Bygg Dashboard i Python

Geir Arne Hjelle / Tekna

18. november 2021

Velkommen til kurset **Bygg Dashboard i Python**. Dette dokumentet beskriver hvordan du kan installere Python, samt nødvendige ekstra pakker på din maskin, forteller litt om bakgrunnen for at Python har blitt et veldig populært programmeringsspråk, og gir deg noen små raske oppgaver.

Oppgavene er hovedsaklig ment som en sjekk på at installasjonen har fungert, samt å gi deg en liten forsmak på noe av det du vil se i kurset. Vær snill og installer Python samt jobb gjennom oppgavene **før kurset starter**. Dermed kan vi bruke kurstiden på å lære Python. Om du har noen problemer med installasjonen eller spørsmål til oppgavene så **send en e-post til kursleder Geir Arne:** `geirarne@gmail.com`

Installer Python

På python.org vil du alltid finne siste versjon av Python. Det er likevel enklere å installere en Python-distribusjon kalt Anaconda. Anaconda inneholder flere nyttige tilleggspakker og programmer:

- Python
- De mest brukte Pythonpakkene
- Conda - En pakkebehandler som brukes for å installere flere støttepakker
- Jupyter - Et miljø som kjører i nettleseren, og som er effektivt for interaktiv programmering og utforskning
- Flere editorer tilpasset Python for å skrive programmer

Anaconda er tilgjengelig for både Windows, Mac og Linux, og kan installeres uten administrative rettigheter.

- Last ned og installer **Anaconda**. Du finner detaljerte instruksjoner for installasjon på docs.anaconda.com/anaconda/install/windows

*Python er allerede installert på Mac og Linux. Du bør **ikke** bruke disse preinstallerte versjonene av Python, fordi de brukes av systemet. Det gjør det vanskelig å oppdatere til nyere versjoner, og om noe skulle gå galt er det fare for at du ødelegger for hele operativsystemet. Bruk Anaconda også på Mac og Linux.*

Etter at du har installert Anaconda er programmet **Anaconda Navigator** tilgjengelig. Du kan bruke dette programmet til å starte forskjellige programmeringsverktøy, inkludert Pythonterminaler, Jupyter og editorer.

Anaconda er en ganske tung installasjon. Om du ønsker en mindre installasjon, hvor du bare installerer pakker og verktøy ved behov, kan du installere Miniconda i stedet. Dette krever dog at du er komfortabel med å jobbe med kommandolinjen.

Oppgave 1: Hei Python

For å skrive Pythonprogrammer trenger du en **editor**. Dette er i praksis et skriveprogram hvor du kan skrive koden din og få hjelp til å utforske den videre. Det finnes mange forskjellige editorer tilgjengelige for Python, inkludert Visual Studio Code og PyCharm. I dette kurset vil vi bruke Spyder. Denne kommer installert som en del av Anaconda som du allerede har installert. Videre er Spyder en god blanding mellom å være ganske oversiktlig samtidig som den gir god støtte i programmeringen.

Start Spyder:

- Start **Anaconda Navigator**
- Finn boksen merket **Spyder**
- Klikk **Launch**

Oversikt over Spyder

Spyder består av flere områder som du kan organisere selv. I standardinstillingen ser du tre hovedområder:

1. **Editorområdet:** Til venstre. Her kan du skrive programmer og andre tekstfiler. Programmene kan du senere kjøre gang på gang.
2. **Informasjonsområdet:** Øverst til høyre. Her finner du flere faner, inkludert *Variable Explorer* for å se verdien av variable og *Plots* som viser figurer og plott du lager.
3. **Konsollet:** Nederst til høyre. Her kan du kjøre enkeltstående Pythonkode. Det er også her du vil se resultatet når du kjører programmer gjennom Spyder.

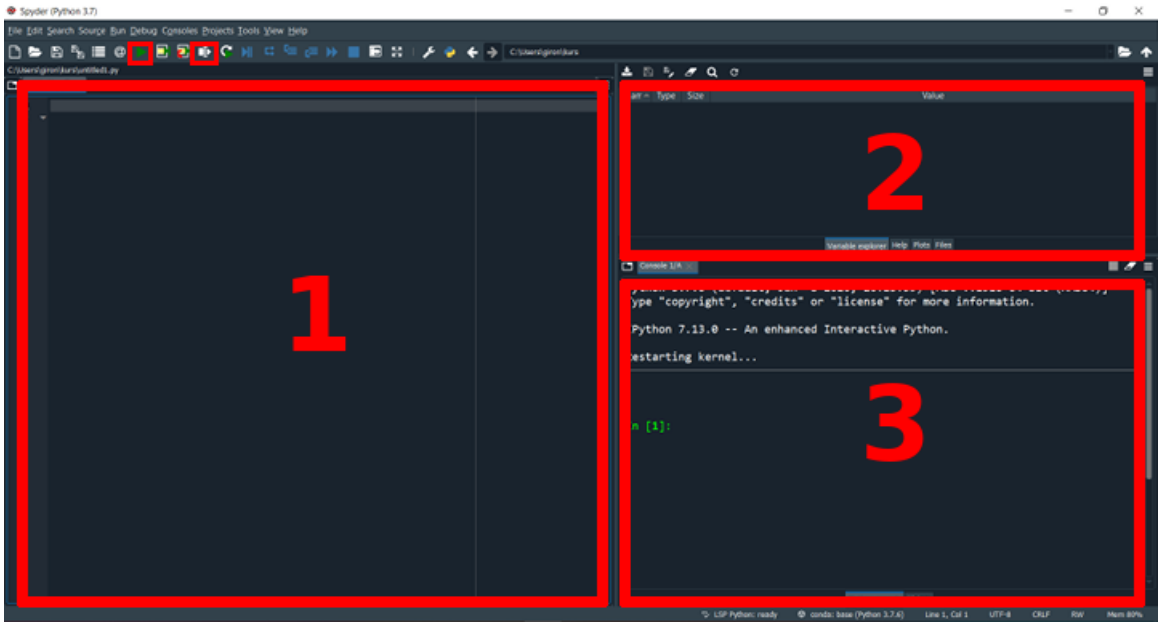

Figure 1: Spyder

Spyder har også en menylinje og en knappelinje som kan brukes til kjøring av programmer og lignende.

Din Første Kommando

Konsollet (3) kan brukes for å teste enkeltkommandoer. Du vil se noe tekst øverst i konsollet. Nederst ser du et *prompt* på formen `In [1]:` Skriv inn følgende kode etter promptet og trykk *Enter*:

Python

```
1 print("Hei Python!")
```

Terminalen skal svare med å skrive ut teksten `Hei Python!`. Hvis den ikke gjør det, men du i stedet får en feilmelding så sjekk nøye at du har skrevet alt riktig og prøv igjen. Det er spesielt viktig å stave `print` riktig, samt å bruke parantesene og fnuttene som beskrevet.

> **Merk:** I denne og andre kodesnutter i dette heftet vil du se linjenummer til venstre for koden. Disse skal du **ikke** skrive inn. De er bare brukt for å kunne referere til enkeltlinjer i kodesnuttene.

Når terminalen svarer `Hei Python!` har du skrevet og kjørt en Pythonkommando! Dette bekrefter også at installasjonen av Python fungerte.

Kort forklaring av koden:

- `print()` er en kommando som brukes for å skrive ting til terminalvinduet. Navnet på kommandoen henger igjen fra gamle dager da man ikke hadde skjerm, men måtte fysisk skrive ut resultater for å se dem.
- `"Hei Python!"` er en tekststreng. Python bruker fnutter (enten enkle, `'`, eller doble, `"`) for å markere tekststrenger. Selve teksten blir ikke tolket av Python slik at du kan skrive (nesten) hva som helst mellom fnuttene.

Trykk *pil opp*. Dette henter tilbake den forrige kommandoen du skrev. Endre tekststrengen til noe annet, for eksempel navnet ditt. Kjør kommandoen en gang til.

Interaktiv Programmering i Konsollet

Konsollet kalles av og til for en **REPL**. Dette er en forkortelse som står for **Read-Eval-Print-Loop**. Det betyr at konsollet **leser** det du skriver, **evaluerer** det som kode og **skriver ut** resultatet til skjermen. Deretter **løkker** det tilbake og begynner på nytt med neste linje du skriver. REPL'en er veldig nyttig for å teste ut små kodesnutter interaktivt, men kan ikke brukes til å skrive fullstendige programmer som skal brukes flere ganger.

- Skriv inn følgende kode i REPL. Du må først skrive linje 1 og svare på spørsmålet som dukker opp. Deretter kan du skrive inn linje 2 og se resultatet:

Python

```python
navn = input("Hva heter du? ")
print(f"Hei {navn}")
```

Kort forklaring av koden:

- `input()` er en annen kommando som brukes for å lese informasjon fra brukeren. Tekststrengen skrives til skjermen. Deretter venter `input()` på at du skriver noe og avslutter med *Enter*.
- `navn` er en variabel. Du bruker `=` for å tilordne verdier til variabler som du kan referere til senere. Her sier du at svaret på `Hva heter du?` skal refereres til som `navn`. Du kan bruke nesten hvilke ord som helst som variabler i Python.
- `print()` kjenner du allerede, den skriver noe til skjermen.

- `f"Hei {navn}"` er en spesiell tekststreng. `f` som står foran tekststrengen markerer den som en **formattert tekst**, ofte kalt en **f-string** på engelsk. Disse er nyttige fordi du kan referere til variabler ved å skrive de mellom krøllparanteser. Legg merke til at `{navn}` byttes ut med svaret du skrev på spørsmålet `Hva heter du?`

Som tidligere, sjekk at du har skrevet ting riktig om det dukker opp feilmeldinger. Eksperimenter ved å endre litt i koden. Hva skjer om du tar bort `f` i `print()` kommandoen: `print("Hei {navn}")`? Hva skjer om du skriver `navn` uten krøllparanteser: `print(f"Hei navn")`

En forskjell mellom REPL og vanlige Pythonskript (som du vil se senere) er at verdien av uttrykk automatisk skrives til skjermen:

Python

```
1 navn
```

Dette vil skrive ut teksten `navn` refererer til. Legg merke til at REPL skriver ut fnutter for å markere at `navn` er en tekststreng. `print()` skriver ikke ut de samme fnuttene.

Du kan også bruke Python som en kalkulator med tegnene `+` (pluss), `-` (minus), `*` (gange), `/` (dele) og `**` (opphøyd i):

Python

```
1 (6048 + 1729) ** 2
```

> Avslutt den kjørende REPL-sessjonen ved å skrive `exit()`. Du kan også bruke *Ctrl-D* som en snarvei. Dette vil starte konsollet på nytt, men nullstiller alle variabler du har laget.

Oppgave 2: Ditt første program

Det kan være nyttig å eksperimentere ved å skrive kode i REPL-miljøet. Men det er oftest mer nyttig å forberede programmer som lett kan kjøres om og om igjen.

Et Python-program er en ren tekstfil som inneholder Pythonkommandoer. Du kan bruke **editorområdet** (1) for å skrive slike tekstfiler. I denne oppgaven skal du skrive og kjøre et Pythonprogram.

> Slett kommentarene som allerede står i editorområdet, og skriv inn den samme koden du skrev tidligere:

Python

```
1 navn = input("Hva heter du? ")
2 print(f"Hei {navn}")
```

- Lagre programmet ditt ved å velge *File* > *Save as ...* i menyen.
- Finn en passende katalog hvor du finner igjen programmet senere. Gi programmet ditt navnet `heisann.py`. **NB:** Du bør lagre programmet på en lokal disk. Noen har opplevd små forskjeller videre i kurset om programmet er lagret for eksempel på en Sharepoint-disk.

Pythonprogrammer bruker endelsen `.py`. Du kan velge ganske fritt hvordan du navngir programmene dine, men det er lurt å unngå mellomrom (bruk understrek, _, i stedet).

- I Spyder kan du kjøre programmet ditt ved å trykke på den grønne ▷ på menylinjen. Du kan også trykke *F5* eller velge *Run* > *Run* i menyen.
- Du vil nå se spørsmålet `Hva heter du?` i konsollet nede til høyre. Skriv inn et svar og trykk `Enter`. Programmet vil nå skrive ut hilsenen direkte. Legg merke til at begge kodelinjene ble utført etter hverandre.

Når du kjører et program kjører alle kommandoene i programmet etter hverandre.

- Prøv å legge til flere kommandoer.
- Om du bare legger til et uttrykk, for eksempel en linje som kun sier `(6048 + 1729) ** 2`, vil ikke denne skrives ut av programmet ditt, selv om den ble skrevet ut i REPL-sesjonen. Du må bruke `print()` for å vise verdien av uttrykk: `print((6048 + 1729) ** 2)`.

Utforskende arbeidsflyt

Du kan bruke Spyder ganske effektivt i en utforskende arbeidsflyt hvor du utforsker dataene dine, og undersøker effekten av forskjellig kode.

- Skriv kode i editorvinduet (1). Klikk *Run file*-knappen (F5) for å kjøre all koden i konsollet (3).
- Bruk *Variable explorer* (2) og *Plots* (2) for å undersøke resultatet av koden.
- Skriv enkle kodelinjer i konsollet (3).
- Kjør deler av koden i editorvinduet (1) ved å merke den og klikke *Run selection or current line*-knappen (F9).

Vær oppmerksom på at konsollet "husker" tidligere kode som har blitt kjørt. Dette er nyttig i forhold til at du kan spare mye tid på for eksempel å slippe å lese inn data gang på gang.

Det kan likevel skape noen problemer. For å være sikker på at du ikke er avhengig av tidligere kode som har blitt slettet, bør du en gang i blant:

1. Restarte konsollet (*Consoles* > *Restart kernel*)
2. Kjøre programmet ditt på nytt (*Run file*-knappen)

Installer Dash

I kurset vil du lære hvordan du kan bygge dine egne dashboardløsninger hvor du kan utforske og presentere data, modeller og annen innsikt. Til dette skal du bruke pakken **Dash**. Dash er utviklet av et firma som heter **Plotly**, som også står bak en kraftig pakke for å lage plott og figurer. Dash er Open Source og fritt tilgjengelig.

Dash er ikke en del av standardinstallasjonen til Anaconda. Du må installere Dash som en ekstra pakke. I **Anaconda Navigator** kan du gjøre følgende:

- Klikk **Environments** i listen til venstre
- Klikk **Channels**
- Klikk **Add...** og legg til URL'en `https://conda.anaconda.org/conda-forge/`
- Trykk **Enter** og klikk **Update channels**
- Velg **All** i nedtrekksmenyen og søk etter **dash**. Du vil få en del treff. Se etter den som kun heter **dash** med en beskrivende tekst som sier noe a la "*A python framework for building reactive web-apps*"
- Marker **dash**
- Finn også raden som sier **dash-bootstrap-components** og marker denne.
- Klikk **Apply**
- Det kan ta et par minutter å installere dash og andre pakker den avhenger av, og du kan bli spurt om å godkjenne installasjonen underveis.

(Ana)conda kan ha problemer med brannmur og lignende. Om installasjonen ser ut til å stoppe helt kan det være et problem med at Anaconda ikke kommer seg på nett. Det kan hjelpe å koble seg til et mer åpent nett hvis mulig.

Du kan sjekke at installasjonen av Dash har gått bra ved å skrive følgende kode i konsollet:

Python

```python
import dash
```

`import` er en Pythonkommando som sier at du ønsker å bruke en gitt pakke, i dette tilfellet `dash`. Dersom (tilsynelatende) ingenting skjer når du kjører denne kommandoen er Dash installert, og du kan gå videre til neste oppgave.

Dersom du fikk tilbake en melding som sier `ModuleNotFoundError: No module named 'dash'` betyr det at du **ikke** fikk til å installere Dash gjennom Anaconda Navigator. Du kan da bruke et kommandovindu i stedet. Fordelen med dette er at det kan gi en litt tydeligere melding om hva som går galt:

- Åpne **Anaconda Prompt** fra startmenyen
- Skriv `conda install dash dash-bootstrap-components -c conda-forge`
- Dersom `conda`-kommandoen gir feilmelding, kan du prøve `python -m pip install dash dash-bootstrap-components` i stedet.

Oppgave 3: Lag et Enkelt Dashboard

Programmer er mer interessante når de brukes til nyttige oppgaver. I dette kurset vil du lære hvordan du bygger et dashboard. Som en enkel introduksjon skal du lage et dashboard leser input og responderer til den.

Din Første Dash-side

Dash fungerer ved at du spesifiserer utseende på dashboarded ditt i en **layout**. Deretter kan du spesifisere hvordan ting kobles sammen gjennom såkalte **callback-funksjoner**. Begynn med å skrive inn følgende kode i en ny fil i editorområdet ditt. Denne definerer en enkel layout og det er foreløpig ingen koblinger. Lagre filen som `dashboard.py`:

Python

```python
import dash
import dash_bootstrap_components as dbc

app = dash.Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP])

app.layout = dash.html.H1("Hei Dash!")

app.run_server(debug=True)
```

Lagre koden, og kjør den som tidligere. Du kan nå legge merke til at konsollet viser noe tekst men det forblir opptatt. Du får ikke tilbake et nytt prompt. Dette er fordi Dash har startet en lokal webserver på din datamaskin som serverer dashboardet som en nettside som kun er tilgjengelig for deg.

Legg merke til linjen:

```
Dash is running on http://127.0.0.1:8050/
```

Denne viser en nettadresse som du kan kopiere og åpne i nettleseren din. Du skal da se hvit side som viser teksten **Hei Dash!** øverst. Adressen `http://127.0.0.1` er ofte også kalt **localhost**.

En feature med Dash er at den automatisk laster dashboardet ditt på nytt når du gjør en endring:

- Endre teksten `"Hei Dash!"` til noe annet
- Lagre det oppdaterte programmet ditt
- Gå til nettleseren din, legg merke til at teksten ble automatisk oppdatert

Om du trenger å avslutte Dash-serveren kan du gjøre det ved å trykke på den røde firkanten øverst til høyre i konsollet. Om dette ikke har noen effekt kan du stenge konsoll-fanen ved å trykke på *x* ved siden av *Console 1/A*-teksten.

Ditt Første Dashboard

Om du fikk opp Hei Dash!-nettsiden har du allerede fått testet at installasjonen din fungerer og du har sett den grunnleggende arbeidsflyten når du jobber med Dash. Du er derfor klar til kurset hvor du vil gå mer i dybden. Denne siste oppgaven er derfor frivillig og noe du kan se på om du vil få inntrykk av noe av det som vil dekkes i kurset. Koden blir derfor ikke forklart i detalj her.

Et dashboard består av forskjellige komponenter som vises sammen. I Dash bruker du en `layout` som sier hvordan ting skal struktureres på siden. Dette er basert på HTML som er språket som definerer strukturen på alle nettsider. I praksis oversetter Dash din Pythonkode til HTML (og CSS og JavaScript) som vises på en nettside.

Komponentene kan være HTML-elementer som `html.H1` som du så ovenfor (`H1` betyr største overskrift, *heading-1*), men det er mer spennende med de interaktive komponentene som er Dash-spesifikke. Du kan se noen av disse på nettsiden `https://dash.plotly.com/dash-core-components`. I dette eksempelet bruker du bare en **Input**-komponent som er et felt du kan skrive inn tekst i.

Oppdater `dashboard.py` slik at koden blir seende slik ut:

Python

```python
import dash
from dash import html, Input, Output
import dash_bootstrap_components as dbc

app = dash.Dash(__name__, external_stylesheets=[dbc.themes.JOURNAL])

app.layout = html.Div(
    [
        html.H1("Hei Dash!"),
        html.Div(
            [
                "Hva heter du? ",
                dbc.Input(id="navn", value="", type="text")
            ]
        ),
        html.Div(id="hilsen"),
    ]
)

@app.callback(
    Output("hilsen", "children"),
    Input("navn", "value"),
)
def si_hei(navn):
    return f"Hei {navn}!"

app.run_server(debug=True)
```

Etter at Dash har oppdatert dashboardet vil du se et tekstfelt hvor du kan skrive inn tekst. Legg merke til at teksten under tekstfeltet endrer seg etterhvert som du skriver i tekstfeltet.

I kurset vil du lære mer om hvordan dette henger sammen, samt hvordan du kan ta i bruk mer spennende komponenter og koble til eksisterende data, modeller og figurer.