IN1000: Obligatorisk innlevering 4

Frist for innlevering: 15.9.26, kl. 23:59 **Sist endret:** 9.9.26

Introduksjon

Det er kun quiz som må godkjennes denne uken, men programmeringsoppgavene er likevel nødvendige både for trening og som et mål på om du følger progresjonen i emnet. Ved å levere løsningene dine får du tilbakemeldinger som hjelper deg videre.

Oppgavene (og neste ukes undervisning) forutsetter at du har jobbet med ukens pensum til du har god kontroll på stoffet. Oversikt over undervisningen i IN1000 inkludert ekstratilbud finner du her. Undervisningsmateriale for hver uke finner du her eller fra Timeplanen (under Forelesninger).

For å få godkjent obligatoriske innleveringer må du innen fristen:

1. Besvare denne quizen i Nettskjema. Det er ingen krav til antall rette.
2. I Devilry (se nederst):
    a. Levere skjermbilde (.jpg) av svarsiden i Nettskjema
    b. Levere filer som angitt i oppgaver merket **obligatorisk** nedenfor (0 eller flere). Merk at koden må kjøre uten feil, eventuelt må du beskrive feilen og hva du tror kan hjelpe (se under "Hvordan levere i Devilry")

Har innleveringen mindre feil eller mangler i obligatoriske oppgaver kan du få et (1) nytt forsøk - dette vil ha kort frist (3 virkedager) for innlevering, så sjekk e-post for tilbakemelding hver dag.

Les gjennom hver oppgave før du begynner å programmere, og forsøk gjerne å løse oppgavene på papir først (da kan det være lettere å konsentrere seg om en fremgangsmåte som gradvis beskrives i Python).

For hvert program du skriver, skal du legge inn en kommentar i toppen av filen som forklarer hva programmet gjør. Videre forventes det at du kommenterer koden underveis, så det blir tydelig hva du har tenkt. Dersom oppgaven sier du skal endre programmet, trenger du kun å levere den endrede løsningen. 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 leser inn tall fra brukeren helt til brukeren taster 0, uten å gjøre noe mer med tallene.
2. Utvid programmet slik at alle tallene brukeren taster inn lagres i en samling.
3. Når brukeren er ferdig med å legge inn tall, skriv da alle tallene ut ett etter ett ved å iterere over samlingen.

4. La programmet regne ut summen av alle tallene brukeren har skrevet inn (uten å bruke sum()-funksjonen).
5. Finn nå samlingens minste og største tall, og skriv disse ut i terminalen (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 i uke 4. 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. Brukeren skal få mulighet til å legge inn informasjon om flere reisemål, hva de skal pakke til hvert sted, og når de skal reise. Problemet må løses på en måte som er oversiktlig og gjør det lett for brukeren å hente ut informasjon etterpå.

1. Be brukeren om å legge inn informasjon for fem ulike reiser. For hver reise trenger vi navnet på reisemålet, hvilke type klær som må pakkes og avreisedatoen. Tenk over hvordan du lagrer denne informasjonen (gjennom bruk av samlinger) på en effektiv og ryddig måte. Det skal være mulig å lagre alle reisedetaljene samlet, men også hente ut informasjon om én spesifikk del av én reise ved hjelp av indekser.
2. Bruk en enkel for-løkke for å skrive ut informasjonen for hver reise. .
3. Nå skal du la brukeren velge hva de ønsker å se ved å oppgi to tall – ett for hvilken av de fem reiser, og ett for hvilken av de 3 kategoriene (reisemål, klær, dato). Håndter situasjoner der brukeren gir ugyldige tall på en god måte. Skriv ellers ut den valgte informasjonen.
4. Prøv da om du kan gjøre programmet enda mer brukervennlig ved å bruke navn på kategoriene i stedet for tall. Hvordan finner du riktig informasjon når brukeren spør etter klær til reise 2?
5. Litt vanskeligere: Brukeren har ombestemt seg! Utvid nå programmet ditt slik at brukeren kan endre en allerede eksisterende reiseplan. Kanskje avreisedatoen for turen til Frankrike blir utsatt?

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 i uke 4.
Syntes du denne oppgaven var enkel? Se Trix-oppgave 4.16.

Oppgave 3: Løkker og lister

Filnavn: *lokker_og_lister.py*

I denne oppgaven skal du bruke både `while`- og `for`-løkker til å løse problemer. Du må selv velge hvordan du bygger opp løsningene, og tenke gjennom hvilken løkke som passer best i ulike situasjoner.

1. Skriv et program som fyller en tom samling gradvis med tall. Disse tallene må starte på 0 og øke med 1 for hvert steg, inntil du har ti tall. Bruk den løkketypen som passer best når du ikke kjenner antall nødvendige steg på

forhånd.
2. Lag nå samme samling, men bruk denne gangen en annen løkketype. Denne typen må være en type som passer godt når du kjenner antall gjentagelser.
3. Hva er forskjellen i hvordan du måtte tenke og skrive koden for de to løkke-typer?
4. En for-løkke forutsetter 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 finnes det allerede en samling når man bruker en for-løkke?
    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 nå et program, som denne gangen legger til tall som øker med 3 for hvert steg, så lenge tallet ikke går over 20. Velg en løkke-type som gir deg god kontroll underveis i iterasjonen.
6. Gå gjennom tallene du har samlet i punkt 5 og skriv dem ut én etter én. Hvilken løkke-type velger du nå?
7. Nå skal du ikke skrive ut tallene, men i stedet posisjonene (indeksene) i samlingen. Hvilken form for for-løkke er best her og hvorfor? (er det forøvrig noen valg av samling i punkt 5 som ville skape problemer når man her skal skrive ut posisjonene?)
8. Fortsett videre med samme kode for å endre innholdet i samlingen fra punkt 5 slik at alle tallene ganges med 10. Hva kan skje dersom man her velger feil strategi for hva for-løkken går gjennom (itererer over)?
9. Skriv ut listen etter den er blitt endret i punkt 8 og sjekk at du får riktige 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

- Programmer som leveres inn må kjøre uten feil. Om du har feil du ikke selv finner ut av kan du få hjelp i gruppetimer, på Fredagspython eller i Discourse. Hvis du leverer programmer med feil, må dette kommenteres i Devilry (se 4.c nedenfor).
- Koden din skal inneholde gode kommentarer som forklarer hva programmene gjør.
- Programmene skal ha gode utskriftsetninger som er enkle å forstå for bruker.

Hvordan levere i Devilry

1. Du skal IKKE lage en .zip-fil

2. Logg inn i Devilry.
3. Last opp alle filer som angitt i besvarte oppgaver.
4. Du **skal** også besvare noen spørsmål i kommentarfeltet i Devilry:
    a. Hvordan syntes du innleveringen var? Hva var enkelt og hva var vanskelig?
    b. Hvor lang tid (ca.) brukte du på innleveringen?
    c. Var det noen oppgaver du ikke fikk til? Hvis ja:
        i. For hvilke(n) oppgave(r) fungerer ikke det innleverte programmet ditt?
        ii. Hvorfor tror du programmet ikke fungerer?
        iii. Hva ville du gjort for å få programmet til å fungere hvis du hadde mer tid?
5. Husk å trykke **«Legg til innlevering eller spørsmål»** og sjekk deretter at innleveringen din er komplett. Du kan levere så mange ganger du vil frem til fristen, men **siste innlevering må inneholde alle filer som skal vurderes**.
6. Innleveringen med tilhørende Trix-oppgaver er minimum av hva du bør ha programmert i løpet av en uke - de fleste vil trenge en del mer trening for å få god kontroll på stoffet. Du finner flere oppgaver for denne uken her.

Finner du feil i oppgaveteksten, send en epost til siriamj@uio.no.