Velkommen :)

...

In1000 - Johanna
johannph@uio.no

Hvem skal jeg kontakte?

Spørsmål om faget:
johannph@uio.no(meg) eller foreleser.

Studieinfo: spørsmål, klager, utsettelse av frister
https://www.mn.uio.no/ifi/studier/kontakt/

UiO forvei: bekymringer, noen å snakke med
https://www.mn.uio.no/studier/forvei/

SiO: fysisk og psykisk helsehjelp
https://www.sio.no/helse
https://www.sio.no/helse/noen-%C3%A5-snakke-med

Praktisk

Håper dere leverte oblig 4! Hvis dere synes obligene er for vanskelige så bare ta kontakt med meg så skal jeg hjelpe så godt jeg kan!

Mattermost: har du joina gruppekanalen?

Se på emnesiden: oversikt over temaer

Uke 4

Eksempel på while-loop

```
tall = 0
while tall < 5:
    print(tall)
    tall += 1


userInput = ""
while userInput != "exit" and userInput != "quit" and userInput != "ja":
    print("Programmet fortsett her kan vi gjøre skøy ting, f.eks. kalle på metoder")
    userInput = input("Vil du avslutte programmet? ").lower()
```

Eksempel på “for in range()”-loop

```
print("Print 1 til 6")
for tall in range(1, 7):
   print(tall)

print("Print 0 til 6, med step 2")
for tall in range(0, 7, 2):
   print(tall)

print("Print 6 til 0")
for tall in range(6, -1, -1):
   print(tall)
```

Eksempel på “for in list”-loop

```
numberList = [1, 2, 3, 4, 5]
nameList = ["Kari", "Mari", "Alfred"]

for number in numberList:
   print(number)

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

# Bruker variabelnavn banan, får samme resultat
for banana in nameList:
   print(banana)
```

For vs. while

- For brukes gjerne når man vet hvor mange ganger noe skal skjer.
- 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.

For- og while-looper

Dere må ha god kontroll på hvordan man bruker dem, og når man burde bruke de ulike loopene.

Break og “while true” er ofte regnet som dårlig stil, unngå å bruke det, eller les om det først! Det er dårlig stil fordi alternativet som regel er mer leselig.

while true vs. while input != 0, hva gir mest informasjon?

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*.

Funksjoner eksempel

Husk return inne i funksjonen, ellers er det ingen funksjon!

Husk å lagre returverdien i en variabel! Her kalt tekst!

```
tekst = "Hallo"

def leggTilHei(tekstInn):
    tekstUt = tekstInn + "hei"
    return tekstUt

tekst = leggTilHei(tekst)
print(tekst)
```

Oblig 4

Vanlige “feil” på oblig

Lage unødvendige lister/ordbøker. Det er ikke gratis å opprette lister og ordbøker.

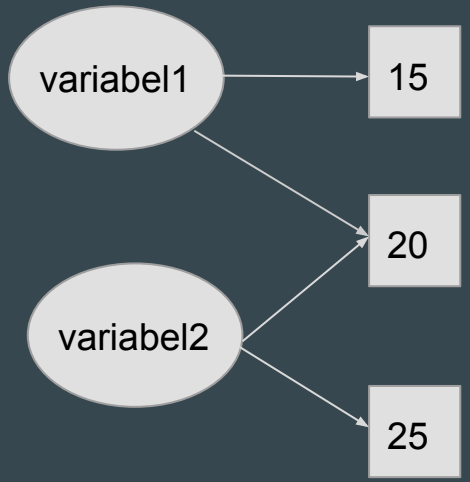
Dårlige variabelnavn (x, y, a, b ,c) (og forsåvidt i1...)

Bruk av while-loop istedenfor for-loop når vi vet hvor mange ganger noe skal skje.

Uke 5

Først litt basic om variabler

Stringer og tall er immutable, altså man kan ikke endre dem!

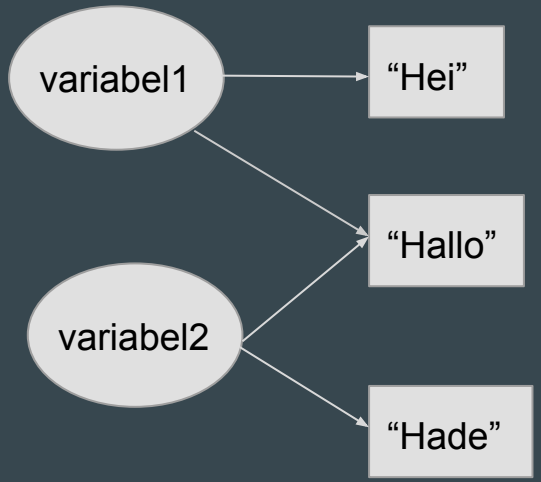



```
variabel1 = 15
variabel1 = 20
variabel2 = variabel1
# Det over er akkuratt det samme
som å si: variabel2 = 20
variabel2 = 25
```

Dette er en animasjon! Gir ikke mening hvis man ikke ser den som animasjon! Last ned pptx-versjon av foilene.

Stringer og tall er immutable!

Samme for stringer:

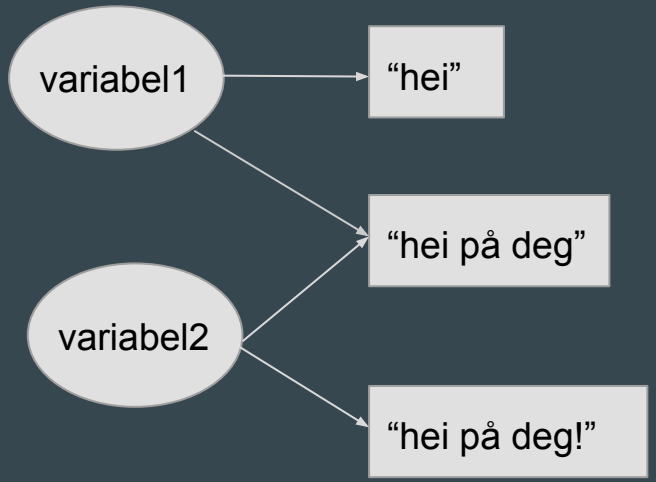



```
variabel1 = “hei”
variabel1 = “hallo”
variabel2 = variabel1
# Det over er akkuratt det samme
som å si: variabel2 = “hallo”
variabel2 = “hade”
```

Dette er en animasjon! Gir ikke mening hvis man ikke ser den som animasjon! Last ned pptx-versjon av foilene.

Stringer og tall er immutable!

Samme hvis vi plusser! Vi lager alltid **helt nye** strenger!




```
variabel1 = “hei”
variabel1 += “på deg”
variabel2 = variabel1
# Det over er akkuratt det samme
som å si: variabel2 = “hei på deg”
variabel2 += “!”
```

Dette er en animasjon! Gir ikke mening hvis man ikke ser den som animasjon! Last ned pptx-versjon av foilene.

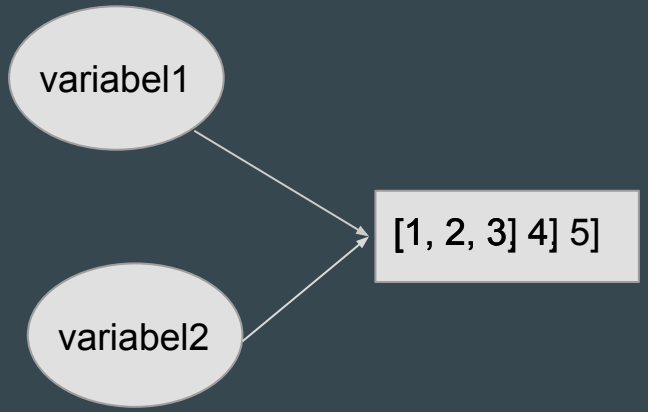
immutble vs. mutable

Så husk: strenger og tall er immutable, de kan ikke endres! Kanskje det ser ut som vi endre en string eller et tall, men egentlig lager vi en ny!

Lister, dictionaries og mange andre typer er derimot mutable, altså vi kan endre dem!

Først litt basic om variabler

Stringer og tall er immutable, altså man kan ikke endre dem!




```
variabel1 = [1, 2, 3]
variabel1 = variabel1.append(4)
variabel2 = vari abel1
# Det over er IKKE det samme som å
si: variabel2 = [1, 2, 3, 4]
variabel2 = variabel2.append(5)
print(variabel1)
# Kontrollspm: Hva printes?
```

Dette er en animasjon! Gir ikke mening hvis man ikke ser den som animasjon! Last ned pptx-versjon av foilene.

immutble vs. mutable

Så husk: strenger og tall er immutable, de kan ikke endres! Kanskje det ser ut som vi endre en string eller et tall, men egentlig lager vi en ny!

Lister, dictionaries og mange andre typer er derimot mutable, altså vi kan endre dem!

Skop

Skop sier noe om hvor en variabel er tilgjengelig!
Hva er skopet til disse variablene?

```
# 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)
>> banana
```

Globale variabler

Unngå så langt det lar seg gjøre å bruke globale variabler, spesielt i prosedyrer/funksjoner:

```
# 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 = gloablVariabel + 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))
```

Lese fra fil!

```
#Syntaks: Lese en fil:
innfil = open(“filnavn.filtype”, “r”)    # “r” står for “read”

# Gitt at det finnes en fil som heter “input.txt”, hvis vi skal lese den skriver vi:
   innfil = open(“input.txt”, “r”)
#    Skrive til en:
   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.
   utfil.close()

# LESE FRA FIL:
   linje = innfil.readline()
# .readline() vil lese en linje fra oppgitt fil som en String, her lagres denne string-verdien i variabelen
“linje”.
```

Gjøre oppgaver!

Veldig lurt å gjøre ukesoppgavene denne uka!! Deler dem på mattermost.