



# GRUPPETIME

## UKE 3

**IN1000 GRUPPE 4 - 08.02.22**



# PLAN FOR GRUPPETIMEN

- Oblig 1 - hvordan gikk det?
- Ojekter tilbyr tjenester
- Jobbe sammen på MetroRetro
- Lister
- Nøstede lister
- Mengder
- Ordbøker



# LÆRDOMMER FRA OBLIG 1

- Setup - done
- Mestringsfølelsen - programmering er gøy
- Forskjellig emåter å løse et problem!
  - Feks dato.py

# LÆRINGSMÅL UKE 3

- Vite om begrepet objekter, samt at objekter tilbyr tjenester
- Kunne bruke samlinger for å holde på mange verdier
- Kjenne til forskjellene mellom lister, mengder og ordbøker, samt være i stand til å benytte egnet type samling for et gitt formål



# OBJEKTER tilbyr tjenester

- Et objekt tilhører en klasse
  - en tekststreng er et **objekt** av klassen `str`
  - `volvo` kan være et objekt av klassen `Bil` - vi kan lage våre egne klasser
- Gjennom **metoder** tilbyr objekter tjenester
  - en metode ligner på en funksjon/prosedyre - men den er alltid knyttet til et objekt
    - `objekt.metode(argumenter)`
      - Altså, vi kaller på en metode med dot-notasjon
        - Feks: `streng.upper()` eller `liste.len()`

# NOEN NYTTIGE METODER SOM STR TILBYR

```
hei = "heisann sveisANN!"  
hei.upper() # STORE BOKSTAVER  
hei.lower() # små bokstaver  
hei.count("n") # Teller forekomster  
hei.find("sveisANN") # Finds first index of occurrence of value  
hei.replace("heisann", "hade") # Erstatter heisann med hade  
hei.title() # Gjør om til overskrift -> Heisann Sveisann
```

```
bare = "      fjern      BARE mellomrom      foran og bak      "  
bare.strip() # Fjerner blanklines foran og bak i strengen
```

```
frokost = "banan, yoghurt, pannekaker, bacon, cereal"  
print(frokost.split(", ")) # Splitter en streng til en liste på x
```

→ Les om flere str metoder [her](#)



# LISTER

- Lister har dynamisk størrelse
  - Dersom man legger til ett element blir listen en størrelse større, dersom man fjerner ett element blir listen en størrelse mindre.
- Man kan ha lister med forskjellige elementer:
  - ```
tall_liste = [2, 4, 6, 7]
navne_liste = ["Anne", "Per", "Lisa"]
misc_liste = ["A", 1, 9.2, "dette er bare ett element i lista"]
tom_liste = []
```



# **JOBBSAMMEN I BREAKOUT ROOMS**

→ uke 3 oppgaver





# OPPGAVE 1: Hvor mange elementer?

```
liste = [0] # 1  
  
liste1 = [1, 2, 3] # 3  
  
liste2 = [3, 3, 5, 7] # 4  
  
liste3 = ["A", "BC", "D", "E", "F"] # 5  
  
liste4 = ["mange elementer i denne listen"] # 1  
  
liste5 = [] # 0/tom
```

→ Vi kan bruke `len(liste)` for å finne antall elementer i en liste.

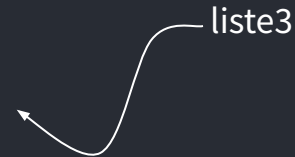
# OPPGAVE 2: liste indeksering

```
# 2.1 Hva ligger på indeksene?  
liste3[4] # "F"  
liste1[0] # 1  
liste4[1] # "mange elementer i denne listen"  
liste3[5] # ingenting, er kun 5 elementer, på indeksene 0-4.
```

```
# 2.2 På hvilken indeks ligger  
liste2[2] # ← 5 i liste2  
liste3[1] # ← "BC" i liste3  
liste1[0] # ← 1 i liste1  
liste3[3] # ← "E" i liste3
```

```
liste = [0]  
liste1 = [1, 2, 3]  
liste2 = [3, 3, 5, 7]  
liste3 = ["A", "BC", "D", "E",  
"F"]  
liste4 = ["mange elementer.."]  
liste5 = []
```

|   |      |
|---|------|
| 0 | "A"  |
| 1 | "BC" |
| 2 | "D"  |
| 3 | "E"  |
| 4 | "F"  |



# OPPGAVE 3: liste metoder

```
# 3.1 Legg til "Kari" i lista 'navn'  
navn = ["Ola", "Martin", "Selma"]  
navn.append("Kari")
```

```
# 3.2 Skriver ut andre elementet i lista  
print(navn[1])
```

```
# 3.2 Legger til Lise og Kai først i lista  
navn.insert(0, "Lise")  
navn.insert(0, "Kai")
```

```
# 3.4 Fjerner Martin fra lista  
navn.remove("Martin") # alternativt navn.pop(3)  
print("Alle navn: ", navn)
```

```
# Bonus: ["Kai", "Lise", "Ola", "Selma", "Kari"]
```

|   |         |
|---|---------|
| 0 | "Kai"   |
| 1 | "Lise"  |
| 2 | "Ola"   |
| 3 | "Selma" |
| 4 | "Kari"  |

listen 'navn' etter alle  
kodelinjene har kjørt

# OPERASJONER PÅ LISTER

```
len(liste) # Gir lengden på listen
liste.insert(index, element) # Sette inn element på angitt index
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) # Da slettes det siste elementet(!)
print("Liste: ", liste) # Skriver ut> Liste: 1, 2, 3
```

→ Les om flere liste metoder [her](#)



# NØSTEDE LISTER

- Kan sees på som lister i lister

## *Koffert eksempel*

- Vi skal ha med oss toalettsaker, klær og en liten bag med diverse
  - `toalettsaker = ["tannborste", "haarborste", "tannkrem"]`
  - `klaer = ["jakke", "bukse", "sokker", "undertoy"]`
  - `div = ["kamera", "lommebok", "pass", "mobillader"]`
- For å samle alle småbaggene når vi skal på tur legger vi dem i en koffert:
  - `koffert = [toalettsaker, klaer, div]`

# NØSTEDE LISTER: koffert eksempel

Hvordan få tak i ett element?

koffert[0][2] → tannkrem

koffert[2][1] → lommebok

koffert[1][1] → bukse

koffert[2][0] → kamera

koffert[0][0] → tannborste

Hvordan henter man frem?

“mobillader” → koffert[2][3]

“sokker” →

“haarborste” →

koffert

|   |              |
|---|--------------|
| 0 | toalettsaker |
| 1 | klaer        |
| 2 | div          |

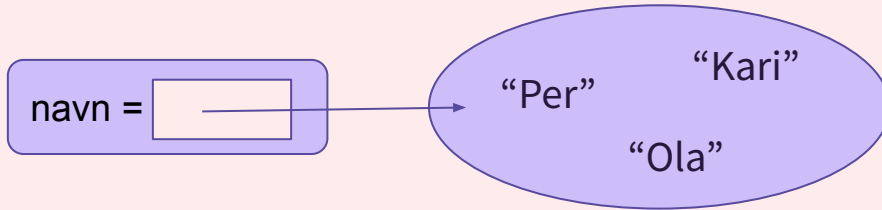
|   |              |
|---|--------------|
| 0 | "tannborste" |
| 1 | "haarborste" |
| 2 | "tannkrem"   |

|   |            |
|---|------------|
| 0 | "jakke"    |
| 1 | "bukse"    |
| 2 | "sokker"   |
| 3 | "undertoy" |

|   |              |
|---|--------------|
| 0 | "kamera"     |
| 1 | "lommebok"   |
| 2 | "pass"       |
| 3 | "mobillader" |

# MENGDER

- Lages nesten likt som en liste. Syntaks:
  - `mengde = {element, element2, element3, ...}`  
`navn = {"Kari", "Per", "Ola"}`
- Mengder har ingen spesifikk rekkefølge på elementene, ingen indeksering.
- Mengder har **INGEN** duplikater



- Tom mengde lages slik: `mengde = set()`
  - **IKKE** `mengde = {}` ← *fordi dette er en tom ordbok!*

# MENGDER VS. LISTER

| MENGDER/SET                                                                                                                                  | LISTER                                                                                                                                                               |
|----------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>navn = {"Kari", "Per", "Ola"}<br/>navn2 = {"Per", "Ola", "Kari"}</pre>                                                                  | <pre>navn = ["Kari", "Per", "Ola"]<br/>navn2 = ["Per", "Ola", "Kari"]</pre>                                                                                          |
| <p>if navn == navn2 : ....</p> <p>evalueres til <b>sant</b> fordi begge inneholder de samme elementene, sett er uavhengig av rekkefølgen</p> | <p>if navn == navn2 : ....</p> <p>evalueres til <b>usant</b> selv om begge inneholder de samme elementene, så er de i ulik rekkefølge og dermed ikke like lister</p> |
| <p>Kan ikke ha duplikater</p> <p>Ingen spesifikk rekkefølge</p> <p>Kan ikke indekseres</p>                                                   | <p>Kan ha duplikater</p> <p>Spesifikk rekkefølge</p> <p>Kan indekseres</p>                                                                                           |



# OPERASJONER PÅ MENGDER

```
mengde = set()
mengde_av_liste = set(liste)
mengde.add(element) # Legge til nytt element i mengden
mengde.remove(element) # Elementet må være i mengden for å fjernes
mengde.discard(element) # Fjerner element fra mengden uansett
mengde.clear() # Fjerner alle elementer i mengden, gjør mengden tom
len(mengde) # Finner antall elementer i mengden.
```

→ Les om flere set metoder [her](#)

# OPPGAVE 4: mengder

```
# Fiks følgende kode slik at det kjører uten feilmelding
min_mengde = {1, 3, 2}
print("Min mengde:", min_mengde)
print("Består av" ,len(min_mengde) ,"elementer")
>> Min mengde: {1, 2, 3}
>> Består av 3 elementer
```

```
# Kjør følgende kode: hva ser du, og hvorfor er verdien slik den er?
print(min_mengde == {1, 2, 3, 2, 3})
>> True
```

```
# Skriv ut tall_liste til terminalen. Hva har skjedd?
tall_liste = [1, 1, 2, 3, 4, 4]
tall_liste = list(set(tall_liste))
print(tall_liste)
>> [1, 2, 3, 4]
```

# ORDBØKER

- Er en beholder som holder på par av nøkler og verdier.
  - Hver nøkkel i ordboken er knyttet til én verdi.
- Syntaks:  
ordbok = {nøkkel : verdi, nøkkel2 : verdi2, .....}
  - kontakter = {"Kari" : 47543234, "Ola" : 76554532, "Martin" : 99543367}
- I stedet for å bruke indeksering som i lister, bruker man nøkkelen (i dette eksempelet navn) for å hente ut verdier.
  - print("Kari sitt telefonnummer er ", kontakter["Kari"])
- tom\_ordbok = {}

# OPERASJONER PÅ ORDBØKER

```
tom_ordbok = {}
grades = {"Per": 98, "Pål": 55, "Espen": 78}

# Legge til nytt element ELLER endre på eksisterende:
ordbok[nøkkel] = verdi
grades["Per"] = 100 # Endrer verdien til Per
grades["Askeladden"] = 34 # Legger til Askeladden i ordboka
grades

# Fjerne nøkkelen og verdien med ordbok.pop(nøkkel)
grades.pop("Per")

# Sjekker om en nøkkel finnes i ordboken
print("Per" in grades)

# ordbok.get() henter verdien til en nøkkel
print(f"Espen fikk karakteren: {grades.get('Espen')}")
```

→ Les om flere ordbok metoder [her](#)

# OPPGAVE 5: ordbøker

```
# 5.1 definerer ordboken brukere
brukere = {"hanjo": "Hanne Johansen", "karsi": "Kari Sirisen", "olha": "Ole Hansen"}

# 5.2 endrer navnet til karsi
brukere["karsi"] = "Kari Marie Sirisen"

# 5.3 registrer deg selv som bruke
brukere["sirisoll"] = "Siri Sollerud"

# 5.4 skriv ut hanjo sitt navn
print(brukere["hanjo"])

# 5.5 fjern Hanne Johansen fra ordboken
brukere.pop("hanjo")
```

# OPPGAVE 6: forbedre koden

```
def f1() :  
    i = float(input("input"))  
    print(i*i)
```

```
def f2() :  
    i = input()  
    print("velkommen til in1000 ", i)
```

```
def f3( )      :  
    print (    "Hei!"    )
```

```
f3()  
f2()  
f1()
```

- Generelt dårlig funksjonsnavn og variabelnavn, lite beskrivende
- f1 kan vi kalle “multipliser()”
  - Gi bedre beskjed i terminalen til bruker om hva input skal være (ett tall)
- f2 kan vi kalle velkommen()
  - Endre variabelnavn i til navn
  - Gi bedre beskjed i terminalen til bruker om hva input skal være (et navn)
- f3 kan vi kalle hei()
  - Mange unødvendige mellomrom inni parenteser, disse syns ikke i utskriften uansett.

# HVA SLAGS SAMLING?

|                                                                    |                                                                                                |
|--------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
| Ordene i en setning                                                | liste med stringer, fordi rekkefølgen er viktig og et ord kan forekomme to ganger i en setning |
| De latinske navnene på alle forskjellige spiselige soppene i Norge | mengde med stringer, fordi vi ikke vil ha duplikater                                           |
| Navnene på alle klassekameratene dine, i alfabetisk rekkefølge     | liste med stringer, fordi rekkefølgen er viktig og et navn kan vises to ganger                 |
| Hvilket måltid du skal spise for hver dag i uken                   | en ordbok med hverdager som nøkkel (string eller heltall) og måltid som verdi (string)         |
| Telle hvor mange fugler du ser av hver art i hagen din             | ordbok med fuglearter som nøkler og tall som verdi                                             |
| Alle primtall under 1000                                           | mengde med heltall, fordi rekkefølgen er ikke viktig og vi vil ikke ha duplikater              |

Fra Trix oppgave [03.11](#) - les mer om lister, mengder og ordbøker [her](#)

# KONTAKT



...

sirisoll@uio.no  
@sirisoll på Matteredmost