



# Alt du burde lært i ITGK

(og litt du lærte men sikkert ikke husker)

<3 hilsen vegard

# Hvorfor er vi her?

- Alle forventer at du kan Python
- Man lærer ikke Python i ITGK
  - Joda, man lærer litt

```
1 import løs_probleme_mine
2
3 mine_problemer = 'Labrapport'
4 løs_probleme_mine.løs(mine_problemer)
```

- NumPy, SciPy => Matte
- Matplotlib => Plotting
- Pandas => Filbehandling
- SymPy => Algebra



# Fillesing

- `np.genfromtxt()` eller `np.loadtxt()` hvis du vil
  - `skip_headers` (optional): Hopp over rader i starten av filen
    - NB: Du får IKKE feilmelding hvis du hopper over for få rader, du får NaN i stedet!
  - `delimiter` (optional): Hvilket tegn er mellom kolonnene?
  - `invalid_raise` (optional): Kræsje hvis det er feil antall kolonner et sted?
  - masse mer, bare sjekk dokumentasjon
- Pandas!
  - Gjør excel filer lett!
    - Husk: Alt excel kan gjøre, kan Python gjøre bedre <3
    - Husk: Alt excel ikke kan gjøre finnes det et python-bibliotek for (prøv "import anitgravity" hvis du tviler)
  - Bittelitt mer styr å lære seg, men gull verdt når du har store, uordnede dataset

# Henger du med?

- For hver av filene 'drittfil.txt', 'kjiip\_fil.csv' og 'excel\_fil.xlsx'
  - Les inn filen
  - Lagre hvert av datasettene t, x og y i en egen variabel
  - Print variablene og merk forskjellen på når du leser inn med np.genfromtxt() og pandas
- For å konvertere en enkelt kolonne i en pandas DataFrame til en python-liste (*ikke np.array*) kan du bruke f.eks.
  - `df = pd.read_excel(excel_data.xlsx)`
  - `t = df['t'].to_list()`



# Numpy

- `np.array()`
  - Matte, vektorer, matriser (ikke bruk math – bruk numpy)
  - `np.dot()`
  - `np.transpose()`
  - `np.linalg`
- `np.linspace()`
- `np.polyfit()` og `np.polyval()`
  - `Scipy.optimize.curve_fit()`
    - Bare bruk denne hvis du har likninger som helt tydelig er ikke-polynomer
    - Husk å gi en brukbar gjetning!

# Er det noen der ute?

- `kjip_fil.csv` inneholder målinger for tid, x- og y-posisjon for en berg- og dalbane
  - Les inn filen
  - Regn ut hastigheten i x- og y-retning ved hvert tidspunkt
  - Skriv x- og y- hastighetene til en excel-fil
- Hint 1: Bruk `np.genfromtxt()` for å lese inn en csv-fil, det er nok enklere enn Pandas når filen inneholder masse mellomrom og sånn.
- Hint 2: `np.diff()` gir deg differansen mellom ett punkt og det neste i en array (husk at du får ett punkt mindre i den resulterende arrayen)
- Hint 3: `df = pd.DataFrame({dict})` lager en ny DataFrame,
  - `df.to_excel(<file_path>)` skriver til en excel-fil



# Matplotlib.pyplot

- `plt.plot()` for å lage linjer
  - `plt.scatter()`, `plt.fill_between()` osv...
  - `color`, `label`, `marker`, `alpha` osv... bare å google
- `plt.scatter()` for å få punkter
- `plt.errorbars()` eller `plt.fill_between()` for å vise usikkerhet
- `plt.xlim()` og `plt.ylim()`
- `plt.xlabel()` og `plt.ylabel()`
- `plt.legend()`
  - `loc = 'upper left', 'lower right', 'upper center', 'center left', 'center right'` osv.
- `plt.savefig()`
  - Husk: `dpi=600`
- **Skriv `r'tekst_her'` for å bruke latex i figurtekster/aksetitler osv.**
- **Forskjellige akser: Bruk `ax.twinx()` eller `ax.twiny()`**

# Er det noen hjemme?

- Les inn dataen i 'excel\_fil.xlsx'
- plot  $x(t)$  og  $y(x)$  med en regresjonslinje, lagre plottet i en egen mappe med `plt.savefig()`
- regn ut og plot  $v_x(t)$  med en regresjonslinje i et annet plot (uten å slette koden fra forrige plot)
  - Pynt så mye du orker på plottet (fænsi linjer, merker, aksetitler osv.)
- lag et errorbar-plot av  $x(t)$  og  $y(t)$  som viser usikkerheten i x- og y-posisjon når usikkerheten i x og y er gitt ved
  - $\text{sigma\_x} = 0.8 * 0.005 * v_x(t)$
  - $\text{sigma\_y} = 1.2 * 0.005 * v_y(t)$
  - $\text{sigma\_t} = 0$



# SciPy

- `scipy.constants`, har Gasskonstanten, Plancks konstant, Boltzmann osv.
- `scipy.optimize.fsolve()`
  - Løser likningen  $f(x) = 0$
  - Kan også brukes på  $f(x,y,z) = 0$
- `scipy.optimize.root()`
  - Løser likningssett
  - Denne lærer dere i prosess (håper jeg)
- `scipy.integrate.quad()`
  - Numerisk integrasjon
  - Fantastiske greier

# Generelt å huske på

- forskjellen på "min\_funksjon" og "min\_funksjon()"
- **Alt** kan brukes som et argument til en Python funksjon
  - Inkludert andre funksjoner!!!



# Feilmeldinger

- `ValueError: operands could not be broadcast together with shapes (2,) (3,)`
  - Du prøver å gjøre matte med to arrays med forskjellig lengde
- `ValueError: x and y must have same first dimension, but have shapes (2,) and (3,)`
  - Du prøver å plotte to lister/arrays med forskjellig lengde
- `TypeError: 'numpy.ndarray' object is not callable`
  - Du har glemt et gangetegn
- `TypeError: 'int' object is not callable`
  - Du har glemt et gangetegn
- `ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or a.all()`
  - Du har skrevet `"if <np.array> == <et eller annet>:"` eller liknende

# Bonusoppgave

- Les inn dataen i `kjip_fil.csv`
- $x$ - og  $y$  beskriver posisjonen til en vogn på en berg- og dalbane ved tiden  $t$ .
- Strekningen den har beveget seg er gitt ved
  - $\int_{x(0)}^{x(t)} \sqrt{(dy/dx)^2 + 1} dx$
- Plot total strekning vognen har beveget seg som funksjon av tid
- Hint: Bruk `scipy.integrate.quad()` til integralet
- Hint: Bruk `curve_fit`, for å tilpasse en funksjon til  $(dy/dx)$ 
  - Eller bruk `curve_fit` på  $y(x)$  og deriver for hånd



# BONUS: SymPy

- Symbolsk algebra med python
- Derivere, integrere osv. symbolsk
- `sympy.Symbol()`, `sympy.symbols()`
- `sympy.diff()`, `sympy.integrate()`
- `sympy.lambify()`
- `sympy.Expr.evalf()`
- `LaplaceTransform()`, `FourierTransform()` og mye, mye mer
  - NB: Det er ikke *alltid* disse klarer transformene