Oppgaver dag 1

Dette er oppgaver som hører til Dag 1 av Programmering med Python, som er et grunnkurs i programmering laget av Kodeskolen i samarbeidet med Folkeuniversitetet.

Målet for dag 1 er å gi dere en grunninnføring i Pythonsyntaks, samt en introduksjon til følgende programmeringskonsepter

- Variabler
- Betingelser
- Løkker

Oppgavene i dette dokumentet dekker disse konseptene. Idéen er at dere skal jobbe med disse oppgavene både iløpet av dag 1, samt oppgaveøkten som legges mellom dag 1 og dag 2 av kurset. Dokumentet inneholder ganske mange oppgaver, og det er ikke nødvendigvis slik at dere vil få tid til å gjøre alle, dette er helt greit. Dere kan selv velge om dere ønsker å gjøre oppgavene i rekkefølge, eller om dere vil hoppe litt frem og tilbake blant oppgaver dere synes virker med interessante.

Det er mye forskjellig å huske på når man lærer seg å programmere. Det er ikke rart om det føles litt overveldende! Det viktigste er at dere prøver dere frem, det er slik man lærer. Ikke vær redd for å gjøre feil, og ikke vær redd for å spørre om hjelp. Husk også at det ligger en *jukselapp* på kursets sider, som kan gjøre det litt enklere å huske hvordan ulike kommandoer og begreper skrives.

Oppgave 1 *Printing*

a) Lag et program som skriver ut teksten *«Hello, World!»* til skjermen.

b) Lag et program der du først lagrer navnet ditt i en variabel, og så få programmet ditt til å skrive ut en hilsen direkte til deg.

c) Python har en innebygget funksjon som heter `len()`, om du bruker denne på variabelen din får du ut antall bokstaver i navnet. Endre programmet ditt så så det også skriver ut denne informasjonen.

d) Endre programmet så det istedet bruker `input()` til å spørre om navnet ditt når koden kjøres, istedenfor at dette er skrevet rett inn i koden.

Oppgave 2 *Konvertering av temperatur*

La oss si du har funnet en kakeoppskrift fra USA som sier at du skal bake kaken ved 350 grader. Det vil si, grader Fahrenheit, 350°F. La oss lage et enkelt program som konverterer en slik temperatur til grader Celsius for oss. Da trenger vi følgende formel

$$C = \frac{5}{9}(F - 32).$$

Der F er antall grader i Fahrenheit, og C blir antall grader i celsius.

a) Start med å opprette en variabel, `F`, som du setter lik 350.

b) På en ny kodelinje kan du nå opprette `C` som resultatet av en utregning ved hjelp av formelen over og `F`.

c) Hvor mange grader celsius er 350°F? Er det rimelig å skulle bake en kake ved denne temperaturen?

Du har nå laget et program som regner om fra F til C. Men hva om vi vil gjøre motsatt? Da må vi først ha en formel for F gitt C.

d) Klarer du å ta formelen over og løse for F? Tips: Om det er lenge siden du har gjort algebra kan du selvfølge spørre om hjelp, eller du kan rett og slett google deg frem til riktig formel!

e) Lag nå et nytt program der man gir en verdi til variabelen `C` og deretter regner ut `F` ved hjelp av formelen du har funnet.

f) Bruk programmet ditt til å finne frysepunktet og kokepunktet til vann i Fahrenheit.

En utvidelse av dette problemet kan nå være å lage et program som slår sammen disse to konverteringene. Da kan du f.eks først bruke `input` til å spørre brukeren om de vil regne fra F til C, eller fra C til F. Deretter kan du bruke en `if`-test til å bruke riktig formel. Dette kan være en utfordring til spesielt interesserte.

Oppgave 3 *Trille terninger*

a) Les følgende kodesnutt linje for linje og prøv å forstå hva koden gjør. Her kan det lønne seg og prøve å forklare koden for noen andre (eller du kan late som du forklarer for noen andre)

```
1 from random import randint
2
3 terning1 = randint(1, 6)
4 terning2 = randint(1, 6)
5
6 print(terning1 + terning2)
```

b) Skriv koden inn og kjør den et par ganger. Oppfører koden seg som du trodde?

c) Klarer du å endre koden så den i tillegg til å skrive ut summen av de to tallene, viser hva de faktisk er? F.eks kan en kjøring nå vises som

```
4 + 3 = 7
```

d) Kan du forklare hvorfor vi bruker funksjonen `randint(1, 6)` to ganger, istedenfor å bruke `randint(2, 12)`?

e) I Monopol er det slik at om man triller det samme på to terninger, så får man kaste på nytt. Klarer du å utvide koden med en `if`-test som skriver ut en beskjed dersom de to terningene har samme verdi?

Oppgave 4 *Finn fire feil!*

Her følger det fire kort kodesnutter som *ikke* fungerer som de skal. Skriv inn programmene på din egen maskin og kjør dem. Les feilmeldingen og prøv å tolke den, Når du skjønner hva som er galt kan du rette opp feilen og kjøre programmet.

a)

```
print "Hello, World!"
```

b)

```
navn = Jonas
print("Hei på deg", navn)
```

c)

```
navn = "Terje"
print("Hei på deg {Terje}!")
```

d)

```
from math import pi

radius = 5
areal = pi*R**2

print(f"Arealet er lik {areal}")
```

Oppgave 5 *Rabattkalkulator*

Lag et program der man skriver inn prisen på en vare (enten i koden, eller ved hjelp av `input`), og en rabatt i antall prosent. Programmet skal så oppgi hva prisen på varen er når rabatten er trukket fra. En kjøring av programmet kan f.eks se ut som følger

```
Original pris på vare: 450
Rabatt i %: 30
Nypris på vare blir: 315
```

Oppgave 6 *Minutter og timer*

Vi gir deg nå et lite program, delt opp i et par ulike biter. For hver bit les koden, prøv å forstå hva den gjør, skriv den inn i programmet ditt og sjekk om den gjør som du tror

a) Hva gjør følgende kodelinje?

```
minutter = int(input("Antall minutter: "))
```

b) De to neste kodelinjene inneholder to matematiske operasjoner du kanskje ikke er veldig kjent med. Her betyr to deletegn (`//`) det vi kaller *heltallsdivisjon*, altså divisjon *uten rest*. Mens prosenttegn `%` betyr *modulus*, som gir resten i en divisjon. Prøv å forstå hva disse operasjonene gjør. Skriv eventuelt variablene for å se om det oppfører seg som du tror.

```
timer = minutter //  60
rest = minutter % 60
```

c) Til slutt legger vi til en `if`-test. Men hva er egentlig formålet med denne testen? Prøv spesielt å forklare hvorfor vi har lagt inn denne *elif*-blokken.

```
if timer > 0 and rest > 0:
    print(f"Det er {timer} timer og {rest} minutter.")
elif timer > 0:
    print(f"Det er akkurat {timer} timer.")
else:
    print("Det er mindre enn 1 time.")
```

d) Implementer hele koden som et program om du ikke allerede har gjort det. Kjør koden slik at hvert av de tre ulike utfallene testes. Hvor lenge er f.eks 275 minutter i timer og minutter? Hva med 3000 minutter?

Oppgave 7 *Gangetabell-Quiz*

a) Les koden under og forklar hva den gjør

```
from random import randint

a = randint(1, 10)
b = randint(1, 10)

svar = int(input(f"Hva er {a} ganger {b}? "))
```

b) Utvid koden med en `if`-test som sjekker om brukeren svarer riktig eller ikke. Om de svarer riktig bør de gratuleres, om de svarer feil bør det riktige svaret skrives ut til skjermen, En utskrift kan se ut som følger

```
Hva er 4 ganger 7? 21
Det er dessverre feil. (4 x 7 = 28).
```

c) (Utfordrende) Klarer du å utvide koden så programmet spør brukeren fem ulike spørsmål etter hverandre og holder tellingen på antallet brukeren svarer rett på? Hint: Her bør du bruke en løkke for å gjenta koden din.

Oppgave 8 *Spambot*

a) Les følgende kode. Hva tror du den gjør?

```
count = 10

while count > 0:
    print(10*"SPAM! ")
    count = count - 1
```

b) Implementer koden selv og kjør den. Forklar hvorfor utskriften blir som den gjør

c) Hvor mange ganger blir det skrevet ut *SPAM!* totalt sett?

Oppgave 9 *Håndhilseproblemet*

I et rom på 20 mennesker, hvor alle skal håndhilse på alle. Hvor mange håndtrykk vil det være totalt? Dette kalles håndhilseproblemet.

For å regne ut dette er det enklest å jobbe person for person. Den første hilser på de 19 andre, den neste i rekka gjør også det, men vi har allerede talt håndtrykket fra den første personen, så vi får 18 nye håndtrykk. Den tredje personen i rekka hilser på 17 nye", og så videre. Så antall håndtrykk vil være

$$19 + 18 + 17 + \ldots + 3 + 2 + 1.$$

Lag et program som regner ut denne summen. Her må du altså først klare å lage en `while`-løkke som går over tallene $19, 18, \ldots, 2, 1$, og på en eller annen måte klarer å legge dem sammen.

Om du sliter med å komme i gang kan du se på skjelettkoden under og fylle inn i denne

```
n = 19
total = 0

while ...:
    total = ...
    n = ...

print(...)
```

Oppgave 10 *Sparekalkulator*

Siri driver å sparer til en ny telefon som koster 6500. Hun sparer 725 kroner i måneden. Fyll inn i følgende kode for å finne ut hvor mange måneder hun må spare

```
spart = ...
måneder = ...

while spart < ...:
    spart += ...
    måneder += ...

print(f"Siri må spare i {...} måneder")
```

Oppgave 11 *Tallrekker*

Som en enkel øvelse for å jobbe med løkker kan du prøve å skrive kode for å skrive ut følgende tallrekker. Denne oppgaven kan løses med både `while`-løkker og `for`-løkker (som vi skal dekke på dag 2). Om du har lært om `for`-løkker kan du godt prøve å løse løkkene på begge måter.

a) Skriv ut tallene 1 til 100

b) Skriv ut 7-gangen fra og med 0 til og med 70

c) skriv ut rekka 1, 4, 7, 10, ..., 31 (Hint: Den begynner på 1 og øker med 3 hver gang)

d) Skriv ut rekka 1, 3, 9, 27, ..., 729 (Hint: Vi ganger med 3 hver gang)

Oppgave 12 *While-løkker for hånd*

Gå igjennom løkkene under for hånd og forutsi hva de kommer til å skrive ut. Her kan det være til stor hjelp og ha penn og papir! Etterpå kan du kjøre dem og se om du hadde rett.

a)

```
x = 1
while x < 20:
    x *= 2
    print(x)
```

b)

```
i = 2
while i < 10:
    i = i**2 + 1
    print(i)
```

c)

```
a = b = c = 0

while c < 10:
    a += 1
    b += a
    c += a + b

print(a, b, c) # Denne er utenfor løkka
```

Oppgave 13 *Fakultet*

Si at du har 5 forskjellige små skulpturer du skal sette opp på rekke. Hvor mange ulike måter kan du gjøre dette på? For den første har du 5 valg, deretter har du 4 valg, så 3, og så videre. Dermed blir antall kombinasjoner til

$$5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120.$$

I matematikken skriver vi dette mer kompakt som 5!, som vi kaller *fakultet*. Å rangere n ting kan altså gjøres på $n!$ måter, som blir

$$n! = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1.$$

For små tall er det enkelt å regne ut fakultet for hånd, men dersom n begynner å bli litt større vil dette ta fryktelig lang tid.

a) Du skal nå skrive et program som kan regne ut fakultet for oss. Du kan enten prøve å gjøre dette helt selv, eller følge vår «steg-for-steg»-instrukser under.

- Spør brukeren om hva n er, lagre svaret i en variabel. Husk å konvertere svaret med `int(input())`.
- Lag en variabel `fakultet`, som du gir verdien 1.
- Lag en while-løkke som fortsetter så lenge n er større enn 0.
- For hver iterasjon av løkka, gang `fakultet`-variabelen din med n, og deretter reduser n med 1.
- Skriv ut det endelige svaret så brukeren kan se det.

b) Test at programmet ditt gir $5! = 120$.

c) En vanlig kortstokk har 52 unike kort. Hvor mange mulige rekkefølger kan vi få om vi stokker en kortstokk?

d) I noen kortspill legger man til 2 jokere i kortstokken, slik at det nå er 54 kort i kortstokken. Hvor mange måter blir det nå å stokke kortstokken på?