IN1000 Innlevering 4

Introduksjon

Innleveringen består av 4 oppgaver. Vær oppmerksom på at innleveringen kan kreve noe mer arbeid enn de du har løst så langt, så sett av nok tid. Les gjennom hver oppgave før du begynner å programmere, og forsøk gjerne å løse oppgavene på papir først! Hvis du sitter fast på en oppgave bør du prøve å løse øvingsoppgavene i Trix (se lenke under hver oppgave) før du spør om hjelp.

Pass på at oppgavene du leverer ligger i riktig navngitte filer, som vist under oppgavetittelen. For hvert program du skriver skal du legge ved en kommentar i toppen av fila som forklarer hva programmet gjør. Videre forventes det at du kommenterer koden underveis, så det blir tydelig hva du har tenkt. Andre viktige krav til innleveringen og beskrivelse av hvordan du leverer finner du nederst i dette dokumentet.

Læringsmål

I denne innleveringen skal du vise at du kan løse problemer med programmerings-verktøy, og du skal kunne bruke løkker, både alene og i kombinasjon med lister, for å gjøre beregninger på en effektiv måte.

Oppgave 1: Regning med løkker

Filnavn: *regnelokke.py*

1. Lag et program som bruker en while-løkke for å lese inn tall fra brukeren helt til brukeren taster 0, uten å gjøre noe mer med tallene.

2. Utvid programmet ved å lage en tom liste før while-løkken. Deretter skal du endre løkken slik at den for hver gjennomkjøring lagrer tallet brukeren taster inn i denne listen.

3. Etter at while-løkken er ferdig, skriv en for-løkke som går gjennom lista og skriver ut hvert enkelt element.

4. Utvid programmet med en variabel *min_sum*. Skriv så en ny for-løkke som går gjennom listen du lagde tidligere, og legger til verdien av hvert tall i *min_sum*. (**Hint:** +=). Skriv deretter ut summen til brukeren.

5. Bruk to separate for-løkker til å finne henholdsvis det minste og det største elementet i lista, og skriv ut disse. Her skal du komme frem til det minste og største tallet i listen uten å bruke *min()* eller *max()*.

Syntes du denne oppgaven var vanskelig? Se Trix-oppgave 4.01, 4.06, 4.09 og/eller se gjennom undervisningsmodulene Løkker, For-løkker og Kombinere løkker og samlinger. Syntes du denne oppgaven var enkel? Se Trix-oppgave 4.15.

Oppgave 2: Reiseplan

Filnavn: *reiseplan.py*

I denne oppgaven skal du lage et program som lar en bruker legge planer for en reise. Dette skal du gjøre ved hjelp av en *nøstet* liste, det vil si en liste av lister.

1. Lag en tom liste *steder*, og gi brukeren mulighet til å fylle den med 5 reisemål ved hjelp av en for-løkke.

2. Lag to lister til ved navn *klesplagg* og *avreisedatoer*, og la brukeren fylle inn disse på samme måte, med fem elementer i hver liste.

3. Nå skal du lage en liste som kan inneholde de andre listene du har skrevet. Opprett en liste *reiseplan*, og legg til *steder*, *klesplagg* og *avreisedatoer* i lista.

4. Bruk en enkel for-løkke for å skrive ut innholdet i *reiseplan*, og se at det skrives ut tre lister med hvert sitt innhold.

5. Nå skal du la brukeren oppgi en plass i *reiseplan* og skrive ut elementet på den gitte posisjonen. For å gjøre dette, kan du følge stegene under:
    a. Først henter du inn en indeks *liste_indeks1* som representerer en av de tre listene i *reiseplan*, der gyldig input vil være mellom 0 og 2 (lengden til *reiseplan* minus 1).

    b. Deretter en indeks *liste_indeks2* som representerer et av de fem elementene i den valgte listen, der gyldig input vil være mellom 0 og 4 (lengden til listen minus 1).

    c. Bruk en if-sjekk til å teste at de to tallene brukeren har oppgitt er gyldige

verdier (med andre ord: om indeksene er innenfor den tillatte rekkevidden). Dersom tallene er mulige plasser i listene skal de brukes til å skrive ut *reiseplan*[*liste_indeks1*][*liste_indeks2*]. Hvis tallene ikke er gyldige plasseringer skal du skrive ut "Ugyldig input!"

Syntes du denne oppgaven var vanskelig? Se Trix-oppgave 3.10, 4.08, 4.10 og/eller se gjennom undervisningsmodulene For-løkker og Nøstede samlinger. Syntes du denne oppgaven var enkel? Se Trix-oppgave 4.16.

Oppgave 3: Løkker og lister

Filnavn: *lokker_og_lister.py*

1. Skriv et program som ved hjelp av en while-løkke lager en liste som inneholder tallene fra 0 til 9.

2. Skriv et program som ved hjelp av en for-løkke lager en liste som inneholder tallene fra 0 til 9.

3. I forelesningen for uke 4 ble det forklart at while-løkker er en fleksibel måte å iterere på, som kan brukes for alle formål som krever repetisjon/iterasjon. Løsningen på oppgave 3.1 er et eksempel på dette. I samme forelesning ble det forklart at for-løkker er mye mer spesialiserte, og kun kan brukes til å gå gjennom (iterere over) hvert element av en samling.

4. I oppgave 3.2 ble du bedt om å lage en samling ved hjelp av en for-løkke. Men en for-løkke forutsetter jo at du allerede har en slags samling. Hvordan kan dette ha seg? Forklar kort, ved å inkludere følgende punkter i forklaringen din:
   a. Hvor i koden din i oppgave 3.2 er det at du i tilfelle har en samling?
   b. Og hvilke elementer (tall) inneholder denne samlingen som du bruker i din for-løkke sammenlignet med samlingen (listen) som du lager? (Hint: skriv "list(range(10))" i Python-tolken og se hva du får).
5. Skriv et program som først lager en tom liste **mine_tall**, og deretter ved hjelp av en while-løkke legger tallet 0, tallet 3, tallet 6 osv. til i listen (3 høyere hver gang) frem til tallet blir høyere enn 20 (18 skal være det siste tallet i listen).

6. I programmet fra oppgave 3.5, legg til en for-løkke som går gjennom (itererer over) **mine_tall** og skriver ut hvert av tallene på en egen linje.

7. Skriv videre på programmet fra oppgave 3.5, ved å legge til en ny for-løkke som skriver ut hver av de *gyldige indeksene* for listen **mine_tall** (altså 0, 1, 2, o.s.v. frem til siste indeks i lista). Vi er altså ikke ute etter elementene i lista, men de gyldige indeksene, og du må velge en for-løkke-variant som passer til formålet.

8. Fortsett å skrive videre på programmet fra 3.5, ved å bruke enda en for-løkke til å *endre* **mine_tall**, slik at hvert tall blir 10 ganger større enn det var tidligere (altså 0, 30, 60, o.s.v). Vi er altså ute etter å endre elementene i lista, og du må velge en for-løkke-variant som passer til formålet.

9. Skriv ut listen **mine_tall** etter at den er blitt endret av koden du skrev i oppgave 3.8. Sjekk at du får ut 0, 30, 60, o.s.v.

10. Måtte du i oppgave 3.7 la for-løkken gå gjennom (iterere over) selve tallene i listen **mine_tall** eller gå gjennom (iterere over) de gyldige indeksene for listen **mine_tall**? Forklart kort hvorfor/hvorfor ikke, og inkluder følgende punkter i forklaringen:
    a. Hvorfor fungerer det bare på én av måtene?
    b. Hva ser man i oppgave 3.8 dersom man i oppgave 3.7 valgte feil strategi for hva for-løkken gikk gjennom (itererte over)?

Hint: husk at vi på forelesning snakket om to varianter av for-løkker, og at det bare er den ene varianten vi kan bruke når vi skal endre verdier.

Oppgave 4: Egen oppgave

1. Skriv oppgavetekst til en oppgave som handler om løkker og samlinger. Et forslag er å programmere et system som lar bruker holde styr på, legge til og skrive ut venners bursdager.

2. Løs oppgaven! Du skal levere både oppgaveteksten og besvarelsen (skriv oppgaveteksten som kommentarer over løsningen din).

Krav til innlevering

- Oppgaven må kunne kjøres på din maskin. Test dette før du leverer!
- Kun *.py*-filene skal leveres inn.
- Spørsmålene til innleveringen skal besvares i kommentarfeltet.
- Koden skal inneholde gode kommentarer som forklarer hva programmet gjør og

kommentarer som forklarer hvordan du har tenkt underveis.

- Programmet skal ha gode utskriftssetninger som gjør det enkelt for bruker å forstå.

Hvordan levere oppgaven

1. Svar på spørsmålene under i kommentarfeltet i Devilry. De **skal** besvares.
    a. Hvordan synes du innleveringen var? Hva var enkelt og hva var vanskelig?
    b. Hvor lang tid (ca.) brukte du på innleveringen?
    Var det noen oppgaver du ikke fikk til? Hvis ja:
        i. Hvilke(n) oppave(r) er det som ikke fungerer i innleveringen?
        ii. Hvorfor tror du at oppgaven ikke fungerer?
        iii. Hva ville du gjort for å få oppgaven til å fungere hvis du hadde mer tid?

2. Logg inn på Devilry.

3. Lever alle *.py*-filene, og husk å svare på spørsmålene i kommentarfeltet.

4. Husk å trykke på *Add delivery or question*. Sjekk deretter at innleveringen din er komplett. Du kan levere så mange ganger du vil frem til fristen. Det er ikke mulig å slette innleveringer i Devilry. *Alle* filer må være med hver gang du leverer, og filene må ha samme navn som oppgitt i oppgaveteksten. Retteren din vil gi tilbakemelding på kun den siste innleveringen, og derfor er det viktig at alle filer er med i siste innlevering.

5. Innleveringen gir deg mulighet for en konstruktiv tilbakemelding på hva du har fått til, i tillegg til forbedringer/misforståelser og om du er omtrent i rute med faget. De fleste vil ha behov for å løse mange flere oppgaver enn de du finner i denne innleveringen. Det får du anledning til i gruppetimene og i Trix. Du finner Trix-oppgaver for denne uken her. Læreboken inneholder også mange oppgaver (noen går litt utenfor vårt pensum). Du finner dessuten mye på nett - og ikke minst kan du finne på egne programmer og utvidelser til oppgaver.