Generelt om sets

Læringsmål:

- Sets
- Lister

Starting Out with Python:

- Kap. 9.2

Generelt om sets

Det kan være lurt å lese gjennom dette før du går videre

Et set inneholder en samling av unike verdier og fungerer på samme måte som et set i matematikken. Forskjellen på et set og en liste er at en liste kan inneholde flere like elementer, for eksempel:

In [ ]:

```
liste = [1,2,3,1,2,3]
```

Altså kan vi finne alle tallene to ganger. Dette går ikke i et set ettersom alle elementene i et set er ulike. Tilsvarende set ser slik ut:

In [ ]:

```
my_set = set([1,2,3])
```

På samme måte som vi kunne opprette en dictionary ved å skrive `my_dict = dict()` , kan vi opprette et set ved å skrive:

In [ ]:

```
my_set = set()
```

For å legge til et element i et set kan vi benytte oss av `add()` . **Husk å kjøre koden over før du kjører koden under**

In [ ]:

```
my_set.add(1)
my_set.add(2)
my_set.add(3)
my_set.add(3)
my_set.add(3)
print(my_set)
```

For å legge til flere elementer på samme gang, kan `update()` benyttes.

For å fjerne et element kan vi benytte oss av `remove()` eller `discard()` .

In [ ]:

```
my_set.remove(2)
print(my_set)
```

De fleste ting fungerer likt mellom sets og lister, og her er noen eksempler:

- iterering gjennom setet/listen
- sjekke om et element er i setet/listen
- finne lengden til setet/listen

Dersom du har hatt sannsynlighet er du kanskje godt kjent med venndiagram. Da har du sikkert hørt om union og snitt. Dersom vi har to sets

In [ ]:

```
set1 = set([1,2,3,4,5,6,7,8])
set2 = set([0,2,4,6,8,10,12])
```

kan vi se at begge setene inneholder 2, 4, 6 og 8. Dette er det som kalles intersection, eller snitt. På figuren under kan vi se at snittet er det feltet hvor sirklene overlapper.

 img

For å finne snittet av set1 og set2 kan vi skrive som under. **Kjør kun koden under dersom du har kjørt forrige kodeblokk som initialiserer set1 og set2**

In [ ]:

```
set3 = set1.intersection(set2)
set3 = set2.intersection(set1)
set3 = set1&set2
print(set3)
```

Alle de tre første linjene i kodeblokken over er ekvivalente.

Union er et annet nyttig ord, og det vil si tallene som enten er i set1 eller set2 eller begge, dvs. alle tallene som er med. For å finne unionen av set1 og set2 kan vi skrive:

In [ ]:

```
set3 = set1.union(set2)
set3 = set2.union(set1)
set3 = set1 | set2
print(set3)
```

Her gjør også alle de tre øverste kodelinjene akkurat det samme.

Tallene som er i set1, men ikke i set2, dvs. 1, 3, 5 og 7 kan vi finne ved å skrive

In [ ]:

```
set3 = set1.difference(set2)
set3 = set1-set2 # denne linja og linja over gjør akkurat det samme
print(set3)
```

For å finne elementene som er i set2 men ikke set1 er det bare å bytte om på set1 og set2 i koden over.

Symmetric difference vil si alle tallene som er i set1 eller set2 (dvs. unionen) minus snittet (tallene som er i begge setene). I dette tilfelle er det 0,1,3,5,7,10 og 12.

Dette finner vi slik:

In [ ]:

```
set3 = set1.symmetric_difference(set2)
set3 = set2.symmetric_difference(set1)
set3 = set1^set2               #set3 = (0,1,3,5,7,10,12)
print(set3)
```

Her gjør også de tre første linjene akkurat det samme.

a)

Lag et tomt set som heter `my_set` , legg til alle oddetallene opp til 20 i setet ved å bruke en for-løkke og print setet

Skriv koden i kodeblokken under

In [11]:

```
my_set = set()
for x in range(20):
    if x%2 != 0:
        my_set.add(x)
print(my_set)
```

```
{1, 3, 5, 7, 9, 11, 13, 15, 17, 19}
```

Har du gjort det riktig skal output være

```
{1, 3, 5, 7, 9, 11, 13, 15, 17, 19}
```

b)

Lag et nytt set som heter `my_set2` og inneholder alle oddetallene frem til 10.

Skriv din koden i kodeblokken under

In [12]:

```
my_set2 = set()
for x in range(10):
    if x%2 != 0:
        my_set2.add(x)
```

Har du gjort det riktig skal kodeblokken under printe `{1, 3, 5, 7, 9}`

In [13]:

```
print(my_set2)
```

```
{1, 3, 5, 7, 9}
```

c)

Lag et setet `my_set3` som inneholder alle tallene som er i setet fra a) men ikke i setet fra b)

Skriv koden i kodeblokken under

In [19]:

```
my_set3 = my_set-my_set2
my_set3
```

Out[19]:

```
{11, 13, 15, 17, 19}
```

Har du gjort det riktig skal kodeblokken under printe `{11, 13, 15, 17, 19}`

In [20]:

```
print(my_set3)
```

```
{11, 13, 15, 17, 19}
```

d)

Dersom du tar snittet av setet fra b) og setet fra c), hva forventer du å få da? Hva med a) og c)?

Svar: b) og c): ingenting, de har ingen felles element a) og c): forventer å få ut lista fra c) ettersom hele c) finnes i a)

e)

Bruk de innebygde funksjonene `len()` og `set()` til å lage en funksjon `allUnique(lst)`, som returnerer `True` om listen `lst` inneholder unike elementer og ellers returnerer `False`.

Skriv koden i kodeblokken under

In [24]:

```
def allUnique(lst):
    listas_set = set(lst)
    return len(lst) == len(listas_set)
```

Har du gjort det riktig kan du teste med koden under (etter å ha kjørt din egen kodeblokk)

In [25]:

```
print(allUnique([1,3,2,6,8]))
print(allUnique([1,3,5,2,3,7]))
```

```
True
False
```

Output fra denne burde være:

```
True
False
```

f)

Bruk de innebygde funksjonene `list()` og `set()` til å lage en funksjon `removeDuplicates(lst)`, som fjerner duplikater fra listen `lst` og returner den modifiserte listen.

Skriv koden i kodeblokken under

In [26]:

```
def removeDuplicates(lst):
    lst_set = set(lst)
    liste = list(lst_set)
    return liste
```

Du kan teste koden din med koden under (etter å ha kjørt din egen kodeblokk)

In [27]:

```
print(removeDuplicates([1,3,5,2,3,7]))
```

```
[1, 2, 3, 5, 7]
```

Har du gjort alt riktig skal output fra denne være:

```
[1, 2, 3, 5, 7]
```