IN1000 Obligatorisk innlevering 4

Sist endret: 23.02.23

Introduksjon

Innleveringen består av 5 oppgaver og de gir 1 poeng hver. Vær obs 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 oppgaven skal du vise at du kan løse problemer med programmeringsverktøy, og du skal kunne bruke løkker, både alene og i kombinasjon med lister, for å gjøre beregninger på en effektiv måte. Du skal også kunne skrive funksjoner som tar imot, manipulerer og returnerer data.

Oppgave 1: Parametere og returverdier

Filnavn: *funksjoner.py*

1. Definer en funksjon `adder(tall1, tall2)` som tar to heltall som parametere. Funksjonen skal returnere summen av parameterne. Kall på funksjonen to ganger med argumenter du selv velger, og skriv ut resultatene med en passende tekst.

2. Be en bruker skrive inn en tekststreng, og deretter en bokstav. Lag en **loop** som teller opp hvor mange ganger den oppgitte bokstaven forekommer i den oppgitte tekststrengen. Skriv ut resultatet med en passende tekst (stor og liten bokstav regnes som ulike - for eksempel vil det si at det er 0 forekomster av "*E*" i ordet *"hei"*).

3. Skriv en funksjon `tellForekomst(minTekst, minBokstav)`. Funksjonen skal telle hvor mange ganger en bokstav `minBokstav` forekommer i teksten `minTekst`, og returnere dette tallet. Skriv deretter om deloppgave 2 til å benytte denne funksjonen, men sørg for at programmet fremstår helt likt for brukeren når det kjøres. Lever programmet slik det ser ut etter at du har gjort denne endringen (du trenger altså ikke å levere to versjoner av programmet).

Synes du denne oppgaven var vanskelig? Se Trix-oppgave 4.02, 4.05, og/eller se gjennom undervisningsmodulene Parametre i prosedyrer og Retur-verdier
Synes du denne oppgaven var enkel? Se Trix-oppgave 4.12, 4.16

Oppgave 2: 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. Listen skal ikke inneholde 0.

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 `minSum`. Skriv så en ny for-løkke som går gjennom listen du lagde tidligere, og legger til verdien av hvert tall i `minSum`. Hint: `+=` Skriv deretter ut summen til brukeren med passende tekst.

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()`.

Synes 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.
Synes du denne oppgaven var enkel? Se Trix-oppgave 4.15, 4.20, 4.22

Oppgave 3: 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 fem reisemål ved hjelp av en for-løkke.

2. Lag to nye lister `klesplagg` og `avreisedatoer`, og la brukeren fylle inn disse på samme måte, med fem elementer i hver liste.

3. Lag en ny liste `reiseplan` som skal inneholde de tre listene du har definert over.

4. Bruk en for-løkke for å skrive ut innholdet i `reiseplan`, slik at det skrives ut en liste per linje.

5. Programmet skal nå la brukeren velge et sted, et klesplagg eller en avreisedato fra listen `reiseplan`.

a) Be brukeren om to indeksverdier, `index1` og `index2`.
   `index1` lar brukeren velge en av de tre listene. `index2` lar brukeren velge et av de fem elementene.
b) Bruk en if-test til å sjekke om indeksene, `index1` og `index2`, ligger innenfor listene. Skriv ut elementet `reiseplan[index1][index2]` dersom det eksisterer, ellers skal programmet skrive ut “Ugyldig indeksverdi!”.

Synes du denne oppgaven var vanskelig? Se Trix-oppgave 3.10, 4.08, 4.10 og/eller se gjennom undervisningsmodulene For-løkker og Nøstede samlinger
Synes du denne oppgaven var enkel? Se Trix-oppgave 4.21, 4.23

Oppgave 4: Å telle bokstaver og ord

Filnavn: *ordtelling.py*

I denne oppgaven skal du definere funksjoner som utfører operasjoner på strenger.
Det kan være lurt å bruke string-metoden '`split()`' i noen av funksjonene, illustrert under:

```
sentence = "hei hei du"
words = sentence.split(" ") → ["hei", "hei", "du"]
```

1. Definer en funksjon som tar inn et ord som parameter. Funksjonen skal returnere antall bokstaver ordet består av.

2. Definer en funksjon som tar inn en setning som parameter. Funksjonen skal returnere antall ord setningen består av. Man anta at hvert ord er adskilt med mellomrom. Hint: `split()`

3. Definer en funksjon som tar inn en setning som parameter. Funksjonen skal returnere ei ordbok hvor hvert (unike) ord i setningen er nøkkelverdien, og innholdsverdien er antall forekomster av ordet. Du skal komme frem til antallet uten å bruke *`count()`*. Hint: `split()`

4. Lag et program som ber en bruker skrive inn en setning. Skriv deretter ut hvor mange ord det er i setningen, hvor mange ganger hvert unike ord forekommer, og hvor mange bokstaver hvert ord består av. Funksjonene du har definert skal brukes.

Skilletegn, som '!', '.' og ',', etc. kan skilles med mellomrom, slik at de kan behandles som ord (slik vist nedenfor).

Eksempel på utskrift:

```
>>>Skriv en setning: >>>Jeg liker kake , jeg .
>>>Det er 6 ord i setningen din.
>>>Ordet 'jeg' forekommer 2 ganger, og har 3 bokstaver
>>>Ordet 'liker' forekommer 1 gang, og har 5 bokstaver
>>>Ordet 'kake' forekommer 1 gang, og har 4 bokstaver
>>>Ordet ',' forekommer 1 gang, og har 1 bokstav
>>>Ordet '.' forekommer 1 gang, og har 1 bokstav
```

Synes du denne oppgaven var vanskelig? Se Trix-oppgave 3.02, 4.04, og avhengig av hva du står fast på se gjerne gjennom undervisningsmodulene Objekter tilbyr tjenester, Parametre i prosedyrer, Retur-verdier,Kombinere løkker og samlinge eller Ordbøker
Synes du denne oppgaven var enkel? Se Trix-oppgave 4.14, 4.22 og 4.24

Oppgave 5: Egen oppgave

1. Skriv en oppgavetekst til en oppgave som benytter løkker og lister. Implementasjonen skal inneholde minst ei liste, og den må inkludere at brukeren både skal kunne legge til og fjerne elementer fra lista/listene. Minst en av deloppgavene må kreve bruk av løkker.

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

Krav til koden og filene

- Kun *.py*-filene skal leveres inn.

- Koden skal inneholde gode kommentarer som forklarer hva programmet gjør.

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

Den obligatoriske innleveringen er minimum av hva du bør ha programmert i løpet av en uke. Du finner flere oppgaver for denne uken her.