IN-KJM1900 — Forelesning 2

Simen Kvaal

Onsdag 1/11/2017

Dagens plan

1. Beskjeder
2. Oppgave 1
3. Exceptions
4. Testfunksjoner
5. Oppgave 2
6. Dictionary
7. Oppgave 3

Oversikt

Frister og timeplan

- Siste forelesning er onsdag 22/11
- Siste gruppetimer er
  - Gruppe 1: torsdag 23/11
  - Gruppe 2: mandag 27/11
  - Samretting: onsdag 29/11
- Siste frist for innlevering I: fredag 10/11, 23:59
- Siste frist for innlevering II: fredag 1/12, 23:59 NB: Foreløpig frist

Trykkfeil rettet

- Last ned siste versjon av oppgaven!
- Oppgave 1: To små trykkfeil rettet
- I testfunksjonene for Newtons metode: $x_0 = 1$ skal konvergere mot $x = 2$, og $x_0 = -1$ skal konvergere mot $x = -2$. (Og ikke omvendt!) Verdien for konstanten $K_H = 2.5 \cdot 10^{-10}$ er rettet. (Gal tidligere verdi var $K_H = 2.5 \cdot 10^{-11}$.)

Oversikt

Oppgaven

Vi tar en kikk på oppgaveteksten.

Hva skal gjøres?

Oppgave 1a:

- Implementere Newtons metode for likninger

$$f(x) = 0$$

- Denne skal legges som en funksjon i en modul, feks `birkutil` i filen `birkutil.py`
- Testfunksjoner, feks i filen `test_newton.py`

Newtons metode

- Iterativ metode:
  1. Start med et gjett x_0 for x_*
  2. Gjenta en formel mange ganger:

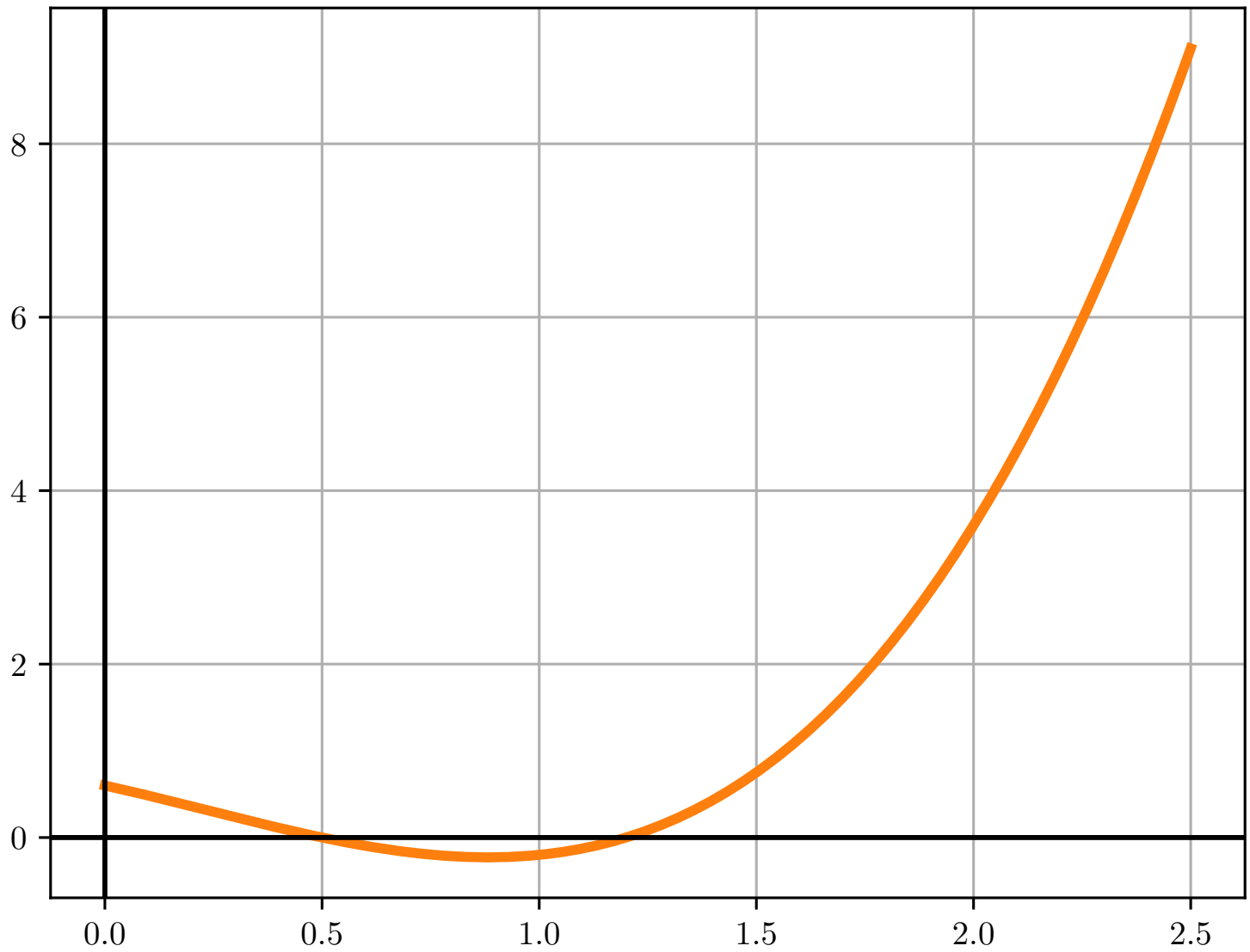
$$x_0 \longrightarrow x_1 \longrightarrow x_2 \longrightarrow \cdots$$

  3. Stopp når x_k har konvergert
- Formelen for Newtons metode er:

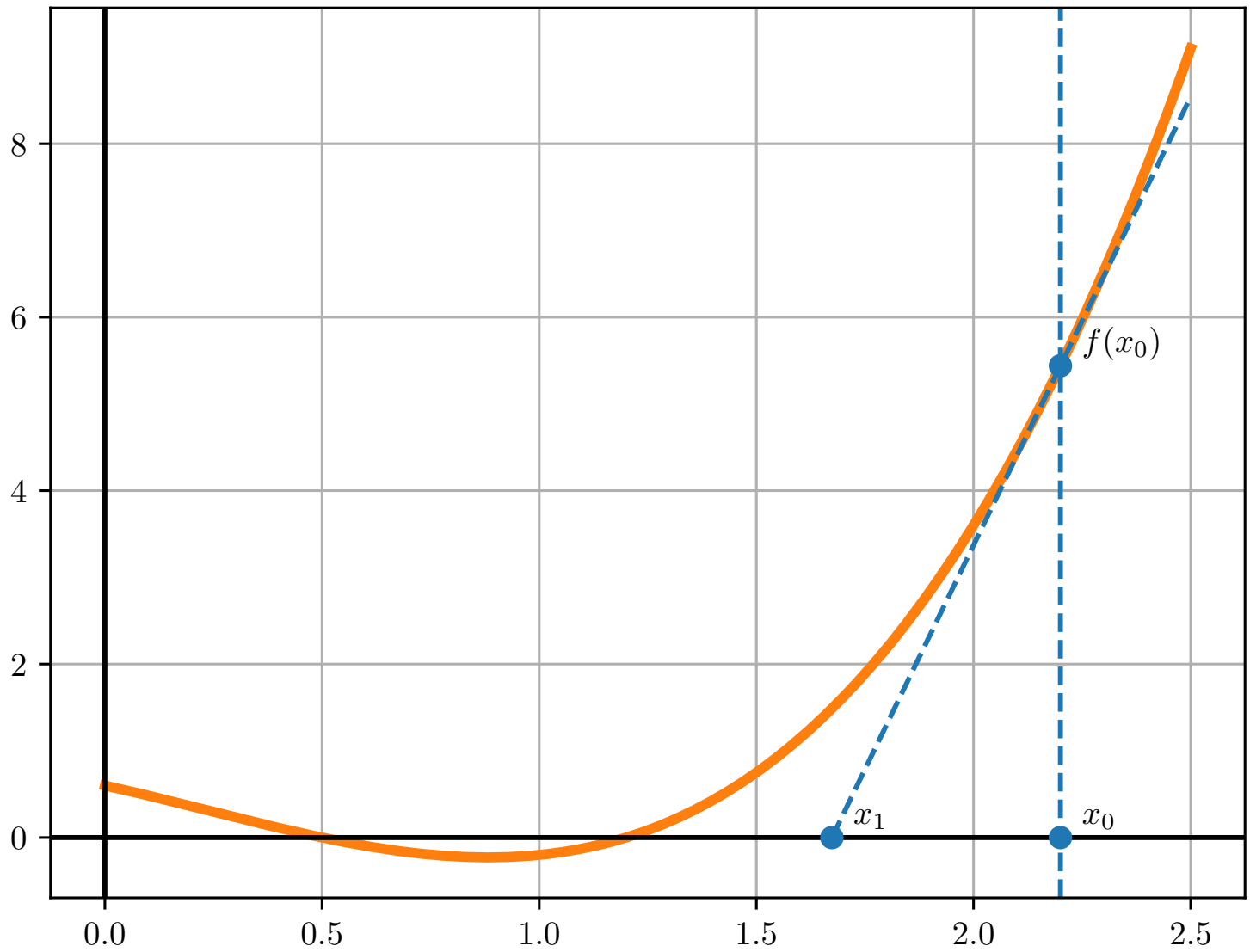
$$x_{k+1} = x_k - f'(x_k)^{-1} f(x_k).$$

- Vi må ha et uttrykk for både $f(x)$ og $f'(x)$.

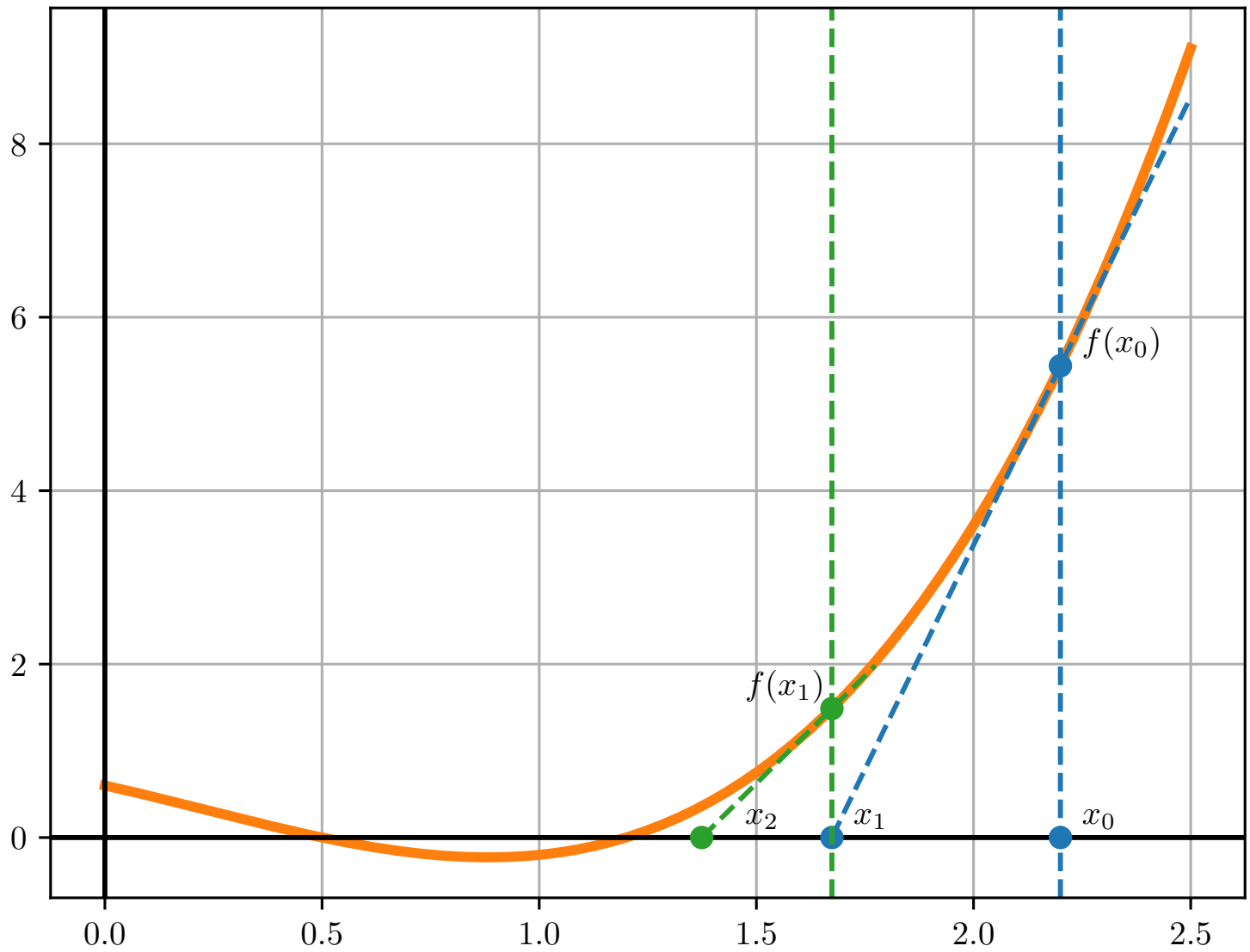
Illustrasjon: $f(x) = x^3 - 0.7x^2 - 1.1x + 0.6$



Illustrasjon: $f(x) = x^3 - 0.7x^2 - 1.1x + 0.6$



Illustrasjon: $f(x) = x^3 - 0.7x^2 - 1.1x + 0.6$



Hva skal gjøres

- Implementere Newtons metode for *generelle funksjoner*
- Denne skal ligge i en modul, feks. `birkutil.py`
- Det vil si at du bare putter definisjonen i denne fila, slik at du kan importere med `import` senere
- Krav til funksjonen:
  - $f(x)$ og den deriverte $f'(x)$ skal være input (liveeksempel: funksjoner som argumenter)
  - Toleranseparameter `epsilon`: terminere når $|f(x_k)| <$`epsilon`
  - Maks antall iterasjoner `maxit`. Terminere dersom `epsilon` ikke er nådd
  - Reise exception når $f'(x_k) = 0$

Testfunksjonene

- Skal teste om din versjon av Newtons metode virker
- 3 tester
- $f(x) = x^2 - 4$, $x = \pm 2$. Starte med $x_0 = \pm 1$.
- Starter man med $x_0 = 0$, så skal din Newton reise exception:

$$f'(0) = 0$$

$$x_1 = 0 - f(0)/f'(0) \qquad ???$$

Hva skal gjøres?

Oppgave 1b:

- Funksjon fra birkenesmodellen (elektronøytralitet):

$$f(x) = 3K_{AlH}x^3 + 2K_{HCa}^{-1}x^2 + x - 2s - K_H x^{-1}$$

$$\textbf{A}: \quad K_{AlH} = 10^9, \quad K_{HCa} = 10^{-2.2}, \quad K_H = 2.5 \cdot 10^{-10}$$

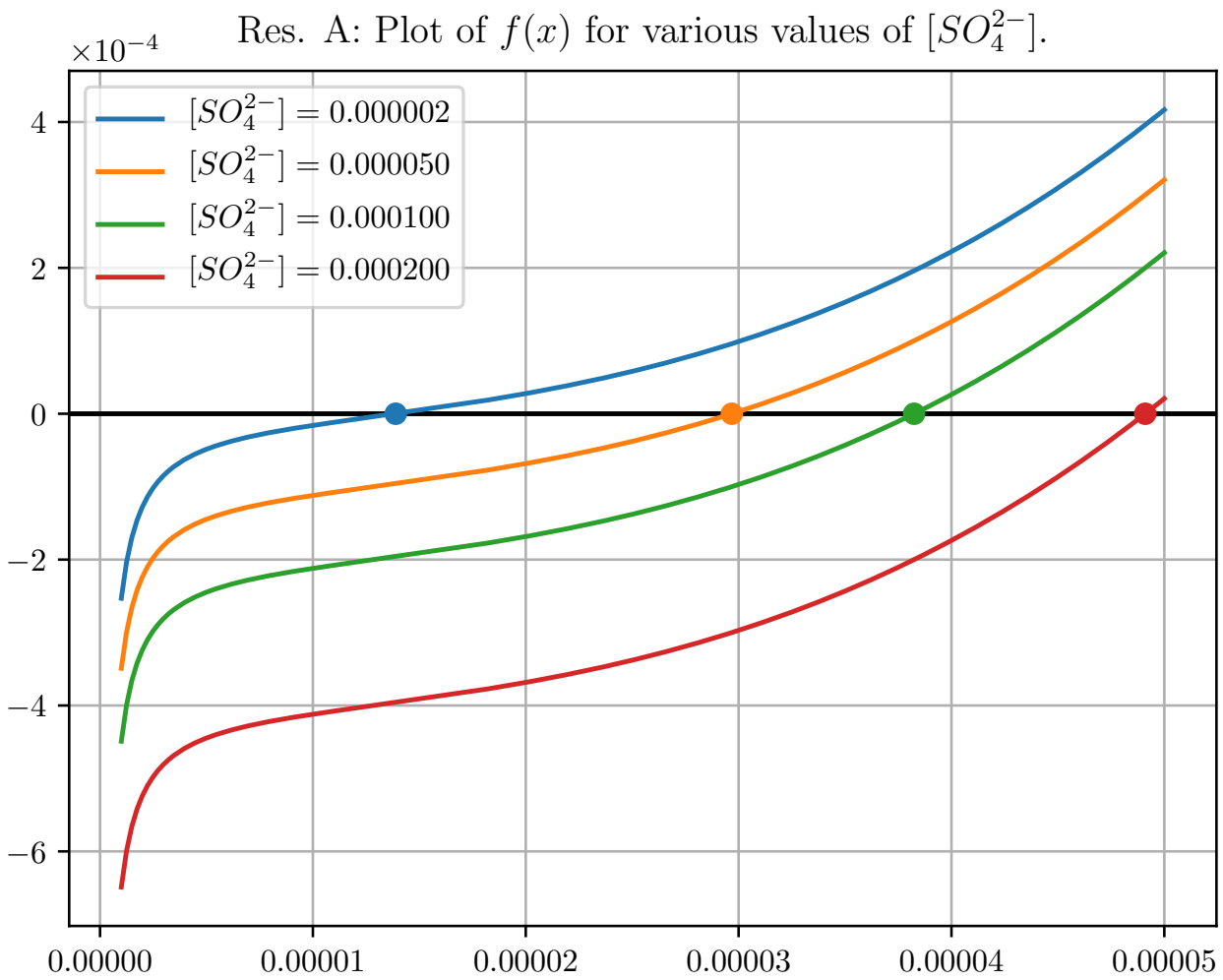
$$\textbf{B}: \quad K_{AlH} = 10^9, \quad K_{HCa} = 10^{-3.2}, \quad K_H = 2.5 \cdot 10^{-10}$$

$$\text{alltid} \quad x > 0$$

- Hver verdi av $s = [SO_4{}^{2-}]$ gir ulik funksjon.
- Studere $f(x)$. Plotting for ulike s.
- Hvorfor? Vi trenger en rimelig verdi for x_0 når vi skal kjøre Newton!

Eksempel

Her er $f(x)$ plottet for ulike verdier av s. Røttene er markert.



Hva skal gjøres? (forts.)

- Skrive modulfunksjon for $f(x) = 0$ i birkenesmodellen

```
from birkutil import finn_konsentrasjon
x = finn_konsentrasjon(s, K_AlH, K_HCa, K_H)
```

- Bruke implementeringen av Newtons metode fra 1a
- Testfunksjoner, feks i `test_oppgave1b.py`
- Plotting. Se eksempler i prosjekttekstens appendiks

Oversikt

Testfunksjoner og exceptions

- I del I av prosjektet: testfunksjoner en del av oppgaven
- Det kan være greit å vite litt om exceptions, siden testfunksjoner baserer seg slike via `assert`

```
# skjematisk testfunksjon
def test_funksjon():

    # teste mange ting her
    assert success, msg   # assert kan kaste exception
```

Hva er exceptions?

- Exceptions er feilmeldinger. Forteller at noe uforutsett har hendt
- Exceptions er den moderne måten for å behandle feilmeldinger og uforutsette hendeler i et program
- En funksjon kan reise/kaste (raise) en exception. Dersom denne ikke er fanget/behandlet (catch) terminerer programmet.
- Vi kan behandle exceptions med en try-except blokk:

```
try:
    # vi forsøker å gjøre dette
    # dersom det kommer en exception, hopper vi til try-delen
except:
    # dette skjer bare dersom det kommer en exception i try-delen
```

- Dersom en exception er ubehandlet, vil programmet terminere

Eksempel

```
#
# Eksempel på en try-except-blokk.
# Vi forsøker å eksekvere noen statements.
# Når ett av de feiler, hopper vi rett ut av try-delen
# og går til except-delen
#
try:
    x = 1.0
    print x
    x = 1.0/2
    print x
    x = 1.0/0  # Kaster en ZeroDivision exception og hopper til except.
    print x    # Dette kommer aldri til å bli eksekvert
    x = 1.0/10 # Ikke dette heller
    print x    # Ikke dette heller

except:
    print "Oops!"
```

Typer exceptions

- Det finnes mange typer exceptions, akkurat som det finnes mange typer feil som kan skje
- `ZeroDivisionError` – ble kastet/reist i eksempelet vårt
- `ValueError` – ganske generell. Brukes feks dersom variabler har feil/ugyldige verdier
- `IndexError` – eksempel: `y[3]` dersom `y` bare har 1 element (liveeksempel)
- `AssertionError` – kommer fra `assert`-kommandoer
- Mange fler ...

`assert` og `AssertionError`

- For testfunksjoner er det spesielt en exception som er viktig: `AssertionError`
- Kastes av `assert`
- `assert` er en enkel måte å sjekke at programmet gjør det du tenker
- Eskempel: du gjør en mengde beregninger og tenker, "nå må `x` være et positivt tall!"

```
# Først mange beregninger som resulterer i et tall x.
# Kanskje x er massen av noe.
assert (x >= 0), "massen er negativ!"  # throws AssertionError
```

- Live-eksempel: `assert_examples.py`

Vi kan også kaste/reise exceptions selv

- Anta at det skjer noe feil, og du ønsker å si fra
- Med `raise` kan vi kaste en exception.
- Dersom denne går ubehandlet, så stopper programmet.
- Eksempel: `raise.py`

Fange exceptions

- Med `except`-blokken kan vi fange exceptions av ulike typer, og foreta ulike handlinger.
- Eksempel: `IndexError` har typisk andre årsaker enn `ZeroDivisionError`, så de må ofte behandles ulikt
- Live-eksempel: `assert_examples2.py`

Oversikt

Testfunksjoner og assert

Live-eksempel: Programmering av løsning av kvadratisk likning

$$ax^2 + bx + c = 0$$

$$x = \frac{1}{2a}\left(-b \pm \sqrt{b^2 - 4ac}\right)$$

Vi har lagt inn en bug ...

`kvadratisk.py` og `test_kvadratisk.py`

Oppgaveteksten, revisited

Vi tar en kikk på oppgaveteksten igjen, nå som vi har vært igjennom mye materiale

Oversikt

Oppgaven

Vi tar en kikk på oppgaveteksten

Hva skal gjøres?

- Skrive modulfunksjon for innlesing av tallverdier fra fil, `birkenes.data`
- La oss ta en kikk på denne fila
- Funksjonen skal lagre verdiene i dictionary

Oversikt

“Dictionary” = “ordbok”

- Vi skal kun gå overfladisk inn på dictionaries her
- En form for array/liste, men indeks kan være en hvilken som helst immutable type, feks strenger.
- Et dictionary består av en samling nøkkel-verdi-par

```
# opprett et tomt dictionary
d1 = {}
# opprett et annet dictionary
d2 = {'fornavn': 'Harry', 'etternavn': 'Potter', 'alder': 13}
# printe
print d2['fornavn']
```

- Som vi ser kan verdiene være av ulik type: `'Harry'`, `'Potter'`, 13.
- Nøklene kan også være av ulik type (!). Vi skal bruke strenger. Dictionaries kan være hendige når man har store datamengder med komplisert struktur.

Liveeksempel

`dictionaries.py`

Dictionaries i oppgave 2

- Lagre hver kolonne i datafilen i et dictionary.
- `data['cpso4'] =` en numpy array med verdier
- Veldig hendig senere! Letter å huske `'cpso4'` enn feks indeks 2, om vi hadde lagret dataene i en liste av lister ...

Oversikt

Oppgaven

La oss se på oppgaveteksten

Hva skal gjøres?

- Skrive et skript — ikke en modulfunksjon denne gangen
- Løse et testproblem med Eulers metode

$$\frac{d}{dt}y^1(t) = \pi y^2(t)$$

$$\frac{d}{dt}y^2(t) = \pi y^1(t)$$

$$y^1(0) = 0, \quad y^2(0) = 1$$

- Vi kan løse denne eksakt:

$$y^1(t) = \sin(\pi t), \quad y^2(t) = \cos(\pi t)$$

  Tegner en sirkel i planet
- Du skal løse ODE med Eulers metode med ulike steglengder. For-loops! Se også eksempel fra forrige forelesning.
- Dere skal se at mindre steglengde gir bedre svar

Liveeksempel: `euler-double-decay.py`

- Vi skriver et skript for å løse

$$\frac{d}{dt}x(t) = -ax(t)$$

$$\frac{d}{dt}y(t) = ax(t) - cy(t)$$

$$x(0) = 1.0, \quad y(0) = 0.0$$

$$a = 1.0, \quad b = 0.5$$

- Dette beskriver en to-stegs radioaktiv henfallsreaksjon, der molekyler av typen X blir til Y ved henfall, og Y henfaller videre.
- Eulers metode med steg h:

$$x_{k+1} = x_k + h \cdot (-ax_k)$$

$$y_{k+1} = y_k + h \cdot (ax_k - by_k)$$