GRUPPETIME
UKE 5

IN1000 GRUPPE 4 - 22.02.21

PLAN FOR GRUPPETIMEN

- Oblig 4 - hvordan gikk det?
- Jobbe sammen i Breakout Rooms
- Repetisjon
- Lese fra og skrive til filer
- Skop: globalt vs. lokalt

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

PROBLEMLØSNING MED KODING

- Les gjennom oppgaven og lag en outline i pseudokode
- Inkrementelle drafts med testing → pass på at en del av koden fungerer før du tester hele programmet
- Når hovedprogrammet er på plass kan vi begynne å teste hele programmet: fungerer det slik som beskrevet i oppgaveteksten?
- Finpuss: kan noe forenkles/forkortes? Har du kommentert underveis?

LÆRINGSMÅL UKE 5

- Kjenne til innlesing fra og utskrift til fil, inkludert organisering av informasjon innad på linjer
- Kunne bruke filer, løkker, samlinger og funksjoner for å løse mer sammensatte problemer

repetisjon

repetisjon

repetisjon

LISTER, MENGDER OG ORDBØKER

Hvis du føler deg usikker på en samling:
gjør Trix oppgaver! Det må til en del øving for at dette skal sitte.

Husk at for mengde/set: rekkefølge og antall forekomster er irrelevant!

```
{1, 2, 3} == {3, 3, 3, 2, 2, 1, 1}
tomt_set = set()
```

```
dict = {"frokost" : "banan",
        "lunch" : "pizza",
        "middag" : "burger"}

for key in dict:
   print(key)
   print(dict[key])

for key, value in dict.items():
   print(key, value)
```

Eksempler på hvordan vi kan iterere gjennom ordbøker. Se meals.py

FOR ELEMENT IN LIST

```
number_list = [1, 2, 3, 4, 5]
name_list = ["Kari", "Mari", "Alfred"]

for number in number_list:
    print(number)

# Bruker variabelnavn name
for name in name_list:
    print(name)

# Samme resultat - variabelnavn har ikke noe å si så
# lenge vi er konsistente, men det er god stil å ha et
# variabelnavn som passer til samlingen
for banana in name_list:
    print(banana)
```

WHILE-LOOP: gjenta til False

```
item_list = []
item = input("Add item or press 's' to stop: ")
while item != "s":
    item_list.append(item)
    item = input("Add item or press 's' to stop: ")
print("Your items are:", item_list)
```

→ *while loops er nyttige når vi ikke vet hvor mange ganger noe skal skje!*

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.

NB: break og “while true” er ofte regnet som dårlig stil - unngå å bruke det, eller les om det først!

FUNKSJONER

- Husk return inne i funksjonen, ellers er det ingen funksjon
- Husk å lagre reutrverdien i en vairabel → her, tekst
- **NB!** En funksjon kan kun returnere én ting!
    - Dersom man ønsker å returnere mer enn 1 ting må man lage en liste, legge tingene i listen og returnere hele listen.

```
hilsen = "Hallo"

def leggtil_hei(tekst_inn):
    tekst_ut = tekst_inn + " hei"
    return tekst_ut

tekst = leggtil_hei(hilsen)
print(tekst)
```

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*

SKOP

- “Skop” vil si den delen av programmet du kan aksessere/få tilgang på en variabel.
  - En variabel som er definert i en prosedyre kalles for en lokal variabel.
  - Defineres variabler utenfor en prosedyre vil den være en global variabel og dermed tilgjengelig for alle.
- Generelt burde man bruke lokale variabler og heller lage prosedyrer som returnerer verdier fremfor globale variabler som oppdateres av mange prosedyrer.

```
hilsen = "Hallo"

def leggtil_hei(tekst_inn):
    tekst_ut = tekst_inn + " hei"
    return tekst_ut

tekst = leggtil_hei(hilsen)
```

GLOBALE VS. LOKALE VARIABLER

```
# Hva skrives ut?

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

a = 3
b = 2
summer()
>> sum: 5
```

```
# Hva skrives ut?

def minus():
    a = 3
    b = 2


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

summer()
>> NameError: name 'a' is not defined
```

JOBB SAMMEN I BREAKOUT ROOMS!

→ uke 5 oppgaver

FILER

Eks. tenk at test.txt består av følgende linjer:

Hei!

Dette er en fil.

- For å **lese** følgende fil:

```
innfil = open("test.txt", "r")
linje = innfil.readline() # Første linje i filen leses, dvs. "Hei!"
```

  - Når denne så lagres i variablen “linje” lagres den som → Hei!\n
  - Når det ikke lenger er noe innhold i filen, dvs. Tom linje (“ ”), så leses dette som “\n” av filleseren.

- **Skrive til fil**

```
utfil.write("Skriv dette til fil\n")
linje = innfil.readline()
while linje != "" :   #  Lese helt frem til tom linje/slutten av filen
      #  <gjøre noe med variabelen "linje"/det som er lest inn>
      linje = innfil.readline() # lese neste linje i filen, repeter while-løkken
```

FILER: syntax

Når vi skal bruke filer, enten lese fra dem eller skrive til dem, må vi åpne og lukke dem:

Åpne og lese en fil:

```
innfil = open("input.txt", "r") # "r" står for "read"
```

Skrive til en fil:

```
utfil = open("filnavn.fitype", "w")    # w står for "write"
```

Skrive til ved å legge til:

```
utfil = open("filnavn.filtype", "a")   # a står for  "append"
```

Lukke fil:

```
innfil.close()   # lukker når man er helt ferdig med filen.
```

Lese fra fil:

.readline() vil lese en linje fra oppgitt fil som en String, her lagres denne string-verdien i variabelen “linje”

```
linje = innfil.readline()
```

FILER

- **Konvertere til tall:**
  Når man leser en linje fra en fil så leses det som string, dersom man ønsker å konvertere til tall skriver man
  - `verdi = float(linje)  # konverterer til flyttall/float`
  - `verdi = int(linje)    # konverterer til heltall/int`
  - \n vil da ignoreres!

lag_dna.py

```
def skriv_dna(filnavn):
    fil = open(filnavn, "w")
    fil.write("O---o\n O-o\n  O\n
o-O\no---O\n")
    fil.close()
skriv_dna("dna.txt")

def les_dna(filnavn):
    fil = open(filnavn, "r")
    for linje in fil:
        linje = linje.rstrip()
        print(linje)
les_dna("dna.txt")
les_dna("dna.txt")
les_dna("dna.txt")

print("\nEn annen måte å lese filer: " )
def les_dna2(filnavn):
    fil = open(filnavn, "r")
    linje = fil.readline().rstrip()
    while linje != "":
        print(linje)
        linje = fil.readline().rstrip()
    fil.close()
les_dna2("dna.txt")
les_dna2("dna.txt")
```

→ *Basert på Trix 5.02*

KONTAKT



...

sirisoll@uio.no
@sirisoll på Mattermost

UKENS MESTERVERK: