IN1000 Innlevering 2

Introduksjon

Denne gangen er det fem oppgaver som skal løses. 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

Du skal nå ha fått en mer presis forståelse av hvordan koden fungerer. Det innebærer blant annet å kunne vise at du forstår kodeflyten i et program og hvordan uttrykk evalueres. Du skal også ha lært om forskjellige datatyper og om gjenbruk av kode ved hjelp av prosedyrer.

Oppgave 1: Utskriftsprosedyre

Filnavn: *utskriftsprosedyre.py*

1. Lag et program som kommuniserer med brukeren. Programmet skal ta inn et navn og et bosted fra terminalen og bruke begge deler i en hilsen som skrives ut til terminalen. Eksempel på hvordan en kjøring av programmet skal se ut:

> python3 utskriftsprosedyre.py
Skriv inn navn: Espen Askeladd
Hvor kommer du fra? Oslo
Hei, Espen Askeladd! Du er fra Oslo.

2. Flytt koden som leser inn informasjon og skriver ut en hilsen til en egen prosedyre. Kall denne prosedyren 3 ganger, slik at du får lest inn og skrevet ut informasjon om 3 personer. I denne oppgaven trenger du kun å levere inn deloppgave 2. Eksempel på kjøring av programmet:

> python3 utskriftsprosedyre.py
Skriv inn navn: Espen Askeladd
Hvor kommer du fra? Oslo
Hei, Espen Askeladd! Du er fra Oslo.
Skriv inn navn: Donald Duck
Hvor kommer du fra? Andeby

Hei, Donald Duck! Du er fra Andeby.

Skriv inn navn: Minni Mus
Hvor kommer du fra? Museby
Hei, Minni Mus! Du er fra Museby.

Syntes du denne oppgaven var vanskelig? Se Trix-oppgave 2.01, 2.09, 2.12
Syntes du denne oppgaven var enkel? Se Trix-oppgave 2.06, 2.17, 2.23

Oppgave 2: Konvertering

Filnavn: *fahrenheit_celsius.py*

Du skal skrive et program for å konvertere en gitt temperatur fra fahrenheit til celsius.

1. Skriv et program med en variabel som er en temperatur i fahrenheit. Bestem selv hva temperaturen skal være, og gi variabelen et beskrivende navn.

2. Skriv ut temperaturen i fahrenheit.

3. Definer en ny variabel som bruker variabelen i punkt 1 til å finne temperaturen i celsius. For å regne om fra fahrenheit til celsius kan du bruke denne formelen:

   ((temperatur i fahrenheit) − 32) ∗ 5/9.

4. Skriv ut temperaturen i celsius.

5. *Endre programmet*, slik at variabelen for fahrenheit blir gitt via brukerinput. **Sørg for at variabelen får en tallverdi**. Sjekk deretter programmet ditt ved å kjøre det og taste inn tallet 100. Får du ca 37,78 grader celsius? Hvis ikke, sjekk at du har fulgt formelen gitt over.

Syntes du denne oppgaven var vanskelig? Se Trix-oppgave 2.02, 2.03, 2.08
Syntes du denne oppgaven var enkel? Se Trix-oppgave 2.06, 2.16, 2.17

Oppgave 3: Kodeflyt

Filnavn: *kodeflyt.pdf*

Dette er en tekstoppgave. Du kan gjøre den med penn og papir og ta et bilde eller scanne inn arket, eller du kan gjøre den på datamaskin. Du skal levere oppgaven som en PDF-fil sammen med kodefilene i Devilry. Det holder ikke å gi filnavnet endelsen .pdf for å gjøre den til en PDF-fil. Du må altså lagre filen som en PDF eller eksportere/konvertere filen til PDF.
Du skal kort forklare programflyten til et program (i hvilken rekkefølge de forskjellige linjene utføres når vi kjører programmet) ved å skrive tall foran linjene i programmet. Tallene beskriver i hvilken rekkefølge programsetningene utføres, slik at 1 er den første setningen som utføres, 2 er den neste, og så videre.

Her er et **eksempel** på hvordan filen din skal se ut (legg merke til linjenummereringen / tallrekkefølgen oppgitt til venstre):

```
1 a = 13
2 if a < 10:
      print("Mindre!")
3 else:
4 print("Ikke mindre!")
```

Vi definerer først en variabel *a* med verdien 13. Så sjekker vi om *a* er mindre enn 10. Ettersom dette ikke stemmer, utføres ikke neste linje, men vi går i stedet videre til *else* og skriver ut linjen vi finner der.

Hvis en linje kjøres flere ganger (for eksempel ved flere kall på samme prosedyre), kan du skrive flere tall foran setningen (med komma mellom hvert tall).

Kopier programmet nedenfor og legg til nummerering av linjene som beskrevet over.
Som du ser tar programmet input fra bruker, og du skal her anta at brukeren under kjøring av programmet taster inn tallet 8.

```
def print_prosa():
    print("Melding til alle gaardeiere:")
    print("Antall dyr paa gaarden: ")

antall_dyr = 4
print_prosa()
print(antall_dyr)
antall_nye_dyr = int(input("Hvor mange nye dyr kommer til gaarden: "))
antall_dyr = antall_dyr + antall_nye_dyr
print_prosa()
print(antall_dyr)

if antall_dyr > 12:
    print("Det er mer enn ett dusin dyr paa gaarden!")
elif antall_dyr == 12:
```

```
    print("Det er ett dusin dyr paa gaarden!")
else:
    print("Det er mindre enn ett dusin dyr paa gaarden!")
```

Syntes du denne oppgaven var vanskelig? Se Trix-oppgave 2.11, 2.14
Syntes du denne oppgaven var enkel? Se Trix-oppgave 2.18, 2.20, 2.22, 2.23

Oppgave 4: Kodeforståelse og feilsøking

Filnavn: *kodeforstaaelse.py*

Dette er en teorioppgave. Besvar teorispørsmålene som kommentarer øverst i *kodeforstaaelse.py*.

Les koden nedenfor grundig og forsøk å svare på deloppgavene under. Test deretter besvarelsen din ved å lime inn koden i *kodeforstaaelse.py* og kjøre programmet. spes

```
a = input("Tast inn et heltall! ")
b = int(a)
if b < 10:
    print (b + "Hei!")
```

1. Vil programmet ovenfor kjøre? Begrunn svaret.

2. Hvilke problemer vil vi kunne møte på når vi kjører denne koden?

Syntes du denne oppgaven var vanskelig? Se Trix-oppgave 2.03, 2.05, 2.13
Syntes du denne oppgaven var enkel? Se Trix-oppgave 2.15, 2.24

Oppgave 5: Egen oppgave

1. Skriv din egen oppgavetekst til en oppgave som har med beslutninger (if/elif/else).

Et forslag er å skrive et quiz-program som stiller spørsmål til bruker og gir tilbakemelding på om svarene er korrekte.

2. Løs oppgaven du har skrevet! Du skal levere både oppgaveteksten og besvarelsen (skriv oppgaveteksten som kommentar over løsningen din).

Krav til innlevering

- Kun *.py*-filene og *kodeflyt.pdf* skal leveres inn.
- Spørsmålene til innleveringen (nedenfor) skal besvares i kommentarfeltet.

- Koden skal inneholde gode kommentarer som forklarer hva programmet gjør og hvordan du har tenkt underveis.

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

Hvordan levere oppgaven

1. Svar 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) 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, *kodeflyt.pdf*, 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. *Alle* filer må være med hver gang du leverer, og filene må ha samme navn. Det er ikke mulig å slette innleveringer i Devilry. Retteren din vil gi tilbakemelding på den siste innleveringen.

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.