Flask Web Application Exercises

Her er noen oppgaver der du kan lage en enkel web-applikasjon ved hjelp av Flask. Oppgavene varierer i vanskelighetsgrad, fra de enkleste uten HTML-maler til mer avanserte oppgaver der du må sende data til serveren via POST.

Oppgave 1: Lag en enkel "Hello, World" Flask-applikasjon

Lag en Flask-applikasjon som viser meldingen "Hello, World!" når en bruker går til rot-URL-en (`/`). Dette er en grunnleggende oppgave for å bli kjent med Flask, og du trenger ikke bruke HTML-maler for denne.

Krav:

- Når brukeren går til rot-URL-en (`/`), skal meldingen "Hello, World!" vises i nettleseren.
- Flask-applikasjonen skal kunne kjøres med `debug=True` for å enklere finne eventuelle feil under utvikling.

Oppgave 2: Lag en Flask-applikasjon med flere ruter

Utvid Flask-applikasjonen fra Oppgave 1 ved å legge til flere ruter. Hver rute skal returnere en forskjellig tekst når den besøkes.

Krav:

- Legg til minst tre nye ruter, for eksempel `/about`, `/contact`, og `/info`, som viser ulik tekst når de besøkes.
- Bruk dekoratører i Flask til å knytte disse rutene til forskjellige funksjoner.

Oppgave 3: Bruk HTML-mal (Template) i Flask

Lag en Flask-applikasjon som viser en HTML-side ved hjelp av `render_template()`-funksjonen i Flask. Du skal lage en enkel HTML-fil og bruke den som mal for å vise dynamisk innhold i nettleseren.

Krav:

- Lag en mappe kalt `templates` i prosjektet ditt.
- Opprett en HTML-fil kalt `index.html` i `templates`-mappen.
- Når en bruker besøker rot-URL-en (`/`), skal Flask bruke HTML-malen til å vise en nettside i nettleseren.
- Du kan inkludere en overskrift og et avsnitt i HTML-filen.

Oppgave 4: Håndtere skjemaer i Flask (GET)

Lag en Flask-applikasjon som har et enkelt skjema hvor brukeren kan skrive inn navnet sitt. Når brukeren sender inn skjemaet, skal Flask-applikasjonen vise en personlig hilsen basert på navnet som ble sendt inn via GET-metoden.

Krav:

- Lag et skjema i HTML der brukeren kan skrive inn navnet sitt.
- Når brukeren sender inn skjemaet, skal Flask hente navnet fra URL-en og vise en hilsen som "Hei, [navn]!".
- Skjemaet skal bruke GET-metoden for å sende data til serveren.

Oppgave 5: Håndtere skjemaer i Flask (POST)

Lag en Flask-applikasjon der brukeren kan fylle ut et skjema med navn og alder. Når skjemaet sendes, skal Flask behandle informasjonen og vise en melding som inkluderer både navnet og alderen til brukeren. I denne oppgaven skal du bruke POST-metoden.

Krav:

- Lag et skjema der brukeren kan fylle ut både navn og alder.
- Når brukeren sender inn skjemaet, skal Flask hente informasjonen fra skjemaet ved hjelp av POST-metoden og vise en melding som "Hei, [navn], du er [alder] år gammel".
- Flask skal bruke POST-metoden til å sende data fra skjemaet til serveren.

Oppgave 6: Dynamiske nettsider med Flask og Jinja2

Lag en Flask-applikasjon som viser en liste over produkter på en nettside. Du skal bruke en HTML-mal med Jinja2 til å vise produktdata som kommer fra Flask-serveren.

Krav:

- Lag en liste over produkter i Flask-applikasjonen din (for eksempel en liste med navn og pris for hvert produkt).
- Bruk en HTML-mal og Jinja2-syntaks til å vise produktinformasjonen på en nettside.
- Hvert produkt skal vises som en egen rad i en tabell.
- Du kan velge om du ønsker å lagre produkter fil 'products.csv' eller om du lager egen database for dette