Samlinger og objekter

Uke 3, time 1.

Gruppe 6

Frøya Lydersen

skagel@uio.no

Contents

Contents

Samlinger 2
Lister 4
Ordbøker 12
Mengder 19
Quiz på menti! 26
Litt ekstra om syntaks 27
Mye, men ikke alt, gjøres med metoder 29
Funksjon, metode, prosedyre .30
Aktiviteter i par! 31
Oppgaver 32
Alias! 33
–Exit ticket– 34

Samlinger

Samlinger

- **lister**, **ordbøker**, og **mengder**, er samlingene våre.
- Samlinger lar oss “lagre flere verdier i en variabel”.

Lister

- Intuitivt er en liste en rekke nummererte bokser vi kan putte verdier i.
- Hver boks må ha noe inni seg, og vi kan ikke ha mellomrom mellom boksene.
- Vi henvender oss til nummeret på boksen vi er interessert i.
- Skrives med `[]` med verdier inni.

Skape en liste

```
1       ["peter", "gulrot", "løk", "potet", "hvitløk"]
```



Skape en liste


```
1  print(["peter", "gulrot", "løk", "potet", "hvitløk"])
```



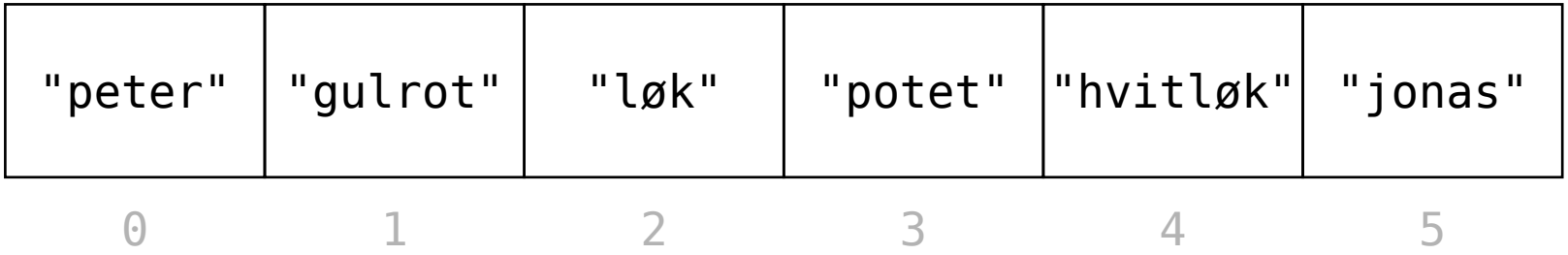
Skape en liste

```
1  x = ["peter", "gulrot", "løk", "potet", "hvitløk"]
```



Utvide en liste


```
2  x.append("jonas")
```



Fjerne plasser fra en liste

Som å ta vekk bokser fra rekken.

```
3  x.pop()
```

```
4  x.pop(0)
```

"peter"	"gulrot"	"løk"	"potet"	"hvitløk"
0	1	2	3	4

"gulrot"	"løk"	"potet"	"hvitløk"
0	1	2	3

Hente ut en verdi fra en liste

Hente ut verdien i første boks:

```
5          x[0]
```

Hente ut en verdi fra en liste

Hente ut og skrive ut verdien i første boks:

```
5  print(x[0]) # -> gulrot
```

Hente ut en verdi fra en liste

Hente ut og skrive ut verdien i første boks:

```
5  print(x[0]) # -> gulrot
```

Tredje boks:

```
6  print(x[2]) # -> potet
```

Overskrive verdi i en liste

Vi kan tilordne verdier direkte til boksene!

```
1  oddetall = [1,2,5,7,9]
```

Overskrive verdi i en liste

Vi kan tilordne verdier direkte til boksene!

```
1  oddetall = [1,2,5,7,9]
2  oddetall[1] = 3
```

Lagre forskjellige typer

- Lister kan lagre andre ting enn bare strenger!

```
1  aldere = [20, 19, 41, 21, 25, 18, 33, 28]
```

- Vi kan ha til og med blande verdier med forskjellige typer i lister!

```
1  x = ["gulrot", 5, False]
```

Vanlige metoder

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Ordbøker

- Lagrer “par” med verdier, som en tabell med to kolonner.
- “nøkler” assosieres med verdier.
- Skrives med `{}`, par med verdier inni separert med `:`
- Kan ikke ha duplikate nøkler.

Hovedstater!

```
1  x = {"Norway":"Oslo", "Sweden":"Stockholm",
   "Denmark":"Copenhagen", "Iceland":"Reykjavik"}
```

Nøkler	Verdier
"Norway"	"Oslo"
"Sweden"	"Stockholm"
"Denmark"	"Copenhagen"
"Iceland"	"Reykjavík"

Eks1

Hovedstater!

```
1    x = {
2    "Norway": "Oslo",
3    "Sweden": "Stockholm",
4    "Denmark": "Copenhagen",
5    "Iceland": "Reykjavik"
6  }
```

Nøkler	Verdier
"Norway"	"Oslo"
"Sweden"	"Stockholm"
"Denmark"	"Copenhagen"
"Iceland"	"Reykjavik"

Eks1

Hovedstater!

```
1    x = {
2    "Norway": "Oslo",
3    "Sweden": "Stockholm",
4    "Denmark": "Copenhagen",
5    "Iceland": "Reykjavik",
6  }
```

Nøkler	Verdier
"Norway"	"Oslo"
"Sweden"	"Stockholm"
"Denmark"	"Copenhagen"
"Iceland"	"Reykjavik"

Pris på varer!

```
1  x = {"Cheese": 110, "Milk":48, "Bread":60}
```

Nøkler	Verdier
"Cheese"	110
"Milk"	48
"Bread"	60

Hente ut en verdi fra en ordbok

```
1  x = {"Cheese": 110, "Milk":48, "Bread":60}
2  print(x["Cheese"])
3  print(x["Milk"])
```

Nøkler	Verdier
"Cheese"	110
"Milk"	48
"Bread"	60

Overskrive verdi i en ordbok

```
1  countries = {"Norway": "Paris", "Sweden": "Stockholm"}
```

Nøkler	Verdier
"Norway"	"Paris"
"Sweden"	"Stockholm"

Overskrive verdi i en ordbok


```
1  countries = {"Norway": "Paris", "Sweden": "Stockholm"}
2  countries["Norway"] = "Oslo"
```

Nøkler	Verdier
"Norway"	"Oslo"
"Sweden"	"Stockholm"

Utvide en ordbok

Samme greia som å overskrive en verdi!

```
1  x = {"Cheese": 110}
2
3
4
```

Utvide en ordbok

Samme greia som å overskrive en verdi!

```
1  x = {"Cheese": 110}
2  x["Milk"] = 48
3
4
```

Utvide en ordbok

Samme greia som å overskrive en verdi!

```
1  x = {"Cheese": 110}
2  x["Milk"] = 48
3  x[input("Hvilken vare: ")] = int(input("Pris: "))
4
```

Utvide en ordbok

Samme greia som å overskrive en verdi!

```
1  x = {"Cheese": 110}
2  x["Milk"] = 48
3  x[input("Hvilken vare: ")] = int(input("Pris: "))
4  x[input("Hvilken vare: ")] = int(input("Pris: "))
5  print(x)
```

Fjerne plasser fra en ordbok


```
1  x = {"Cheese": 110, "John":49}
```

```
2
```

Fjerne plasser fra en ordbok


```
1  x = {"Cheese": 110, "John":49}
2  y = x.pop("John")
```

Fjerne plasser fra en ordbok

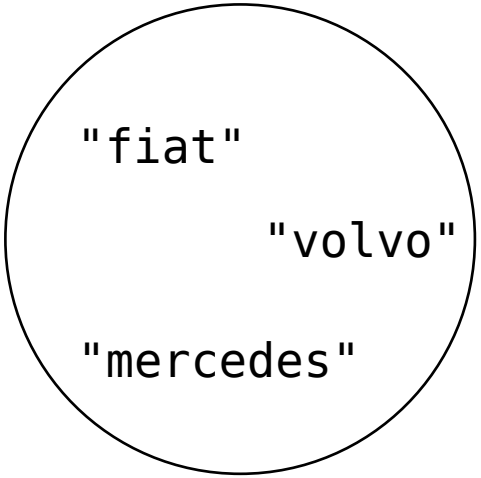

```
1  x = {"Cheese": 110, "John":49}
2  y = x.pop("John")
3  print(y, x.pop("Cheese"))
```

Mengder

- Representerer en gjeng med verdier, med ingen duplikater. F.eks:
  - De veganske rettene på menyen.
  - Navnene til gjestene dine.
- En verdi har ingen indeks eller nøkkel i en mengde, den bare er eller er ikke med i mengden.
- Samme som mengder i matte!
- Skrives.. også.. med `{}`. Kommaseparert liste med verdier inni!

Skape en mengde


```
1  bil_produsenter = {"volvo", "fiat", "mercedes"}
```



Skape en mengde

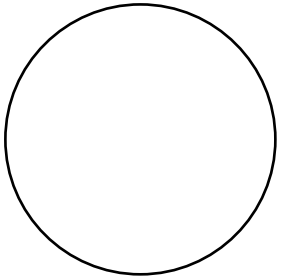
Hovedstater!

```
1    x = {
2    "Norway": "Oslo",
3    "Sweden": "Stockholm",
4    "Denmark": "Copenhagen",
5    "Iceland": "Reykjavik"
6  }
```

Nøkler	Verdier
"Norway"	"Oslo"
"Sweden"	"Stockholm"
"Denmark"	"Copenhagen"
"Iceland"	"Reykjavik"

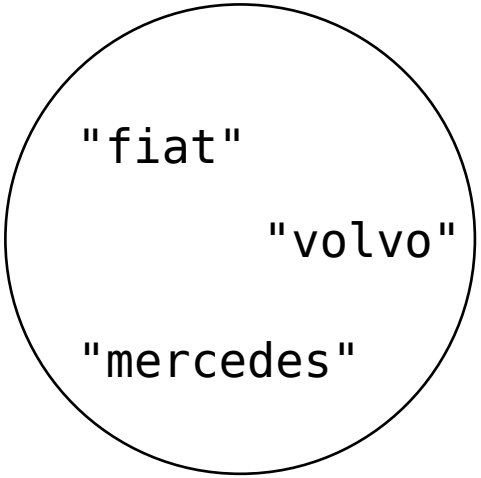
Skape en tom mengde


```
1  x = set()
```



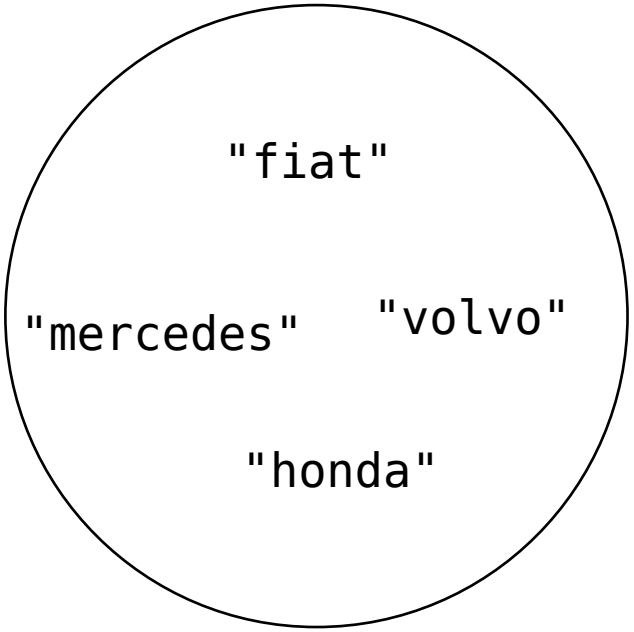
Legge noe til en mengde

```
1  bil_produsenter = {"volvo", "fiat", "mercedes"}
```



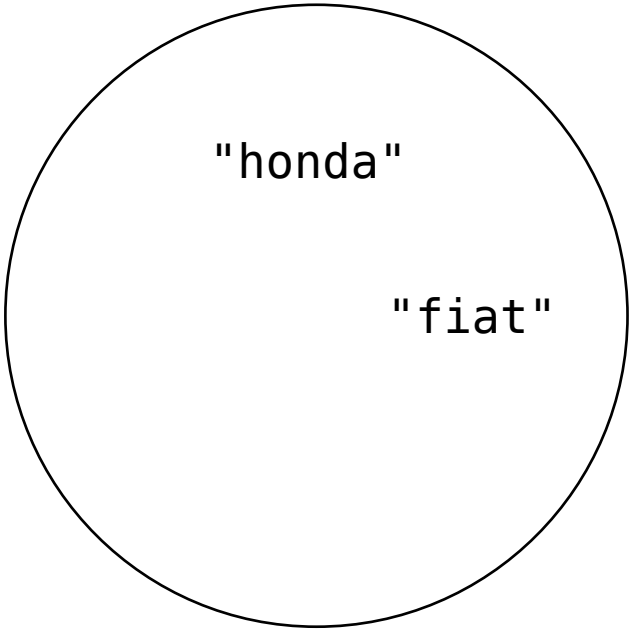
Legge noe til en mengde

```
1  bil_produsenter = {"volvo", "fiat", "mercedes"}
2  bil_produsenter.add("honda")
```



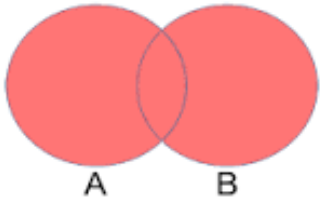
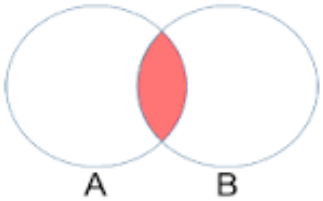
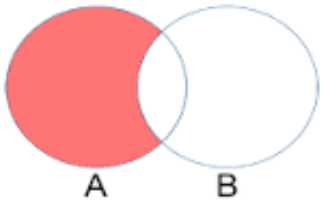
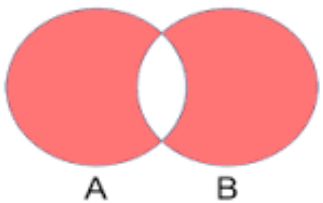
Fjerne fra en mengde

```
1    bil_produsenter = {"volvo", "fiat", "mercedes", "honda"}
2    bil_produsenter.remove("volvo")
3    bil_produsenter.pop()
```



?

Mengder tilbyr mengdeoperasjoner :D

Set Operation	Venn Diagram	Interpretation
Union		$A \cup B$, is the set of all values that are a member of A, or B, or both.
Intersection		$A \cap B$, is the set of all values that are members of both A and B.
Difference		$A \setminus B$, is the set of all values of A that are not members of B
Symmetric Difference		$A \triangle B$, is the set of all values which are in one of the sets, but not both.

Method	Shortcut	Description
add()		Adds an element to the set
clear()		Removes all the elements from the set
copy()		Returns a copy of the set
difference()	-	Returns a set containing the difference between two or more sets
difference_update()	-=	Removes the items in this set that are also included in another, specified set
discard()		Remove the specified item
intersection()	&	Returns a set, that is the intersection of two other sets
intersection_update()	&=	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()		Returns whether two sets have a intersection or not
issubset()	<=	Returns True if all items of this set is present in another set
	<	Returns True if all items of this set is present in another, *larger* set
issuperset()	>=	Returns True if all items of another set is present in this set
	>	Returns True if all items of another, *smaller* set is present in this set
pop()		Removes an element from the set
remove()		Removes the specified element
symmetric_difference()	^	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	^=	Inserts the symmetric differences from this set and another
union()	|	Return a set containing the union of sets
update()	|=	Update the set with the union of this set and others

Mengder, praktisk eksempel


```
1  coop_kunder = {
2      "Peter", "Javad", "Aleksandra", "Natalia", "Alex", "Jonas"
3  }
4  rema_kunder = {
5      "Ulrik", "Natalia", "Svein", "Aleksandra", "Mohammed"
6  }
7  print(coop_kunder.union(rema_kunder))
8  print(coop_kunder.intersection(rema_kunder))
9  print(coop_kunder.difference(rema_kunder))
```

Quiz på menti!

Litt ekstra om syntaks

Litt ekstra om syntaks

```
1  l = ["Eli","Peter","Ost"]
2  l[2] = "Marianne"
3  print(l[2]) # Marianne
4  l.append("Eli")
5  l.pop()
```

```
1  d = {"Eli":422, "Peter":701, "Marianne":48}
2  d["Eli"] = 511
3  print(d["Eli"]) # 511
4  d["Alex"] = 91
```

```
1  m = {"Eli","Peter","Marianne"}
2  m.remove("Eli")
3  m.pop()
4  m.add("Alex")
```

Mye, men ikke alt, gjøres med metoder


```
1  l = [5,9,2,3]
2  print(len(l)) # 4
3  print(sorted(l)) # 2,3,5,9
4  print(9 in l) # True
```

```
1  d = {"Norway":"Oslo", "Sweden":"Stockholm"}
2  print("Norway" in d) # True
```

```
1  print(len("8st epop!")) # 9
2  print(len("❄")) # 3
```

```
1  a = "HamBURGer"
2  print(a.lower()) # hamburger
3  print(a.upper()) # HAMBURGER
```

Funksjon, metode, prosedyre

Funksjonskall:

```
1  print("hei!")
2  x = input("navn? ")
3  y = [90,5,20,3]
4  sorted(y)
```

Metodekall:

```
1  a = "Ost"
2  print(a.lower())
3  b = {"Rød", "Blå", "Hvit"}
4  b.pop()
```

Prosedyrekall:

```
1  min_prosedyre()
2  skriv_feilmelding()
```

Aktiviteter i par!

Oppgaver

siden for gruppe6:

		02	Slides, Oppgaver
03	samlinger	01	Oppgaver
		02	

Alias!



–Exit ticket–