Hva er et boolsk uttrykk?

Et boolsk uttrykk evaluerer til enten `True` eller `False`

Alle disse evaluerer til en boolean:

```
"a" == "a"   >>True
1 != 2       >>True
1 == "1"     >>False
1 < 1        >>False
1 <= 1       >>True
```

“!=” leses “er ikke lik”

“<” leses “er mindre enn”

“<=” leses “er mindre enn eller lik”

Not

Dersom et uttrykk er usant, vil not() av uttrykket være sant, og omvendt.

```
not(False) > True

not(True) > False

not(5 == 5) > False

not(5 > 1) > False

not(5 > 20) > True

not(5 == 1) > True
```

Diskuter hva disse blir:

```
not("a" == "a")

not(1 != 2)

not(1 == "1")

not(1 < 1)

not(1 <= 1)
```

Sette sammen boolske uttrykk

Vi kan sette sammen flere boolske uttrykk, og vi skal lære to måter å gjøre det på:

1) And
2) Or

And

Begge uttrykkene må være sanne, både a og b!

```
False and False > False
```

```
False and True > False
```

```
True and True > True
```

Diskuter disse:

```
(5 == 1) and (5 > 20)  >
```

```
(5 == 1) and (5 > 1)  >
```

```
(5 == 5) and (5 > 1)  >
```

Or

Minst ett av uttrykkene må være sanne, enten a, eller b, eller begge!

```
False or False > False
```

```
False or True > True
```

```
True or True > True
```

Diskuter disse:

```
(5 == 1) or (5 > 20)  >
```

```
(5 == 1) or (5 > 1)  >
```

```
(5 == 5) or (5 > 1)  >
```

lf-tester

...

If-tester sjekker boolske uttrykk

Husk at alle boolske uttrykk evaluerer til enten `True` eller `False`! If testene sjekke boolske uttrykk og utfører koden hvis det boolske uttrykket er `True`.

```
if(True):
    #Do this
elif(True):
    #Do this
else:
    #Do this

#Man kan også ha en if-sjekk uten
en elif-sjekk eller else-sjekk:

if(True):
    #Do this
```

Eksempler

```
if(5 < 1):
   print("Hei på deg!")
else:
   print("Hade!")

# “Hade!” printes alltid
```

```
if(5 == 5):
   print("Hei på deg!")
else:
   print("Hade!")

# “Hei på deg!” printes alltid
```

Diskuter

```
# Hva er verdien til variabelen tekst etter at
følgende kode er utført?
tall = 7
tekst = "a"
if tall>10:
 tekst = tekst + "b"
elif tall<5:
 tekst = tekst + "c"
else:
 tekst = tekst + "d"
```

Løkker

Repeterende kode

Loops avslutter når uttrykket ikke evaluerer til True

```
for True:
    # Do something
# Avslutter når uttrykket etter for ikke lenger er True

while True:
    # Do something
# Avslutter når uttrykket etter while ikke lenger er True
```

Et godt tips er å bruke penn å papir til å sjekke hva som skjer i koden din, skriv ned hva som skjer for hver loop

Diskuter

```
# Hva skrives ut på skjermen
når følgende kode utføres?
tallene = [ ]
a = 0
b = 1
while a<4:
    tallene.append(b)
    b = b*2
    a = a+1
print(tallene[0] +
tallene[3])
```

```
# Hva er verdien til
variabelen a etter at
følgende kode er utført?
a = 0
for b in [2,4,1]:
    a = 2*a + b
```

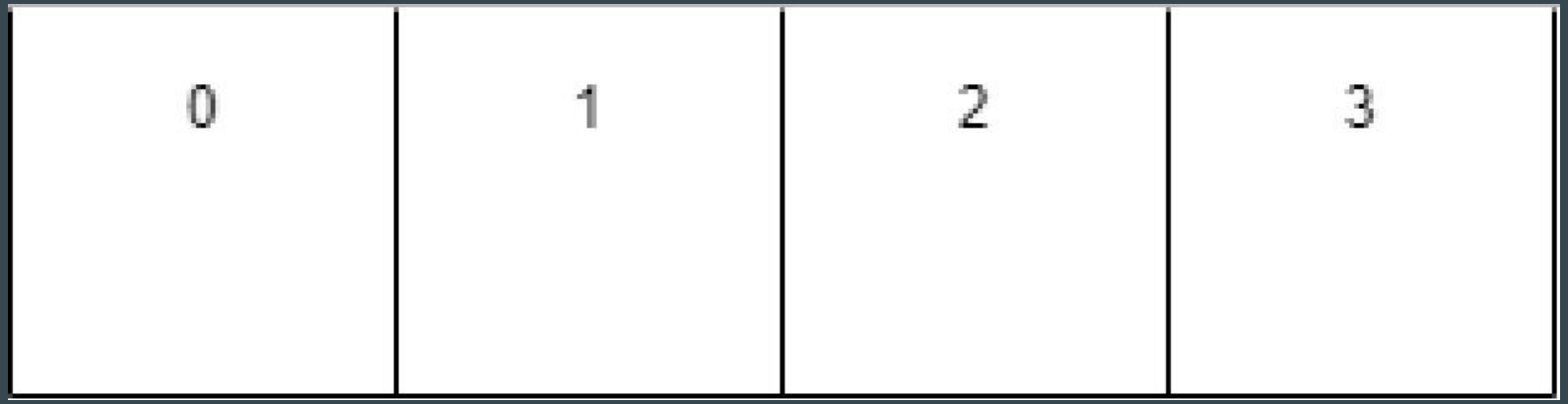
Samlinger

...

[0, 1, 5, 3, -2]

Hva er en liste?

- Som en parkeringsplass med plassnummer.
- Hver plass har plass til en bil.
- Kan dermed holde på flere verdier.
- Kan legge til verdier eller ta verdier ut av lista.



Nyttig funksjonalitet

- Lage en liste. (index) 0 1 2 3

  parkeringsplass = [“ledig”, “Volvo”, “BMW”, “ledig”]
- Indeksering - bestemme hvilken parkeringsplass vi vil se på.

  parkeringsplass[2] # Får ut verdien på index 2 → “BMW”
- Legg til elementer:

  parkeringsplass.append(“Skoda”)

Ordbok

Som en telefonbok

- En nøkkelverdi - analogi -> Navn i en telefonbok
- En innholdsverdi - analogi -> Telefonnummeret som hører til navnet

Lage en ordbok:

```
telefonbok = {
#   Nøkkel       Innholdsverdi
   "Lise"   :   41554365,
   "Ole"    :   49865732,
   "Mor"    :   47849302
}
```

Nyttig funksjonalitet

```
telefonbok = {
#   Nøkkel       Innholdsverdi
    “Lise”   :   41554365,
    “Ole”    :   49865732,
    “Mor”    :   47849302
}
```

- Hente ut en verdi:
  - telefonbok[<nøkkelverdi>]
  - telefonbok[“Mor”] # → henter ut verdien 47849302
- Legge til element:
  - telefonbok[<ny nøkkel>] = <ny innholdsverdi>
  - telefonbok[“Far”] = 91432453
  - print(telefonbok[“Far”]) #Skriver nå ut 91432453
- Kan ikke ha flere like nøkkelverdier, da overskriver man kun innholdsverdien.
- len(telefonbok) # henter antall elementer i ordboken.

Nøstet liste

Husker dere at vi kan ha liste i liste?

Veldig viktig at dere har god kontroll på hvordan nøstet liste ser ut.

1. Illustrer hovedliste på papir og finn "koordinatene" til hvert element.

Hvordan gjør man dette? Skriv kode på papir:

1. Print hvert element
2. Print "heidi" ved å aksessere det elementet i listen.

```
hovedliste = []
delliste1 = ["a", "b", "c"]
delliste2 = ["6", "5", "4"]
delliste3 = ["heidi", "kari",
"maria"]



hovedliste.append(delliste3)
hovedliste.append(delliste2)
hovedliste.append(delliste1)
```

Funksjoner, prosedyre og metoder

...

Funksjon vs prosedyre vs metode - diskuter forsjellene

Funksjon	En funksjon som defineres med *def*, som ikke er del av en klasse (ordet *self* brukes ikke). Funksjoner har alltid en returverdi.
	Eks:
	`def sum(a, b):`
	`  c = a + b`
	`  return c`
	På engelsk kalles dette *function*.

Prosedyre	Tilsvarende funksjon, men uten returverdi. Bruker aldri ordet *self*, og er ikke del av en klasse.
	Eks:
	`def superprint(ord):`
	`  print("ordet er", ord)`
	På engelsk kalles dette *procedure*.

Metode	Tilsvarende funksjon, men som del av en klasse. Har alltid *self* som første parameter. Kan, men må ikke, ha en returverdi.
	Eks:
	`def areal(self):`
	`  firkant_areal = self.lengde * self.bredde`
	`  return firkant_areal`
	På engelsk kalles dette *method*.

Diskuter i grupper

For hvert eksempel, hva printes?

```
johannasFunksjon()
```

1)

```
def johannasFunksjon():
    for i in range(0,10)
        print(i)
        if i > 5
            return i
        return i
```

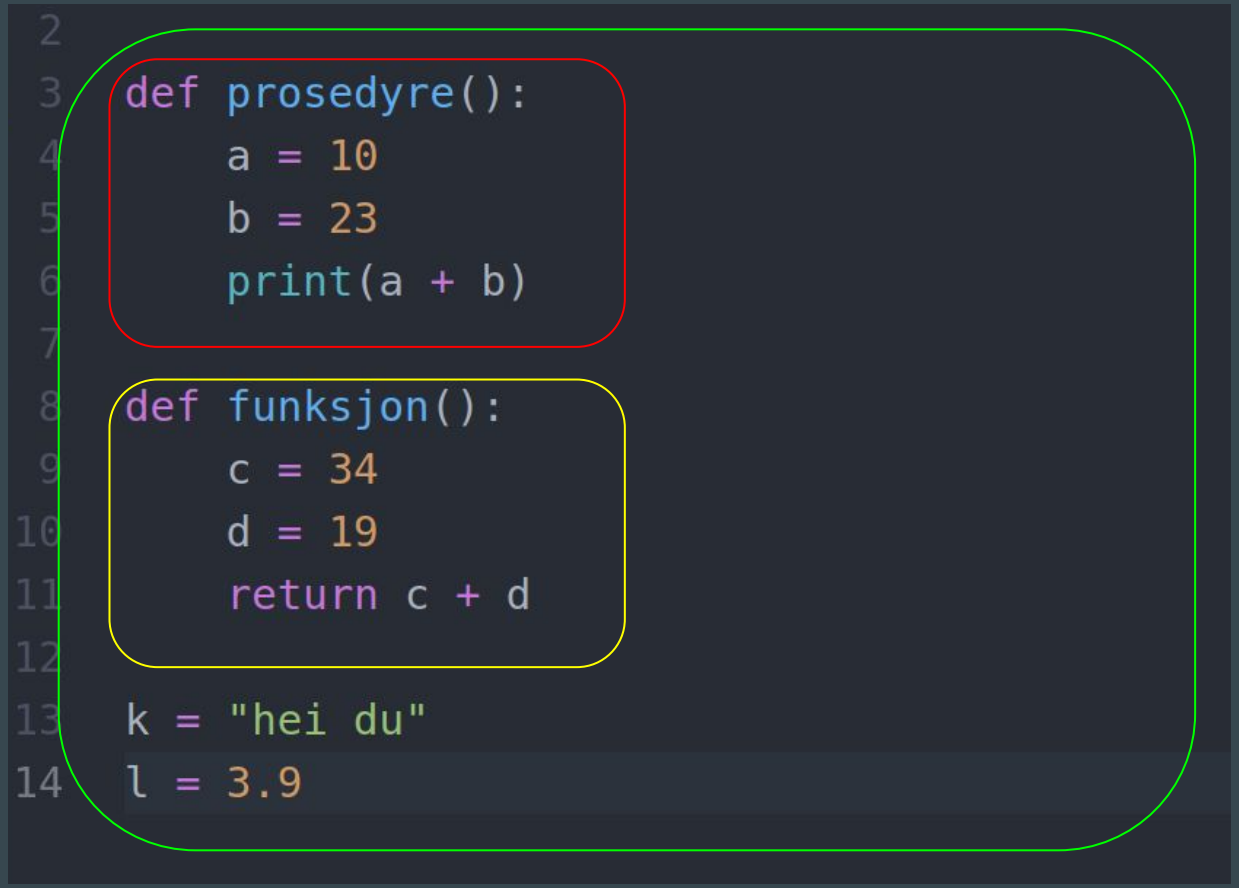
2)

```
def johannasFunksjon():
    for i in range(0,10)
        print(i)
        if i > 5
            return i
    return i
```

Skop

Kort sagt har alle tilgang på variablene som ligger i det globale skopet, mens det er en variabel inne i en prosedyre/funksjon er det kunne denne prosedyren/funksjonen, som har tilgang og kan bruke denne variabelen

Skop

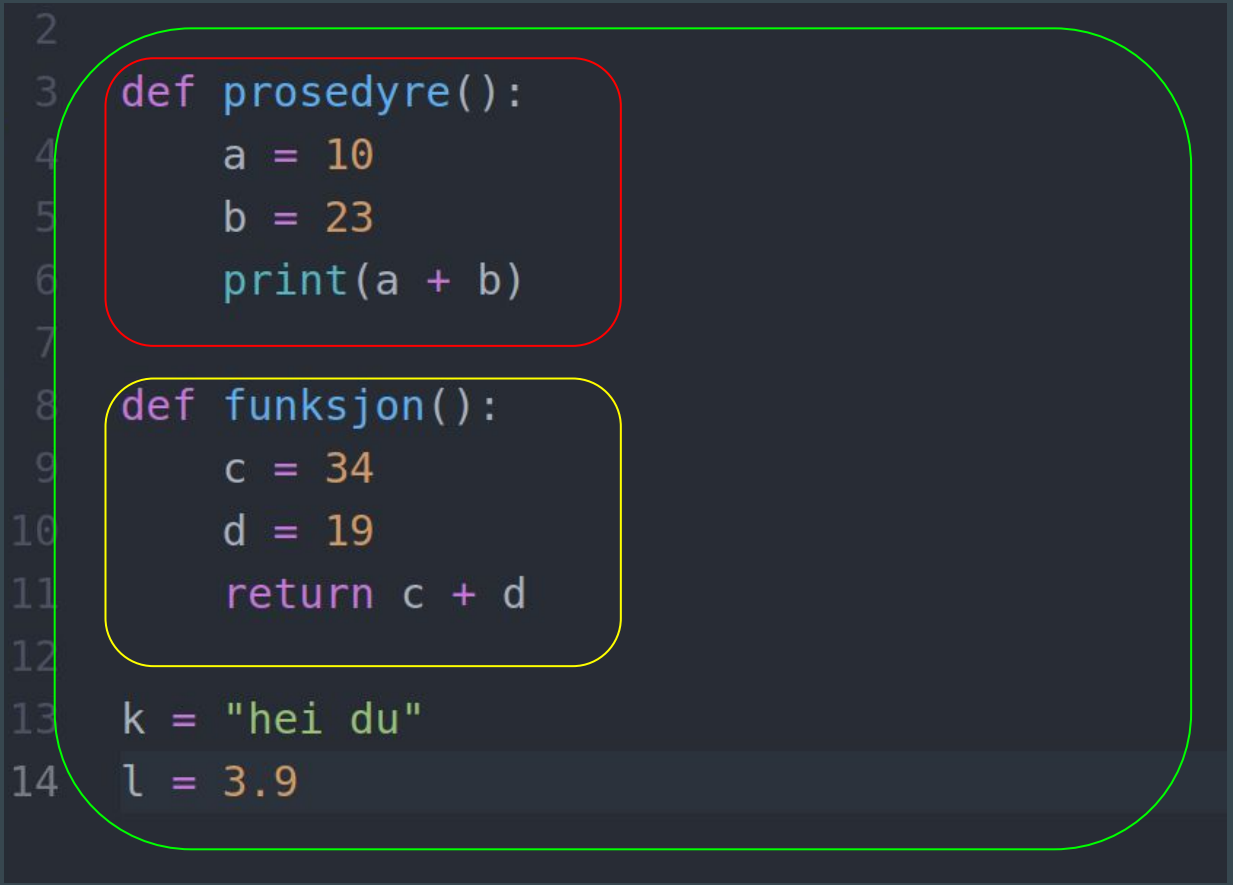

def prosedyre():
a = 10
b = 23
print(a + b)
def funksjon():
c = 34
d = 19
return c + d
k = "hei du"
l = 3.9

Skop

prosedyre kan bruke variablene: a,b,k,l

funksjon kan bruke variablene: c,d,k,l

det globale skopet kan kun bruke variablene: k og l



Diskuter

```
# Vi har en funksjon kalkuler som vist
nedenfor:
def kalkuler(tall):
    tall = tall + 1
    return tall*2
# Hva skrives ut på skjermen når følgende
kode utføres?
print( kalkuler(2) + kalkuler(4) )
```

Oppgaver

...

Dere kan velge mellom to oppgaver

Mål: øve på nøstet liste

1. Turbuss
2. Trikk (litt vanskelig)

None

Vi bruker None når vi vil si at “Her er det ingenting”.

None er en egen datatype, sånn som også string og Integer/int er datatyper.

En prosedyre har ingen returverdi, hvis vi prøver å printe returverdien til en prosedyre får vi None.

[]

```
def funksjon():
    return 5

def prosedyre():
    #Her kan det skje mange ting,
     men det er ingen return i en
     prosedyre
    tall = 5 + 6
    string = "heihie"

print(funksjon())
print(prosedyre())
```

Turbuss

1. Les hele oppgaven og lag en datastruktutegning
2. Lag en klasse turbuss.
3. Den skal ha en konstruktør som tar inn antall rader i bussen.
4. Den skal ha en instansvariabel antallPlasserPerRad som alltid er 4.
5. Lag metodene _lagBuss() og __str__(), disse skal være slik som metoden _lagTrikk() og __str__() i trikkeoppgaven, se den filen.
6. Lag en metode leggTilPassasjer(passasjer), som legger til en passajer på første ledige sete, denne er lik som gaaPaaTrikk() i trikkeoppgaven.
7. Lag en privat metode _finnNaboer(), denne er lik som _finnNaboer() i trikkeoppgaven
8. Legg til passasjerer manuelt for å teste programmet deres, bruk klassen PassasjerBuss.

Datastruktur

:Trikk

_ruteNummer =

_plasserPerRad = 5

_plasser =

Datastruktur

```
class Trikk:
  def __init__(self, rutenr, antallRader):
```

```
trettenTrikken = Trikk(13, 10)
```

:Trikk

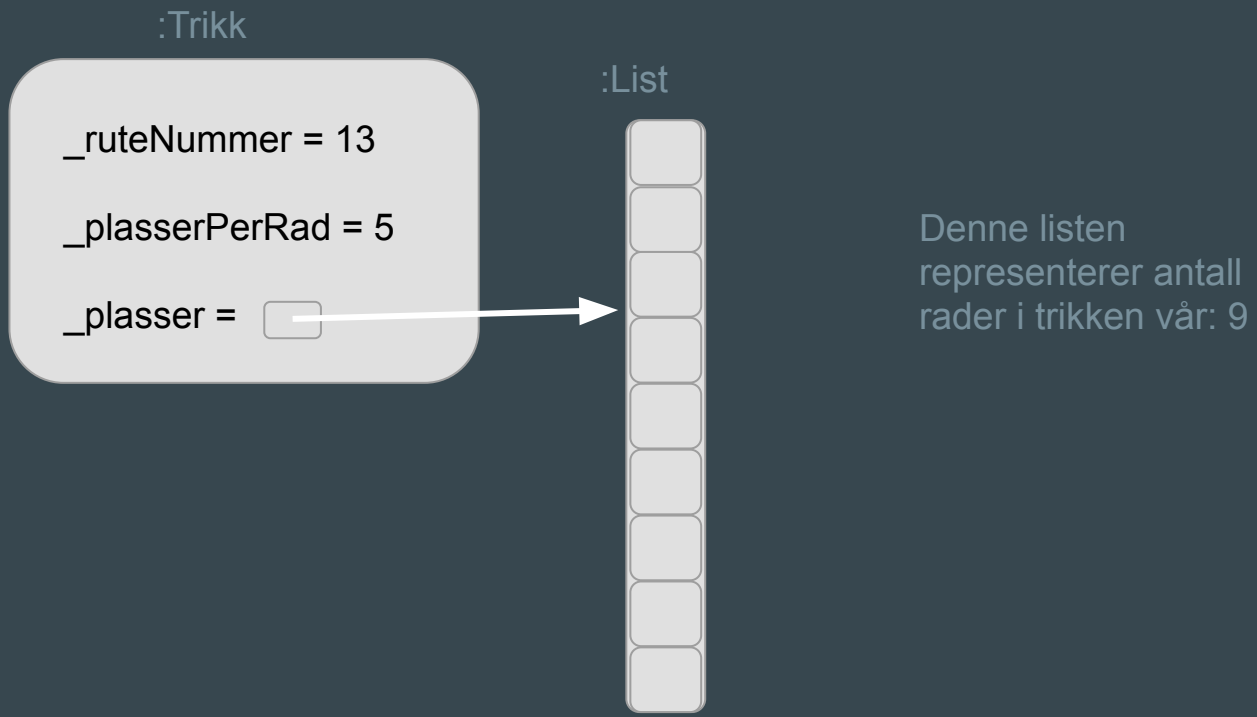
_ruteNummer = 13

_plasserPerRad = 5

_plasser = 

Datastruktur

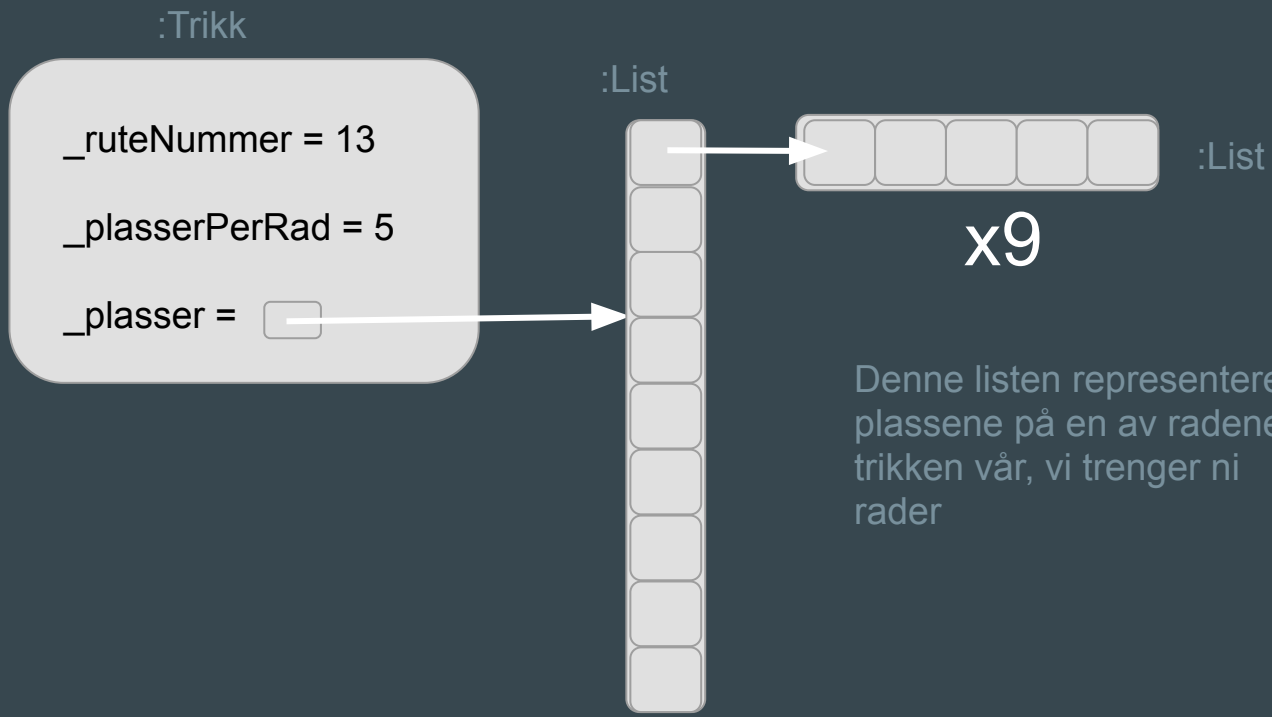
```
class Trikk:
  def __init__(self, rutenr, antallRader):

trettenTrikken = Trikk(13, 9)
```



Denne listen representerer antall rader i trikken vår: 9

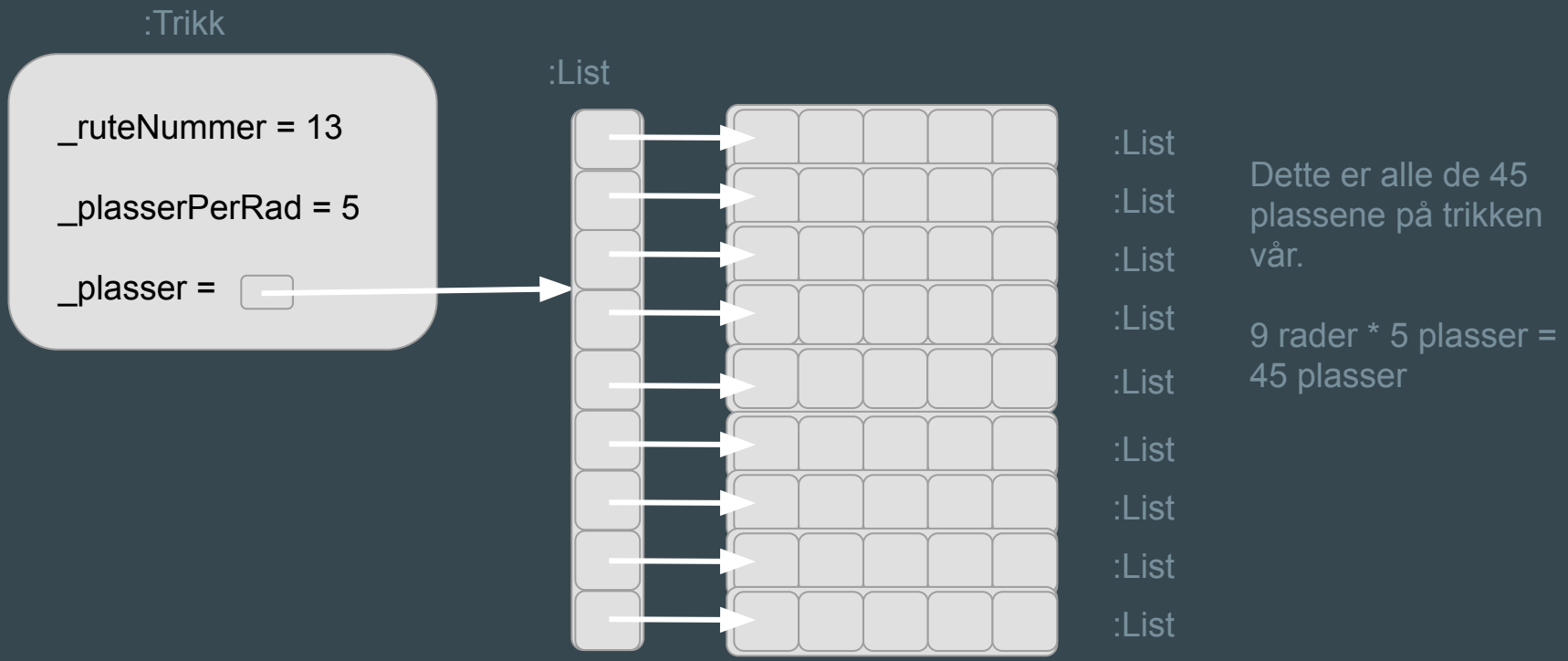
Datastruktur

```
class Trikk:
  def __init__(self, rutenr, antallRader):

trettenTrikken = Trikk(13, 10)
```



Datastruktur

```
class Trikk:
  def __init__(self, rutenr, antallRader):

trettenTrikken = Trikk(13, 10)
```



Dette er alle de 45 plassene på trikken vår.

9 rader * 5 plasser = 45 plasser

Datastruktur

```
class Trikk:
  def __init__(self, rutenr, antallRader):

trettenTrikken = Trikk(13, 10)
```