Forelesning IN1900 – 5 September 2022

Ole Christian Lingjærde
Institutt for Informatikk, Universitetet i Oslo

Uke: 5 September – 11 September, 2022

Forrige forelesning på en foil

Lage sekvenser av tall:

[1,2,5]

range(5), range(2,7), range(2,7,2).

[0]*n

[2*i+1 for i in range(5)]

Liste-indeksering:

a[: stopp]

a[start :]

a[start : stopp]

Tupler:

som lister, men kan ikke endres

Tema i dag

Lese to lister i parallell med zip

Lister av lister

If-tester

Funksjoner

Plenumsoppgaver (så langt vi rekker, resten på onsdag):
2.1, 2.3, 2.4, 2.7, 2.8, 2.14, 2.15, 3.20, 3.23, 3.28

Løpe gjennom to lister samtidig

Anta at A og B er lister av samme lengde.

Vi kan gå gjennom dem i parallell slik:

```
A = [1, 3, 5]
B = [1, 9, 25]
for i in range(len(A)):
    print(A[i], B[i])
```

og svaret blir da:

```
1 1
3 9
5 25
```

Funksjonen `zip`

Alternativ metode: bruke **zip**

```
A = [1, 3, 5]
B = [1, 9, 25]

for a,b in zip(A,B):
    print(a,b)
```

og svaret blir som før:

```
1 1
3 9
5 25
```

Hva gjør egentlig `zip`?

Anta at `a` og `b` er lister av samme lengde.

`zip(a,b)` samler listeelementer med samme plassering i et tuppel:

```
zip([4,5,6], [7,8,9])  # (4,7), (5,8), (6,9)
```

Vi kan også bruke zip på tre eller flere lister:

```
a = [0,1,2]
b = [3,4,5]
c = [11,12,13]
zip(a,b,c)             # (0,3,11), (1,4,12), (2,5,13)
```

Lister av lister (tabeller)

Elementene i en liste kan selv være lister:

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

# A[0] er listen [1, 3, 5]
# A[1] er listen [2, 4, 6]

# A[0][0] er 1
# A[0][1] er 3
# A[0][2] er 5
# A[1][0] er 2
# A[1][1] er 4
# A[1][2] er 6
```

Naturlig å tolke A i eksemplet over som en matrise/tabell:

$$A = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

Kort oppsummering av lister

Kommando	Mening
`a = []`	Lag en tom liste
`a = [1, 4.4, 'run.py']`	Lag en liste med angitte verdier
`a.append(4.4)`	Utvid listen a med verdien 4.4
`a +[1,3]`	Slå sammen (konkatenér) to lister
`a.insert(i, 99)`	Sett inn 99 i listen a slik at den får indeks i
`a[0]`	Verdien med indeks 0 (førsteplassen)
`a[2]`	Verdien med indeks 2 (tredjeplassen)
`a[-1]`	Verdien som ligger sist i listen
`a[1:3]`	Ny liste med to elementer a[1], a[2]
`del a[3]`	Fjern elementet med indeks 3
`a.index(99)`	Finn første indeks som inneholder verdien 99
`99 in a`	Er verdien 99 i listen a? (True eller False)
`a.count(99)`	Hvor mange ganger forekommer 99 i listen?
`len(a)`	Hvor lang er listen a?
`min(a)`	Minste element i listen a
`max(a)`	Største element i listen a
`sum(a)`	Beregn summen av alle elementer i a
`sorted(a)`	Returner en sortert versjon av a
`reversed(a)`	Returner en reversert versjon av a
`isinstance(a, list)`	Er a en liste? (True eller False)
`type(a) is list`	Er a en liste? (True eller False)

If-tester

Vi skal lese et tall fra terminal og skrive ut logaritmen til tallet. Vi lager et program logx.py:

```
import math
print("x = ?")
x = eval(input()) # Leser x fra terminal
logx = math.log(x)
print(f"log({x}) = {logx}")
```

Gir dette programmet alltid fornuftig output?

If-tester (forts.)

Vi prøver programmet med en positiv x:

```
> python logx.py
x = ?
2.7
log(2.7) = 0.9932517730102834
```

Vi prøver igjen, nå med en negativ x:

```
> python logx.py
x = ?
-11
Traceback (most recent call last):
File "<ipython-input-63-b644e331e7a0>", line 4, in <module>
logx = math.log(x)
ValueError: math domain error
```

If-tester (forts.)

Vi utvider programmet med en **if-test** slik at det bare regner ut logaritmen når x > 0 og gir feilmelding ellers:

```
import math
print("x = ?")
x = eval(input())
if x > 0:
    logx = math.log(x)
    print(f"log({x})={logx}")
else:
    print("log(x) er bare definert for x > 0")
```

If-tester (forts.)

Vi prøver programmet med en positiv x:

```
> python logx.py  x = ?
2.7
log(2.7) = 0.9932517730102834
```

Vi prøver igjen, nå med en negativ x:

```
> python logx.py  x = ?
-11
log(x) er bare definert for x > 0
```

→ Fornuftig utskrift uansett fortegn av x.

If-tester (forts.)

Programmet vil fortsatt feile dersom brukeren skriver inn noe annet enn et tall:

```
> python logx.py
x = ?
femti
Traceback (most recent call last):

File "<ipython-input-5-b6b12d30e678>", line 3, in <module>
    x = eval(input())

  File "<string>", line 1, in <module>

NameError: name 'femti' is not defined
```

If-tester (forts.)

Vi utvider programmet slik at det først tester om input er numerisk og gir feilmelding hvis ikke, og deretter tester om x > 0:

```
import math
print("x = ?")
answer = input()
if answer.isnumeric():
    x = eval(answer)
    if x > 0:
        logx = math.log(x)
        print(f"log({x}) = {logx}")
    else:
        print("x kan ikke være 0")
else:
    print("x må være et positivt tall")
```

If-tester (forts.)

```
import math
print("x = ?")
answer = input()
if answer.isnumeric():
    x = eval(answer)
    if x > 0:
        logx = math.log(x)
        print(f"log({x}) = {logx}")
    else:
        print("x kan ikke være 0")
else:
    print("x må være et positivt tall")
```

Merk: isnumeric returnerer True når alle tegn er sifre, så -2 er ikke numerisk ifølge denne funksjonen.

If-tester (forts.)

```
import math
print("x = ?")
answer = input()
if answer.isnumeric():
    x = eval(answer)
    if x > 0:
        logx = math.log(x)
        print(f"log({x}) = {logx}")
    else:
        print("x kan ikke være 0")
else:
    print("x må være et positivt tall")
```

If-tester (forts.)

Nå fungerer programmet for alle typer input:

```
> python logx.py
x = ?
50
log(50) = 3.912023005428146

> python logx.py
x = ?
0
x kan ikke være 0

> python logx.py
x = ?
-50
x må være et positivt tall

> python logx.py
x = ?
femti
x må være et positivt tall
```

If-tester: de tre variantene

If:

```
if x < 0:
  setninger
```

If-else:

```
if x < 0:
  setninger
else:
  setninger
```

If-elif-else:

```
if x < 0:
  setninger
elif x < 5:
  setninger
else:
  setninger
```

Quiz

Hva skrives ut her?

```
x = 3.14

if x < x**2:
   print("A", end="")

if x < 0:
   print("B", end="")
else:
   x = -x
   print("C", end="")

if x < x**2:
   print("D", end="")
else:
   print("E", end="")
```

Hindrer ny linje etter utskrift

Quiz

Hva skrives ut her?

```
x = 3.14

if x < x**2:
   print("A", end="")

if x < 0:
   print("B", end="")
else:
   x = -x
   print("C", end="")

if x < x**2:
   print("D", end="")
else:
   print("E", end="")
```

Svar: ACD

Quiz

Vi kjører dette programmet:

```
x = 0
if x >= 0:
    x = x + 2
if x-2 <= 0:
     x = x * 2
     if x%2 != 0:
          x = x - 1
     else:
          x = x + 1
else:
    x = x * 6
```

Hvilken verdi har nå x?

Quiz

Vi kjører dette programmet:

```
x = 0
if x >= 0:
    x = x + 2
if x-2 <= 0:
     x = x * 2
     if x%2 != 0:
          x = x - 1
     else:
          x = x + 1
else:
    x = x * 6
```

Hvilken verdi har nå x?

Svar: 5

Quiz

Vi kjører dette programmet:

```
x = 0
for k in range(6):
   if k % 2 == 0:
      x = x + k
   else:
      x = x - k
```

Hvilken verdi har nå x?

Quiz

Vi kjører dette programmet:

```
x = 0
for k in range(6):
   if k % 2 == 0:
      x = x + k
   else:
      x = x - k
```

k	k%2==0	x = ...	x

Quiz

Vi kjører dette programmet:

```
x = 0
for k in range(6):
   if k % 2 == 0:
      x = x + k
   else:
      x = x - k
```

k	k%2==0	x = ...	x
0	True	x=x+0	0

Quiz

Vi kjører dette programmet:

```
x = 0
for k in range(6):
   if k % 2 == 0:
      x = x + k
   else:
      x = x - k
```

k	k%2==0	x = ...	x
0	True	x=x+0	0
1	False	x=x-1	-1

Quiz

Vi kjører dette programmet:

```
x = 0
for k in range(6):
   if k % 2 == 0:
      x = x + k
   else:
      x = x - k
```

k	k%2==0	x = ...	x
0	True	x=x+0	0
1	False	x=x-1	-1
2	True	x=x+2	1

Quiz

Vi kjører dette programmet:

```
x = 0
for k in range(6):
   if k % 2 == 0:
      x = x + k
   else:
      x = x - k
```

k	k%2==0	x = ...	x
0	True	x=x+0	0
1	False	x=x-1	-1
2	True	x=x+2	1
3	False	x=x-3	-2

Quiz

Vi kjører dette programmet:

```
x = 0
for k in range(6):
   if k % 2 == 0:
      x = x + k
   else:
      x = x - k
```

k	k%2==0	x = ...	x
0	True	x=x+0	0
1	False	x=x-1	-1
2	True	x=x+2	1
3	False	x=x-3	-2
4	True	x=x+4	2

Quiz

Vi kjører dette programmet:

```
x = 0
for k in range(6):
   if k % 2 == 0:
      x = x + k
   else:
      x = x - k
```

k	k%2==0	x = ...	x
0	True	x=x+0	0
1	False	x=x-1	-1
2	True	x=x+2	1
3	False	x=x-3	-2
4	True	x=x+4	2
5	False	x=x-5	**-3**

Funksjoner i Python

Python har innebygd en lang rekke funksjoner slik at vi kan utføre komplekse operasjoner med én enkelt programsetning:

```
import math
x = math.sin(0.24)      # Sinus til 0.24
y = math.atanh(0.2)     # Invers hyberbolsk tangens til 0.2
z = sorted([6,2,3,7]) # Lager en sortert liste
a = list(range(50))     # Lager liste med verdiene 0,1,...,49
```

Bak hver av funksjonene **math.sin**, **math.atanh**, **sorted** osv. ligger det mange linjer med programkode som er skjult for oss.

Hvorfor skjule programkode?

Gode grunner til å skjule programkode. Ta for eksempel funksjonen **math.sin(x)**:

- Må i praksis beregnes hver gang, f.eks. ved å bruke de første leddene av Taylorrekken

$$sin(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$$

- Programkoden for dette er ofte lang og ikke særlig interessant hvis vi bare vil vite svaret

- Koden kan være i et annet programmerings-språk (ofte C) og kan variere avhengig av operativsystem og Python-versjon

Eksempel

Program for å regne ut sinus med formelen over:

```
x = 0.15
y = x
kfac = 1
sign = 1
for k in range(3, 50, 2):
  kfac = kfac * (k-1) * k
  sign = -sign
  y = y + sign * x**k / kfac
print(f"sin({x}) = {y:.6f}")
```

Enklere (hvis vi bare ønsker svaret):

```
x = 0.15
import math
y = math.sin(x)
print(f"sin({x}) = {y:.6f}")
```

Eksempel

Program for å regne ut sinus med formelen over:

```
x = 0.15
y = x
kfac = 1
sign = 1
for k in range(3, 50, 2):
  kfac = kfac * (k-1) * k
  sign = -sign
  y = y + sign * x**k / kfac
print(f"sin({x}) = {y:.6f}")
```

Enklere (hvis vi bare ønsker svaret)

```
x = 0.15
import math
y = math.sin(x)
print(f"sin({x}) = {y:.6f}")
```

Selvdefinerte funksjoner

Lett å definere egne funksjoner i Python:

```
def fnk(x,y,z):
  programsetninger
```

Her er:

- `fnk`: navnet vi har gitt funksjonen
- `x,y,z`: variablene til funksjonen (null, en eller flere)
- `programsetninger`: det vi ønsker skal utføres

Variablene i en Python-funksjon kalles også argumenter, input-variabler og formelle parametre.

Eksempel 1: Regne ut annengradspolynom

Funksjon som regner ut verdien til annengradspolynomet $x^2 + 3x - 5$ for en gitt x :

```
def f(x):
    value = x**2 + 3*x - 5
    return value
```

Vi har gitt funksjonen navnet `f`, men den kunne også vært kalt `g` eller `funksjon_som_beregner_verdi_av_et_polynom`.

Eksempel på bruk:

```
print(0.5, f(0.5))
```

```
0.5 -3.25
```

Komplett program

```
def f(x):
   value = x**2 + 3*x - 5
   return value

print("x = ?")
x = eval(input())
print(f"Hvis x = {x} så er x^2+3x-5 = {f(x)}")
```

Kjøreeksempel:

```
x = ?
6
Hvis x = 6 så er x^2+3x-5 = 49
```

Eksempel 2: Løse annengradslikning

Løsningene av annengradslikningen $ax^2 + bx + c = 0$:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Python-funksjon som finner løsningene:

```
def solve(a, b, c):
  from math import sqrt
  x1 = (-b - sqrt(b**2 - 4*a*c)) / (2*a)
  x2 = (-b + sqrt(b**2 - 4*a*c)) / (2*a)
  return x1, x2
```

Eksempel: vi løser $3x^2 + 2x - 1 = 0$ med

```
print(solve(3, 2, -1))
```

Komplett program

Her er et komplett program for å løse likningen $ax^2 + bx + c = 0$ når vi kjenner verdiene til a, b, c:

```
def solve(a, b, c):
    from math import sqrt
    x1 = (-b - sqrt(b**2 - 4*a*c)) / (2*a)
    x2 = (-b + sqrt(b**2 - 4*a*c)) / (2*a)
    return x1, x2

print("a, b, c = ?")
a, b, c = eval(input())
x1, x2 = solve(a, b, c)
print(f"Løsningene er x1={x1} og x2={x2}")
```

Kjøreeksempel:

```
a, b, c = ?
4, 2, -2
Løsningene er x1=-1.0 og x2=0.5
```

Eksempel 3: Funksjon som leter i en liste

Kan vi lage en funksjon som finner ut om en liste a inneholder en bestemt verdi `x`?

Idé: løp gjennom alle elementene i listen med løkke, test om noen av elementene er lik x.

```
def findvalue(a, x):
    found = False
    for i in range(len(a)):
        if a[i] == x:
            found = True
    return found
```

Komplett program

```
# Vi definerer letefunksjonen:
def findvalue(a,x):
   found = False
   for i in range(len(a)):
      if a[i] == x:
         found = True
   return found


# Sjekk om listen L=[11, 2, 56, 3] inneholder x=56:

L = [11, 2, 56, 3]
found = findvalue(L,56)
if found:
   print(f"L inneholder verdien {x}")
else:
   print(f"L inneholder ikke verdien {x}")
```

Funksjoner uten argumenter

Vi kan lage funksjoner uten input-variabler, f.eks:

```
def printstars():
    for i in range(50):
        print("*", end="")
```

NB: må fortsatt ha med parentesene både i definisjonen og ved senere bruk:

```
printstars()
```

Eksempel: kalender

Funksjon som finner tidspunktet programmet kjøres:

```
def day():
  import time
  day = time.gmtime().tm_yday
  year = time.gmtime().tm_year
  text = f"Det er nå dag nummer {day:g} i år {year:g}"
  return text
```

Svaret som returneres avhenger av tiden på datamaskinen:

```
print(day())
```

```
Det er nå dag nummer 249 i år 2021
```

Returverdier

Funksjoner kan ha null, en eller flere returverdier:

```
def gratulasjon(navn):
    print(f"Gratulerer {navn}, du har vunnet 1 million kroner!")

def kvadrat(x):
    return x**2

def potenser(x):
    return x, x**2, x**3, x**4
```

Plenumsoppgaver

Exercise 2.1: Make a Fahrenheit-Celsius conversion table
Write a Python program that prints out a table with Fahrenheit degrees 0, 10, 20, . . . , 100 in the first column and the corresponding Celsius degrees in the second column.

Hint Modify the `c2f_table_while.py` program from Sect. 2.1.2.
Filename: `f2c_table_while`.

Plenumsoppgaver

Exercise 2.1: Make a Fahrenheit-Celsius conversion table
Write a Python program that prints out a table with Fahrenheit degrees 0, 10, 20, . . . , 100 in the first column and the corresponding Celsius degrees in the second column.

Hint Modify the `c2f_table_while.py` program from Sect. 2.1.2.
Filename: `f2c_table_while`.

Fra tidligere har vi følgende formel for å regne om fra Celcius til Fahrenheit:

$$F = (9/5)C + 32$$

Dermed er

$$C = (5/9)(F - 32)$$

og vi kan løse oppgaven med en enkel for-løkke eller while-løkke.

Plenumsoppgaver

Exercise 2.3: Work with a list

Set a variable `primes` to a list containing the numbers 2, 3, 5, 7, 11, and 13. Write out each list element in a `for` loop. Assign 17 to a variable `p` and add `p` to the end of the list. Print out the entire new list.

Filename: `primes`.

Plenumsoppgaver

Exercise 2.4: Generate odd numbers

Write a program that generates all odd numbers from 1 to `n`. Set `n` in the beginning of the program and use a `while` loop to compute the numbers. (Make sure that if `n` is an even number, the largest generated odd number is `n-1`.)

Filename: `odd`.

Plenumsoppgaver

Exercise 2.7: Generate equally spaced coordinates
We want to generate $n + 1$ equally spaced x coordinates in $[a, b]$. Store the coordinates in a list.

a) Start with an empty list, use a `for` loop and append each coordinate to the list.

Hint With n intervals, corresponding to $n + 1$ points, in $[a, b]$, each interval has length $h = (b - a)/n$. The coordinates can then be generated by the formula $x_i = a + ih, i = 0, \ldots, n + 1$.

b) Use a list comprehension as an alternative implementation.

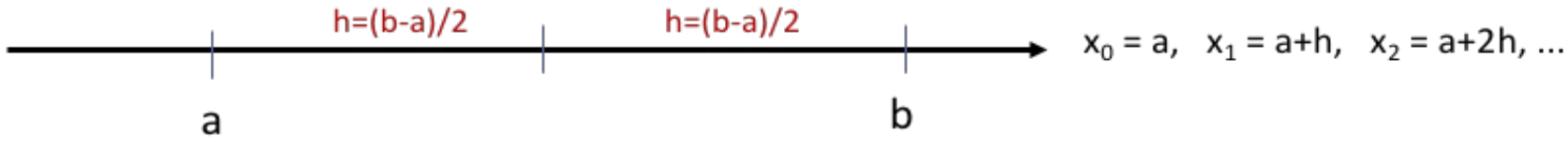
Filename: `coor`.

Oppgave 2.7: Løsningsskisse

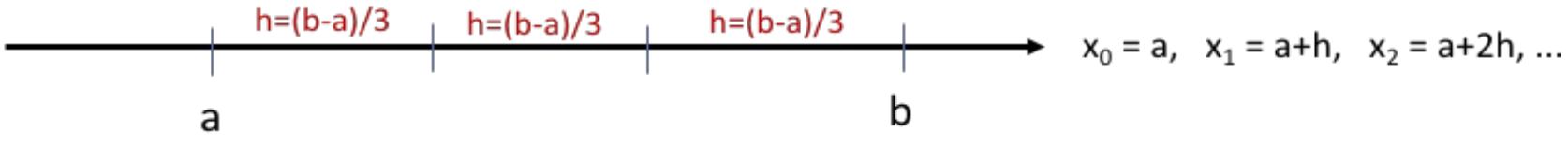
Vi ønsker å dele opp intervallet $[a, b]$ i n like lange delintervaller. Da må hvert delintervall ha lengde $h = (b - a)/n$. Intervallene blir:

$[a, a + h]$, $[a + h, a + 2h]$, $[a + 2h, a + 3h]$, ...

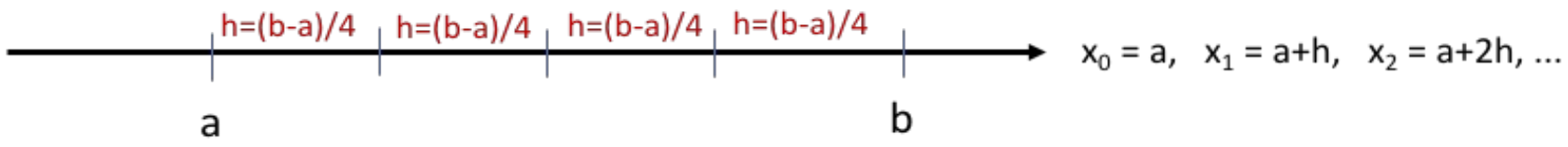
n = 2 intervaller:



n = 3 intervaller:



n = 4 intervaller:



Plenumsoppgaver

Exercise 2.8: Make a table of values from a formula

The purpose of this exercise is to write code that prints a nicely formatted table of t and $y(t)$ values, where

$$y(t) = v_0 t - \frac{1}{2} g t^2 .$$

Use $n + 1$ uniformly spaced t values throughout the interval $[0, 2v_0/g]$.

a) Use a `for` loop to produce the table.
b) Add code with a `while` loop to produce the table.

Hint Because of potential round-off errors, you may need to adjust the upper limit of the `while` loop to ensure that the last point ($t = 2v_0/g$, $y = 0$) is included.
Filename: `ball_table1`.

Oppgave 2.8: Løsningsskisse

Vi lager først en liste med alle t-verdiene. Vi skal ha $n + 1$ verdier med jevn avstand på $[0, b]$ hvor $b = 2v_0/g$.

Avstanden mellom nabopunkter blir $h = (b - 0)/n = b/n$.

Punktene blir dermed:

$$t = 0, 0 + h, 0 + 2h, \ldots, 0 + nh$$

De tilhørende y-verdiene finner vi med formelen som er oppgitt.

Plenumsoppgaver

Exercise 2.14: Explore Python documentation
Suppose you want to compute with the inverse sine function: $\sin^{-1} x$. How do you do that in a Python program?

Hint The `math` module has an inverse sine function. Find the correct name of the function by looking up the module content in the online Python Standard Library[7] document or use `pydoc`, see Sect. 2.6.3.
Filename: `inverse_sine`.

Oppgave 2.14: Løsningsskisse

Mange muligheter, f.eks.:

- Google "inverse sine python"
- Søk på https://docs.python.org/3/library
- Skriv pydoc math i kommandovinduet Skriv
- !pydoc math i Ipython-konsollet i Spyder

Plenumsoppgaver

Exercise 2.15: Index a nested list

We define the following nested list:

```
q = [['a', 'b', 'c'], ['d', 'e', 'f'], ['g', 'h']]
```

a) Index this list to extract 1) the letter `a`; 2) the list `['d', 'e', 'f']`; 3) the last element `h`; 4) the `d` element. Explain why `q[-1][-2]` has the value `g`.

b) We can visit all elements of `q` using this nested `for` loop:

```
for i in q:
    for j in range(len(i)):
        print i[j]
```

What type of objects are `i` and `j`?

Filename: `index_nested_list`.

Plenumsoppgaver

Exercise 3.20: Write functions

Three functions, `hw1`, `hw2`, and `hw3`, work as follows:

```
>>> print hw1()
Hello, World!
>>> hw2()
Hello, World!
>>> print hw3('Hello, ', 'World!')
Hello, World!
>>> print hw3('Python ', 'function')
Python function
```

Write the three functions.

Filename: `hw_func`.