GRUPPETIME UKE 4

IN1000 GRUPPE 25 - 20.09.21

PLAN FOR GRUPPETIMEN

- Oblig 3 - hvordan gikk det?
- Jobbe sammen på MetroRetro
- Løkker
- Kombinere løkker og samlinger
- For-løkker
- Parametre i prosedyrer
- Retur-verdier

LÆRDOMMER FRA OBLIG 3

- Oppgave 3.3.
  - fant du den logiske feilen i oppgaven?
- Oppgave 4.3
  - forklar antagelser og begrunn!
- .pop() → sletter siste elemente i en liste
- Når du begynner å repetere kode
  - Prøv å forenkle koden med løkke og/eller prosedyre/funksjon

HVA SLAGS SAMLING?

Brukernavn på alle IN1000 studentene	Mengde (navn str) - ønsker ikke duplikater
Brukernavn og antall poeng på innlevering 3 for alle studentene på IN1000	Ordbok med brukernavn som nøkkel og poeng som verdi - knytte poeng til brukernavn
Alle vinnere i Lotto siste år (kun navn)	Liste (navn str) - kan være duplikater
All mat noen gjester i et selskap er allergisk mot (for å planlegge menyen)	Ordbok med navn som nøkkel og allergi som verdi - da kan vi lage ulik mat til gjestene

Prøv Trix oppgave 03.11

LÆRINGSMÅL UKE 4

- Kjenne til skrivemåte for while-løkker og for-løkker
- Kunne bruke løkker for å løse problemer
- Kunne bruke løkker sammen med samlinger
- Ha kjennskap til parametre i prosedyrer og parameteroverføring
- Kjenne til funksjoner og kunne bruke de for å få unngå redundans og få mer strukturerte programmer

WHILE LØKKER

- Er det sant? → Ja, gå inn og kjør kodeblokk → Er det sant? → Nei, gå videre

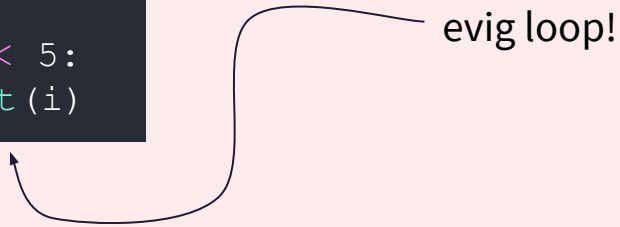
```
x = 5     # kan også endre slik at x er input fra bruker
i = 0     # i er telleren vår, veldig vanlig å bruke
          # variabelnavnet i for teller i løkker.

while i < x:       # kan leses som "så lenge i er mindre enn 5"
    print(i)       # skriver ut telleren vår
    i = i + 1      # deretter øker vi variabelen i med 1
                   # kan alternativt skrives som i += 1
                   # så gjentas løkken
```

Usikker hvordan koden kjøres? Prøv PythonTutor!

WHILE LØKKER

- I en while-løkke må det boolske uttrykket være modifiserbart
  - Dvs. at det bli evaluert til false på et eller annet tidspunkt, ellers så vil vi havne i en evig løkke
- Nyttig bruk for while-løkke er for eksempel en kommandoløkke, der man ikke avslutter før brukeren taster inn en spesifikk kommando.

```
i = 3
while i < 5:
    print(i)
```



Jobb sammen på Metro Retro

HVA SKRIVES UT?

```
tekst = ["hadet", "på", "badet", "din", "gamle",
"sjokolade"]
indeks = 0

while indeks < len(tekst):
    print(tekst[indeks])
    indeks += 2
```

>> hadet

>> badet

>> gamle

Indekseringen øker med 2 for hver runde.

HVA SKRIVES UT?

```
def skriv_historie(forst, andre, tredje) :
    print(forst, "dro paa ferie med ", tredje,
    "de ville dra uten ", andre, "  men ",
    andre, "snek seg med i  bagasjerommet..")

navn1 = "Silje"
navn2 = "Ole "
navn3 = "Jakob"
navneliste = ["Emilie", "Haakon", "Yulai"]
skriv_historie("Kari", "Per", "Martin")
skriv_historie(navneliste[0], navn3, navn1)
skriv_historie(navn2 + navn3, navneliste[1],navneliste[2])
```

>Kari dro paa ferie med Martin de ville dra uten Per, men Per snek seg med i bagasjerommet..

>Emilie dro paa ferie med Silje de ville dra uten Jakob, men Jakob snek seg med i bagasjerommet..

>Ole Jakob dro paa ferie med Yulai de ville dra uten Haakon, men Haakon snek seg med i bagasjerommet..

FOR LØKKER

- *for element in collection:*
  do this codeblock
- Kan bruke range() for å iterere x antall ganger
  - for i in range(3) *# Kjører 3 ganger → i er 0, 1, 2*
  - for i in range(2, 5) *# Kjører 3 ganger → i er 2, 3, 4*
  - for i in range (2, 15, 3) *# Kjører 5 ganger → i er 2, 5, 8, 11, 14*
- div = ["kamera", "lommebok", "pass", "mobillader"]
  for ting in div :
  print("Pakket med: ", ting)

FOR VS. WHILE

For brukes gjerne når man vet hvor mange ganger noe skal skje, feks:

- range()
- Iterere gjennom en samling

While kan gjerne brukes når man ikke vet hvor mange ganger noe skal skje, men feks.

- Hvis man vil lagre et ukjent antall verdier en bruker taster helt til brukeren taster 0, da er det fordelaktig å bruke while.

Funksjon	En funksjon som defineres med *def*, som ikke er del av en klasse (ordet *self* brukes ikke). Funksjoner har alltid en returverdi.
	`def sum(a, b):`
	`  c = a + b`
	`  return c`
	På engelsk kalles dette *function*

Prosedyre	Tilsvarende funksjon, men uten returverdi. Bruker aldri ordet *self*, og er ikke del av en klasse.
	`def superprint(ord):`
	`  print("ordet er", ord)`
	På engelsk kalles dette *procedure*

Metode	Tilsvarende funksjon, men som del av en klasse. Har alltid *self* som første parameter. Kan, men må ikke, ha en returverdi.
	`def areal(self):`
	`  firkant_areal = self.lengde * self.bredde`
	`  return firkant_areal`
	På engelsk kalles dette *method*

MED PARAMETERE

- Til prosedyrer/funksjoner kan man også sende med parametre, som er en type variabler.
  - Variablene man sender med “blir” de som er med som argument i prosedyren.
- *def prosedyre_navn(parameter1, parameter2, …)*
  gjør ting
- def summer(a, b):
  print("Sum: ", a + b)
  x = 3
  y = 2
  summer(x, y)

NB! Når prosedyren summer er ferdig så vil ikke a og b “beholde” verdiene sine. Fordel med prosedyrer -> kan gjenbruke kode! Kan kalle på en prosedyre flere ganger

FUNKSJON

```
def summer(a, b):
    return a + b

x = 3
y = 2
total = summer(x, y)
```

Når vi skriver “return” så “avsluttes” funksjonen.
Videre linjer i funksjonen kjøres da ikke.

Kjekt å vite: skop

PENSUM HITTIL...

print()
Variabler
Typer
assert()
input() og konvertering av typer
Forståelse av feilmeldinger
Prosedyrer
Lister → spesielt eksempler med lister og løkker
while-løkker og for løkker
Dictionary/ordbok, mengder

KONTAKT



...

sirisoll@uio.no
@sirisoll på Mattermost

UKENS MESTERVERK: “Ur-Venus”

- ***Om det er lov***

Bonus mesterverk fra gruppe 27: “Dinosaur”