



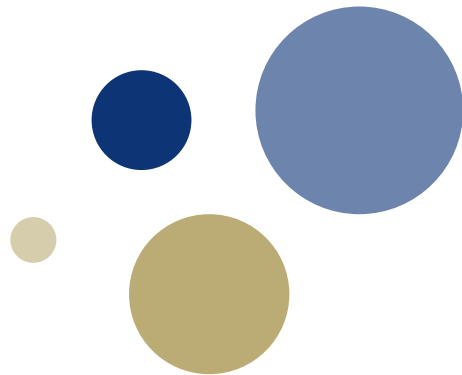
Kunnskap for en bedre verden

TDT4110 ITGK – Uke 34

Intro

Dag Olav Kjellemo

Institutt for datateknologi og informatikk (IDI)



Fagstab

- Emneansvarlig: Atle Nes
- Foreleser: Dag Olav Kjellemo
- Vit.ass. og øvingsforeleser: Viljan
- 5 “undasser” som bidrar i kulissene

I tillegg er det 70 undervisningsassistenter («studasser») som hjelper dere med øvinger og står for godkjenning av disse.



Læringsmål

Læringsutbytte: Kunnskaper

- Kan forklare grunnleggende prinsipper for digital representasjon av informasjon.
- Kan forklare virkemåten til sentrale mekanismer for prosedyreorientert programmering i Python.
- Kan forklare grunnleggende algoritmer innen programmering, samt for enkle numeriske beregninger.

Læringsmål

Læringsutbytte: Ferdigheter



- Kan løse problemer ved å skrive fungerende prosedyreorienterte programmer, og ved å komplettere kode hvor noen fragmenter mangler.
- Kan bruke relevante programmeringsverktøy til utvikling, testing og feilsøking av programkode.
- Kan forklare egen kode for andre (virkemåte, tankegang) og gi konstruktive tilbakemeldinger på andres kode.

Læringsmål

Generell kompetanse: Evne til å reflektere over bruk av programmering til beregninger knyttet til egen fagdisiplin.

I flere studieretninger er det ønske om å tidlig å ta i bruk programmering/Python. Koding gir mange ekstra muligheter for å jobbe med mange fag.

Med litt andre ord

Faglig innhold

- Prosedyreorientert programmering i Python
- En del fokus på kode for beregninger,

Se også Pensum-oversikt i Blackboard



Eksamen

Eksamen er for det meste praktisk programmering, og litt teori (Python-forståelse, numerikk)















Situasjonen nå er at vi IKKE kan kjøre kode under eksamen. Vi ser på muligheter for å kunne gjøre dette.

Hvorfor lærer vi dette?

- Nyttig både videre i studiet, og i yrkeslivet
- Programmering brukes over alt
 - Styre roboter, kontrollsystemer, maskiner, skip
 - Design og utvikling av produkter, utvinning av naturressurser
 - Vitenskapelige beregninger, simuleringer
- Norge kan ikke konkurrere på lave lønnskostnader
 - Må konkurrere på infrastruktur, smartheit, kreativitet
- Python er fleksibelt og utvidbart
 - Mange moduler for ulike behov (numerikk, plotting, AI, mm)
 - Relativt lav terskel som nybegynnerspråk
 - Ulempe: Ikke spesielt raskt
 - Men kan kalle kode i C og FORTRAN hvis nødvendig

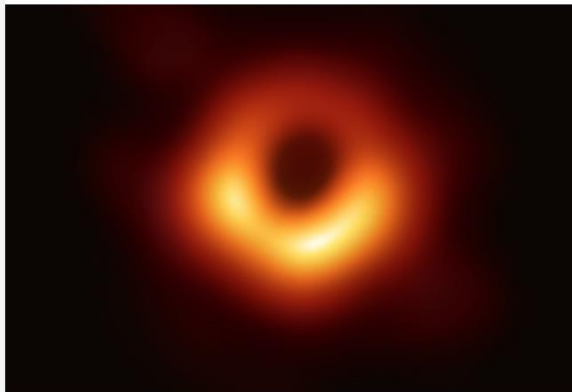


Eksempler (fra numpy.org)

Quantum Computing	Statistical Computing	Signal Processing	Image Processing	Graphs and Networks	Astronomy Processes	Cognitive Psychology
						
QuTiP PyQuil Qiskit	Pandas statsmodels Xarray Seaborn	SciPy PyWavelets python-control	Scikit-image OpenCV Mahotas	NetworkX graph-tool igraph PyGSP	AstroPy SunPy SpacePy	PsychoPy
Bioinformatics	Bayesian Inference	Mathematical Analysis	Chemistry	Geoscience	Geographic Processing	Architecture & Engineering
						
BioPython Scikit-Bio PyEnsembl ETE	PyStan PyMC3 ArviZ emcee	SciPy SymPy cvxpy FEniCS	Cantera MDAnalysis RDKit	Pangeo Simpeg ObsPy Fatiando a Terra	Shapely GeoPandas Folium	COMPAS City Energy Analyst Sverchok

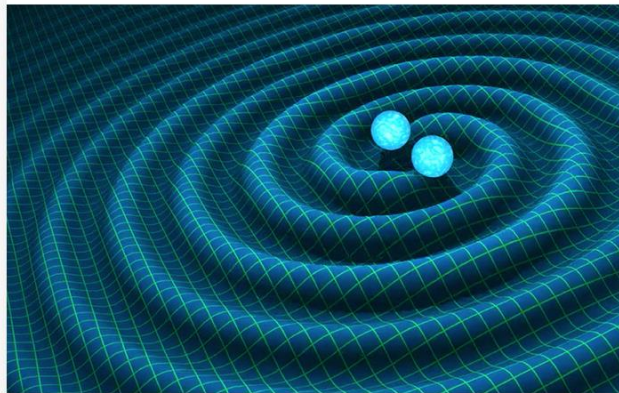
Eksempler (fra numpy.org)

FIRST IMAGE OF A BLACK HOLE



How NumPy, together with libraries like SciPy and Matplotlib that depend on NumPy, enabled the Event Horizon Telescope to produce the first ever image of a black hole

DETECTION OF GRAVITATIONAL WAVES



In 1916, Albert Einstein predicted gravitational waves; 100 years later their existence was confirmed by LIGO scientists using NumPy.

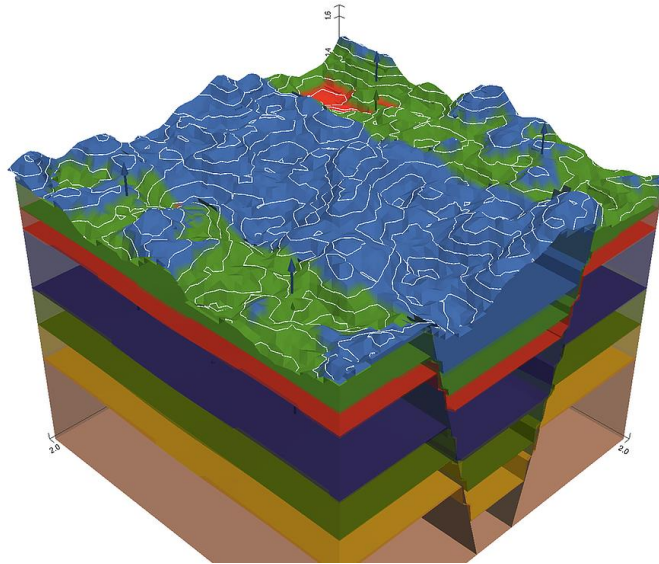
Eksempler: geologi, petroleumstek.

GemPy

Open-source 3D geological modeling

GemPy is a tool for generating **3D structural geological models** in **Python**. As such, it enables you to create complex combinations of stratigraphical and structural features such as **folds, faults, and unconformities**. It was furthermore designed to enable **probabilistic modeling** to address parameter and model uncertainties.

Best of all: GemPy is completely **free** and **open-source**!



Eksempler: kjemi (ChemPy)



Balancing stoichiometry of a chemical reaction

```
>>> from chempy import balance_stoichiometry # Main reaction in NASA's booster rockets
>>> reac, prod = balance_stoichiometry({'NH4ClO4', 'Al'}, {'Al2O3', 'HCl', 'H2O', 'N2'})
>>> from pprint import pprint
>>> pprint(dict(reac))
{'Al': 10, 'NH4ClO4': 6}
>>> pprint(dict(prod))
{'Al2O3': 5, 'H2O': 9, 'HCl': 6, 'N2': 3}
>>> from chempy import mass_fractions
>>> for fractions in map(mass_fractions, [reac, prod]):
...     pprint({k: '{0:.3g} wt%'.format(v*100) for k, v in fractions.items()})
...
{'Al': '27.7 wt%', 'NH4ClO4': '72.3 wt%'}
{'Al2O3': '52.3 wt%', 'H2O': '16.6 wt%', 'HCl': '22.4 wt%', 'N2': '8.62 wt%'}
```

Eksempler: bygg

Python for Civil Engineers

28/12/2020 by eradityanath@gmail.com, 0 Comments, in [Latest blog](#)

See Also

[Civil-Design-Engineer Jobs](#)[Civil Engineers Wanted](#)[Civil Engineering Job Openings](#)[Civil Engineering Construction](#)[Top 10 Engineering Colleges](#)[Civil Engineering Courses](#)

How can python be helpful for civil engineers? This question had raised the eagerness of every other civil engineer. Now it's time to resolve each and every uncertainty that civil engineers have in their minds.

Like every other field, applications of data science are also involved in Civil Engineering. Python in this regard is considered to be one of the most in-demand and favored programming languages. Using Python, we have seen a lot have favorable outcomes. A glimpse of these accomplishments is shown below:

Eksempler: elektro, fysikk, marin

<https://www.allaboutcircuits.com> › ... ▼ [Oversett denne siden](#)

When Can Electrical Engineers Use Python? - All About Circuits

5. mai 2019 — As an **EE**, one of the big advantages of using **Python** is controlling and automating test equipment. It's becoming more and more common to find ...

<http://www-personal.umich.edu> › c... ▼ [Oversett denne siden](#)

Computational Physics with Python - umich.edu and www ...

15. jul. 2013 — The **Python** programming language is an excellent choice for learning, teaching, or doing computational **physics**. It is a well-designed, ...

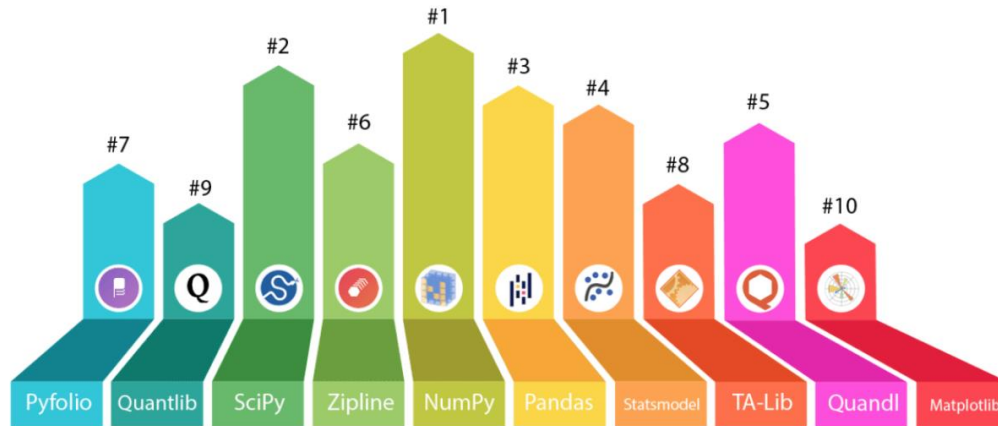
<https://www.python.org> › tribon ▼ [Oversett denne siden](#)

The Need for a Ship Building API - Python Success Stories ...

... and construction of a ship involves a high degree of concurrent engineering. ... He joined the company in 1997, after studies in **naval architecture**.

Eksempler: finans

Top 10 Python Packages For Finance And Financial Modeling



Hvordan lære dette?

- Forelesning (2x45 min): oversikt, motivasjon, gjennomgang av det viktigste Ikke alt dekkes i forelesning
- Øvingsforelesning (to paralleller, hver 2 x 45 min):
 - løsning forrige ukes oppgaver, tips til kommende øving,
- Digitale læringsressurser:
 - Videoer, Jupyter Notebook tutorials
- **Øvingsoppgaver: Programmering**
 - Viktigste aktivitet: Gjøre programmeringsoppgaver!
 - Forelesninger og ressurser bidrar med forståelse av konsept

Hvordan lære dette?

- Mange studenter sliter med å lære intro programmering
 - Globalt fenomen, mye forskning
- Hva gjør at noen lykkes, andre ikke?
 - Eks.: Liao et al. (2019)

Table 2: Summary of Differences Encountered Between Student Groups

	Higher-Performing	Lower-Performing
Exam study strategy	<ul style="list-style-type: none">• execute targeted review of areas of perceived weakness (4.1.2)• create new questions to solve (4.1.2)• review course notes	<ul style="list-style-type: none">• execute untargeted review (reviewing everything) (4.1.2)• memorize existing code (4.1.1)
Getting help	<ul style="list-style-type: none">• solicit help, successfully addressing their challenges (4.2.3, 4.2.4)	<ul style="list-style-type: none">• solicit help, only sometimes addressing challenges (ask friends for help, 4.1.3)
Addressing confusion	<ul style="list-style-type: none">• refer back to their own course notes for answers (4.1.2)• seek out extra resources (4.1.2)• double-check solutions with others (4.2.4)• keep notes on areas of confusion (4.1.2)	<ul style="list-style-type: none">• give up if cannot find answer (4.1.1, 4.2.1)• get correct answers from others, but do not seek to understand why the answers are correct (4.1.1)• through debugging, can get the code to work but may not understand why (4.2.2, 4.2.3)
Post-deadline behaviors	<ul style="list-style-type: none">• will continue working on assignments (4.1.4, 4.2.5)	–
Procrastination	<ul style="list-style-type: none">• when they do, it does not impact them significantly as they can still complete assignments (4.2.1)	<ul style="list-style-type: none">• when they do, they are unable to find help when they need it to overcome problems (4.2.1)

Hvordan jobbe med faget?

Det er viktig å *gjøre* og *reflektere* over det du gjør.

- Gjør mange oppgaver, skriv egen kode
- Bruk gjerne Copilot/ChatGPT, men tving deg selv til å prøve først selv.
- Prioriter å forstå hva koden gjør, særlig hvis du har fått hjelp (personer, ...)
 - Gjør endringer og prøv å forutse hva som skjer
 - Diskuter løsninger med andre
- Ikke vær redd å gjøre feil! Prøve og feile er en programmerers hverdag.

Referansegruppe

- FORMÅL: Kvalitetssikring av emnet
 - Tilbakemelding på pensum og læringsaktiviteter
 - Kortsiktige forbedringsforslag
 - Langsiktige forbedringsforslag
- Ikke arbeidskrevende (~4 timer totalt)
- Får attest
- Interessert? Send epost til dag.o.kjellemo@ntnu.no
 - Emne: Refgruppe ITGK
 - Oppgi navn og studieprogram
 - Skriv 1-2 setninger om hvorfor du ønsker å sitte i referansegruppa

Nødvendige ting som ikke er pensum



For å kode (effektivt), så skal vi bruke noen verktøy (programvare) som ikke er pensum per se.

Dette er programvare som vi bruker for å skrive og kjøre, og jobbe med kode, som å finne feil.

Vi skal bruke JupyterLab og VSCode. Dere kan velge hva dere vil bruke, men det er disse vil demonstrere og legge til rette for å bruke.

Utføre Python kode

VI skal se hvordan vi kan utføre Python-instruksjoner

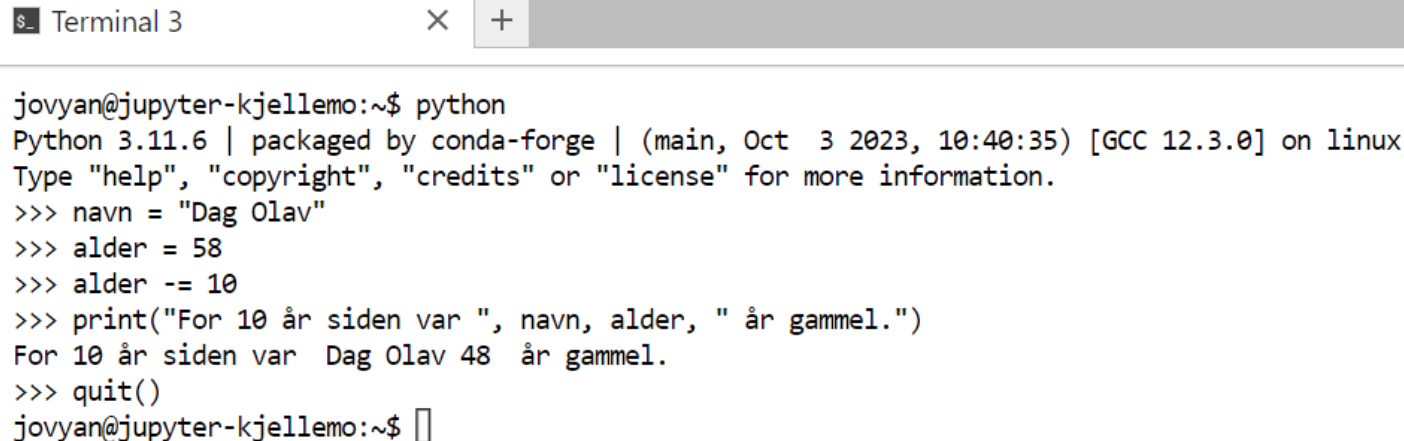
- Interaktivt, hvor kode skrives og utføres linje for linje.
- Batch, hvor vi kjører alle linjene i en tekstfil (.py) fortløpende
- Jupyter Notebook, hvor som er en slags miks av de to over

Interaktiv Python

Python tolkeren støtter en interaktiv modus.

I JupyterHub'en så start en Terminal fra "Launcher", eller velg File -> New -> Terminal.

Skriv python og trykk enter. Når du er ferdig, skriv quit() og trykk enter.



```
jovyan@jupyter-kjellemo:~$ python
Python 3.11.6 | packaged by conda-forge | (main, Oct  3 2023, 10:40:35) [GCC 12.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> navn = "Dag Olav"
>>> alder = 58
>>> alder -= 10
>>> print("For 10 år siden var ", navn, alder, " år gammel.")
For 10 år siden var  Dag Olav 48 år gammel.
>>> quit()
jovyan@jupyter-kjellemo:~$
```

Utføre kode fra .py fil fra terminalvindu

Først må vi skrive filen: Velg

File -> New -> Python file

Skriv kode og lagre filen

Fra terminalvindu, skriv

python <filnavn>

Terminalvindu brukes til
input/output.

Når koden er ferdig kjørt, så
stoppes tolkeren.

```
temperatur = int(input("Hva er  
tempen? "))
```

```
if temperatur >= 25:  
    print("Oj, varmt i dag!")  
elif temperatur < 15:  
    print("Småkjølig, ja")  
else:  
    print("Ganske passe")
```

Utføre kode fra .py fil fra IDE

Det er en fordel å bruke et egnet program som hjelper oss å skrive og kjøre kode, en såkalt IDE (Integrated Development Environment).

Vi skal demonstrere **Visual Studio Code** (ikke forveksles med Visual Studio). Det finnes også andre gode alternativer (PyCharm, Spyder)

Dette kommer vi tilbake til. Se også info på Blackboard.



JupyterNotebooks

En Jupyter Notebook er en egen type fil, som inneholder kode, tekst og også andre elementer.

Når vi jobber med en Notebook i et program som støtter dette (f.eks. JupyterLab), så kan vi velge hvilke deler av koden som skal kjøres.

I bakgrunnen startes en Python-tolker (en såkalt “kjerne” eller “kernel”).

Denne holdes aktiv mens vi jobber med Notebooken. Når vi kjører koden i en celle, så er det kun denne koden som blir sendt til tolkeren. Når denne er utført, går tolkeren i “hvilemodus”, men beholder sin tilstanden. Neste kode vi kjøres kjøres med den tilstanden som utgangspunkt.

- Øvingene kommer som Jupyter Notebooks.



Oppsummering

- Hva bør du lære/gjøre denne uka?
 - Les igjennom informasjonen på Blackboard
 - Påse at du har tilganger til JupyterHub, via lenke
 - Installer programvare på egen maskin om du ønsker det. Det vil gjøre dere uavhengig av å være online. Dere kan få hjelp på øvingsforelesningen.
- Gå til Blackboard og
 - Se videoer
 - Les / jobb deg gjennom Jupyter Notebooks som ligger der
 - Øving 1 kommer i løpet av uka.

Aktuelt stoff:

- Se på alt som ligger under uke 34 under «Undervisningsmaterielt»
 - Hvordan skrive aritmetiske uttrykk i Python
 - Variable: tilordning og bruk – hvordan virker =
 - Hvordan definere enkle funksjoner
 - (Hvordan lage enkle plott av funksjoner)
 - Enkel inn/ut av tekst med input() og print()
 - Mikse fast og variabel tekst med f-strenger
- Begynn å gjør øving 1, den blir lagt ut snart. Frist for innlevering er i uke 36.

Forkunnskapstest

Husk å gjøre forkunnskapstest:

<https://nettskjema.no/a/439505>

