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

Pls lever oblig4!! Send meg en melding på mattermost/mail hvis jeg kan hjelpe med noe!

Mattermost: har du joina gruppekanalen?

Se på uke4 på emnesiden

Opprop!

Kahoot: repetisjon

Uke 3

Lister

tallListe = [2, 4, 6, 7]

navneListe = [“Anne”, “Per”, “Lisa”]

tomListe = [ ]

Lister er dynamiske, du kan endre størrelse på dem ved å legge til eller fjerne et element.

Operasjoner på lister

len(<liste>) # gir lengden på listen

liste.insert(index, element) # sette inn element på angitt indeks

liste.append(element) # setter inn nytt element på slutten av listen

liste.remove(element) # fjerne angitt element fra listen

liste.pop(index) # fjerner (og returnerer) element på angitt indeks
(liste.pop(-1): sletter alltid det siste elementet(!))

print(“Liste: “, liste) # skriver ut> Liste: 1, 2, 3

Mengder/sets

Rekkefølge og antall forekomster er irrelevant.

tomMengde = set()

navnMengde = {“Per”, “Pål”, “Espen”}

tallMengde = {1, 2, 3}

Da rekkefølge og antall forekomster ikke betyr noe får vi at:
{1, 2, 3} == {1, 3, 2} == {1, 1, 1, 3, 3, 3, 2, 2} == {3, 2, 1}

Operasjoner på mengder

ny_mengde = set() eller mengde_av_liste = set(liste)

menge.add(element) # legge til nytt element i mengden

mengde.discard(element) # fjerner element fra listen, gjør ingenting dersom elementet ikke finnes i listen.

mengde.remove(element) # fjerner element fra listen

mengde.clear() # fjerner alle elementer i mengden, gjør mengden tom.

len(<mengde>) # finner antall elementer i mengden.

Ordbøker/dictionaries

Tenk på en vanlig ordbok, typ norsk-engelsk-ordbok.

Har en nøkkel og en verdi.

F.eks. hund: dog

ordbok = {nøkkel : verdi, nøkkel2 : verdi2, ...}

Kan fylles med alt mulig:

quizSpm = {“Er en banan en frukt eller grønnsak (frukt/grønnsak)” : “frukt”, “Flyter bananer i vann? (ja/nei)”: “ja”}

Operasjoner på ordbøker/dictionaries

tomOrdbok = {}

ordbok[nokkel] = verdi # legge til nytt element i

ordboken ELLER endre på eksisterende.

Eks: kontakter[“Per”] = 45667990 # dersom “Per” allerede finnes i ordboken vil

verdien til “Per” endres.

ordok.pop(nøkkel) # fjerne nøkkelen og verdien.

Eks: kontakter.pop(“Per”)

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 1")
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.

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*.

Oppgaver

Diskuter oppgaver i par

La alle tenke litt før dere diskuterer

Skriv gjerne ned det du tror

Diskuter til slutt

Skriv ned det dere kommer fram til på et ark

Hva printes?

1)

```
def adder(parameter):
    parameter = 5 + parameter

tall = 10
tall = adder(tall)
print(tall)
```

2)

```
def adder(parameter):
    return 5 + parameter

tall = 10
adder(tall)
print(tall)
```

3)

```
def adder(parameter):
    return 5 + parameter

tall = 10
tall = adder(tall)
print(tall)
```

Hva printes?

```
minListe = [["a", "b"], ["c", "d"], ["e", "f"]]
print(minListe[0])
print(minListe[0][1])
print(minListe[1])
print(minListe[1][1])

indreListe = minListe[0]
print(indreListe[0])
```

Hva printes?

```
x = 3

while x < 10:
    x += 1

print(x)
```

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

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

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

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

Parprogrammering

Skriv et program med en prosedyre printHei, prosedyren skal skrive ut teksten “Hei” til skjermen. Definer en variabel a med en verdi 5 og lag en while-løkke, løkken skal kalle prosedyren printHei. Bruk variabelen a slik at løkken er ferdig etter du har kalt print_hei 5 ganger.

Skriv prosedyren printHei. Deretter løs oppgaven både ved hjelp av while-løkke og for-løkke.

Parprogrammering

1. Skriv en prosedyre “storst_av_to” som tar imot to tall som parametre og skriver ut verdien til det største tallet.
2. (Du skal nå endre på oppgave 8): Skriv en funksjon “storst_av_to” som tar imot to tall som parametre og returnerer det største tallet. Deretter skrives tallet ut etter funksjonskallet.