Oversikt

Vår implementasjon av treningsdagboken er et tekstbasert program skrevet i Python. Programmet består av to filer: hovedprogrammet, `workoutjournal.py`, og en fil med hjelpemetoder, `helper.py`. Programmet er ikke objektorientert, så det er ingen klasser. Istedenfor er alle de forskjellige funksjonalitetene separert i sine egne funksjoner.

`workoutjournal.py`

`def insert_workout(db)`

Denne metoden loggfører en treningsøkt. Brukeren blir først spurt hvor lenge treningsøkten varte, og hvor bra form og prestasjon var på en skala fra 1 til 10. Brukeren blir så spurt om to ting:

1. om den ønsker å legge til en øvelse (eller flere)
2. om den ønsker å legge til et notat

Dersom brukeren velger å legge til en øvelse legges treningsøkten inn i databasen, og `WorkoutID` lagres. Deretter kalles metoden `insert_exercise(db, wo_id)` med denne ID'en. Dette repeteres så lenge brukeren ønsker å legge til flere øvelser.

Etter at brukeren er ferdig med å legge til øvelser, kan den velge å legge til et notat. Dette gjøre ved å kalle metoden `insert_note(db, wo_id)` med den samme ID'en.

`def insert_exercise(db, wo_id: int)`

Denne metoden loggfører en øvelse. Brukeren blir først spurt om øvelsen er på apparat eller uten, og deretter navnet på øvelsen (*use-case 1*).

- dersom brukeren velger apparat, kalles metoden `insert_exerciseondevice(db, ex_id)`
- dersom brukeren velger uten, kalles metoden `insert_excercisefree(db, ex_id)`

Etter at én av disse metodene er kalt, kobles treningsøkten med id `wo_id` og øvelsen med id `ex_id` sammen med metoden `insert_exerciseinworkout(db, ex_id, wo_id)`.

Brukeren blir deretter spurt om den ønsker å legge øvelsen i en gruppe. Dersom svaret er ja, kalles metoden `insert_exercise_in_group(db, ex_id)`.

`def insert_exerciseinworkout(db, ex_id: int, wo_id: int)`

Denne metoden kobler sammen en treningsøkt og en øvelse i tabellen `ExerciseInWorkout`

`def insert_exerciseondevice(db, ex_id: int)`

Denne metoden kalles når brukeren har valgt å legge til en øvelse på apparat. Følgende skjer:

1. Alle tidligere loggførte apparater listes med navn og id.
2. Brukeren velger om den ønsker å bruke et tidligere loggført apparat, ved å skrive inn id'en, eller et nytt ved å skrive inn 0.
    - Dersom brukeren velger et nytt apparat, blir den spurt om navnet på apparatet samt en beskrivelse. Dette apparatet blir så lagt inn i tabellen `Device`
3. Brukeren blir så spurt om vekt og antall repetisjoner.

Denne informasjonen legges så inn i tabellen `ExerciseDevice`.

`def insert_exercisefree(db, ex_id: int)`

Denne metoden kalles når brukeren har valgt å legge til en øvelse uten apparat. Brukeren blir spurt om en beskrivelse av øvelsen, og dette legges inn i tabellen `ExerciseFree`.

`def insert_exercise_in_group(db, ex_id)`

Denne metoden kobler en øvelse til øvelsesgruppe. Følgende skjer:

1. Alle tidligere loggførte øvelsesgrupper listes med navn og id.
2. Brukeren velger om den ønsker å bruke en tidligere loggført gruppe, ved å skrive inn id'en, eller en ny, ved skrive 0.
    - Dersom brukeren velger å lage en ny gruppe, blir brukeren spurt om navnet på gruppen. Hvis brukeren skriver inn et gruppenavn som allerede er loggført, blir øvelsen lagt inn i den eksisterende gruppen.
3. Øvelsen legges inn i den valgte gruppen.

`def insert_note(db, wo_id: int)`

Denne metoden loggfører et notat tilhørende en treningsøkt. Brukeren blir spurt om målet med treningsøkten og refleksjoner. Dette legges inn i tabellen `ExerciseNote`.

`def get_exercises(db, wo_id: int) -> list`

Denne metoden gir en liste med øvelser som tilhører treningsøkten med id `wo_id`.

`def get_note(db, wo_id: int)`

Denne metoden henter notatet tilhørende treningsøkten med id `wo_id`.

`def delete_workout(db)`

Denne metoden sletter en treningsøkt fra databasen. Alle treningsøkter listes med id og tidspunkt. Brukeren blir så spurt om hvilken treningsøkt den ønsker å slette, ved å skrive inn id'en. Brukeren blir så spurt om den er helt sikker, og treningsøkten slettes dersom den sier ja.

`def list_workouts(db)`

Denne metoden lister brukerens treningsøkter, med tilhørende øvelser og notat (*use-case 2*). Brukeren spørres først hvor mange, *n*, av de siste øvelsene den ønsker å se. De *n* siste treningsøktene hentes så ut fra databasen. For hver av treningsøktene hentes tilhørende øvelser og notat, og disse skrives ut dersom de eksisterer. Dersom ingen treningsøkter er loggført, informeres brukeren om dette.

`def list_devices(db)`

Denne metoden lister brukerens mest brukte apparater (*use-case 5*). Apparatene listes med navn, og antall ganger brukt, sortert fra mest til minst.

`def list_groups(db)`

Denne metoden lister brukerens loggførte øvelsesgrupper (*use-case 4*). Hver av gruppene listes med navn og id. Brukeren kan så se alle øvelsene i en gruppe ved å skrive inn id'en.

`def list_exercise_results(db)`

Denne metoden gir brukeren informasjon om resultatet for en øvelse i et gitt tidsintervall (*use-case 3*). Øvelsene har ikke noe eget mål på prestasjon, men treningsøktene som øvelsen inngår i har det. Brukeren velger en øvelse og et tidsintervall, og får se hvordan formen og prestasjonen var på treningsøktene hvor øvelsen ble utført, i det gitte tidsintervallet.

`def choose_action(db)`

Denne metoden fungerer som en meny. Brukeren får følgende valg:

0. Exit
1. Show workouts
2. Insert a workout
3. Delete a workout
4. Show most used devices
5. Show excercise groups

6. List exercise results

Hver av valgene 1 til 6 er knyttet til sin tilhørende funksjon:

- Show workouts → `list_workouts`
- Insert a workout → `insert_workout`
- Delete a wokrout → `delete_workout`
- Show most used devices → `list_devices`
- Show exercise groups → `list_groups`
- List exercise results → `list_exercise_results`

Når brukeren velger et tall, kalles den tilhørende funksjonen.

`def main()`

Hovedprogrammet. Brukeren blir først møtt av en innlogging, hvor den må skrive inn MySQL-brukernavn og passord. Dersom det er første gang brukeren bruker programmet, lages alle tabellen med metoden `execute_script(db, filename)` fra `helper.py` , som kjører SQL-skriptet `maketables.sql`. Metoden `choose_action` blir så kalt så lenge brukeren ikke skriver inn 0 (Exit).

`helper.py`

`def wrap_indent(text, amount, first=' ', ch=' ')`

Denne metoden brukes for å skrive ut lang tekst, som for eksempel treningsnotater. Teksten indenteres med `amount`, og hver linje wrappes ved lengde `terminal_width()- 20`, slik at det er lesbart uavhengig av bredden på terminalen som programmet kjøres i.

`def int_parse(text, default=0)`

Denne metoden brukes for å verifisere at brukerinput er et tall. Dersom det ikke er det, returneres en default verdi.

`def str_parse(text, selection, default)`

Samme som `int_parse`. Brukerinput sammenlignes mot en ønsket type (`selection`), og en default verdi returneres dersom det ikke er en match.

`def date_parse(text)`

Sjekker at brukerinput for en dato er på formen `yyyy-mm-dd`.

`def time_parse(text)`

Sjekker at brukerinput for et tidspunkt er på formen `hh:mm`.

`def terminal_width()`

Denne metoden finner bredden av terminalen som programmet kjøres i. Brukes i `wrap_indent`, samt for å printe separasjonslinjer.

`def print_menu(menu)`

En hjelpemetode for å printe menyen (`choose_action`) i to kolonner.

`def execute_script(db, filename)`

Utfører SQL-skriptet `filename`. Brukes for å opprette alle tabeller ved å kalle funksjonen med `maketables.sql`-skriptet fra innlevering 1.