UKE 5 SEMINAR

FILINNLESING, FUNKSJONER OG SKOP

LESE FRA FIL

- Filen må alltid åpnes og lukkes
    - fil = open(«filnavn.filtype»)
    - fil.close()

- .strip() fjerner linjeskift, \n, og whitespace
- .split() skiller på mellomrom og returnerer en liste

LESE FRA FIL

- To måter å iterere over en fil
- «r» betyr «read»

```
fil = open("test.txt", "r")

# Leser hver linje
for linje in fil:
    linje = linje.strip()
    print(linje)

fil.close()
```

```
fil2 = open("test.txt", "r")

# Leser første linje
linje = fil2.readline()
while linje != "":
    linje = linje.strip()
    print(linje)
    # Leser neste linje
    linje = fil2.readline()

fil2.close()
```

LØS I PAR

- Anta at du har filen "historie.txt".
- Les inn alle linjene og lagre dem i en liste.
- Første linje skal ligge først i listen, osv. (husk å åpne og lukke filen..)

```
# Løsningsforlag
innfil = open("historie.txt", "r")
historie_liste = []
linje = innfil.readline()
while linje != "" :
    linje = linje.strip()
    historie_liste.append(linje)
    linje = innfil.readline()

innfil.close()
```

```
# Løsningsforlag
innfil = open("test.txt", "r")
historie_liste = []
for linje in innfil:
    linje = linje.strip()
    historie_liste.append(linje)

innfil.close()
```

LØS I PAR

- Skriv en funksjon som tar i mot to sannhetsverdier (True/False).
  Funksjonen skal returnerer 1 hvis begge verdier er sanne og 0 hvis begge verdier er usanne.
  Bruk assert til å teste funksjonen for alle mulige tilfeller.

- Hva tar ikke funksjonen høyde for?
  Hva kan gå "galt"?

```
def er_det_sant(bool1, bool2):
    if bool1 and bool2:
        return 1
    elif bool1 or bool2:
        return -1
    return 0

assert er_det_sant(True, False) == -1
assert er_det_sant(True, True) == 1
assert er_det_sant(False, False) == 0
```

FUNKSJONER

Funksjon	En funksjon som defineres med *def*, som ikke er del av en klasse (ordet *self* brukes ikke). Funksjoner har alltid en returverdi. Eks: *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. Eks: *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. Eks: *def areal(self):* *firkant_areal = self.lengde * self.bredde* *return firkant_areal* På engelsk kalles dette *method*.

Metoder kommer på pensum om noen uker!

LØS I PAR

- Skriv et program som definerer en liste [2, 3, 6, 8].
  Bruk en for-løkke til å skrive ut alle verdiene i listen.

- Skriv om til en prosedyre som tar en liste som input/parameter og som skriver ut alle verdiene i listen.

- Endre prosedyren til en funksjon som returnerer den laveste verdien i listen i stedet for å skrive dem ut.
  Hint: her kan det være lurt å ha en variabel som mellomlagring av den minste verdien.

LØSNINGSFORSLAG

```
liste = [2, 3, 6, 8]
for tall in liste:
    print(tall)

def skrivUtListe(liste):
    for tall in liste:
        print(tall)
```

```
def hentMinsteTall(liste):
    minsteTall = liste[0]
    for tall in liste:
        if tall < minsteTall:
            minsteTall = tall
    return minsteTall
```

LØS I PAR

- Skriv en funksjon som tar i mot to tall og returnerer det største av de to tallene (ikke skriv det ut)
- Lag en main prosedyre som kaller på funksjonen din

```
# Løsningsforlag
def storst_av_to(tall1, tall2):
    if tall1 > tall2:
        return tall1
    else :
        return tall2

# Et bedre løsningsforlag
def storst_av_to(tall1, tall2):
    if tall1 > tall2:
        return tall1
    return tall2
```

```
def main():
    storst_av_to(3, 4)

main()
```

SKOP

Skop	Sier noe om hvor en variabel er tilgjengelig.
Lokal variabel	En variabel som er definert inne i en prosedyre. Kun tilgjengelig inne i prosedyren.
Global variabel	En variabel som er definert utenfor en prosedyre. Tilgjengelig for alle.

Generelt burde man bruke lokale variabler og heller lage prosedyrer som returnerer verdier fremfor globale variabler som oppdateres av mange prosedyrer.

Skop er veldig viktig, siden det sier noe om hva deler av programmet "ser"

SKOP

Skop sier noe om hvor en variabel er tilgjengelig.

```
# Dette funker:
globalVariabel = 1
def printVariabel():
    print(globalVariabel)
printVariabel()
# >> 1

# Dette funker ikke:
globalVariabel = 1
def lagVariabel():
    lokalVariabel = 2
print(lokalVariabel)
# >> Error
```

```
# Dette funker:
globalVariabel = 1
def printVariabel1():
    print(globalVariabel)
def printVariabel2():
    print(globalVariabel)

# Dette funker ikke:
def lagVariabel():
    lokalVariabel = 2
def printVariabel():
    print(lokalVariabel)
# >> ERROR
```

```
# Dette funker ikke
def lagVariabel():
    lokalVariabel = "banan"
    return lokalVariabel
def printVariabel():
    nyVariabel = lagVariabel()
    print(lokalVariabel)
# >> ERROR lokalVariabel undefined

# Dette funker:
def lagVariabel():
    lokalVariabel = "banan"
    return lokalVariabel
def printVariabel():
    lokalVariabel = lagVariabel()
    print(lokalVariabel)
# >> banan
```

GLOBALE VARIABLER

Unngå så langt det lar seg gjøre

å bruke globale variable.

```
# Dette funker, men dårlig stil!
globalVariabel = 1
def printVariabel():
    print(globalVariabel)
printVariabel()

# Dette funker ikke:
def lagVariabel():
    lokalVariabel = 2
lagVariabel()
print(lokalVariabel)
# >> ERROR
```

```
# Dette er dårlig stil:
globalVariabel = 5
def sumAvVariabler(lokalVariabel1, lokalVariabel2):
  summen = gloablVariabl + lokalVariabel1 + lokalVariabel2
  return summen
print(sumAvVariabler(1, 2))

# Dette er enda verre:
globalVariabel1 = 5
globalVariabel2 = 1
globalVariabel3 = 2
def sumAvVariabler():
  summen = gloablVariabl + globalVariabel2 + globalVariabel3
  return summen
print(sumAvVariabler())

# Dette er bra stil:
def sumAvVariabler(lokalVariabel1, lokalVariabel2, lokalVariabel3):
  summen = lokalVariabel1 + lokalVariabel2 + lokalVariabel3
  return summen
print(sumAvVariabler(1, 2, 5))
```

EN LITT VANSKELIGERE OPPGAVE

- Denne oppgaven er åpen. Dere får et problem og kan velge hvordan dere vil løse den selv.

- Oppgave:
  Gitt filen gruppe<x>Navn.txt spør programmet (i terminalen) om studentene er tilstede.
  Hvis brukeren taster inn x er studenten der, ellers er studenten ikke stede.
  Oppmøtet skal registreres i en annen fil som heter uke<ukenummer>.txt (du må spørre brukeren om hvilken uke det er). Her skal alle navnene skrives til fil, men hvis brukeren er til stede skal det stå en x bak navnet.

- Filen som man skriver ut til skal f.eks. se slik ut:
  Tiril Torgren: x
  Sandra Frogn:
  Emil Orgensbø: x

TRIX OG OBLIG