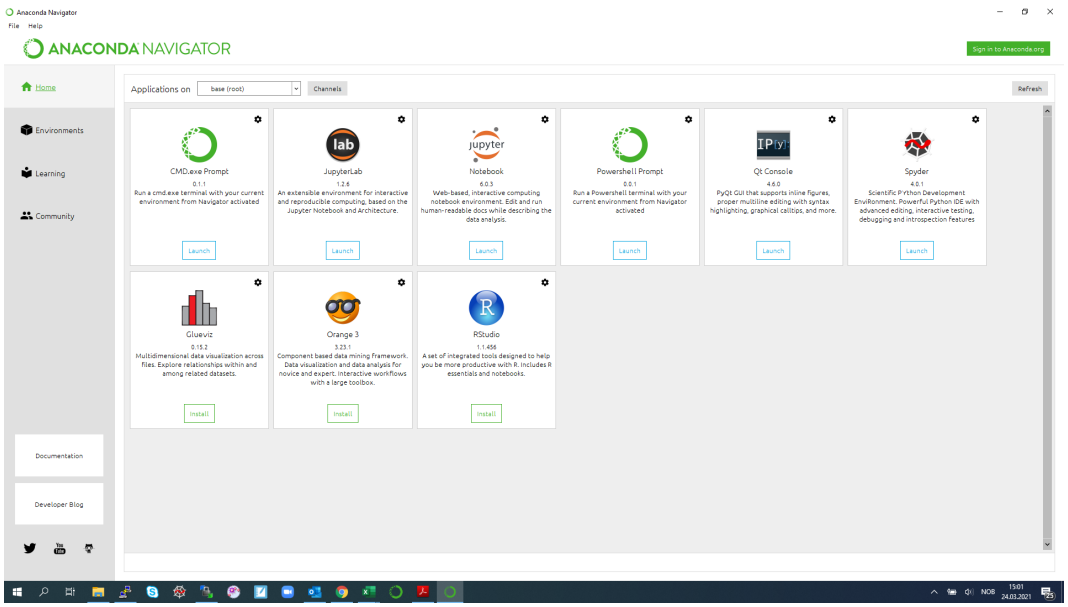
Python – Installering og et par enkle anvendelser

1. Gå til `https://docs.anaconda.com/anaconda/install/#`.

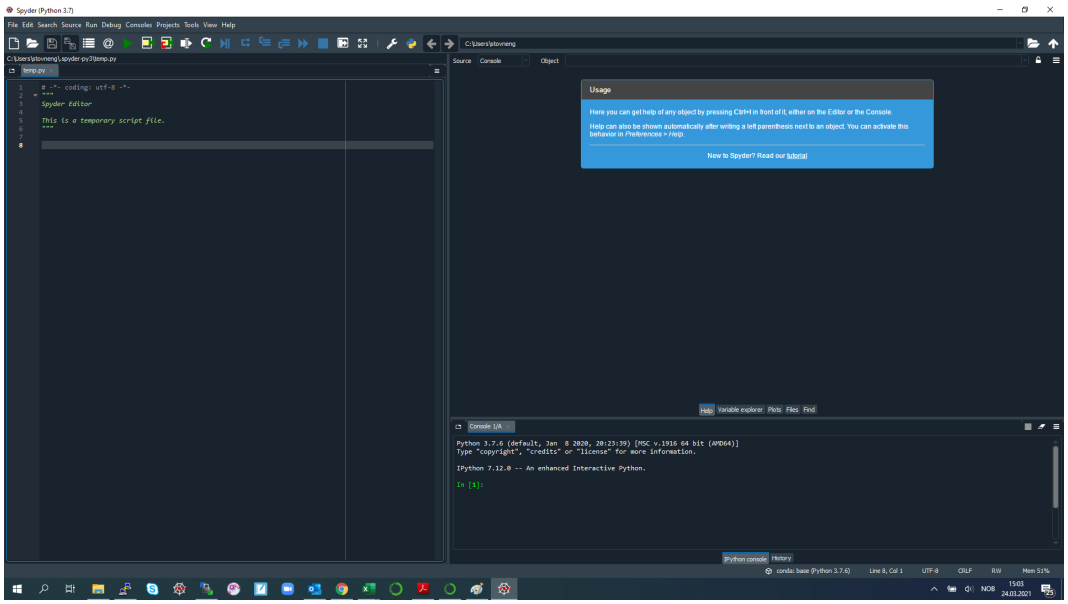
2. Last ned og installer anaconda for ditt operativsystem.

3. Start programmet med å dobbeltklikke ikonet for Anaconda-Navigator på skjermen. Slik ser det da ut hos meg (Windows):



Figur 1: Windows skjermbilde etter oppstart av Anaconda-Navigator.

4. Anaconda inneholder et knippe ulike program. Finn Spyder og klikk *Launch*:



Figur 2: Windows skjermbilde etter oppstart av Spyder.

Du kan nå begynne å skrive inn python-kommandoer på kommandolinja i vinduet til høyre, f.eks:

```
In [1]: 2+2
Out[1]: 4
In [2]: 7**(0.5)
Out[2]: 2.6457513110645907
```

Dvs, python kan brukes som en enkel kalkulator. Som regel ønsker vi å skrive et program og lagre dette i ei fil.

5. I editor-vinduet til venstre har du et forslag til heading i et program som foreløpig er kalt temp.py. Her kan du skrive inn python-kommandoer, f.eks:

```
a=6
b=7
answer=a*b
print(answer)
```

Velg *File - Save As* og lagre programmet (dvs disse fire kommandoene) med et passende filnavn (som du velger selv) og *extension* .py (som er nødvendig for at python skal skjønne at det er et python-program), f.eks addisjon.py. For å holde litt orden på saker og ting oppretter jeg mappa `python` under `C:\Users\stovneng\FY6013_2021`. Nå kan programmet kjøres (dvs de fire kommandoene utføres, en etter en) ved å trykke på den grønne pila i menyen (*Run file*). I vinduet nede til høyre får jeg nå svaret

```
42
```

Med andre ord, python setter variablene a og b lik hhv 6 og 7, variabelen answer settes lik produktet mellom a og b, dvs 42, og til slutt skrives verdien av answer ut til det som gjerne kalles *standard output* (som her er skjermen, vinduet nede til høyre) med kommandoen print.

6. Du ser allerede nå at python forstår en god del matematikk "uten videre", f.eks at "+" betyr addisjon og at "**" betyr "opphøyd i". Hva med litt mer avanserte matematiske funksjoner som f.eks sinus og cosinus? La oss prøve. Lag ei ny fil med kommandoen *File - New file*. Denne heter i utgangspunktet untitled0.py, så vi lagrer den like godt som sinus.py med det samme. Skriv inn de tre linjene

```
a=1.5708
b=sin(a)
print(b)
```

og lagre. Med forventning om at python skal returnere et tall i nærheten av 1 trykker vi på den grønne pila. Skuffelsen er kanskje stor når python i vinduet nede til høyre returnerer diverse tekst, som avsluttes med

```
NameError: name 'sin' is not defined
```

Python skjønner med andre ord ikke uten videre hva kommandoen "sin" betyr. En rød sirkel med et kryss i til venstre for kommandoen `b=sin(a)` gir strengt tatt et signal om at noe er galt allerede før vi kjører programmet. Flytter du markøren bort til denne sirkelen, kommer det da også opp en melding om at `sin` er udefinert. Vi må importere et passende *bibliotek* av programmer som inneholder sinus-funksjonen, og vi velger biblioteket numpy ("numerical python") ved å inkludere kommandoen

```
import numpy as np
```

i programmet. Dessuten må vi fortelle python at funksjonen sin skal hentes fra numpy-biblioteket, som vi nå har gitt kortnavnet np:

```
b=np.sin(a)
```

Hele programmet blir da seende slik ut:

```
import numpy as np
a=1.5708
b=np.sin(a)
print(b)
```

Nå går det mye bedre! *Grønn pil* gir oss utskrift av verdien

```
0.9999999999932537
```

Biblioteket numpy inneholder de fleste matematiske funksjoner som du kan tenke deg å få bruk for. I tillegg legger det til rette for å utføre operasjoner på hele tabeller (og matriser) med tall "i en smekk". Vi skal se at det er svært nyttig. (Det finnes også et bibliotek som heter math, som vi alternativt kunne ha importert i stedet for numpy, men da hadde vi ikke fått med oss tabell-funksjonaliteten på kjøpet, slik vi gjør med numpy.)

7. Vi har rett som det er behov for å framstille våre resultater ved å plotte en eller flere funksjoner (grafer). Da må vi først importere modulen pyplot fra biblioteket matplotlib. Vi importerer også numpy, slik at vi kan lage tabeller med tall:

```
import matplotlib.pyplot as plt
import numpy as np
```

La oss ta et konkret eksempel: Vi ønsker å plotte funksjonen $\cos(x)$ på intervallet $0 < x < 2\pi$. Vi lager da først en tabell x med et antall verdier, f.eks 100, mellom 0 og 2π:

```
x = np.linspace(0,2*np.pi,100)
```

Med andre ord, `x = np.linspace(start,slutt,antall)` medfører at `x` blir en tabell med `antall` elementer, slik at 1. element får verdien `start`, siste element får verdien `slutt`, og resten av elementene er jevnt fordelt mellom `start` og `slutt`. Med kommandoen

```
y = np.cos(x)
```

blir `y` nå automatisk en tabell med like mange elementer som `x`, dvs 100, og slik at 1. element får verdien $\cos(0) = 1$ og siste element får verdien $\cos(2\pi) = 1$. Nå kan vi plotte $y(x)$ med kommandoene

```
plt.plot(x,y)
plt.show()
```

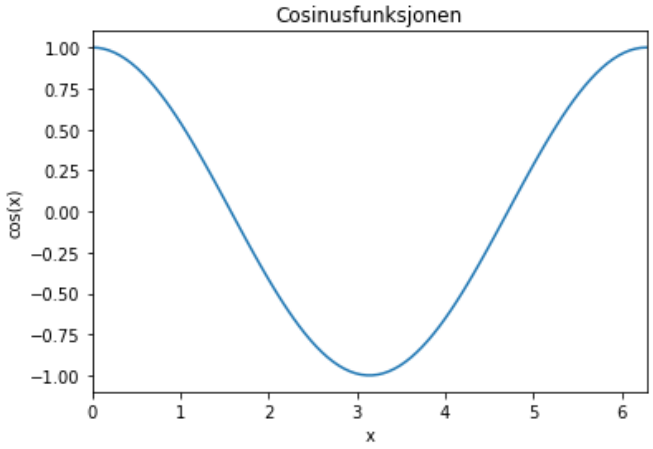
der siste kommando er nødvendig for å få opp figuren på skjermen. Vi kan pynte på figuren (der de ulike kommandoene bør tale for seg):

```
plt.title('Cosinusfunksjonen')
plt.xlabel('x')
plt.ylabel('cos(x)')
```

Enda penere ser det ut hvis vi avgrenser x-aksen til 2π:

```
plt.xlim(0,2*np.pi)
```

Skriv disse 10 linjene og lagre programmet i fila cosinusplott.py. (Pass på å ha show-kommandoen til slutt.) Kjøring av programmet gir figur 3.



Figur 3: Plotting av cos(x) med kommandoene plot og show fra modulen pyplot i matplotlib.