Forelesning IN1900 – 5 okt 2022

Ole Christian Lingjærde
Institutt for Informatikk, Universitetet i Oslo

Uke: 3 Oktober – 9 Oktober, 2022

Denne ukens agenda

Dictionaries (mandag)

Oppgave A.14 og 5.14 i Langtangens bok (i dag)

Tekststrenger (i dag)

Oppvarming til midtveiseksamen (i dag)

Exercise A.14: Find difference equations for computing sin x

The purpose of this exercise is to derive and implement difference equations for computing a Taylor polynomial approximation to $\sin x$:

$$\sin x \approx S(x;n) = \sum_{j=0}^{n} (-1)^j \frac{x^{2j+1}}{(2j+1)!}\,. \tag{A.60}$$

To compute $S(x;n)$ efficiently, write the sum as $S(x;n) = \sum_{j=0}^{n} a_j$, and derive a relation between two consecutive terms in the series:

$$a_j = -\frac{x^2}{(2j+1)2j} a_{j-1}\,. \tag{A.61}$$

Introduce $s_j = S(x; j-1)$ and a_j as the two sequences to compute. We have $s_0 = 0$ and $a_0 = x$.

a) Formulate the two difference equations for s_j and a_j.

Exercise A.14

Trinn 1

Vi tar utgangspunkt i Taylorrekken til sin(x):

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

Trinn 2

Vi kan bruke formelen over til å lage et program som finner en god tilnærming til sin(x), f.eks. slik:

```
from math import factorial
y = 0     # Skal holde approksimasjonen
m = 1     # Fortegn (enten -1 eller 1)
for i in range(1,9,2):
    y += m * x**i / factorial(i)
    m = -m
```

Exercise A.14

Trinn 3

Programmet i trinn 2 er ikke veldig effektivt. La oss se hvilke matematiske operasjoner vi gjør:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

Ledd nr	Operasjoner
1	Ingen
2	x^3 , 3! , divisjon
3	x^5 , 5! , divisjon
4	x^7 , 7! , divisjon

Kan vi gjøre dette mer effektivt?

Hint: gjenbruk

Exercise A.14

Trinn 4

Vi observerer at hvert ledd i rekken er ganske lik det foregående leddet. Mer presist:

$$\frac{x^3}{3!} = x \cdot \frac{x^2}{2 \cdot 3}$$

Exercise A.14

Trinn 4

Vi observerer at hvert ledd i rekken er ganske lik det foregående leddet. Mer presist:

$$\frac{x^3}{3!} = x \cdot \frac{x^2}{2 \cdot 3}$$

$$\frac{x^5}{5!} = \frac{x^3}{3!} \cdot \frac{x^2}{4 \cdot 5}$$

Exercise A.14

Trinn 4

Vi observerer at hvert ledd i rekken er ganske lik det foregående leddet. Mer presist:

$$\frac{x^3}{3!} = x \cdot \frac{x^2}{2 \cdot 3}$$

$$\frac{x^5}{5!} = \frac{x^3}{3!} \cdot \frac{x^2}{4 \cdot 5}$$

$$\frac{x^7}{7!} = \frac{x^5}{5!} \cdot \frac{x^2}{6 \cdot 7}$$

Exercise A.14

Trinn 4

Vi observerer at hvert ledd i rekken er ganske lik det foregående leddet. Mer presist:

$$a_0 = x$$

$$\frac{x^3}{3!} = x \cdot \frac{x^2}{2 \cdot 3}$$

$$\frac{x^5}{5!} = \frac{x^3}{3!} \cdot \frac{x^2}{4 \cdot 5}$$

$$\frac{x^7}{7!} = \frac{x^5}{5!} \cdot \frac{x^2}{6 \cdot 7}$$

Exercise A.14

Trinn 4

Vi observerer at hvert ledd i rekken er ganske lik det foregående leddet. Mer presist:

$$a_0 = x$$

$$\frac{x^3}{3!} = x \cdot \frac{x^2}{2 \cdot 3} \qquad a_1 = a_0 \cdot \frac{x^2}{2 \cdot 3}$$

$$\frac{x^5}{5!} = \frac{x^3}{3!} \cdot \frac{x^2}{4 \cdot 5}$$

$$\frac{x^7}{7!} = \frac{x^5}{5!} \cdot \frac{x^2}{6 \cdot 7}$$

Exercise A.14

Trinn 4

Vi observerer at hvert ledd i rekken er ganske lik det foregående leddet. Mer presist:

$$a_0 = x$$

$$\frac{x^3}{3!} = x \cdot \frac{x^2}{2 \cdot 3} \qquad a_1 = a_0 \cdot \frac{x^2}{2 \cdot 3}$$

$$\frac{x^5}{5!} = \frac{x^3}{3!} \cdot \frac{x^2}{4 \cdot 5} \qquad a_2 = a_1 \cdot \frac{x^2}{4 \cdot 5}$$

$$\frac{x^7}{7!} = \frac{x^5}{5!} \cdot \frac{x^2}{6 \cdot 7}$$

Exercise A.14

Trinn 4

Vi observerer at hvert ledd i rekken er ganske lik det foregående leddet. Mer presist:

$$a_0 = x$$

$$\frac{x^3}{3!} = x \cdot \frac{x^2}{2 \cdot 3} \qquad a_1 = a_0 \cdot \frac{x^2}{2 \cdot 3}$$

$$\frac{x^5}{5!} = \frac{x^3}{3!} \cdot \frac{x^2}{4 \cdot 5} \qquad a_2 = a_1 \cdot \frac{x^2}{4 \cdot 5}$$

$$\frac{x^7}{7!} = \frac{x^5}{5!} \cdot \frac{x^2}{6 \cdot 7} \qquad a_3 = a_2 \cdot \frac{x^2}{6 \cdot 7}$$

Exercise A.14

Trinn 5

Vi oppsummerer:

$$\sin(x) \approx a_0 + a_1 + \cdots + a_n$$

hvor

$$a_0 = x$$

og

$$a_j = -a_{j-1} \cdot \frac{x^2}{2j(2j+1)} \qquad j = 1, 2, 3, \ldots, n$$

Exercise A.14

Trinn 6

Vi setter navn på delsummene i Taylorrekken:

$$\sin(x) \approx \underbrace{a_0}_{s_1} + \underbrace{a_1}_{s_2} + \cdots + \underbrace{a_n}_{s_{n+1}}$$

Exercise A.14

Trinn 7

Vi setter navn på delsummene i Taylorrekken:

$$\sin(x) \approx \underbrace{a_0}_{s_1} + \underbrace{a_1}_{s_2} + \cdots + \underbrace{a_n}_{s_{n+1}}$$

og får differenslikningene (svar på oppgave a):

$$a_j = -a_{j-1} \cdot \frac{x^2}{2j(2j+1)} \qquad a_0 = x$$

$$s_{j+1} = s_j + a_j \qquad s_0 = 0$$

Exercise A.14

Trinn 7

Vi setter navn på delsummene i Taylorrekken:

$$\sin(x) \approx \underbrace{a_0}_{s_1} + \underbrace{a_1}_{s_2} + \cdots + \underbrace{a_n}_{s_{n+1}}$$

og får differenslikningene (svar på oppgave a):

$$a_j = -a_{j-1} \cdot \frac{x^2}{2j(2j+1)} \qquad a_0 = x$$

$$s_{j+1} = s_j + a_j \qquad s_0 = 0$$

b) Implement the system of difference equations in a function `sin_Taylor(x, n)`, which returns s_{n+1} and $|a_{n+1}|$. The latter is the first neglected term in the sum (since $s_{n+1} = \sum_{j=0}^{n} a_j$) and may act as a rough measure of the size of the error in the Taylor polynomial approximation.

c) Verify the implementation by computing the difference equations for $n = 2$ by hand (or in a separate program) and comparing with the output from the `sin_Taylor` function. Automate this comparison in a test function.

d) Make a table or plot of s_n for various x and n values to illustrate that the accuracy of a Taylor polynomial (around $x = 0$) improves as n increases and x decreases.

Exercise 5.14: Plot data in a two-column file

The file `src/plot/xy.dat`[12] contains two columns of numbers, corresponding to x and y coordinates on a curve. The start of the file looks as this:

```
-1.0000      -0.0000
-0.9933      -0.0087
-0.9867      -0.0179
-0.9800      -0.0274
-0.9733      -0.0374
```

Make a program that reads the first column into a list `x` and the second column into a list `y`. Plot the curve. Print out the mean y value as well as the maximum and minimum y values.

Hint Read the file line by line, split each line into words, convert to `float`, and append to `x` and `y`. The computations with `y` are simpler if the list is converted to an array.

Filename: `read_2columns`.

Tekstbehandling - kjapp repetisjon

```
s = "min fil.txt"

# Splitte i enkeltord:
s.split()   # ['min', 'fil.txt']
```

Tekstbehandling - kjapp repetisjon

```
s = "min fil.txt"

# Splitte i enkeltord:
s.split()   # ['min', 'fil.txt']

# Splitte i enkeltord med selvvalgt skilletegn:
s.split(".")    # ['min fil', 'txt']
s.split(" fil")  # ['min', '.txt']
```

Tekstbehandling - kjapp repetisjon

```
s = "min fil.txt"

# Splitte i enkeltord:
s.split()   # ['min', 'fil.txt']

# Splitte i enkeltord med selvvalgt skilletegn:
s.split(".")    # ['min fil', 'txt']
s.split(" fil")  # ['min', '.txt']

# Finne plassering av substreng:
s.index(".")   # 7
```

Tekstbehandling - kjapp repetisjon

```
s = "min fil.txt"

# Splitte i enkeltord:
s.split()   # ['min', 'fil.txt']

# Splitte i enkeltord med selvvalgt skilletegn:
s.split(".")    # ['min fil', 'txt']
s.split(" fil")  # ['min', '.txt']

# Finne plassering av substreng:
s.index(".")   # 7

# Sjekke om substreng er tilstede:
"fil" in s      # True
"FIL" in s      # False
```

Tekstbehandling - kjapp repetisjon

```
s = "min fil.txt"

# Splitte i enkeltord:
s.split()   # ['min', 'fil.txt']

# Splitte i enkeltord med selvvalgt skilletegn:
s.split(".")    # ['min fil', 'txt']
s.split(" fil")  # ['min', '.txt']

# Finne plassering av substreng:
s.index(".")   # 7

# Sjekke om substreng er tilstede:
"fil" in s      # True
"FIL" in s      # False

# Plukke ut ett tegn:
s[0]    # 'm'
s[1]    # 'i'
s[2]    # 'n'
s[3]    # ' '
```

Plukke ut substreng

```
s = "Dette er en tekststreng"

# Alt unntatt første tegn
s[1:]  # "ette er en tekststreng"
```

Plukke ut substreng

```
s = "Dette er en tekststreng"

# Alt unntatt første tegn
s[1:]  # "ette er en tekststreng"

# Alt unntatt første og siste tegn
s[1:-1]  # "ette er en tekststren"
```

Plukke ut substreng

```
s = "Dette er en tekststreng"

# Alt unntatt første tegn
s[1:]  # "ette er en tekststreng"

# Alt unntatt første og siste tegn
s[1:-1]  # "ette er en tekststren"

# Tegnene med indeks 2,3,4
s[2:5] # "tte"
```

Plukke ut substreng

```
s = "Dette er en tekststreng"

# Alt unntatt første tegn
s[1:]  # "ette er en tekststreng"

# Alt unntatt første og siste tegn
s[1:-1]  # "ette er en tekststren"

# Tegnene med indeks 2,3,4
s[2:5] # "tte"

# Alt fra og med en substreng
s[s.index("tekst"):]  # "tekststreng"
```

Plukke ut substreng

```
s = "Dette er en tekststreng"

# Alt unntatt første tegn
s[1:]  # "ette er en tekststreng"

# Alt unntatt første og siste tegn
s[1:-1]  # "ette er en tekststren"

# Tegnene med indeks 2,3,4
s[2:5] # "tte"

# Alt fra og med en substreng
s[s.index("tekst"):]  # "tekststreng"

# Fjern blanke foran og bak
s = "     A B C     "
s.strip()   # "A B C"
s.lstrip()  # "A B C     "
s.rstrip()  # "     A B C"
```

Slå sammen tekststrenger

```
a = ["I", "am", "happy"]


# Slå sammen listeelementer
s = "".join(a)    # "Iamhappy"
```

Slå sammen tekststrenger

```
a = ["I", "am", "happy"]


# Slå sammen listeelementer
s = "".join(a)    # "Iamhappy"

# Slå sammen listeelementer med blanke i mellom
s = " ".join(a)   # "I am happy"
```

Slå sammen tekststrenger

```
a = ["I", "am", "happy"]


# Slå sammen listeelementer
s = "".join(a)    # "Iamhappy"

# Slå sammen listeelementer med blanke i mellom
s = " ".join(a)   # "I am happy"

# Slå sammen listeelementer med "--" i mellom
s = "--".join(a)  # "I--am--happy"
```

Bytt ut substreng

```
s = "Dette er en tekststreng"

# Erstatt alle blanke med "X"
t = s.replace(" ", "X")
   # 'DetteXerXenXtekststreng'
```

Bytt ut substreng

```
s = "Dette er en tekststreng"

# Erstatt alle blanke med "X"
t = s.replace(" ", "X")
   # 'DetteXerXenXtekststreng'

# Erstatt en substreng med en annen
t = s.replace("Dette", "Her")
   # 'Her er en tekststreng'
```

Bytt ut substreng

```
s = "Dette er en tekststreng"

# Erstatt alle blanke med "X"
t = s.replace(" ", "X")
   # 'DetteXerXenXtekststreng'

# Erstatt en substreng med en annen
t = s.replace("Dette", "Her")
   # 'Her er en tekststreng'

# Erstatt alt foran "tekst" med noe annet
t = s.replace(s[:s.index("tekst")], "Ny ")
   # 'Ny tekststreng'
```

Bytt ut substreng

```
s = "Dette er en tekststreng"

# Erstatt alle blanke med "X"
t = s.replace(" ", "X")
   # 'DetteXerXenXtekststreng'

# Erstatt en substreng med en annen
t = s.replace("Dette", "Her")
   # 'Her er en tekststreng'

# Erstatt alt foran "tekst" med noe annet
t = s.replace(s[:s.index("tekst")], "Ny ")
   # 'Ny tekststreng'

# Erstatt alt fra og med "tekst" med noe annet
t = s.replace(s[s.index("tekst"):], "setning")
   # 'Dette er en setning'
```

Ny linje i tekststrenger

Her er det forskjeller mellom operativsystemer!

```
# Unix/Linux/Mac:
s = "\n".join(["Line A", "Line B", "Line C"])
s.split("\n")

# Windows:
s = "\r\n".join(["Line A", "Line B", "Line C"])
s.split("\r\n")

# Alle operativsystemer:
s.splitlines()
```

Noen flere tekstfunksjoner

```
# Test om en string bare består av sifre
s = "314"
s.isdigit()    # True
s = "  314"
s.isdigit()    # False
s = "3.14"
s.isdigit()    # False
```

Noen flere tekstfunksjoner

```
# Test om en string bare består av sifre
s = "314"
s.isdigit()    # True
s = "  314"
s.isdigit()    # False
s = "3.14"
s.isdigit()    # False

# Endre alt til små eller til store bokstaver
s = "ABC def"
s.lower()  # "abc def"
s.upper()  # "ABC DEF"
```

Noen flere tekstfunksjoner

```
# Test om en string bare består av sifre
s = "314"
s.isdigit()    # True
s = "  314"
s.isdigit()    # False
s = "3.14"
s.isdigit()    # False

# Endre alt til små eller til store bokstaver
s = "ABC def"
s.lower()  # "abc def"
s.upper()  # "ABC DEF"

# Test om en string starter/slutter med en gitt string
s = "Dette er en string"
s.startswith("Dette er")    # True
s.endswith("Dette er")     # False
```

Eksempel: Ekstrahere tall i en tekstfil

Anta at vi ønsker å lese en fil med følgende format:

```
(1.3,0)    (-1,2)     (3,-1.5)
(0,1)      (1,0)      (1,1)
(0,-0.01)  (10.5,-1)  (2.5,-2.5)
```

Algoritme:

1. Les én linje av gangen
2. For hver linje: splitt opp i ord
3. For hvert ord: fjern parenteser og splitt på komma

Implementasjon

```
pairs = []

with open("pairs.dat", "r") as infile:
    for line in infile:
        words = line.split()
        for w in words:
            w = w[1:-1]   # Fjern parentesene rundt
            numbers = w.split(",")
            pair = (float(numbers[0]), float(numbers[1]))
            pairs.append(pair)
```

Resultat

```
pairs:

 [(1.3, 0.0),
  (-1.0, 2.0),
  (3.0, -1.5),
  (0.0, 1.0),
  (1.0, 0.0),
  (1.0, 1.0),
  (0.0, -0.01),
  (10.5, -1.0),
  (2.5, -2.5)]
```

Eksempel: Finne vanligst forekommende ord

Vi ønsker å lage et program som finner de vanligst forekommende ordene i en tekstfil.

Innledende strategi:

- Lage en tom liste `words` for å holde alle ordene
- Lese filen linjevis med `readline()` og finne enkeltordene med `split()`
- Legge ordene fortløpende inn i `words`

Eksempel: Finne vanligst forekommende ord

Problem: Hvordan unngå at spesialtegn og store/små bokstaver skaper problemer? Anta vi har tekstfilen:

```
Sola skinner. Det er veldig varmt i
sola i dag. Hvis det ikke hadde vært
for sola, ville det ha vært veldig
kaldt.
```

Her vil `split()` fange opp blant annet ordene

```
Sola
sola
sola,
```

og vi ønsker å telle disse som tre forekomster av samme ord.

Løsningsstrategi:

1) Trekke ut bokstavene i teksten med `isalpha()`
2) Konvertere til små bokstaver med `lower()`.

Eksempel: Finne vanligst forekommende ord

Hvordan teller vi opp antall forekomster av hvert ord?

Løsningsstrategi A:

- Innføre en ny liste `counts` som er like lang som `words` og som inneholder antall forekomster vi har sett til nå av hvert ord

Løsningsstrategi B:

- Erstatte listen `words` med en dictionary `wordcounts`
- Første gang vi ser et ord, legger vi det inn som en ny nøkkel i dictionaryen `wordcounts` med tilhørende verdi satt til 1
- Hvis samme ord dukker opp senere i teksten, søker vi det opp i dictionaryen og øker tilhørende verdi med 1.

Eksempel: Finne vanligst forekommende ord

```
filename = input("File name? ")
file = open(filename)

# Read words and add to dictionary with count
wordcounts = {}
for line in file:
   words = line.split()
   for word in words:
       w = "".join([e for e in word if e.isalpha()])
       w = w.lower()
       if w in wordcounts:
           wordcounts[w] = wordcounts[w]+1
        else:
           wordcounts[w] = 1

# Put all words in list and sort on counts
words = list(wordcounts.keys())
words.sort(key=lambda x:wordcounts[x], reverse=True)

# Print 30 most frequent words with counts
for i in range(0,min(len(words),30)):
print(f"{i+1:3d} {words[i]:10s} {wordcounts[words[i]]}")
```

Oppgave 1

Hva skrives ut?

```
a = []
for k in range(2,5):
    a = a + [k,k+1]
print(a)
```

Oppgave 1

Hva skrives ut?

```
a = []
for k in range(2,5):
    a = a + [k,k+1]
print(a)
```

[2, 3, 3, 4, 4, 5]

Oppgave 2

Hva skrives ut?

```
a = [8, 9, 10, 11]
print(a[1:1])
print(a[1:-1])
print(a[-1:1:1])
```

Oppgave 2

Hva skrives ut?

```
a = [8, 9, 10, 11]
print(a[1:1])
print(a[1:-1])
print(a[-1:1:1])
```

[]
[9, 10]
[]

Oppgave 3

Hva skrives ut?

```
for i in range(2,5):
    print(i, end=" ")
    for j in range(i-1, i+1):
        print(j, end=" ")
```

Oppgave 3

Hva skrives ut?

```
for i in range(2,5):
    print(i, end=" ")
    for j in range(i-1, i+1):
        print(j, end=" ")
```

2 1 2 3 2 3 4 3 4

Oppgave 4

Hva skrives ut?

```
def f(x,y):
    return x-y

def g(x):
    return abs(x) - x

x = 4
y = 3
print(g(f(y,x)))
```

Oppgave 4

Hva skrives ut?

```
def f(x,y):
    return x-y

def g(x):
    return abs(x) - x

x = 4
y = 3
print(g(f(y,x)))
```

2

Oppgave 5

Hva skrives ut?

```
x = [1, 3, 6, 9]
m = list(range(1,len(x)))
y = [x[k]-x[k-1] for k in m]
print(y[1])
```

Oppgave 6

Hva skrives ut?

```
def cubes(n):
    return sum([k**3 for k in range(n+1)])

def test_cubes():
    expected = sum([0, 1, 8, 27])
    computed = cubes(4)
    tol = 1e-12
    success = abs(computed-expected) < tol
    assert success, "Testen feilet"

test_cubes()
```

Oppgave 7

Vi har en liste `a` med tall og programmet

```
s = 0
for i in range(1,len(a)):
   if a[i] < a[i-1]:
      s += 1
```

Fyll inn det som mangler i dette programmet for at det skal gi samme resultat, dvs gi samme verdi til `s`:

```
# Program A
s = 0
i = 1
while a[i] < a[i-1]:
   <fyll inn kode her>
```

Oppgave 7

Vi har en liste `a` med tall og programmet

```
s = 0
for i in range(1,len(a)):
   if a[i] < a[i-1]:
      s += 1
```

Fyll inn det som mangler i dette programmet for at det skal gi samme resultat, dvs gi samme verdi til `s`:

```
# Program B
s = 0
i = 0
while i < len(a)-1:
   <fyll inn kode her>
```