Velkommen til gruppetime i IN1000 ☺

18. september 2020

Jessie Yue Guan

Planen for i dag

- Vanlige for-løkker versus for-each-løkker
- Skopet til if/elif/else, for/while, prosedyrer/funksjoner
- Globale versus lokale variabler
- Assert, uttrykk, og evalueringer
- Innlesing av data fra filer
- Nøstede lister

For-løkker som vi har sett på hittil

- Hittil har vi for det meste sett på for-løkker med følgende syntaks:

- **for i in range(0, 10):**
  - **print(i)**

- Disse kaller vi bare vanlige for-løkker

For-løkker som vi skal se litt mer på

- Men fra nå av kommer vi også til å bruke for-løkker med følgende syntaks:

- **for element in liste:**
  - **print(element)**

- Disse kaller vi for-each-løkker

Hvilken for-løkke skal man bruke?

- Vanlige for-løkker egner seg spesielt godt til å gjøre det samme om og om igjen et visst antall ganger

- For-each-løkker egner seg spesielt godt til å gjøre det samme om og om igjen for hver verdi i en kolleksjon (liste, ordbok, osv.)

Hva er skop?

- Følgende programmeringskonsepter har skop:
  - If/elif/else-setninger
  - For/for-each/while-løkker
  - Prosedyrer/Funksjoner/Metoder

Hva er skop?

- Skopet til if/elif/else-setninger er all den koden som er indentert under dem
  - Det er altså koden som kjøres én gang hvis uttrykket i if/elif/else-setningen evaluerer til True
- Skopet til for/for-each/while-løkker er all den koden som er indentert under dem
  - Det er altså koden som kjøres flere ganger hvis uttrykket i for/for-each/while-løkken evaluerer til True
- Skopet til prosedyrer/funksjoner/metoder er all den koden som er indentert under definisjonen av prosedyren/funksjonen/metoden
  - Det er altså koden som kjøres en eller flere ganger hvis vi kaller på prosedyren/funksjonen/metoden

Skopet til if/elif/else-setninger

```
a = 5
b = 10

if (a > b):
    print("a er større enn b\n")
elif (b > a):
    print("b er større enn a\n")
else:
    print("a og be er like store\n")
```

Skopet til prosedyrer/funksjoner

```
def prosedyre(x, y):
    print("Summen er", x + y)

prosedyre(3, 7)

def funksjon(x, y):
    return x + y

print(funksjon(4, 6))
```

Skopet til for/for-each/while-løkker

```
a = 5
b = 10

while (a < b):
    a += 1;
    print("a er nå", a)

print()

for c in range (0, 3):
    print(c)
    print(c*10)

print()

d = [3, 6, 9]

for tall in d:
    print(tall, "er i lista d")
```

Globale versus lokale variabler

- **a = «Hei!»**
- **for a in range (0, 10):**
  - **print(a)**
- Hvordan er dette mulig? Hvorfor får vi ikke en feilmelding?

Globale versus lokale variabler

- Globale variabler er de variablene som er definert innenfor skopet til «hovedprogrammet»
- Lokale variabler er de variablene som er definert innenfor skopet til et av følgende:
  - if/elif/else-setninger
  - For/for-each/while-løkker
  - Prosedyrer/funksjoner/metoder
- Lokale variabler kan ikke brukes utenfor dette skopet
- For eksempel, hvis du definerer en variabel som er inni en for-løkke som igjen er inni en annen for-løkke, så kan du kun bruke variabelen i den innerste løkka og ikke i den ytterste løkka eller i «hovedprogrammet»

Hva er uttrykk og evalueringer?

- Et uttrykk er en påstand som enten kan være sann (True) eller usann (False)

- Det kan være uttrykk med relasjonelle operasjoner:
  - **2 < 3 #True**
  - **6 > 6 #False**
  - **9 == 9 #True**
  - **5 <= 4 #False**
  - **0 >= 0 #True**
  - **1 != 1 #False**
  - **7 != 8 #True**

Hva er uttrykk og evalueringer?

- Et uttrykk er en påstand som enten kan være sann (True) eller usann (False)

- Det kan være uttrykk med logiske operasjoner:
  - **Not True** **#False**
  - **Not False** **#True**
  - **True and True** **#True**
  - **True and False** **#False**
  - **False and False** **#False**
  - **True or True** **#True**
  - **True or False** **#True**
  - **False or False** **#False**

Hva er uttrykk og evalueringer?

- Et uttrykk er en påstand som enten kan være sann (True) eller usann (False)

- Det kan være uttrykk med boolske variabler:
  - **a = True**
  - **b = False**
  - **a and b** **#False**
  - **a or b** **#True**
  - **not a** **#False**
  - **not b** **#True**

Hva er uttrykk og evalueringer?

- Et uttrykk er en påstand som enten kan være sann (True) eller usann (False)

- Det kan være uttrykk med en kombinasjon av logiske og relasjonelle operasjoner:
  - **7 < 7 or True** **#True**
  - **2 >= 2 and False** **#False**
  - **Not True or 0 == 0** **#True**
  - **False and 4 != 6** **#False**
  - **a = False**
  - **b = True**
  - **b or 3 > 1** **#True**
  - **a and False** **#False**

Hva er uttrykk og evalueringer?

- Med andre ord, det er…
- Alt som dere har skrevet etter if/elif/else og før kolon
- Alt som dere har skrevet etter while og før kolon
- … når koden har fungert og programmet kjører uten feilmeldinger

Assert

- Assert sjekker om en antakelse du har om koden din stemmer eller ikke, for eksempel:
  - **assert summer(7, 3) == 10**
- Merk at det som kommer etter assert er et **UTRYKK** som evaluerer til True eller False
- Merk også at vi bruker dobbel likhetstegn, akkurat som i if-setninger
- Hvis antakelsen evaluerer til True vil ingenting skje eller printes når man kjører programmet
- Hvis antakelsen evaluerer til False vil man få en feilmelding og beskjed om hvilken assert som feilet

Hva er IKKE uttrykk og evalueringer?

- assert 2 #DETTE GIR INGEN MENING!
- assert «Velkommen!» #DETTE GIR INGEN MENING!
- assert 9.99 #DETTE GIR INGEN MENING!

- variabel1 = 5
- variabel2 = «Jessie»
- variabel3 = 3.75

- assert variabel1 #DETTE GIR INGEN MENING!
- assert variabel2 #DETTE GIR INGEN MENING!
- assert variabel3 #DETTE GIR INGEN MENING!

- Hva er poenget med disse assert'ene???

Hva er IKKE uttrykk og evalueringer?

- assert 2 and 3 #DETTE GIR INGEN MENING!
- assert «Velkommen!» or «Hei!» #DETTE GIR INGEN MENING!
- assert not 9.99 #DETTE GIR INGEN MENING!

- variabel1 = 5
- variabel2 = «Jessie»
- variabel3 = 3.75

- assert variabel1 and variabel3 #DETTE GIR INGEN MENING!
- assert variabel2 or «Hadet!» #DETTE GIR INGEN MENING!
- assert not variabel3 #DETTE GIR INGEN MENING!

- Logiske operatorer (and, or, not) skal helst KUN brukes med boolske verdier!!!

Hva er IKKE uttrykk og evalueringer?

- variabel5 = «Bla bla»
- variabel6 = «Shalala»

- assert «Hei!» > «Hadet!» #DETTE GIR INGEN MENING!
- assert variabel4 < «Hadet!» #DETTE GIR INGEN MENING!
- assert «Hei!» >= variabel6 #DETTE GIR INGEN MENING!
- assert variabel4 <= variabel6 #DETTE GIR INGEN MENING!

- Relasjonelle operatorer (<, >, <=, >=, ==, !=) skal helst KUN brukes med tall!!!

Hvorfor lese inn data fra filer?

- Man kan jobbe med større datamengder
- Man kan lettere strukturere større datamengder
- Man skiller mellom data (hva) og kode (hvordan)
- Man slipper å skrive dataene på nytt fordi de blir lagret og kan hentes fram igjen
- Man kan se på filer som en samling av linjer og linjer som en samling av tegn

Hvordan lese inn linjer fra filer?

- Steg 1: Åpne filen (og velg les)
- Steg 2: Gå gjennom hver linje i filen
- Steg 3: Gjør noe med hver linje i filen

- **min_fil = open(«mittFilNavn.txt»)**
- **for linje in min_fil:**
  - **min_liste.append(linje)**

Hvordan lese inn tabeller fra filer?

- Steg 1: Åpne filen (og velg les)
- Steg 2: Gå gjennom hver rad i filen
- Steg 3: Splitte raden etter kolonne
- Steg 4: Gå gjennom hver rute i raden
- Steg 5: Gjøre noe for hver rute i tabellen

```
min_fil = open(«mittFilNavn.xlsx»)
for linje in min_fil:
    rad = linje.split(«:»)
    for kol in rad:
        min_liste.append(kol)
```

Hvordan skrive ut linjer til filer?

- Steg 1: Åpne filen og velg skriv
- Steg 2: Skriv det du vil i filen
- Steg 3: Lukk filen

- **min_fil = open(«mittFilNavn.txt», «w»)**
- **min_fil.write(«La la la\nHa ha ha»)**
- **min_fil.close()**

2-dimensjonale/nøstede lister

listeception.py

```
1  hamsterbur = [
2      ["Hamtaro", "Spencer", "Max"],
3      ["Bobby", "Cupcake", "Cookie"],
4      ["Angel", "Per", "Olav"]
5  ]
6
7
8  for liste in hamsterbur:
9      for navn in liste:
10
11         print(navn)
12     print()
13
14 print("Listeindeks 1, navneindeks 2: " + hamsterbur[1][2])
15
```

På tide å sjekke om dere har fulgt med!

- Kahoot! :D