IN1000 Obligatorisk innlevering 5

Frist for innlevering: 30.09 kl 23:59

Introduksjon

Innleveringen består av 5 oppgaver, der hver oppgave teller 1 poeng. 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 utfordringer som omhandler data ved hjelp av programmering. Du skal kunne lese inn verdier fra tekstfiler og bruke disse for å gjøre beregninger. Du skal også forstå hvor en gitt variabel er synlig og tilgjengelig (skop) og hvordan verdier kan overføres mellom forskjellige skop.

Oppgave 1: Regnefunksjoner

Filnavn: *regnefunksjoner.py*

1. Skriv en funksjon *addisjon* som har to parametre. Funksjonen skal returnere summen av disse. Skriv en kodelinje som kaller på *addisjon* med heltallsargumenter du velger selv, og skriv ut resultatet.

2. Skriv tilsvarende funksjoner for *subtraksjon*, der du trekker det andre parameteret fra det første, og *divisjon*, der du deler det første parameteret på det andre. Du skal teste disse funksjonene også, men denne gangen ved hjelp av *assert*. Skriv minst 3 *assert*-uttrykk for hver funksjon, der du sjekker at kall på funksjonen med forskjellige heltall du velger selv får verdien du forventer (minst ett kall med to positive tall, ett med to negative tall og ett kall med ett positivt og ett negativt tall).

3. Lag en funksjon *tommerTilCm* med ett parameter *antallTommer*. Øverst i denne funksjonen skal du bruke *assert* for å sjekke at *antallTommer* er større enn 0. Deretter skal funksjonen returnere hvor mange centimeter *antallTommer* tilsvarer. For å regne om fra tommer til centimeter kan du gange *antallTtommer* med 2.54. Test funksjonen din.

4. Lag en prosedyre *skrivBeregninger*. Den skal ikke ta imot noen argumenter, men i stedet bruke *input* for å hente inn verdier fra bruker til kall på funksjonene du skrev over. Ta høyde for at brukeren kan skrive inn flyttall.

a. Først skal prosedyren hente inn to tall fra bruker. Disse tallene skal brukes som argumenter i kall på *addisjon*, *subtraksjon* og *divisjon*. Hver av disse funksjonene skal kalles og resultatet skrives ut til terminal.

b. Deretter skal prosedyren hente inn et nytt tall som skal konverteres fra tommer til centimeter. Også dette resultatet skal skrives ut.

5. Test *skrivBeregninger*. Utskriften bør se omtrent slik ut:

```
Utregninger:
Skriv inn tall 1: 12
Skriv inn tall 2: 5

Resultat av summering: 17.0
Resultat av subtraksjon: 7.0
Resultat av divisjon: 2.4

Konvertering fra tommer til cm:
Skriv inn et tall: 23
Resultat: 58.42
```

Synes du denne oppgaven var vanskelig? Se Trix-oppgave 5.01.
Synes du denne oppgaven var enkel? Se Trix-oppgave 5.02.

Oppgave 2: Temperatur

Filnavn: *temperatur.py*

For å løse denne oppgaven skal du lese inn data fra en fil. Oppgaven kan løses i både Linux, MacOS og Windows, men vær klar over at det er noen forskjeller i hvordan filstier brukes. Legg for sikkerhets skyld *temperatur.txt* i samme mappe som *temperatur.py.*

1. Vi har en fil *temperatur.txt* som inneholder 12 linjer hvor hver linje er gjennomsnittstemperaturen for en måned i et år. Første linje tilsvarer januar, andre linje tilsvarer februar, osv. Vi har rundet av temperaturene til heltall. Les inn temperaturene en etter en og putt verdiene inn i en liste.

2. Skriv en funksjon som tar i mot en liste og returnerer gjennomsnittet av verdiene i listen ved hjelp av en for-løkke. Kall denne funksjonen med listen av temperaturer som argument, og skriv ut resultatet.

Synes du denne oppgaven var vanskelig? Se Trix-oppgave 6.01.
Synes du denne oppgaven var enkel? Se Trix-oppgave 6.02.

Oppgave 3: Skop

Filnavn: *skop.py*

I denne oppgaven skal du beskrive programflyt. **Følgende kodesnutt og påfølgende beskrivelse er kun et eksempel:**

```
def funk():
	y = 2
	y = y * 2
	return y

def hovedprogram():
	x = 5
	z = 4
	x = funk()

hovedprogram()
```

*"Først defineres funksjonen **funk**, som ikke skal ta imot noen parametere. Deretter defineres prosedyren **hovedprogram**, som heller ikke tar noen parametre. Deretter kalles hovedprogram. Inne i hovedprogram opprettes en variabel **x** med verdi 5. Deretter oppretter vi en variabel **z** med verdien 4. Så kaller vi på funk. I funk opprettes en variabel **y** med verdien 2. Så blir y lik seg selv ganger 2. Deretter returneres y, og x får tilordnet denne returverdien, dvs. 4."*

Selve oppgaven: Beskriv med ord hva som skjer når følgende program kjøres. Forsøk gjerne å løse oppgaven før du kjører programmet.

```
def minFunksjon():
	for x in range(2):
	    c = 2
	    print(c)
	    c += 1
	    b = 10
	    b += a
	    print(b)
	return(b)

def hovedprogram():
	a = 42
	b = 0
	print(b)
	b = a
	a = minFunksjon()
	print (b)
	print (a)

hovedprogram()
```

Synes du denne oppgaven var vanskelig? Se Trix-oppgave 6.04 og side 282-285 i pensumboka.

Oppgave 4: Repetisjon

Filnavn: *repetisjon1.py*

1. Lag en tom liste *mineOrd*.

2. Definer en funksjon som heter *slaaSammen*. Funksjonen skal ta imot to strenger, konkatenere dem (slå dem sammen) og returnere den sammenslåtte verdien.

3. Definer en prosedyre som heter *skrivUt*. Prosedyren skal ta imot en liste og skrive ut hvert element i listen på hver sin linje ved hjelp av en for-løkke.

4. Lag en while-løkke som skal kjøres så lenge brukeren ønsker å fortsette (unngå å bruke while-løkker som alltid evaluerer til True f. eks. while True:). I løkken skal programmet ta i mot en streng fra brukerinput:

    a. Hvis brukeren skriver inn ”i” skal programmet be om to tekststrenger og deretter kalle på funksjonen *slaaSammen* med disse som parametere. Resultatet av funksjonskallet skal legges inn i listen *mineOrd*.

    b. Hvis brukeren skriver inn ”u” skal du kalle prosedyren *skrivUt* med *mineOrd* som parameter.

    c. Hvis brukeren skriver inn ”s” skal vi gå ut av løkken og avslutte programmet.

Synes du denne oppgaven var vanskelig? Se Trix-oppgave 6.05.
Synes du denne oppgaven var enkel? Se Trix-oppgave 6.03.

Oppgave 5: Egen oppgave

1. Skriv oppgavetekst til en oppgave som handler om innlesing fra fil og funksjoner. Eller du kan følge dette forslaget: Skriv et beregningsprogram for skreddere med en funksjon som leser inn en fil (som du lager selv og leverer sammen med de andre filene) der hver linje beskriver et navn på et mål og selve målet i tommer. Formatet vil se slik ut:

    ```
    Skulderbredde 4
    Halsvidde 3.2
    Livvidde 10
    ```

    Hint: du kan bruke funksjonen *.split()* for å gjøre dette.

    La programmet legge disse målene i en ordbok med navn på målet som nøkkelverdi og returner ordboken. Lag deretter en prosedyre som tar imot en liste av mål og benytter seg av *tommerTilCm* som du skrev tidligere for å skrive ut målene i centimeter

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 og *.txt*-filen fra oppgave 5 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 datafilene programmene åpner, 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.