Appendix

Appendix: Useful Functions and Methods

Built-in:

format(numeric_value, format_specifier)

- Formats a numeric value into a string according to the format specifier, which is a string that contains special characters specifying how the numeric value should be formatted. Examples of various formatting characters are “f=floating-point, e=scientific notation, %=percentage, d=integer”. A number before the formatting character will specify the field width. A number after the character “.” will format the number of decimals.

%

- Remainder (modulo operator): Divides one number by another and gives the remainder.

len(s)

- Return the length (the number of items) of a string, tuple, list, dictionary or other data structure.

int(x)

- Convert a string or number to a plain integer.

float(x)

- Convert a string or a number to floating point number.

str([object])

- Return a string containing a nicely printable representation of an object.

String methods:

s.isalnum()

- Returns true if the string contains only alphabetic letters or digits and is at least one character of length. Returns false otherwise.

s.isalpha()

- Returns true if the string contains only alphabetic letters, and is at least one character in length. Returns false otherwise.

s.isdigit()

- Returns true if the string contains only numeric digits and is at least one character in length. Returns falseotherwise.

s.isspace()

- Returns true if the string contains only whitespace characters, and is at least one character in length. Returns falseotherwise. (Whitespace characters are spaces, newlines (\n), and tabs (\t)).

s.ljust(width)

- Return the string left justified in a string of length width.

s.rjust(width)

- Return the string right justified in a string of length width.

s.lower()

- Returns a copy of the string with all alphabetic letters converted to lowercase.

s.upper()

- Returns a copy of the string with all alphabetic letters converted to uppercase.

s.strip()

- Returns a copy of the string with all leading and trailing white space characters removed.

s.strip(char)

- Returns a copy of the string with all instances of char that appear at the beginning and the end of the string removed.

s.split(str)

- Returns a list of all the words in the string, using str as the separator (splits on all whitespace if left unspecified).

s.endswith(substring)

- The substring argument is a string. The method returns true if the string ends with substring.

s.startswith(substring)

- The substring argument is a string. The method returns true if the string starts with substring.

s.find(substring)

- The substring argument is a string. The method returns the lowest index in the string where substring is found. If substring is not found the method returns -1.

s.replace(old, new)

- The old and new arguments are both strings. The method returns a copy of the string with all instances of old replaced by new.

List operations:

s[i:j:k]

- Return slice starting at position i extending to position j in k steps. Can also be used for strings.

item in s

- Determine whether a specified item is contained in a list.

min(list)

- Returns the item that has the lowest value in the sequence.

max(list)

- Returns the item that has the highest value in the sequence.

s.append(x)

- Append new element x to end of s.

s.insert(index,item)

- Insert an item into a list at a specified position given by an index.

s.index(item)

- Return the index of the first element in the list containing the specified item.

s.pop()

- Return last element and remove it from the list.

s.pop(i)

- Return element i and remove it from the list.

s.remove(item)

- Removes the first element containing the item.

s.reverse()

- Reverses the order of the items in a list.

s.sort()

- Rearranges the elements of a list so they appear in ascending order.

Dictionary operations:

d.clear()

- Clears the contents of a dictionary

d.get(key, default)

- Gets the value associated with a specific key. If the key is not found, the method does not raise an exception. Instead, it returns a default value.

d.items()

- Returns all the keys in a dictionary and their associated values as a sequence of tuples.

d.keys()

- Returns all the keys in a dictionary as a sequence of tuples.

d.pop(key, default)

- Returns the value associated with a specific key and removes that key-value pair from the dictionary. If the key is not found, the method returns a default value.

d.popitem()

- Returns a randomly selected key-value pair as a tuple from the dictionary and removes that key-value pair from the dictionary.

d.values()

- Returns all the values in dictionary as a sequence of tuples.

Files:

open()

- Returns a file object, and is most commonly used with two arguments: open(filename, mode). Mode can be ‘r’ (read only), ‘w’ (writing only), ‘a’ (appending), ‘r+’ (both reading and writing).

f.read(size)

- Reads data from file and returns it as a string. Size is an optional and if left out the whole file will be read.

f.readline()

- Reads a single line from the file (reads until newline character (\n) is found), and returns it as a string.

f.readlines()

- Reads data from the file and returns it as a list of strings.

f.write(string)

- Writes the contents of string to file.

f.close()

- Close the file and free up any system resources taken up by the open file.

Plots:

import matplotlib.pyplot as plt

- Imports the plotting library

plt.plot(x,y,'CM')

- Plots the points (x[i],y[i]) in order with a style specified by the string 'CM', where C denotes colour and M marker style. Example: plt.plot(x,y,'g^') plots the points as green triangles.
- Some colours:
  - 'b': blue
  - 'g': green

- 'r': red
- 'y': yellow
- 'k': black
- 'm': magenta

- Some styles:
  - '.': points
  - 'o': circles
  - '-': lines
  - '^': triangles
  - 'x': crosses

plt.xlabel(string)

- Labels the x axis of the plot with the text in the string.

plt.ylabel(string)

- Labels the y axis of the plot with the text in the string.

plt.title(string)

- Sets a title text for the plot specified by the string.

plt.legend(listOfStrings)

- Creates a legend in the figure with titles for the data sets given by the list of strings.

plt.show()

- Shows the plot on-screen.

1 Oppgave 1: Flervalg - 15%

Stoppekriterier behøves i numeriske algoritmer for å...
Velg ett alternativ

- vite når en integrasjonsalgoritme som f.eks. trapesmetoden er ferdig.
- unngå å måtte sjekke likhet mellom flyttall.
- stoppe en direkte ligningsløser som f.eks. Gauss-eliminasjon i tide.
- unngå at en iterativ ligningsløser som f.eks. Newtons metode fortsetter i det uendelige.

Hva er sant om tilnærming av bestemte integraler?
Velg ett alternativ

- Jo færre intervaller man bruker i en kompositttmetode, jo mer nøyaktig resultat får man.
- Feilestimat for algoritmene kan brukes til å gi tilnærmede svar med garanterte øvre grenser for feil.
- Midtpunktmetoden baserer seg på å tilnærme funksjonen som skal integreres med kubiske polynomer.
- Implisitt Euler er ypperlig egnet til å beregne integraler.

Hva er sant om flyttall (floating point numbers)?

Velg ett alternativ

- Flyttall er spesialkonstruerte tall som beskriver fluidstrømninger.
- Ethvert reelt tall kan representeres nøyaktig som et dobbelpresisjons flyttall (double).
- Man bør unngå bruk av flyttall i numeriske beregninger på grunn av problemer med presisjon.
- Et flyttall er representert ved et fortegn, en eksponent og en mantisse.

Anta at du skal løse en ligning i én variabel, og at funksjonen din og startpunktet ditt oppfyller alle konvergensbetingelser for biseksjons- og sekantmetodene samt Newtons metode. Rangér metodene etter konvergensfart. Her betyr a > b > c at a er raskere enn b som igjen er raskere enn c.

Velg ett alternativ

- Newton > biseksjon > sekant
- Newton > sekant > biseksjon
- biseksjon > sekant > Newton
- sekant > biseksjon > Newton

Hva er en Jacobi-matrise?

Velg ett alternativ

- En matrise som er på radredusert form.
- En matrise som består av de andreordens partiellderiverte til en funksjon med flere komponenter.
- En matrise der alle oppføringene er 0.
- En matrise som består av de førsteordens partiellderiverte til en funksjon med flere komponenter.

Hva er sant om forskjellene mellom eksplisitt og implisitt Euler?

Velg ett alternativ

- Implisitt Euler krever at man løser en ligning i hvert tidssteg, det gjør ikke eksplisitt Euler.
- Implisitt Euler er den sensurerte versjonen av eksplisitt Euler.
- Eksplisitt Euler krever at man løser en ligning i hvert tidssteg, det gjør ikke implisitt Euler.
- Eksplisitt Euler brukes til å finne nullpunkt til funksjoner, implisitt Euler brukes til å løse ordinære differensialligninger.

Maks poeng: 6

2 Oppgave 2: Kodeforståelse - 8%

a)

Hva blir variabelen *answer* satt til etter at Kodesnutt 1 er utført?

Forklar med en setning hva funksjonen gjør.

Kodesnutt 1:

```
def secret(m):
    s = ''
    for i in range(len(m)):
        if len(m[i])>2:
            s += str(m[i][2])
        else:
            s += str(m[i][0])
    return s

m = [[4,5],[7,8,2,5]]
answer = secret(m)
```

Skriv ditt svar her

---

b)

Hva blir skrevet ut til skjerm når koden i Kodesnutt 2 blir kjørt?

Forklar med en setning hva funksjonen gjør.

Kodesnutt 2:

```
def mystery(y):
    s = str(y)
    r = 0
    while len(s) and not y<0:
        s = s[:len(s)-1]
        x = (10**len(s))
        r = y//x
        y -= r*x
        print(r,x)

mystery(145)
```

Skriv ditt svar her

---

c)

Hva blir variabelen *answer* satt til etter at Kodesnutt 3 er utført?

Forklar med en setning hva funksjonen gjør.

Kodesnutt 3:

```
def doing_something(lst,ch):
    new_lst = []
    for el in lst:
        if ch in el:
            new_lst.append(el.lower())
        elif ch.upper() in el.upper():
            new_lst.append(el.upper())
    s = ''
    for el in new_lst:
        s += el[0]
    return s

turtle_colors = ['PeAcH pUfF','CorNfloWer bLuE','azUre','NaVy','DarK OrcHid','iNDiGo','Alice Blue']
answer = doing_something(turtle_colors,'a')
```

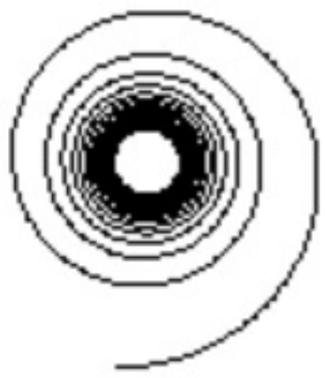
Skriv ditt svar her

---

d)

Hvilken av kodesnuttene under (Kodesnutt 4-6) vil gi figuren i Bilde 1?

Bilde 1:



Kodesnutt 4:

```
from turtle import *
hideturtle()
def draw():
    for i in range(16):
        forward(60-i*4)
        left(90)

draw()
```

Kodesnutt 5:

```
from turtle import *
hideturtle()
def draw():
    for i in range(2000):
        forward(1)
        left(1+(i/360))

draw()
```

Kodesnutt 6:

```
from turtle import *
hideturtle()
def draw():
    i = 0
    while True:
        forward(3)
        left(0+i)
        i += 1

draw()
```

hideturtle() skjuler pila fra tegningen

Skriv tekst her

Maks poeng: 10

3 Oppgave 3: Strenger - 5%

Denne oppgaven omhandler tautonymer.

> *A tautonym is a scientific name of a species in which both parts of the name have the same spelling, for example Gulo gulo.*

I oppgave b) kan det være lurt å bruke funksjonen fra oppgave a). Denne kan du bruke selv om du ikke har gjort oppgave a).

Oppgave a)

I denne deloppgaven skal du lage en funksjon *isTautonym* som tar inn en streng som argument, og sjekker om dette er et tautonym. Funksjonen skal returnere *True* om strengen er et tautonym, og *False* ellers. Ta hensyn til at strengen ikke nødvendigvis består av to ord.

Eksempler:

Ved kjøring av koden under:

```
print(isTautonym("Phocoena"))
print(isTautonym("Caracal caracal"))
print(isTautonym("Spermophilus citellus"))
print(isTautonym("Bubo bubo bubo"))
print(isTautonym("Lynx lynx isabellinus"))
```

skal dette skrives ut til skjerm:

```
False
True
False
True
False
```

Skriv ditt svar her

```
1
```

Maks poeng: 10

4 Oppgave 3b) - 5%

Du skal nå lage en funksjon som tar inn en liste *lst* med strenger, og returnerer en ny liste *new_lst* med strengene som er tautonymer. Om et tautonym forekommer flere ganger i *lst*, skal det ikke legges inn mer enn én gang i *new_lst*.

Eksempel:

Ved kjøring av koden under:

```
mammals = ['Axis axis', 'Prionailurus bengalensis iriomotensis', 'Axis axis', 'Genetta genetta',
'Neofelis nebulosa', 'Hyaena hyaena barbara', 'Axis axis', 'Nasua nasua']
print(tautonyms(mammals))
```

skal dette skrives ut til skjerm:

```
['Axis axis', 'Genetta genetta', 'Nasua nasua']
```

Skriv ditt svar her

```
1
```

Maks poeng: 10

5 Oppgave 4a) - 8%

Birgitte, Sebastian og Neva har mange ligninger å løse og vil bruke numeriske metoder for å automatisere løsningsprosessen. De har bestemt seg for å implementere en algoritme hver slik at de kan kombinere dem til et stort løsningsprogram til slutt.

Først ut er Birgitte som har implementert biseksjonsmetoden i funksjonen **bisection_method(f,a,b,epsilon)**. Den skal ta inn en funksjon, to tall som utgjør startintervallet [a,b] og en toleranse epsilon slik at iterasjonene stopper når intervallet har lengde mindre enn epsilon. Når iterasjonene stopper, returnerer funksjonen midtpunktet i det siste intervallet.

Dessverre mister Birgitte PCen sin i bakken når hun jobber med koden, og kodelinjene flyr overalt. Kan du hjelpe henne med å sette kodelinjene i riktig rekkefølge igjen?

Hint: Husk at i hver iterasjon skal a byttes ut med c = (a+b)/2 dersom f(a) har samme fortegn som f(c). Ellers skal b byttes ut med c.

Oppgave: *Dra og slipp linjene så de havner i riktig rekkefølge og med riktig innrykk, slik at funksjonen vil virke som den skal. Alle kodelinjene skal brukes.*

NB! For å velge riktig innrykk må du passe på at midten av kodelinjen er over riktig boks når den slippes. Kodelinjen vil da automatisk settes på plass.

```
c = (b+a)/2
if f(a)*f(c) > 0:
    return c
    b = c
def bisection_method(f,a,b,epsilon):
    a = c
while abs(b-a) > epsilon:
else:
```

Maks poeng: 8

6 Oppgave 4b) - 6%

Nestemann er Sebastian. Hans yndlingsalgoritme er sekantmetoden, og han går i gang med å implementere den. Plutselig blir han usikker på om han egentlig husker hvordan det er med stoppebetingelser og oppdateringer og sånt. Kan du hjelpe Sebastian med å gjøre riktige valg slik at koden kjører som den skal?

Hint: Sekantmetoden er gitt ved formelen

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$$

Oppgave: *Velg riktige alternativer fra menyene slik at* **secant_method(f, x_0, x_1, delta, epsilon)** *implementerer sekantmetoden anvendt på funksjonen f med de to startpunktene x_0 og x_1, slik at den stopper når* $|x_k - x_{k-1}| <$ *delta eller* $|f(x_k)| <$ *epsilon.*

```
def secant_method(f,x_0,x_1,delta,epsilon):
    x_old = x_0
    x_k = x_1
    while [ ] ((abs(x_k - x_old) > delta) and (abs(f(x_k)) < epsilon), (abs(x_k - x_old) < delta) and (abs(f(x_k)) > epsilon), (abs(x_k - x_old) > delta) and (abs(f(x_k)) > epsilon), (abs(x_k - x_old) < delta) and (abs(f(x_k)) < epsilon)):
        x_new = [ ] (x_old + f(x_old)*(x_k - x_old), x_k - f(x_k) * (x_k - x_old) / (f(x_k) - f(x_old)), x_k - f(x_k)/f(x_old), x_k - f(x_k))
        [ ] (x_old = x_k; x_k = x_new, x_k = x_old; x_k = x_new, x_old = x_new; x_j = x_old, x_old = x_k; x_k = x_old)
    return x_new
```

Maks poeng: 3

7 Oppgave 4c) - 6%

Neva vil implementere Newtons metode. Etter å ha skrevet koden ønsker hun å teste den for å se den navngjetne farten den skal ha. Det resulterer i feil svar. Hun spør Birgitte om hjelp og Birgitte ser

umiddelbart tre feil i koden. Kan du hjelpe Neva å finne feilene?

Den matematiske funksjonen hun ønsker å teste metoden på er

$$f(x) = 3 + x^2 + 2x^3$$

Hint: Newtons metode er gitt ved

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Oppgave: *Finn tre feil i koden nedenfor og beskriv dem i tekstfeltet under. Funksjonen* **Newton_method(f,df,x,epsilon)** *skal ta inn en funksjon f og dens deriverte df, et startpunkt x og en toleranse epsilon. Den skal stoppe når |f(x)| < epsilon eller når det har gått 100 iterasjoner og returnere et tilnærmet nullpunkt for f.*

```
def f(x):
    return 3 + x**2 + 2*x*3

def f_derivative(x):
    return 2*x + 6*x**2

def Newton_method(f, df, x, epsilon):
    N = 0
    while (abs(f(x)) > epsilon) and (N < 100):
        x = x - f(x)*df(x)
        N = N + 1
    return x

def main():
    x = 2
    epsilon = 1E-14
    y = Newton_method(f, f, x, epsilon)
    print(y)
    print(f(y))

main()
```

Skriv ditt svar her...

Maks poeng: 10

8 Oppgave 4d) - 6%

Etter å ha fått til implementasjonen av Newtons metode vil Neva gjøre en test av hvordan den fungerer.

Neva har laget en testkode som skal velge 100 tilfeldige startverdier mellom 2 og 10 og kjøre Newtons metode som implementert i forrige deloppgave med disse startverdiene. For hver startverdi skal det sjekkes om Newtons metode returnerer et punkt y som tilfredstiller stoppebetingelsen |f(y)| < epsilon. Hvis startverdien fører til et svar som ikke tilfredsstiller stoppebetingelsen skal det føres opp i en liste som inneholder dårlige startverdier.

Oppgave: *Fullfør koden nedenfor ved å velge riktige kodelinjer i nedtrekksmenyene slik at koden fungerer som beskrevet ovenfor. Du kan anta at funksjonene* **Newton_method**, **f** *og* **f_derivative** *er implementert fra før.*

```
import random
```

```
bad_starting_values = []
epsilon = 1E-10
for j in range(100):
    x_0 =  (random.uniform(2,10), uniform(0,1), uniform(2,10),
random.uniform(0,1))
    y = Newton_method(f,f_derivative,x_0,epsilon)
    if  (f(y) > epsilon, f(y) < epsilon, abs(f(y)) > epsilon, abs(f(y)) < epsilon):
         (bad_starting_values = x_0,
bad_starting_values = y, bad_starting_values.append(y),
bad_starting_values.append(x_0))
print(bad_starting_values)
```

Maks poeng: 3

9

Oppgave 4e) - 10%

Med alle tre algoritmene på plass er det på tide å samle dem i én stor løsningsalgoritme. Funksjonen **equation_solver(f,df)** skal ta inn en funksjon og dens deriverte (om den deriverte ikke er tilgjengelig, send inn 0). Deretter skal programmet spørre brukeren om hvilken metode som skal brukes.

1. Hvis svaret er "Newton" skal brukeren spørres om x_0 og epsilon, deretter skal **Newton_method(f,df,x_0,epsilon)** brukes.

2. Hvis svaret er "sekant" skal brukeren spørres om x_0, x_1, delta og epsilon, deretter skal **secant_method(f, x_0, x_1, delta, epsilon)** brukes.

3. Hvis svaret er "biseksjon" skal brukeren spørres om a, b og epsilon. Deretter skal det sjekkes om startintervallet er gyldig; altså om f(a)*f(b) < 0. Dersom dette holder skal **bisection_method(f,a,b,epsilon)** brukes. Hvis ikke, skal programmet skrive "Ugyldig startintervall!" til skjermen og avslutte.

Hvis brukeren ikke skriver noen av disse alternativene skal funksjonen skrive til skjerm "Denne metoden er ikke implementert!" og avslutte. Ellers skal funksjonen returnere det numeriske svaret gitt av den valgte metoden.

Oppgave: *I tekstfeltet under, skriv en funksjon* **equation_solver(f,df)** *som oppfyller spesifikasjonene gitt ovenfor.*

Skriv ditt svar her...

```
1
```

Maks poeng: 10

10 Oppgave 4f) - 8%

Brynjar vil bruke de andres funksjon til å løse en ligning, og i likhet med Neva liker han Newtons metode best. Han skulle gjerne likt å ha informasjon om hvordan iterasjonene utvikler seg og vil derfor foreta et par endringer i Nevas implementasjon.

Brynjar ønsker å endre koden til **Newton_method** slik at iterasjonene og funksjonsverdiene blir lagret i hver sin liste i stedet for å overskrives og at funksjonen skal returnere disse to listene. I tillegg ønsker han seg mer kontroll over maks antall iterasjoner, så den nye funksjonen, **Newton_method_list,** skal ta inn et tall N_max som sier hvor mange iterasjoner som maks kan tas.

Oppgave: *Implementér funksjonen* **Newton_method_list.** *Du skal ta utgangspunkt i koden under og fylle ut de fem feltene merket med "...". Funksjonen skal kunne variere maks antall iterasjoner basert på input og returnere to lister - en med de forskjellige verdiene til x (inkludert startverdien) og en med de tilhørende funksjonsverdiene.*

```
def Newton_method_list(f,df,x,epsilon,N_max):
    x_list = [x]
    ...
    N = 0
    while (abs(f(x)) > epsilon) and (...):
        x = x - f(x)/df(x)
        ...
        f_list.append(...)
        N += 1
    return ...
```

Skriv ditt svar her...

```
1
```

Maks poeng: 10

11 Oppgave 4g) - 8%

Etter å ha gjort om på Newton-funksjonen ønsker Brynjar å lage automatisk informasjon om hvordan

funksjonen har kjørt.

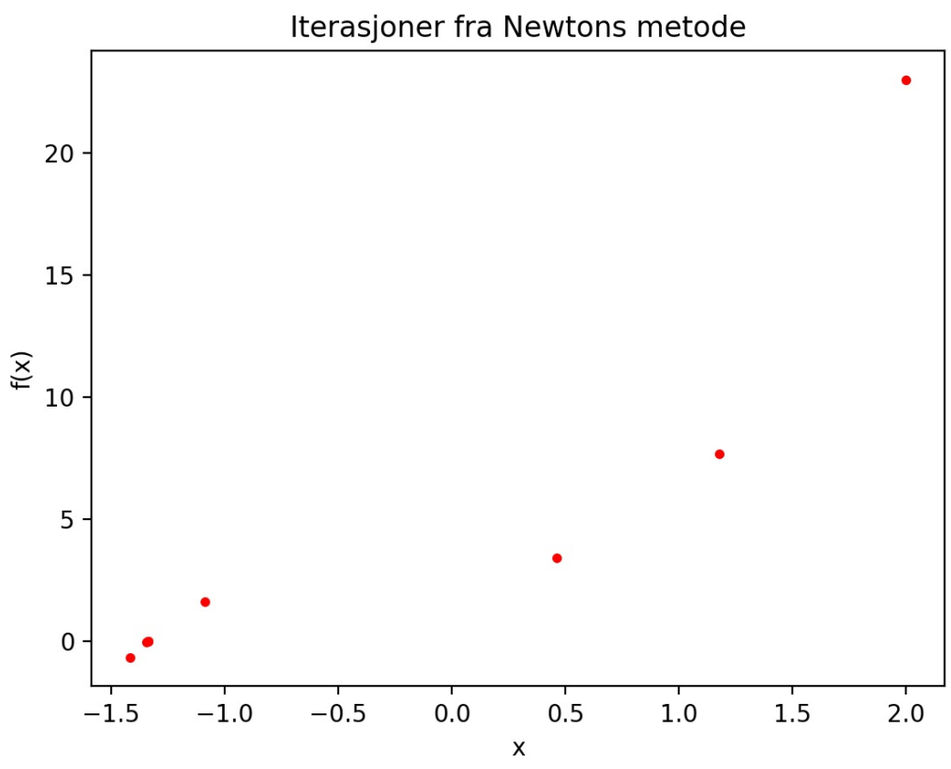
Brynjar vil lage en funksjon som skriver til skjerm hvor mange iterasjoner som er gjort i Newtons metode og hvilken funksjonsverdi som ble oppnådd i siste iterasjon ved bruk av **Newton_method_list**. Han vil også at alle iterasjonene skal plottes som punkter i en figur slik at han får visualisert hvordan funksjonen oppfører seg.

Oppgave: *Implementér en funksjon* iteration_info(x_list,f_list) *som tar inn en liste med punkter beregnet med Newtons metode og en liste med funksjonsverdiene i disse punktene. Funksjonen skal skrive ut informasjon om iterasjonene på formatet:*

"Newtons metode brukte (...) iterasjoner og endte til slutt opp med løsningen x = (...) med funksjonsverdi f(x) = (...)."

Altså skal (...) her fylles inn automatisk med informasjon fra listene. Husk at den første oppføringen i x_list er startverdien som ikke telles som en iterasjon.

I tillegg skal alle punktene og tilhørende funksjonsverdier plottes som røde punkter (x_k,f(x_k)) slik som i figuren under. Sett også tekst på x- og y-akser samt en tittel på plottet. Du finner informasjon om plotting i appendix før oppgave 1.



Skriv ditt svar her...

```
1
```

Maks poeng: 10

12 # Oppgave 5a) - 3%

I løpet av oppgave 5 kan det være nyttig å bruke funksjoner som er laget i tidligere deloppgaver. Dette kan du gjøre selv om du ikke har løst den aktuelle deloppgaven.

Oppgave a)

Lag en funksjon *num_el* som tar inn to argumenter *el* og *lst*, og finner og returnerer antall forekomster av elementet *el* i listen *lst*. Funksjonen skal IKKE endre på listen *lst*.

Eksempel:

Ved kjøring av koden under:

```
letters = ['a', 'b', 'b', 'c', 'a', 'b', 'd', 'e', 'b']
print("Det er", num_el('b', letters), "b'er i", letters)
```

skal dette skrives ut til skjerm:

```
Det er 4 b'er i ['a', 'b', 'b', 'c', 'a', 'b', 'd', 'e', 'b']
```

Skriv ditt svar her

```
1
```

Maks poeng: 10

13 # Oppgave 5b) - 5%

Lag en funksjon *remove_duplicates* som tar inn en liste *lst*, fjerner alle duplikater i listen, og returnerer samme liste uten duplikater.

Eksempel:

Ved kjøring av koden under:

```
letters = ['a', 'b', 'b', 'c', 'a', 'b', 'd', 'e', 'b']
remove_duplicates(letters)
print(letters)
```

skal, for eksempel, følgende skrives ut til skjerm:

```
['c', 'a', 'd', 'e', 'b']
```

Skriv ditt svar her

```
1
```

Maks poeng: 10

14 Oppgave 5c) - 7%

Lag en funksjon *sorted_list* som tar inn en streng og returnerer en liste med ordene i strengen i alfabetisk rekkefølge. Ta hensyn til at ordene i strengen både kan ha små og store bokstaver (dvs. at *sorted(lst)* ikke vil fungere).

Eksempel:

Ved kjøring av koden under:

```
words = "Dette Er et FLOTT eksempel Med små og STORE bokstaver."
print(sorted_list(words))
```

skal dette skrives ut til skjerm:

```
['bokstaver.', 'Dette', 'eksempel', 'Er', 'et', 'FLOTT', 'Med', 'og', 'små', 'STORE']
```

Auditorieøving 2 - Numerikk

Skriv ditt svar her

```
1
```

Maks poeng: 10