Uke 6, time 1

Onsdag 24. september

Introduksjon til Python

Konseptforståelse

Skop

Kodestil

Skop

Skop – hvor noe vi har gitt navn (variabler) til eksisterer

Vi skiller mellom globalt og lokalt skop

Kalles også navnerom (namespace)

Global

- Alt som er definert utenfor prosedyrer, funksjoner og klasser
- Tilgang til overalt, men kan lett føre til kode som er vanskelig å lese, og overskygges av lokale variabler
- Lett å overskrive med uhell, oppdateres fra feil sted
- Gjør det vanskelig å lage kode som kan skaleres, er modulær og gjenbrukbar
- Eneste som er anbefalt er ved bruk for konstanter eller "konfigurasjon"
- Vanskeligere å teste

Lokal

- Variabler inne i funksjoner / prosedyrer
- "Lever" bare inne i og så lenge vi kjører funksjonen
- Vi trenger ikke passe på at variabel-navnet brukes på nytt i andre funksjoner eller prosedyrer
- Gjør programmet modulært og kode kan lett gjenbrukes
- Anbefaling å alltid bruke lokalt skop; bruk parametere og returverdier for å overføre verdier

Ubundet/udefinert variabel

Hvis vi bruker beslutninger som f.eks. if/elif/else eller løkker, så må vi passe på ekstra når vi bruker variabler inne i disse blokkene, og før og etter.

```
teller = 10
if teller > 20:
    svar = "Her er jeg"
print(svar)
```

Her er `svar` ubundet (altså eksisterer ikke), og vil bare eksistere om if blokken kjører, dvs. hvis teller er over 20

Lokal main funksjon

Et vanlig mønster i Python er å lage en hoved-prosedyre som er det eneste som kalles fra det globale skopet

```
def hoved():
    print("Her er hoved prosedyren")

hoved()
```

Ikke så viktig for IN1000, men noe som er vanlig "overalt":

```
def main():
    print("Her er hoved prosedyren")

if __name__ == "__main__":
    main()
```

På denne måten kan du sørge for at du kan lage god modularitet, og holde "alt" i et lokalt skop

Sørger også for at program-filen ikke kjøres hvis den importeres

Global

Eksempler

```
telleren = 10
dyr = "hest"
pris = 25
MAKS_VERDI = 1_000_000
PI = 3.1415
telleren = 20     # re-definisjon
```

Lokal

```
def velg_dyr(dyr):
    if dyr == "sau":
        return "Sauer er best"
    elif dyr == "hund":
        return "Hund er best"
    else:
        return "Sikkert et fint dyr"

def dyr_spiser(dyr):
    if dyr == "sau":
        return "Sauen spiser gress"
    elif dyr == "hund":
        return "Hunden spiser hundemat"
    else:
        return "Spiser sikkert noe"

def hovedprogram():
    dyr = input("Velg et dyr (sau/hund)")
    print(velg_dyr(dyr))
    print(dyr_spiser(dyr))

hovedprogram()
```

```
def addisjon(tall_a, tall_b):
    svar = tall_a + tall_b
    return svar

def multiplikasjon(tall_a, tall_b):
    svar = tall_a * tall_b
    return svar

def utskrift():
    tall_a = 5
    tall_b = 2
    svar = addisjon(tall_a, tall_b)
    print(f"{tall_a} + {tall_b} =", svar)
    svar = multiplikasjon(tall_a, tall_b)
    print(f"{tall_a} * {tall_b} =", svar)

svar = 0                # Dette er
utskrift()              # i det globale
print("Svar", svar)     # skopet
```

Kodestil

Python har en "offisiell" stil-guide som heter PEP8

Noen av de viktigste punktene:

- Bruk 4 mellomrom som innrykk
- Hold linjer kortere enn 80 tegn
- Ha litt luft, altså et blankt linjeskift, men bare mellom logiske deler av koden, etter funksjoner, og evt. to mellom f.eks klasser, og større grupper med kode
- Navn på prosedyrer, funksjoner, metoder, objekter og variabler med småbokstaver og understrek mellom (snake_case)
- Konstanter bør være bare store bokstaver med understrek mellom
- Klassenavn med stor forbokstav for hvert ord, uten understrek eller andre tegn for mellomrom
- navn som begynner (og slutter) med a) 1 understrek regnes som "skjult" b) 2 understreker (double underline = dunder) er mer skjult og regnes som "interne" som ikke skal brukes eksternt

Men, det viktigste er at man er konsekvent

Oppgaver

Oppgave 1

```
def minus():
    a = 3
    b = 2

def summer():
    print("sum: ", a + b)

summer()
```

a) Hvorfor fungerer ikke programmet ovenfor? Diskuter.

b) Endre programmet slik at det fungerer. Dere skal ikke bruke globale variabler.

c) Se deretter kjøringen av det steg for steg i Python Tutor. Sørg for å detaljert forklare og virkelig forstå hva som skjer i hver programlinje.

Oppgave 2

```
def legg_til_bokstav(ordliste, bokstav):
    for ord in ordliste:
        ord += bokstav
    return ordliste

def hovedprogram():
    ordliste = ["Je", "de", "se"]
    ordliste = legg_til_bokstav(ordliste, "g")
    print(ordliste)

hovedprogram()
```

a) Hvorfor endres ikke ordene i programmet ovenfor? Diskuter.

b) Endre programmet slik at ordene endres. Dere skal ikke bruke globale variabler.

c) Se deretter kjøringen av programmet steg for steg i Python Tutor. Sørg for å detaljert forklare og virkelig forstå hva som skjer i hver programlinje.

Oppgave 3

```
def hovedprogram():
    grense = -5
    grenseteller = 0

    while grenseteller <= 0:
        print("Har vi nådd grensen?", grenseteller == grense)
        grenseteller -= 1

hovedprogram()
```

a) Hvorfor slutter programmet ovenfor aldri å kjøre? Diskuter.

b) Endre programmet slik at det stopper (terminerer) når grensa (grense) er nådd. Dere skal ikke bruke globale variabler.

c) Se deretter kjøringen av programmet steg for steg i Python Tutor. Sørg for å detaljert forklare og virkelig forstå hva som skjer i hver programlinje.

Oppgaver

Oppgave 4

```
def bygg_samling(antall_rader, antall_kolonner):
    samling = []

    while antall_rader > 0:
        rad = []

        for kolonnenr in range(antall_kolonner):
            rad.append(kolonnenr)

        samling.append(rad)
        antall_rader -= 1

    return samling

def hovedprogram():
    samling = bygg_samling(7, 5)

hovedprogram()
```

a) Hva slags samling er det programmet ovenfor bygger, og hvordan ser samlingen ut? Diskuter uten å kjøre programmet.

b) Hva er oppgaven til while-løkka og hva er oppgaven til for-løkka? Diskuter.

c) Utvid programmet med en funksjon, finn_antall, som tar inn den bygde samlingen og et tall. Funksjonen skal returnere antall ganger det oppgitte tallet finnes i samlingen. Hovedprogrammet skal skrive ut dette antallet.

Oppgave 5

```
def hent_navn(personbok, personid):
    if personid in personbok:
        return personbok[personid]
    return None

def hovedprogram():
    personbok = {
        "01020300000": "Hjørdis Olsen",
        "02030400000": "Martin Halvorsen",
        "03040500000": "Kristin Andersen"
    }

    personid = "02030400000"
    navn = hent_navn(personbok, personid)
    startbokstav = "M"

    if navn:
        print("Fant person ved navn", navn + ".")
        if navn[0] == startbokstav:
            print("Navnet starter på", startbokstav + ".")
    else:
        print("Fant ingen person med id", personid + ".")

hovedprogram()
```

a) Hva skrives ut i programmet ovenfor? Diskuter først uten å kjøre koden.

b) Se deretter kjøringen av programmet steg for steg i Python Tutor. Sørg for å detaljert forklare og virkelig forstå hva som skjer i hver programlinje.

c) Hva skrives ut hvis personid ikke finnes i personbok?

d) Endre programmet, slik at det ikke lenger inkluderer nøstede beslutninger. Programmet skal likevel gjøre det samme etter endringen som før endringen.