En (lynrask) introduksjon til (akkurat så mye) Python (som du trenger for TMA4135 Matematikk 4D)

Gard Spreemann
Institutt for Matematiske Fag

26. august 2013

Hva er Python og NumPy/SciPy?

Python...

- Programmeringsspråk
- Enkel, grunnleggende syntaks, som er lett å lære på kort tid
- Egnet for vår bruk; små programmeringsoppgaver som illustrerer matematikken i TMA4135

Hva er Python og NumPy/SciPy?

Python...

- Programmeringsspråk
- Enkel, grunnleggende syntaks, som er lett å lære på kort tid
- Egnet for vår bruk; små programmeringsoppgaver som illustrerer matematikken i TMA4135

NumPy...

- Utvidelser for Python for numerisk og vitenskapelig kode ("Matlab-killer")
- Vår bruk (NumPy): Matriser, vektorer, matriseoperasjoner

Hva er Python og NumPy/SciPy?

Python...

- Programmeringsspråk
- Enkel, grunnleggende syntaks, som er lett å lære på kort tid
- Egnet for vår bruk; små programmeringsoppgaver som illustrerer matematikken i TMA4135

NumPy...

- Utvidelser for Python for numerisk og vitenskapelig kode ("Matlab-killer")
- Vår bruk (NumPy): Matriser, vektorer, matriseoperasjoner

SciPy bygger igjen på NumPy og gir et vidt spekter av funksjonalitet for vitenskapelig programmering. Det kan hende vi vil benytte enkelte elementer i en større øving mot slutten av faget.

“Hva må vi kunne?”

- Matematikk 4D er **ikke** *et programmeringskurs!*

“Hva må vi kunne?”

- Matematikk 4D er **ikke** *et programmeringskurs!*
- Matematikk 4D inneholder dog *algoritmer og beregningsmetoder som hører hjemme på en datamaskin!*

“Hva må vi kunne?”

- Matematikk 4D er **ikke** *et programmeringskurs!*
- Matematikk 4D inneholder dog *algoritmer og beregningsmetoder som hører hjemme på en datamaskin!*
- Ved å gi små programmeringsoppgaver på noen av øvingene får en dermed prøvd seg “hands on” med noe av matematikken i kurset.

"...men...hva *må vi kunne??*"

Svar: Uttrykke algoritmene og de numeriske metodene i faget i Python.

"...men...hva *må vi kunne??*"

Svar: Uttrykke algoritmene og de numeriske metodene i faget i Python.
Det innebærer bruk av grunnleggende elementer som:

- Funksjoner (`def`, `return`)
- Tilordning av og operasjoner på variabler (her: tall)
- Conditionals (`if`, `else`)
- Løkker (`while`, `for`)
- Fra NumPy: Operasjoner på matriser (oppretting, lesing/skriving til enkeltelementer, uthenting av rader/kolonner, og de grunnleggende matriseoperasjonene fra lineæralgebra)

Grunnleggende syntaks

```
# Dette er en kommentar. Alle symboler fra og
# med # på en linje blir ignorert.

def sum_av_to_tall(x, y):
    z = x + y # Kodeblokken som tilhører funksjonen
    return z  # sum_av_to_tall må innrykkes med
              # mer enn utenforliggende blokk.

def minste_tall(x, y):
    if x <= y:    # Første innrykksnivå
        return x # Andre innrykksnivå
    else:
        return y

def si_hei():
    print "Hei!"
```

Eksempler!

Oppgåve 6 To metodar for å rekne ut ei løysing til likninga

$$x^4 - 2x - 1 = 0 \qquad (*)$$

er implementert som følgjer i `Python`. Vi kan anta som kjent at likninga har ei løysing i intervallet $[1,2]$.

```
def metodeEin(N):
    x = 1.39

    def g(x):
        return 0.5*(x**4 - 1)        # x**4 betyr x^4

    for n in range(0, N):            # 0 <= n <= N - 1
        x = g(x)
    return x


def metodeTo(N):
    x = 1.39

    def g(x):
        return (1 + 2*x)**0.25       # (...)**0.25 betyr (...)^0.25

    for n in range(0, N):            # 0 <= n <= N - 1
        x = g(x)
    return x
```


Kva for ein av dei to metodane kan vi garantere at konvergerer mot løysinga? Grunngi svaret.

Bruk den metoden du mener konvergerer mot svaret og rekn ut løysinga til $(*)$ (til fem signifikante siffer) der du bruker same startverdi som i koden.

Anta at

$$x^4 - 2x - 1 = 0$$

har én løsning i intervallet $[1, 2]$. La oss skrive et program som approksimerer løsning av ligningen ved å gjøre N fikspunktiterasjoner fra startpunktet $x_0 = 1{,}39$.