Introduksjon til Jupyter Notebook

I dette dokumentet gir vi en kort innføring i programmet *Jupyter notebooks*, som er verktøyet vi skal bruke for å skrive koden vår i dette kurset. Vi gir først en overordnet forklaring på hva Jupyter er, hvorfor vi ønsker å bruke det, og hvem andre som bruker det. Deretter går vi igjennom noe av funksjonaliteten til Jupyter notebook.

Hva er Jupyter notebook?

Når vi skal programmere i Python, så går det ut på å skrive Pythonkode som instruerer datamaskinen til å gjøre det vi ønsker. Denne koden må så *kjøres* for at noe faktisk skal skje. Det er ikke noe eget verktøy som medfølger Python vi bruker til å skrive denne koden, istedet finnes det en lang rekke verktøy der ute vi kan velge fra og bruke, noen er gratis, andre er ikke det. Det vanlige er å bruke en *tekst editor* for å skrive koden, lagre den til en Pythonfil (disse har .py filendelse) og deretter kjører vi filen. Det finnes ekstremt mange tekst editorer der ute. Notepad er en tekst editor som medfølger windows, og denne kan man bruke til å skrive Pythonkode, men den er veldig enkel, så de fleste velger å bruke andre programmer. Noen populære teksteditorer for Python er Notepad++, Spyder, Sublime, Emacs, vim, og listen bare fortsetter.

I dette kurset velger vi å *ikke* bruke en standard tekst editor til å skrive koden vår. Istedet bruker vi en *notebook*. Et notebook program lar oss kombinere koden vi skriver med annet innhold som brødtekst, figurer, matematikk og lignende. Samtidig kan vi kjøre koden og få resultatene inne i samme dokument som koden vår. På denne måten kan en notebook føles mer helhetlig enn å bruke en tekst editor.

Hvorfor velger vi å bruke notebook?

I dette kurset ønsker vi å fokusere på bruk av programming som et verktøy i faglige problemstillinger. Dette betyr at det er viktig å se koden vi skriver i en større kontekst. På denne måten kan vi plassere kode rett etter en forklarende tekst, og vi kan gå rett over på å tolke og analysere det koden gjør etter vi har kjørt den. Notebook er også en god måte å utvikle og dele lærerressurser, ettersom at man kan kombinere forklarende tekst med eksempelkode som leseren faktisk kan kjøre og endre på for å jobbe med stoffet. Med tradisjonelle læremidler kan man vise eksempelkode, men det kan være vanskligere å forstå for leseren når man ikke får kjørt programmene.

Notebook kan også være et godt verktøy for å jobbe prosjektbasert. Om man for eksempel skal lage en kort rapport av et programmeringsprosjekt i Word må man finne en måte å få koden sin i Word, for eksempel ved å ta et skjermbilde. Resultatene etter å ha kjørt koden må også inn, også gjerne som bilder. Om man ønsker å endre på koden sin og kjøre på nytt må man nå først endre koden, så få denne endringen inn i Word. Om de som så senere skal lese rapporten ønsker å dobbeltsjekke noe ved koden må de først skrive av koden og så kjøre den. En løsning på dette er at man leverer rapporten og koden hver for seg, men dette gjør det vanskligere å forstå konteksten. Kort fortalt kan det bli litt av en prosess.

Med en notebook derimot, kan vi skrive en rapport der koden naturlig inngår som en del av dokumentet. Man kan ha en introduserende tekst som beskriver problemstillingen, så kan man introdusere litt kode, så kan man drøfte resultatene, ha litt mer kode, osv. Om man ønsker å endre koden er det bare å gjøre dette og kjøre, og resultatene i rapporten oppdateres automatisk. Til slutt får man én enkelt fil man kan dele med andre, og de som leser har mulighet til å lese, men også å kjøre koden.

Dersom man skal dele dokumentet sitt med noen som ikke har eller ønsker å installere Jupyter, eller man ønsker å skrive ut til papir, så kan Jupyter enkelt lage pdf og html-filer som kan deles med alle. Vi kommer tilbake til dette mot slutten av dokumentet.

Jupyter brukes over hele verden

Jupyter notebook er bare ett av mange verktøy som lages av Jupyterprosjektet. Selve prosjektet er open source, som betyr at de deler all programvaren og kildekoden åpent og gratis for alle. Verktøyene som utvikles av Jupyter, inkludert notebbok, brukes idag av mange store tekonologibedrifter som for eksempel Google, Microsoft, IBM og NASA. Veldig nylig ble Jupyter tildelt en gjev softwarepris (ACM Software System Award), som deles ut for programvare som ser en bred brukerbase og som fører til en varig forbedring av IKT over hele verden. Det er mange kjemper blant tidligere vinnere, som for eksempel WWW (world wide web), Java (et populært programmeringsspråk), UNIX (grunnlaget for Linux og Mac OS). Det er altså tydlig at Jupyter og Jupyter notebook er programvare som er av høy kvalitet og som ser ut til å bli mer og mer populært over tid.

Det er ikke bare innen IKT sektoren at Jupyter ser mye bruk, det er også meget populært i realfagene, både til forskning og undervisning. Ved universiteter over hele verden utvikles og deles kursmaterialer i Jupyter notebooks, og forskere bruker Jupyter til å analysere, vise frem og dele forskningen sin. For eksempel har nobelprisvinnerene i Fysikk fra 2017 delt datasettene fra oppdagelsen de gjorde av gravtiasjonsbølger. Dette gjorde de åpent for alle, og valgt og gjøre det i Jupyter notebooks.

Samtidig blir Jupyter mye brukt innen realfagene, både til forskning og undervisning. Ved universiteter over hele verden utvikles og deles kursmaterialer i Jupyter, og forskere bruker Jupyter til å vise frem, dele og analysere datasettene sine. For eksempel har nobelprisvinnerene i Fysikk fra 2017 delt all dataen fra oppdagelsen sin av gravitasjonsbølger åpent, og har valgt å gjøre dette i Jupyter notebooks.

Hvordan bruke Jupyter notebook

Før vi kan bruke Jupyter, må vi installere det på maskinen vår. For at Jupyter skal fungere må vi også installere Python på maskinen vår. Den anbefalte måten å gjøre begge deler samtidig på er å installere en pakke som heter *Anaconda*, som er en samlepakke som kombinerer Python, Jupyter og andre småprogrammer som kan være nyttige. Anaconda er platformuavhengig, som vil si de har støtte for Windows, Linux og Mac. For å lase ned Anaconda kan du gå til

https://www.anaconda.com/download/ (https://www.anaconda.com/download/)

Om du har en arbeidspc kan det være du kan, eller må, laste ned Anaconda via en applikasjonsportal.

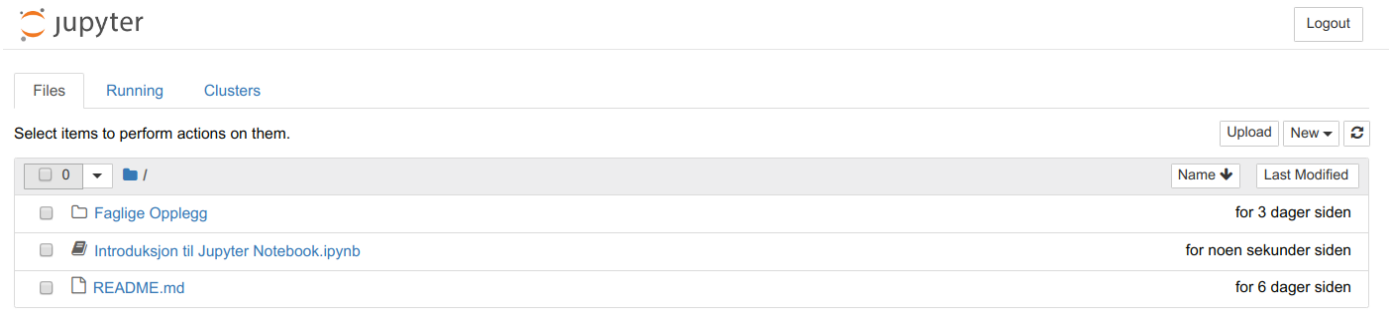
Å starte Jupyter notebook

Når du har installert Anaconda skal du kunne finne `Jupyter notebook` som et program i startmenyen, om du kjører Windows. Om du har Mac eller Linux må du åpne en terminal og skrive kommandoen `jupyter notebook` for å starte programmet.

Når Jupyter notebook kjøres, så starter datamaskinen opp en server. Dette skjer i bakgrunnen, og vi trenger ikke ta stilling til hva som foregår bak kulissene her. Samtidig åpner Jupyter opp en nettleser som man bruker til å interagere med denne serveren. Her vil Jupyter bruke standardnettleseren din, og de fleste moderne

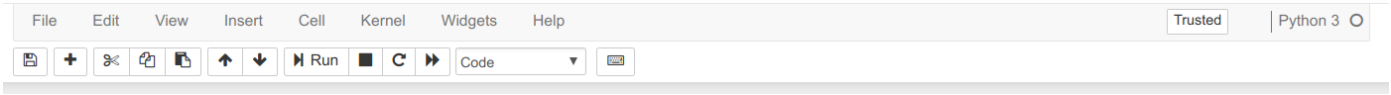
nettlesere skal fungere helt fint. Om det er noe som ikke fungerer som det skal derimot, kan det være en idé å oppdatere nettleseren din. Det er altås ikke et egen "Jupyter vindu" som åpnes, men det blir en fane i nettleseren din.

Når vi kjører Jupyter notebook får vi først opp en navigasjonsmeny. Denne viser filer som er lagret på datamaskinen. Her kan man navigere seg inn og ut av mapper som i en vanlig filutforsker. Meningen med navigasjonsverktøyet er å velge seg ut en eller flere notebooks man vil åpne, eller å lage nye og legge disse riktig sted. Notebook-filer har filendelse ".ipynb", og i navigasjonsmenyen vises disse med et spesielt ikon av en liten bok på venstresiden. Om du klikker på en slik fil vil notebooken åpne seg i en ny fane. Andre filer vises og, men med et annet filikon på venstre siden. Om du klikker på disse andre filene vil Jupyer prøve å vise dem frem. Om det for eksempel er en bilde fil går dette ofte bra, men mer kompliserte filtyper vil den ikke forstå. Derfor er det stort sett Jupyter notebook filer vi ønsker å åpne. Fra navigasjonsmenyen kan vi også rydde opp i filene våres ved å velge dem i av avhukingsboksene på venstre side og så velge operasjoner som "Flytt" eller "Slett" på toppen - slik rydding kan også gjøres i en vanlig filutforsker om man ønsker det.



Vi kan også lage en helt ny notebook ved å klikke på "New" på høyre siden. Når man klikker new får man valg om å lage forskjellig type dokumenter, og her vil vi velge `Notebook Python 3`, ettersom at vi kommer til å programmere i Python 3.

Når vi går inn på en notebook, enten den er ny eller finnes fra før, åpnes denne som en ny fane. Denne fanen er noe annerledes enn navigasjonsmenyen, og har en annen type verktøyslinje på toppen.



I notebook-fanen vår kan vi skrive, lese og endre på dokumentet vårt. Vi kan også skrive, endre og kjøre kode. Resultatene fra en kjøring kommer rett ut i notebooken og blir værende der, slik at alle som leser notebooken kan se dem senere. Ved hjelp av verktøyene på toppen har vi tilgang til en del forskjellige funksjoner vi kan bruke for å gjøre diverse operasjoner. La oss gå igjennom og se på hvordan vi kan lage dokumenter.

Celler

Celler er små vinduer med innhold, og en notebook er bare en lang rekke med slike *celler*. Det finnes hovedsakelig to type celler, *kodeceller* og *tekstceller* (disse heter Markdown i Jupyter). Kodeceller kan inneholde Pythonkode, og vi kan kjøre disse cellene, da tolkes koden linje for linje og *outputen* fra koden vises under cellen. Tekstceller derimot kan inneholde formatert tekst, det betyr at vi kan ha overskrifter, linker, tabeller, kursiv tekst, fet tekst, og så videre. Så langt i denne notebooken har alt du har lest så langt vært tekstceller. La oss vise et eksempel på en kodecelle:

In [3]:

```
1 name = "Mary"
2 print("Hello there, {}!".format(name))
```

```
Hello there, Mary!
```

Når vi ser på et notebook vises alle cellene under hverandre, som paragrafer i en tekst. Vi kan enkelt endre på rekkefølgen, kopiere, klippe og lime celler. Når vi oppretter en ny notebook vil den være helt tom, med unntak av én enkelt kodecelle som er helt tom. Vi kan lage nye celler ved å klikke på "+"-ikonet i verktøyslinja, eller å velge `Insert -> New Cell Below` i menyen på toppen.

Velge og Endre på celler

Om vi ønsker å endre på innholdet i en celle må vi første velge den. Dette kan vi gjøre ved å klikke på den med musa, eller ved å bruke piltastene opp og ned. Når en celle er valgt ser vi det er en boks rundt cellen, som gjør at vi kan se hvilke celle vi har valgt og hvor stor den er. Merk at det ikke er noen begresninger på hvor stor, eller hvor liten, en celle er. Vi kunne teknisk sett ha skrevet alt så langt i dette dokumentet som én enkelt celle. Derimot lønner det seg gjerne å dele cellene opp i logiske bolker, litt som avsnitt i en tekst.

Når en celle er valgt finnes det to moduser vi kan være i:

- I ***command mode*** kan vi bare påvirke cellen som helhet, og bruker kommandoer som å kjøre en kodecelle, flytte cellen opp og ned iforhold til andre cellen, duplisere cellen osv.
- I ***edit mode*** kan vi endre på innholdet i cellen ved å skrive nytt inhold, slette inhold, skrive kode, eller lignende

Boksen rundt cellen er fargekodet for å gi deg informasjon om hvilken modus du er i. *Blå* boks betyr command mode, og *grønn* boks betyr edit mode. I tilegg vil du se en blinkende markør som viser hvor i cellen det du skriver dukker opp om du er i edit mode.

Når du velger en ny celle vil du alltid være i command mode. Om du ønsker å gå over i edit mode må du dobbeltklikke på cellen. Alternativt kan du trykke `Enter` for å gå inn i edit mode på den cellen du har valgt. For å gå ut av edit mode må du klikke med musepekeren utenfor cella, eller du kan trykke `Esc` på tastaturet.

Kjøre en celle

Når du har valgt en celle og er i command-mode kan du *kjøre* denne cellen ved å klikke på `Run` knappen i verktøyslinja, eller ved å bruke `Ctrl + Enter` eller `Shift + Enter`. Når du kjører en kodecelle vil koden tolkes og utføres linje for linje som om det var et Pythonprogram, og outputen av programmet skrives under cella. Hva som skjer når du kjører en tekstcelle kommer vi tilbake til.

La oss se på et eksempel av en kodecelle som regner ut arealet av en sirkel. Prøv å kjør denne koden ved å velge den og så trykke `Ctrl + Enter`, eller ved å trykke på `Run` knappen øverst. Prøv så å endre på radiusen på sirkelen og kjør koden på nytt.

In [8]:

```
pi = 3.14
radius = 2
area = pi*radius*radius

print("Sirkelen har et areal på {}".format(area))
```

```
Sirkelen har et areal på 12.56
```

Når cellen kjøres vil Pythonkoden tolkes og utføres, som her går ut på å først regne ut arealet av en sirkel, og så skrive det ut til skjermen så brukeren kan se det. Merk at resultatet blir stående, og vil også være der om vi deler notebooken med noen andre. Om vi endrer på radiusen til sirkelen og kjører på nytt overskrives det gamle resultatet med det nye resultatet. På venstre side av cellen står det først `In [1]:` , fordi dette er den første cellen vi kjører. Neste gang vi kjører går denne opp til `In [2]:` osv. Merk at tallet er antallet celler vi har kjørt i hele notebooken totalt, ikke bare denne cellen - vi kommer tilbake til hvorfor dette er viktig når vi har begynnt å lære mer programmering.

En siste detalj: merk at vi kan kjøre cellen så lenge den er valgt - uavhengig om vi er i edit-mode eller command-mode.

Tekstceller

Kodeceller er ikke så veldig hokus pokus. Du skriver koden i cellen, og så kan du kjøre den og se på resultatet. Utfordringen ligger i å kunne Python, slik at man vet hvilken kode man skal skrive. I tekstceller derimot, så er det litt mer muligheter og triks man kan kjenne til. Tekstceller skrives nemlig i *markdown*. Markdown er et tekstformat som brukes mye på nett, om du for eksempel skriver i et kommentarfelt, på et forum eller lignende, så er det nok en eller annen form for markdown.

Grunntanken i markdown er at man kan skrive vanlig tekst rett frem, og om man ønsker noe spesiell formatering eller innhold, for eksempel overskrifter, fet skrift eller tabeller, så har man et par enkle symboler vi kan bruke for å få dette inn. Dette er en litt annen fremgangsmåte enn i for eksempel Word, der man gjerne bruker knapper for formattering. For eksempel, om vi ønsker fet skrift i Word velger vi ordet og trykker på fet skrift knappen. I markdown skriver vi istedet to stjernesymboler på hver side av ordet `**fet skrift**` .

Når vi er i edit-mode ser vi innholdet i en tekstcelle som markdown, det vil si at vi ser alle symbolene, og ikke formatteringen. Når vi så kjører tekstcellen, på samme måte vi kjører en kodecelle, vil cellen formateres og vises frem på en penere måte. Om vi går tilbake til edit mode ser vi igjen markdown-koden og vi kan endre på denne.

Markdown er en annen tankegang å måte å jobbe på en for eksempel Word, og det kan være litt uvant i starten, men man blir fort vandt til det og man kan skrive tekster raskt og effektivt. Merk også at det ikke er viktig å kunne markdown for å lage en notebook som kjører som den skal, det er mest for kosmetiske grunner. Vi går nå igjennom en rekke funksjonalitet vi kan få til med Markdown, og ikke alt er like viktig, det er mest ment som et referanseverk.

Tekstformattering

For å få **fet** eller *kursiv* tekst bruker vi `*` eller `_` :

- `*Kursiv**` eller `_kursiv_` -> *Kursiv*
- `**Fet**` -> **Fet**

- `* **Fet og kursiv** *` eller `_**Fet og Kursiv**_` -> ***Fet og kursiv***

Om vi ønsker å skrive kode kan vi bruke appostrofer rundt teksten, så vises det som om det var kode:

- `print("Hello, World!")`

blir til

- `print("Hello, World!")`

Dette kan være fint å bruke når man skal referere til konkrete funksjoner eller variabler. For eksempel: "I Python så vil `print`, skrive ut tekst til skjermen."

Overskrifter

Man kan lage overskrifter ved å starte linjen med én eller flere `#`. Jo fler man bruker jo mindre blir overskriften, og man kan bruke opp til 6.

Overskrift 1

```
# Overskrift 1
```

Overskrift 2

```
## Overskrift 2
```

Overskrift 3

```
### Overskrift 3
```

Overskrift 4

```
#### Overskrift 4
```

Overskrift 5

```
##### Overskrift 5
```

Overskrift 6

```
###### Overskrift 6
```

Punktlister

Punktlister lages ved å starte hver linje med *

```
* Appelsin
* Banan
* Eple
```

Blir til

- Appelsin
- Banan
- Eple

Eller vi kan bruke tall

```
1. Norge
2. Sverige
3. Danmark
```

Blir til

1. Norge
2. Sverige
3. Danmark

Lenke

Ettersom at Jupyter bruker nettleseren til å redigere på dokumentet vårt føles det veldig naturlig å bruke hyperlenker for å hoppe til andre notebooks eller nettsider. Dette gjør vi som følger:

```
[lenketekst](nettaddresse)
```

Der nettaddressen ikke vises til brukeren, men man kan se den om man holder over linken uten å klikke på den.

Eksempel:

```
Som du kan lese i for eksempel [Aftenposten]
(https:www.aftenposten.no)
```

Blir til

Som du kan lese i for eksempel Aftenposten (https://www.aftenposten.no)

For å linke til andre notebooks bruker du navnet på notebook-filen istedetfor nettadressen.

```
Vi skal snart begynne å se på [Pythonprogrammering](Introduksjon til
Python.ipynb)
```

Vi skal snart begynne å se på Pythonprogrammering (Eksempelnotebook.ipynb)

Merk at du må ha med filendelsen (".ipynb"). Dette fungerer også bare hvis filen ligger i samme mappe på datamaskinen, ellers må vi ha med mappestrukturen i filnavnet

```
 [digitale bilder og filtere](Faglige Opplegg/Opplegg 2 - Digitale
Bilder og Filtere.ipynb)
```

[digitale bilder og filtere (Faglige Opplegg/Opplegg 2 - Digitale Bilder og Filtere.ipynb)]

Bilder

Vi kan legge inn bilder i tekstceller. For å gjøre dette bruker vi HTML istedetfor Markdown (Jupyter skjønner HTML godt). Syntaksen for å vise et bilde i HTML er som følger:

```
<img src="bildeaddresse">
```

Der bildeaddressen enten kan være en nettadresse, eller en filplassering på egen maskin. Om du i brosweren din høyreklikker et bilde på internet vil du kunne velge `Kopier Bildenettaddresse` eller noe lignende. Vi kan for eksempel gå på Wikipedia og finne et bilde av en rød panda (https://en.wikipedia.org/wiki/Red_panda#/media/File:RedPandaFullBody.JPG)

Om vi kopierer bildeaddressen vil bildet gjengis i original størrelse, som kan bli veldig stort eller veldig lite:

Men vi kan endre på størrelsen ved å gi bredden eller høyden som et ekstra argument (bildet skaleres likt i begge dimensjoner

```
<img src="..." width=400>
```

gir

Om bildet ligger på egen maskin, og ikke på nett, skriver vi filnavnet i `src`, og vi må ha med filendelsen. Om bildet ligger i samme mappe som notebookfilen kan vi bare skrive mappenavnet, ellers må vi ha med mappestrukturen. Om den for eksempel ligger i en mappe som heter "bilder" kan vi skrive

```
<img src="bilder/mitt_bilde.png">
```

Merk at når vi bruker HTML til vise bilder på denne måten så vil vi alltid refere til et bilde utenfor notebooken. Om de ligger på nett er dette greit, for alle med nettilgang vil se dem når de åpner notebooken. Med egne bildefiler derimot, må disse sendes med ekstra.

Man kan også legge til bilder i en notebook på en måte der de følger med notebooken ved å dra bildet og sleppe det i en markdown celle - men dette er ikke helt idéelt, da det tar mye mer plass og gjør at notebooken bruker mer tid på å laste inn. Samtidig får vi mindre kontroll over bildet, da vi for eksempel ikke kan velge størrelsen. Om man lager notebooks man ønsker å dele med andre kan det være en idé å laste opp bilder til nett først, og så linke til dem. Her kan man for eksempel bruke imgur (https://imgur.com/), men pass isåfall på at du har rett til å bruke bildene du laster opp, og at de ikke inneholder noe sensitivt.

Andre eksempler

Det er mange muligheter til hva vi kan inkludere i Jupyter, og vi dekker ikke alt. Her viser vi bare et par eksempler på det vi kan gjøre. Her kan du kopiere eksempelet vårt eller gjøre et kort google søk om du ønsker å gjøre noe lignende.

Tabell

Land	Innbyggere	Areal
Norge	5.2 millioner	385 tusen km^2
Sverige	9.8 millioner	447 tusen km^2
Danmark	5.7 millioner	43 tusen km^2

Youtube

Vi kan ikke legge youtube-videoer rett inn i tekstceller, men vi kan inkludere et bilde som fører til videoen om man klikker på den

(http://www.youtube.com/watch?feature=player_embedded&v=dU1xS07N-FA)

Om vi ønsker å få videoen inn i selve notebooken kan vi gjøre det med en litten Python kodesnutt

In [1]:

```
from IPython.display import HTML
HTML('<iframe width="560" height="315" src="https://www.youtube.com/embed/dU1xS
```

Out[1]:

Matematikk i Jupyter

Matematikk og programmering er knyttet tett sammen, og vi ønsker ofte å inkludere matematikk i notebookene våres - dette finnes det god støtte for. Jupyter bruker noe som heter *MathJax*, som er laget for å vise matematikk i browsere. For å skrive matematikk bruker vi samme kommandoer som i *LaTeX*, om du kjenner til dette.

Kort fortalt bruker man `$` for å indikere at man skriver matematikk. Om vi bruker enkle dollartegn sier vi det er *inline* matematikk, det holder seg altså på samme linje som resten av teksten. For eksempel blir `$a = \pi r^2$` om til $a = \pi r^2$. Bruker vi doble dollartegn, så vil det istedet blir formatert på en egen linje `$$ a = \pi r^2$$` blir til:

$$a = \pi r^2$$

Som du ser fra eksemplene kan vi bruke `\pi` for å få greske bokstaver og `^` for opphøyd i, vi kan bruke `_` index, `H$_2$0` blir til H_2O. Merk at alle symboler i matte uttrykk automatisk blir kursiv, dette er fordi variabler i matematikk skal være i kursiv. Om vi ønsker å ha tekst, kan vi bruke `\text{}` rundt teksten vår

$$\text{areal} = \pi r^2.$$

Vi kan bruke `\frac{}{}` for å lage brøker, der den første parantesen blir nevner og den andre blir telleren

$$\text{areal av kule} = \frac{4\pi r^3}{3}.$$

Vi kan bruke `\sqrt{}` for kvadratrot:

$$a^2 + b^2 = c^2 \quad \Rightarrow \quad c = \sqrt{a^2 + b^2}.$$

Tastatursnarveier

Det finnes en lang rekke snarveier man kan bruke i Jupyter. Om du jobber en del med Jupyter og skriver noen lengre notebooks er det veldig verdt å merke seg et par av disse. Du kan se hele listen ved å gå på `Help -> Keyboard Shortcuts`