## Forberedelser til workshop for Python og KI

## Veiledning med integrasjon mot eget Github repository. For viderekomne.

**Steg 1 – 5 er en éngangsoperasjon vi må gjennom for å sette opp et egnet analysemiljø.**

**Har utarbeidet en detaljert veiledning, men har forståelse for at det fortsatt kan være utfordrende for noen. Gjør ditt beste, og send meg gjerne en epost (**[**even.sannes.riiser@telemarkfylke.no**](mailto:even.sannes.riiser@telemarkfylke.no)**) hvis du står fast! 😊**

Forberedelsene innebærer følgende steg:

1. Installere nødvendig programvare
2. Sette opp editoren hvor vi skal skrive kode (med god hjelp av KI)
3. Koble oss til Github (slik at vi kan jobbe i «skyen»)
4. Forberede et «virtuelt miljø» i Python med nødvendige programvare.
5. Opprette ditt eget virtuelle miljø hvor du skal behandle og analysere data.
6. **Installasjon av programvare**
7. **Visual Studio Code (VSC)** (<https://code.visualstudio.com>)
   1. Har en integrert AI-assistent (**Github Copilot**) som hjelper deg å skrive kode raskere og enklere.
   2. Ny funksjon, **Github Copilot Agent**, tilbyr mer omfattende hjelp. En «sparringspartner» som kan diskutere metoder, kjenner hele «kodebasen» din (alle filer), kan foreslå ny eller endret kode og gjøre endringer i flere filer samtidig.

Github Copilot kan brukes gratis med et *begrenset* antall interaksjoner, dette krever en «[Free plan](https://github.com/github-copilot/signup)» (krever at du har en [Github-konto](https://github.com/signup), som er gratis).

Tips: Sjekk først om arbeidsgiver (IT) har bedriftsabonnement på Github Copilot, det er i hvertfall tilfelle i Telemark fylkeskommune. Alternativt kan du prøve [én måned](https://github.com/features/copilot/plans) med Github Copilot til 10 USD. Anbefales, men ikke et must.

1. **Anaconda** (= Python) (<https://www.anaconda.com/download>) (må oppgi epost)
2. **“Git for Windows”** (<https://gitforwindows.org>)
3. **Sette opp kode-editoren Visual Studio Code (VSC)**
4. Installere følgende utvidelser/extensions (ikonet ):

* Python
* Jupyter
* Github copilot (Logg så inn på Extension)
* indent-rainbow
* Rainbow CSV
* Material Icon Theme. Klikk “Set file icon theme” og velg “Material Icon Theme”: Et bilde som inneholder tekst, skjermbilde, Font

  Automatisk generert beskrivelse

1. **Viktig**: File --> Preferences --> Settings: Sørg for at “When pressing Shift + Enter (…)” er huket av (**enabled**)! (Tips: Søk deg fram til innstillingen i søkefeltet.)



1. **Sette opp kobling mot Github**

*GitHub sørger for sikker lagring av prosjektene dine i skyen, samtidig som det gjør csv-filer ol. tilgjengelig via en url. Dette kan være nyttig i mange sammenhenger. GitHub gjør det også enkelt å samarbeide på kode ved å spore endringer, og alle i teamet får tilgang til det samme området (repositoriet).*

1. *Hvis du ikke allerede har det*, opprett et repository på din Github-konto
   1. Logg inn på din Github (i nettleseren)
   2. Repositories --> New

Et “repository” er et **område på Github** som er tilknyttet et prosjekt. Her havner blant annet filer og mapper tilknyttet prosjektet.

1. I VSC, velg fane “Source control” () i verktøylinjen langs venstre side av skjermen, “Clone repository” og “Clone from Github”. Fyll inn brukernavn og passord for å logge inn på egen Github-konto. Da kan du velge blant egne repositories.

Et bilde som inneholder tekst, skjermbilde, Font

KI-generert innhold kan være feil.

Et bilde som inneholder tekst, skjermbilde, Font, programvare

KI-generert innhold kan være feil.

Alternativt kan du bruke url (..git) som du finner i repositoriet i nettleseren.

Et bilde som inneholder tekst, skjermbilde, Font, programvare

KI-generert innhold kan være feil.

1. En lokal kopi av repositoriet ditt åpnes nå i mappesystemet i VSC. Noter deg plasseringen (banen) til din lokale kopi av Github-repositoriet. **Tips:** Velg en av mappene på øverste nivå, høyreklikk og velg “Reveal in File Explorer”.

Eks: Mitt repository på Github heter “Telemark”, og mappen på min lokale PC er **(…)/OneDrive - Telemark fylkeskommune/Github/Telemark**

1. Sett opp følgende mappestruktur i ditt repository (filer i **rødt**, mapper i **svart**). Filer og mapper er **vedlagt denne veiledningen** som en zip-fil.

For ordens skyld: I rota ligger det altså én fil (.gitignore) og én mappe (Python).

Et bilde som inneholder tekst, Font, diagram, skjermbilde

KI-generert innhold kan være feil.

**Kjekt å vite:** Når github er satt opp vil alle endringer du gjør i kode/filer/mapper registreres, og kan synkes til repositoriet under fane “Source control” () ved å legge inn en “Message” og trykke “Commit” --> “Sync”. Eksempel:

Et bilde som inneholder tekst, skjermbilde, programvare, Multimedieprogramvare

KI-generert innhold kan være feil.

1. Filen .gitignore definerer innhold vi ikke ønsker å laste opp på Github, f. eks. filer med sensitivt innhold som nøkkelkoder, passord osv. Sjekk derfor at filen inneholder:

Et bilde som inneholder tekst, Font, skjermbilde, nummer

KI-generert innhold kan være feil.

1. Nå er et passende tidspunkt å commite/laste opp endringer til Github. Gå til “Source control” (), skriv en valgfri kommentar, og trykk “Commit” --> “Sync”. Sjekk gjerne om de nyeste endringene er oppdatert på nettsidene for ditt repository (på github.com).
2. For at et script skal kunne jobbe med filer som allerede ligger på Github (eks. “data.csv”) må vi opprette en “token” (nøkkelkode). Dette fordi brukernavn og passord ikke egner seg i kode/scripts.
   1. I Github, trykk på ditt profilbilde øverst til høyre, velg “Settings (dvs. for *konto*, ikke repository) --> Developer settings --> Personal access tokens --> Tokens (classic) --> Generate new token --> Generate new token (classic)”.
   2. Sett ønsket varighet (f. eks. evig), huk av for “repo” og copy/paste din token inn i en fil du kaller “**token.env**”.

GITHUB\_TOKEN="din\_token\_mellom\_anførselstegnene"

* 1. Legg “token.env” i mappen “Python”.

**PS:** Nøkkelen i “token.env” er sensitiv informasjon. Siden vi skrev inn “Python/token.env” i fila “.gitignore” vil Github “ignorere” nøkkelen når den overvåker filer og endringer. “token.env” blir derfor kun liggende på din lokale PC.

1. **Forberede et virtuelt Python-miljø (conda environment) hvor vi vil jobbe.**

*Environments i conda er frittstående, isolerte miljøer hvor man kan installere en spesifikk versjon av Python og diverse software-pakker. Programmer (= pakker eller libraries) i Python (eks. pandas) er ofte avhengig av andre programmer, og miljøet sørger for at avhengighetene går knirkefritt, også ved oppdateringer av enkeltprogrammer. I et environment kan man installere akkurat de programmene og verktøyene man trenger. For å sette opp et conda environment må vi:*

1. Sørge for at conda er lagt til i Windows miljøvariabel “PATH”. Her forteller vi Windows hvor den skal lete etter vår “Python interpreter”. Startmeny --> Skriv “miljøvariabler” --> Velg “Rediger miljøvariabler for kontoen din”. Under brukervariabler for din bruker, merk “Path” og velg “Rediger”. Trykk “Ny”, og lim inn bane til mappe “anaconda3”. Finnes som regel enten under

“**C:\ProgramData\anaconda3**” eller “**C:\Users\brukernavn\AppData\Local\anaconda3**”. Trykk OK x2.

1. Forberede installasjon av environment vha. fila “environment.yaml”. Denne inneholder:
   1. Alle pakker vi ønsker å installere (flere kan installeres senere).
   2. Noen viktige variabler i Python-miljøet vårt. Disse peker til nyttige mapper vi bruker ofte. Slik slipper vi å “hardcode” full filbane i scriptene hver gang vi ønsker å peke til mye brukte filer og mapper. Vi er heller ikke så sårbare for tilfeller hvor vi har behov for å flytte scripts, fordi vi slipper å endre på banene.

Åpne “environment.yaml” og endre de tre siste linjene til å matche mappestrukturen på din lokale PC (**se pkt. 3C**). **Merk doble backslasher!:** \\

**Tips: Studer innholdet i fila nøye.**

**(NB:** Hvis dere er flere som deler samme repository: Alle lager en mappe med sitt navn i mappen “Python”. Her legger man så sin “token.env” med personlig nøkkelkode, samt sin “environment.yaml” med siste 3 linjer tilpasset egen mappestruktur. Pek til korrekte baner i pkt. 3E og 5C. Ta kontakt ved spørsmål!)

1. **Sette opp et “conda environment” i Windsurf.**
2. Åpne kommandopalett i VSC ved å trykke “Ctrl” + “Shift” + “P”, skriv/velg “Python: Select interpreter”, vent evnt. noen sekunder, og velg “Python X.XX.X (‘base’)”.
3. Åpne en ny terminal (øverst på skjermen: Terminal --> New terminal). Viktig: Sørg for at det er en vanlig “Command prompt” (cmd) terminal. Hvis ikke, åpne en slik nederst til høyre:

Et bilde som inneholder tekst, skjermbilde, programvare

Automatisk generert beskrivelse

1. Du skal nå stå i mappen [ditt repository]. Åpne mappen “Python” ved å skrive “cd Python”. (Viktige kommandoer: **dir**: vis innhold, **cd mappenavn**: gå inn i mappe, **cd..** : gå tilbake)
2. Nå skal vi opprette analysemiljøet ("analyse") vi skal jobbe i, hvor alle programmer (pakker) installeres. All info vi trenger ligger i "environment.yaml". Opprettelsen kan ta noen minutter.

Skriv følgende i terminalen: **conda env create --file environment.yaml**

Trykk “Y” ved forespørsel. Vent til prosessen er fullført.

1. Gå inn i miljøet ved å skrive **conda activate analyse** i terminalen. Ønsker man å se installerte pakker, skriver man **conda list**.
2. Åpne kommandopalett i VSC ved å trykke “Ctrl” + “Shift” + “P”, skriv/velg “Python: Select interpreter”, vent evnt. noen sekunder, og velg “Python X.XX.X (‘**analyse’**)”.

**Godt jobbet, du har nå fullført oppsettet! Vi sees i Kristiansand! 😊**