Øving 9: Prosjekt del 1

Læringsmål

Du skal lære å skrive et litt større program. Du skal lære å bruke versjonskontroll (Git og Github). Du skal lære å jobbe med en programmeringsoppgave i gruppe.

Godkjenning

Denne øvingen og øving 10 skal gjøres i grupper på inntil 4 studenter. Øvingen skal godkjennes ved å demonstrere det gruppa har lagd på samme vis som for de tidligere øvingene. I tillegg skal dere vise studentassistenten Github repo-et deres. Dere trenger bare å demonstrere en gang for at alle i gruppa skal få godkjent.

Prosjektoppgave: Enkel avtalebok

Dere skal implementere en enkel tekstbasert avtalebok, inkludert mulighet for å lagre avtalelista til fil og hente den fra fil. Avtaleboka skal kunne registrere avtaler med tittel, sted, start-tidspunkt og varighet

Deloppgaver

a) En av dere lager et Github repository for denne øvingen. De andre inviteres med i dette repoet.
b) Hver student i gruppa skal klone dette Github repositoryet til sin lokale datamaskin og jobbe i denne mappa med øvingen. Koden dere skriver i denne øvingen skal lastes opp til Github repositoryet.
c) Lag en fordeling av oppgaver til de ulike medlemmene i gruppa. Fordel resten av deloppgavene i øving 9 slik at alle i gruppa har noen deloppgaver de skal gjøre.
d) Lag en klasse for en avtale. En avtale skal minimum ha:
    a. En tittel som sier hva denne avtalen er (streng)
    b. Et sted (streng)
    c. Et starttidspunkt (datetime objekt, se hint nederst)
    d. En varighet i minutter (int)
    e. **Frivillig:** Andre egenskaper som for eksempel «kategori»
e) Lag en __str__ metode for avtaler som returnerer en streng som kan skrives ut med en print-setning slik at du får skrevet ut avtalen med alle egenskapene til avtalen på et leselig format for brukeren.
f) Lag en funksjon som lar brukeren skrive inn en ny avtale. Funksjonen skal bruke input-funksjonen til å lese inn egenskapene til avtalen og skal sjekke at det brukeren skriver er gyldig, for eksempel at varighet er et tall. Funksjonen skal returnere et avtale-objekt
g) Lag en funksjon som skriver ut ei liste med avtaler til skjermen. Funksjonen skal minimum skrive ut indeksen til avtalen i lista og tittel til avtalen. Den kan alternativt skrive ut indeksen til avtalen og så hele avtalen som definert i __str__ metoden. Funksjonen skal ha en frivillig parameter «overskrift» som skal være en overskrift som funksjonen skriver ut før avtalene i lista. Funksjonen skal inkludere indeksen til hver avtale i utskriften.
h) Lag en funksjon som lagrer ei liste med avtaler til ei tekstfil. Tenk over hva som vil være et fornuftig format for ei slik tekstfil.

i) Lag en funksjon som leser inn ei liste med avtaler fra ei tekstfil på samme format som dere definerte for funksjonen som skriver fila med avtaler.
j) Lag en funksjon som tar inn ei liste med avtaler og en dato og returnerer ei liste med alle avtalene som foregår på denne datoen. Funksjonen trenger bare å sjekke om datoen stemmer med dato-delen av starttidspunktet til avtalen.
k) Lag en funksjon som tar inn ei liste med avtaler og en streng, og returnerer ei liste med alle avtaler hvor tittelen inneholder strengen. Dere kan bruke find-metoden for strenger til å finne en delstreng i en større streng.
    a. **Frivillig:** Sørg for at det ikke spiller noen rolle om man bruker store eller små bokstaver i dette søket. Det gjelder både avtaletittel og søkestrengen.
    b. **Frivillig:** Lag et søk som søker etter ei liste med nøkkelord heller enn en enkelt delstreng. Funksjonen tar inn ei liste med nøkkelord (strenger). For å passe med søket må avtalen skal inneholde alle nøkkelordene, men ikke nødvendigvis i den gitte rekkefølgen eller rett etter hverandre. I menysystemet seinere, lag et menyvalg som lar brukeren kjøre et slikt søk og hvor brukeren skriver inn en serie med nøkkelord med mellomrom mellom, likt liknende et Google søk. Merk at faktiske Google søk er mye mer avanserte enn dette, måten Google-søk fungerer på er pensum i master-emnet DAT640 informasjonsgjenfinning og tekstutvinning.
l) Lag et menysystem. Når brukeren kjører scriptet skal det gi brukeren valg om å lese inn avtaler fra fil, skrive avtalene til fil, skrive inn en ny avtale, skrive ut alle avtalene, samt avslutte. Menysystemet skal skrive ut alle valgene samt hva brukeren skal skrive inn for å velge dette. Dere velger selv om dere bruker tall, enkeltbokstaver, eller lengre kommandoer for hvert menyvalg. Etter at menysystemet har skrevet ut valgene skal brukeren bli bedt om et valg med en input-setning. Menysystemet skal bruke funksjonene fra tidligere deloppgaver til å utføre valgene. Etter å ha utført det brukeren har valgt skal menysystemet spørre på nytt. Programmet skal avslutte først når brukeren velger menyvalget «avslutt».
m) Lag et menyvalg for å slette en avtale. Dette menyvalget skal skrive ut avtalelista og deretter la brukeren velge indeksen til avtalen som skal slettes og til slutt slette avtalen.
n) Lag et menyvalg for å redigere en avtale. Lag en funksjon som lar brukeren redigere en avtale. En minimumsløsning er følgende: Skriv ut avtale-lista. La brukeren velge en avtale med indeks. Skriv ut avtalen. La deretter brukeren skrive inn nye verdier for egenskapene til avtalen.
    a. **Frivillig:** Gjør denne mer avansert ved å lage et lite indre meny-system hvor brukeren velger en av egenskapene til avtalen og skriver inn ny verdi for bare den egenskapen.
o) **Frivillig:** Hvis brukeren avslutter uten å ha lagret først, spør om brukeren ønsker å lagre. Hvis brukeren skriver «ja», lagre først og deretter avslutte. Ellers avslutte. Dette krever at du har en variabel som sier om avtalene er endret siden sist lagring. Dette kan vær en boolsk verdi som settes lik True når du legger inn eller redigerer avtaler og til false når du lagrer eller leser inn. Dere kan også sjekke om det er avtaler i lista når du leser inn og spør om bekreftelse på at de skal overskrives.
p) **Frivillig, avansert:** Lag en funksjon som tar inn en avtale A og ei liste med avtaler L og finner ut om avtalen A kolliderer med noen av avtalene i L. Funksjonen skal returnere ei liste med avtaler fra L som kolliderer. To avtaler kolliderer hvis de overlapper i tid. For eksempel en avtale som starter 26.9 2022 klokka 10 og varer i 2 timer kolliderer med en avtale som starter 26.9 2022 klokka 9 og varer i 2 timer. Legg til en sjekk for kollisjoner enten i funksjonen som legger til nye avtaler eller som et eget punkt i menysystemet.

Hint

For å lagre starttidspunktet til en avtale kan dere bruke et datetime objekt. Importer modulen datatime i Python scriptet deres. Eventuelt skriv «from datetime import datetime» for å bare importere selve datetime klassen og slik slippe å skrive datetime to ganger.

Det er tre måter å lage datetime objekter på:

- Konstruktøren datetime.datetime(år, måned, dag_i_måneden, timer, minutter, sekunder). Denne tar inn år,måned og så videre som enkelttall.
- Datetime.datetime.now() Lager et nytt datetime objekt for nåværende tidspunkt basert på systemklokka i datamaskina.
- Datetime.datetime.fromisoformat(datostreng) Tar inn en datostreng på formen «ÅÅÅÅ-MM-DD TT:MM:SS» og konverterer denne til et datetime objekt. For eksempel blir 16. september 2022 klokka 13.15 til «2022-09-16 13:15:00» i dette formatet. Den aksepterer noen varianter av formatet, men er ganske fintfølende. For eksempel krever den at måned alltid har to siffer,så «2022-9-16 13:15:00» er ugyldig! Den tanker imidlertid «2022-09-16 13:15»