IN1000: Obligatorisk innlevering 2

Frist for innlevering: 1.9.26, kl. 23:59 **Sist endret:** 25.08.26

Introduksjon

Velkommen til andre obligatoriske oppgave i IN1000! 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 (punkt 2c 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 epost 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

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. Du trenger bare å levere inn programmet slik det ble etter at du her har flyttet koden inn i en prosedyre (du trenger ikke levere inn slik koden var etter deloppgave 1) 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 som først tar en temperatur gitt i Fahrenheit som input fra brukeren av programmet.
2. Konverter nå denne gitte temperaturen fra Fahrenheit til Celsius.
   a. Hint: formelen for å konvertere fra Fahrenheit til Celsius er ((temperatur i Fahrenheit) − 32) * 5/9.
3. Skriv ut resultatet til brukeren.
4. Utvid nå programmet ditt slik at det også kan konvertere fra Celsius til Fahrenheit. La brukeren først oppgi et tall som er temperaturen og la brukeren deretter legge inn temperaturenheten (brukeren skriver først et tall og trykker enter, og deretter skriver brukeren F eller C og trykker enter). Konverter da den opprinnelige temperaturen til Celsius hvis den oppgitte enheten er ‘F’, eller til Fahrenheit hvis enheten er ‘C’.
   a. Hint 1: formelen for å konvertere fra Celsius til Fahrenheit er ((temperatur i Celsius)/5) * 9 + 32
   b. Hint 2: Sørg for at den oppgitte temperaturen blir en tallverdi
   c. Hint 3: implementer programmet slik at det velger hvilken formel som må brukes basert på den gitte temperaturenheten.
5. Sjekk deretter programmet ditt ved å kjøre det og taste inn tallet 100 og deretter taste inn F. Får du da ca 37,78 grader som svarer? 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 2.16

Oppgave 3: Kodeflyt

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. Det kan her være noen ulike måter å tenke på når man nummererer, f.eks. om man skal nummerere selve linjene som begynner med “def” eller ikke. Det er her greit å velge enten det ene eller andre - det viktige er den overordnede flyten i programmet.

```
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!")
```

Synes du denne oppgaven var vanskelig? Se Trix-oppgave 2.11, 2.14, 2.22
Synes du denne oppgaven var lett? Se Trix-oppgave 2.18, 2.20, 2.21

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 - først helt uten å kjøre programmet. Test deretter besvarelsen din ved å lime inn koden i en fil *kodeforstaaelse.py* og kjøre programmet.

```
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 2.05, 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

- 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.
- 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
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.
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.