Python

Python

Python er et programmeringsspråk som relativt enkelt å lære seg (sammenlignet med for eksempel C++) og som likner på mange andre relevante språk (for eksempel MatLab). Dette er en introduksjon som setter deg i gang om som går gjennom alle relevant kommandoer som vi skal bruke i TFY 4125

Installasjon

Gå til https://www.enthought.com/downloads/ Trykk *Download Canopy 1.1* og følg instruksjonene.

Klikk *entought canopy* snarveien som dukker opp på skrivebordet (dette kan sikker variere avhengig av operativesystem, men du finner nok frem til hvor programmet er installert). Velg *editor* i vinduet som kommer opp

[bilde av vindu]

Du er nå klar til å programmere

(Du må gjerne installere en hvilken som helts annen python versjon så lenge den har nødvendig funksjonalitet, biobliotekene *matlibplot* og *numpy*)

Tutorials

Her er noen lenker til ressurser på nettet. Det fins et hav av disse (gi gjerne tips om gode nettsteder)

- Python
  - http://docs.python.org/2/tutorial/
- Matlibplot
  - http://matplotlib.org/users/pyplot_tutorial.html
- Numpy
  - http://wiki.scipy.org/Tentative_NumPy_Tutorial#head-d3f8e5fe9b903f3c3b2a5c0dfceb60d71602cf93

Python undervises i et av parallellene i IT Intro. Foilene finner du her:
http://itgk.idi.ntnu.no/foiler/index.php?parallell=1

Programmering

Ok, la oss sett i gang. Åpne et nytt skript (ny-fil ikon oppe til venstre)

[bilde]

Variabler

I et dataprogram lagrer man informasjon i variabler. `Print` kommandoen gir deg verdien av variablene i output-vinduet. Skriv kommandoene

- `a = 2`
- `b = 3`
- `print a`
- `print b`
- Trykk på Run (grønn trekant)

Dersom tallet skal brukes i en utregning som gir desimaltall kan det være lurt å skrive a = 1.0 så ikke programmet tror vi er ute etter ett heltall (integer) og avrunder resultatet. Se forskjellen på disse

- `3/2 = 2`
- `3.0/2 =1.5`

Regning

Python fungerer fint som en kalkulator. Skriv følgende funksjoner. #-symbolet brukes for å skrive inn kommentarer i koden.

```
c = a + b  #Addisjon
d = b-a          #Subtraksjon
e = 2*a*b-1      #Multiplikasjon
f = sqrt(e)      #Rot
g = a**2   #Eksponent
h = pi           #Tallet π
print(c,d,e,f,g,h)
```

Rekke, tabell (array)

Lagre det forrige skriptet ditt om du ønsker (med endelsen .py). Åpne et nytt skript (open).

I fysikk bruker vi datamaskiner for å analysere eller simulere måledata. Data kommer i lange serier som lagres i en tabell (array). Vi bruker arrays fra *Numpy* modulen og hvis vi da importerer modulene med (sett denne komandoen først i skriptet ditt),

- `import numpy as np`

må alle numpy kommandoer begynnes med `np.`

For å lage en rekke med tall som du skriver inn selv kan du skrive

- `a = np.array([1,2,3])`
- `print(a)`

Trykk run

Om vi har veldig lange rekker blir det slitsom å skrive dem inn selv. Derfor fins funksjoner som automatisk genererer rekker med en ordnet struktur. F.eks for å generere alle tall mellom 1 og 100:

- `b = np.arange(1,101,1)`
- `print(b)`

Trykk run. Forståelsen er `np.arange` (*begynn med dette tallet, slutt før dette tallet, øk med dette tallet for hver gang*).

Dersom du ikke vet intervallet men vet hvor du vil begynne og stoppe og vet hvor mange tall du skal ha kan du skrive

- `c = np.linspace(-3,3,50)`
- `print(c)`

Trykk run. Dette gir 50 tall mellom -3 og 3

Å hente ut ett eller flere elementer fra en tabell kalles indeksering. For å hente ut det fjerde elementet i en tabell skriver man

- `d = b[3]`
- `print(d)`

Trykk run. Merk at det første elementet har indeksen 0 slik at det fjerde elementet har indeksen 3 (og så videre)

For å hente ut en rekke med tall fra en tabell brukes syntaksen

- `e = b[0:4]`
- `print(e)`

Trykk run.

Man kan også regne direkte med tabeller, hvor da den aritmetiske operasjonen gjøres elementvis. For eksempel for å finne differansen mellom to elementer i en rekke (dette kommer du til å bruke for numerisk å regne ut den deriverte)

- `f = c[1:49]-c[0:48]`

Trykk run

Figurer

Lagre forrige fil om du ønsker og åpne en ny fil. Å tegne grafer med et dataverktøy er svært nyttig og effektivt. Mye raskere enn å tegne opp manuelt og du kan raskt se effekten av å endre parametere. *Matplotlib* er en modul som inneholder mange kommandoer for å lage figurer.

Skriv følgende i begynnelsen av skriptet

- `import matlibplot.pyplot as plt`
- `import numpy as np`

Vi skal nå plott funksjonen x^2. Lag først en tabell med x-verdier der du vil plotte funksjonen.

- `x = np.linspace(-5,5,100)`

Regn så ut funksjonsverdien (når du bruker en tabell i en utregning får du ut en tabell)

- `f = x**2`

For å plott funksjonen bruker vi kommandoen plot. Første argumentet er x aksen og andre er y-aksen

- `plt.plot(x,f)`

For å få frem et vindu med figuren må skrive

- `plt.show()`

Denne figuren kan du så lagre i ulike billedformater som du kan skrive ut. I øvingene vil du bli spurt om å legge ved diverse grafer. Skriv også alltid ut koden du har brukt for å lage dataene dine når du leverer øvingene.

Importering av data

Når man bruker dataverktøy for å analyser data må man ofte hente inn data fra et måleverktøy som er lagret i en datafil. Dette vil du måtte gjøre i laboratorieøvingene.

Lag en tekst fil (.txt) hvor du skriver inn en rekke tall og trykker linjeskift etter hvert tall (slik at det blir en kolonne med tall). Legge denne filen i samme mappen som skriptet ditt er (Mulig du må skifte working directory – se bilde)

- `a = np.genfromtxt(filname).`
- `print(a)`