IN1000 Obligatorisk innlevering 4

Introduksjon

Innleveringen består av 6 oppgaver, hvor oppgave 3 er frivillig og de andre 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. Skriv et program med en funksjon *adder (tall1, tall2)* som tar to heltall som parametere. Funksjonen skal summere tallene og returnere resultatet. Kall på funksjonen to ganger med argumenter du selv velger, og skriv ut resultatene med en passende tekst.

2. Be så brukeren om å skrive inn en tekststreng, og deretter en bokstav. Programmet skal så skrive ut hvor mange ganger den oppgitte bokstaven forekommer i den oppgitte tekststrengen. Du skal ikke regne stor og liten bokstav som like (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 kodelinjene fra punkt 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.

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.

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 laget tidligere og legger verdien til *minSum*. 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.

Synes du denne oppgaven var vanskelig? Se Trix-oppgave 4.01 og 4.03.
Synes du denne oppgaven var enkel? Se Trix-oppgave 4.02 og 4.04.

Oppgave 3: Kjede av sirkler - FRIVILLIG

Filnavn: *sirkler.py*

Denne oppgaven er frivillig, og kommer ikke til å gi poeng, men du kan få tilbakemelding.

I denne oppgaven skal du ta i bruk modulen *ezgraphics.py* som du brukte i forrige obligatoriske innlevering. Legg modulen i samme mappe som *sirkler.py*. Hvis du har glemt hvordan du bruker biblioteket kan du ta en ny titt på de relevante sidene i pensumboka (side 63-69).

1. Lag et grafikkvindu *win* med et kanvas ved navn *can* som du kan tegne på som beskrevet i pensumboka.

2. Opprett to variabler: En *teller* med verdien 0, og en *x_pos* med verdien 10.

3. Skriv en while-løkke som fortsetter så lenge *teller* er mindre enn 9. For hver gang løkken kjører skal det tegnes en sirkel, slik:

   ```
   can.drawOval(x_pos, 100, 50, 50)
   ```

   … der de tre siste parametrene betegner henholdsvis y-posisjon, høyde og bredde på sirkelen. Etter at en sirkel er tegnet skal du øke *x_pos* med et tall du velger selv.

Husk også å inkrementere *teller*.

4. Lag en variabel *stoerrelse* før løkken din, med en valgfri verdi. Endre løkken din slik at høyde og bredde på sirkelen bestemmes av *stoerrelse*. La størrelsen økes med 5 for hver gjennomkjøring av løkken.

Synes du denne oppgaven var vanskelig? Se Trix-oppgave 4.05 og 4.06.
Synes du denne oppgaven var enkel? Se Trix-oppgave 4.07.

Oppgave 4: 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. Følg disse stegene for å la brukeren oppgi en plass i *reiseplan* og skrive ut elementet på den oppgitte plassen.

    a. Først henter du inn en indeks *i1* som representerer en av de tre listene i *reiseplan*, der gyldig input vil være mellom 0 og lengden til *reiseplan* minus 1 (som vi har gått gjennom starter indeksen på 0, og en liste med 5 elementer vil derfor ha 4 som høyeste indeks).

    b. Deretter en indeks *i2* som representerer et av de fem elementene i den valgte listen, der gyldig input vil være mellom 0 og lengden til listen minus 1.

    c. Bruk en if-sjekk til å teste at de to tallene brukeren har oppgitt er gyldige verdier. Dersom tallene er mulige plasser i listene skal de brukes til å skrive ut *reiseplan* [i1][i2]. Hvis tallene ikke er gyldige plasseringer skal du skrive ut “Ugyldig input!”

Synes du denne oppgaven var vanskelig? Se Trix-oppgave 4.08.
Synes du denne oppgaven var enkel? Se Trix-oppgave 4.09.

Oppgave 5: Å telle bokstaver og ord

Filnavn: *ordtelling.py*

I denne oppgaven skal dere lage to funksjoner som begge utfører operasjoner på strenger. Så skal dere lage et program som bruker disse funksjonene på input fra brukeren, og skriver ut informasjon i terminalen. Foruten tjenester gjennomgått på forelesning kan du benytte tjenesten "split" som også tilbys av tekst-objekter: Om man har en variabel **tekst="hei du"** kan man skrive **ordene = tekst.split()** for å få en liste med hvert ord. Listen vil se sånn her ut: [“hei”, “du”] siden split deler opp på mellomrom.

1. Lag en funksjon som, gitt et ord, returnerer antall bokstaver i ordet.
2. Lag en funksjon som, gitt en setning, returnerer ei ordbok hvor hvert (unike) ord i setningen er en nøkkelverdi, med antall ganger det ordet finnes i setningen, som innholdsverdi.

3. Lag et program som tar inn en setning fra brukeren. Bruk så funksjonene fra deloppgave 1 og 2. Skriv også til brukeren hvor mange ord det er i setningen, og bruk så resultatene fra de to funksjonene til å skrive hvor mange ganger hvert unike ord forekommer, og hvor mange bokstaver hvert ord består av.

Dere kan skrive tegnsetting som “!”, ”.” og ”,”, etc. med mellomrom, og dere kan også behandle disse tegnene som ord.

Eksempel på bruk og utksrift:

```
>>>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 4.21.

Oppgave 6: Egen oppgave

1. Skriv oppgavetekst til en oppgave som handler om løkker og lister. 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 IFI sine maskiner. 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.
- Programmet skal inneholde gode utskriftssetninger som gjør det enkelt for bruker å forstå.

Hvordan levere oppgaven

1. Kommenter på følgende spørsmål i kommentarfeltet i Devilry. Spørsmålene **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) oppgave 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?
        iv. Hva vil du ha tilbakemelding på?
2. Logg inn på Devilry.
3. Lever alle *.py*-filene, og husk å svare på spørsmålene i kommentarfeltet.
4. Husk å trykke lever/add delivery og sjekk deretter at innleveringen din er komplett.
5. Den obligatoriske innleveringen er minimum av hva du bør ha programmert i løpet av en uke. Du finner flere oppgaver for denne uken her.