Gruppetime uke 6
IN1000

Plan for timen

- Skop
- Programmeringsstafett?
- Egenjobbing/Repetisjon

Skop

Hva er skop

- Skop sier noe om hvor du kan bruke/aksessere en variabel i et program
- En variabel er kun tilgjengelig innenfor skopet den er definert
- Hver funksjon/prosedyre har sitt eget skop

En variabel kan ligge innenfor to type skop

GLOBAL VARIABEL	LOKAL VARIABEL
• En variabel som defineres på utsiden av en prosedyre eller funksjon • Når en variabel er definert utenfor en funksjon/prosedyre sier man at variabelen ligger innenfor det globale skopet	• En variabel som defineres inni en prosedyre eller funksjon • Når en variabel er definert inni en funksjon/prosedyre, sier man at variabelen ligger innenfor det lokale skopet til prosedyren/funksjonen

Lokale variabler

```
1 def storst_av_to(tall1, tall2) :
2     if tall1 > tall2 :
3         return tall1
4     return tall2
5 storst_av_to(3, 4)
```

- Parametrene til funksjonen over kan bare akesseres innenfor funksjonen
- Det er feks ikke mulig å printe ut tall1 eller tall2 etter linjen storst_av_to(3, 4) eller noe som helst annet sted utenfor funksjonen

Lokale variabler

- Variabelen x er her definert innenfor funksjonen min_funksjon, og kan da kun aksesseres innenfor denne funksjonen.
- Det er feks ikke mulig å printe ut x utenfor funksjonen
- x tilhører altså det lokale skopet til prosedyren

```
8  def min_funksjon():
9      x = 300
10     print(x)
11
12 min_funksjon()
```

Globale variabler

- Vi kan aksessere variablene a og b innenfor og utenfor funksjonen summer(), fordi disse ligger i det globale skopet
- Som vi ser kan globale variabler også brukes av alle funksjoner og prosedyrer i et program
- MEN: unngå helst å bruke globale variabler i funksjoner/prosedyrer

```
19 def summer():
20     print("sum: ", a + b)
21
22 a = 3
23 b = 2
24 summer()
```

Globale variabler: bedre løsning

- Selv om globale variabler i prinsippet kan brukes innenfor funksjoner/prosedyrer, er det bedre praksis å sende inn disse variablene som argumenter til funksjonen/prosedyren
- På denne måten kan vi gjenbruke prosedyren med forskjellige argumenter
- Forhindrer bugs:
  - Hvis du forandrer på en global variabel i en funksjon, vil dette gå utover alle andre funksjoner som bruker denne variabelen

```
19 def summer(n1, n2):
20     print("sum: ", n1 + n2)
21
22 a = 3
23 b = 2
24 summer(a, b)
```

Oppgave: lokalt skop

```
29 def minus():
30     a = 3
31     b = 2
32
33 def summer():
34     print("sum: ", a + b)
35
36 summer()
```

- Hvorfor går ikke dette?

Oppgave: lokalt skop

- Hvorfor går ikke dette?
- Løsning:
  - a og b er kun definert i det lokale skopet til minus prosedyren. Altså kan vi ikke bruke disse utenfor minus(), og a og b er dermed udefinerte i summer() prosedyren

```
29 def minus():
30     a = 3
31     b = 2
32
33 def summer():
34     print("sum: ", a + b)
35
36 summer()
```

Variabler med samme navn

```
39 x = 300
40
41 def min_funksjon():
42     x = 200
43     print(x)
44
45 min_funksjon()
46
47 print(x)
```

Variabler med samme navn

Globalt skop:

x= 300

```
39 x = 300
40
41 def min_prosedyre():
42     x = 200
43     print(x)
44
45 min_prosedyre()
46
47 print(x)
```

Variabler med samme navn

Globalt skop:

x= 300

min_prosedyre (lokalt skop):



```
39 x = 300
40
41 def min_prosedyre():
42     x = 200
43     print(x)
44
45 min_prosedyre()
46
47 print(x)
```

Variabler med samme navn

Globalt skop:

x= 300

min_prosedyre (lokalt skop):



```
39 x = 300
40
41 def min_prosedyre():
42     x = 200
43     print(x)
44
45 min_prosedyre()
46
47 print(x)
```

Variabler med samme navn

Globalt skop:

x= 300

min_prosedyre (lokalt skop):

x= 200



```
39 x = 300
40
41 def min_prosedyre():
42     x = 200
43     print(x)
44
45 min_prosedyre()
46
47 print(x)
```

Variabler med samme navn

Globalt skop:

x= 300

min_prosedyre (lokalt skop):

x= 200

Output:
>200



```
39 x = 300
40
41 def min_prosedyre():
42     x = 200
43     print(x)
44
45 min_prosedyre()
46
47 print(x)
```

Variabler med samme navn

Globalt skop:

x= 300

min_prosedyre (lokalt skop):

x= 200

Output:
>200
>300



```
39 x = 300
40
41 def min_prosedyre():
42     x = 200
43     print(x)
44
45 min_prosedyre()
46
47 print(x)
```

Programmeringsstafett

Oppgaver programmeringsstafett

1. Lag en funksjon som tar inn en string og reverserer strengen før den returneres

2. Skriv en funksjon som tar inn en streng og sjekker om strengen er et pangram (Et pangram er en setning som inneholder alle bokstavene i alfabetet. Eksempel: Jeg begynte å fortære en sandwich mens jeg kjørte taxi på vei til quiz). Skal returnere True hvis strenger er et pangram False ellers

3. Lag en funksjon som sjekker om to tall er ved siden av hverandre på en tallrekke med heltall. Returner True om det stemmer.

4. Lag en funksjon som sjekker om tallene i en liste parvis er ved siden av hverandre på en tallrekke med heltall/"nabotall".. Returner hvor mange par som er nabotall i lista. Det vil si at om man har en liste med [1, 2, 4, 6, 4, 3, 9] så vil 1 og 2 være par, 4 og 6 vil være et par og 4 og 3 vil være et par. 9 ser vi bort ifra ettersom at den ikke er med et par. 1,2 og 4,3 er "nabotall", så 2 skal returneres fra denne funksjonen med [1, 2, 4, 6, 4, 3, 9] som parameter. (hint: bruk løsningen fra forrige oppgave)

5. Lag en prosedyre som tar inn et tall, og skriver tallet ut i streng format. Så 1 blir "en". 19 blir "en ni".

Jobbe med innlevering/Trix

- Ukens pensum:
  - Trix oppgave 06.01-06.14
    - https://trix.ifi.uio.no/course/8?tags=uke6
  - Innlevering 5

Temaer gjennomgått så langt :

Uke 1:
- Variabler
- Feilmeldinger
- Input fra tastatur
- Beslutninger

Uke 2:
- Data-typer
- Evaluering av utrykk
- Kodeflyt
- Prosedyrer

Uke 3:
- Samlinger og lister
- Mengder
- Ordbøker
- Nøstede samlinger

Uke 4:
- Løkker
- Kombinere løkker og samlinger
- For løkker
- While løkker

Uke 5:
- Lese fra og skrive til filer
- Parametere i prosedyrer
- Retur verdier

Uke 6:
- Skop

Neste uke: Objektorientert programmering