

Sets (verzamelingen)

Naast lijsten bestaan er nog andere collectie types in Python. Het volgende collectie type dat we bespreken is een verzameling, of *set*. Een set is erg gelijkaardig aan een lijst in gebruik, maar heeft ook een aantal verschillen:

- een verzameling kan elk element maar 1 maal bevatten. Als we meermaals hetzelfde element toevoegen, zal het toch maar 1 maal in de verzameling zitten;
- verzamelingen zijn ongeordend. De elementen zullen niet noodzakelijk in de zelfde volgorde worden teruggegeven als ze werden toegevoegd. Ook de operaties die een ordening verwachten, zoals *L.sort()* en *L[i]* hebben geen equivalent bij sets;
- de operatie *in* om te testen of een element in een lijst zit, bestaat ook voor een verzameling, maar is efficiënter voor een verzameling. Voor de voorbeelden die we in deze cursus behandelen maakt dat geen verschil, maar voor grotere projecten gaat het mogelijk wel van belang zijn om het meest geschikte type te gebruiken.

Nieuwe verzameling maken en elementen toevoegen en verwijderen

Een nieuwe verzameling maak je aan als volgt:

```
In [1]: S=set() # Lege verzameling
        S2={1,2,3} # verzameling met elementen 1,2,3
```

Elementen toevoegen gaat met de functie *add*:

```
In [2]: S2={3,1,2}
        S2.add(4)
        S2.add(1)
        print(S2)

{1, 2, 3, 4}
```

Het aantal elementen van een set krijgen we via *len*

```
In [3]: print(len(S2))
```

4

Verwijderen doe je als volgt:

```
In [4]: S={1,2,3}
        S.remove(3)
        print(S)
```

```
{1, 2}
```

S.remove(x) werkt enkel als *x* werkelijk in de set voorkomt. Komt *x* niet voor, dan krijg je een foutmelding. *S.discard(x)* daarentegen, verwijdert *x* ook uit de lijst maar als *x* niet voorkomt, gebeurt er niets:

```
In [5]: S={1,2,3}
        S.discard(4)
        S.remove(4)
```

```
-----
KeyError                                Traceback (most recent call last)
<ipython-input-5-13e5966ca59d> in <module>
      1 S={1,2,3}
      2 S.discard(4)
----> 3 S.remove(4)

KeyError: 4
```

Waarom hebben we *remove* dan in hemelsnaam nog nodig? Om redeneerfouten die je maakte bij het schrijven van het programma snel te detecteren!

Wanneer je code schrijft, zal het vaak voorkomen dat je 100% zeker bent dat een element *x* dat je wil verwijderen uit een set *S* ook daadwerkelijk in de set voorkomt. Gebruik dan *S.remove(x)*. Het resultaat is hetzelfde, behalve wanneer je een redeneerfout maakte. Door *remove* te gebruiken gaat zulk een redeneerfout niet onopgemerkt voorbij!

Een gouden regel bij het programmeren is: **zorg ervoor dat je fouten zo snel mogelijk vindt**. Op die manier vermijd je erg eigenaardige fouten te krijgen waarnaar je uren op zoek moet gaan, omdat ze het gevolg zijn van een gevolg van een gevolg van een gevolg van een domme kleine redeneerfout. Daarom dus: ben je zeker dat het element dat je wil verwijderen in de set voorkomt? Gebruik dan *remove*! Gebruik *discard* enkel om volgende regels code te vervangen:

```
In [6]: S={1,2,3}
        x=int(input())
        if x in S:
            S.remove(x)
        print(S)
```

```
{1, 2, 3}
```

Enkel in dit geval is het gebruik van *discard* aan te raden:

```
In [8]: S={1,2,3}
        x=int(input())
        S.discard(x)
        print(S)
```

```
{1, 2}
```

Itereren over de elementen van een verzameling

Itereren doe je net als bij een lijst, met *for*. Zoals eerder al gezegd heb je echter geen controle over de volgorde:

```
In [9]: S={4,5,8,1,2}
        for x in S:
            print(x)
```

```
1
2
4
5
8
```

In dit voorbeeld wordt de set geordend teruggegeven, maar dat hoeft niet per se zo te zijn. Verder test je met *in* of een element in de lijst voorkomt:

```
In [10]: klinkers={"a","e","i","o","u"}
        s="Wie heeft er nu klinkers nodig ..."
        zonder=""
        for letter in s: # herinner: we kunnen over een string itereren alsof het een
                           lijst is
            if letter not in klinkers:
                zonder=zonder+letter
        print(zonder)
```

```
W hft r n klnkrs ndg ...
```

Set-operaties

Verder kan je met verzamelingen alle standaard bewerkingen met verzamelingen doen; een goede kennis van deze functies kan jouw code vaak veel leesbaarder en korter maken:

- $s|t$: unie van de verzamelingen s en t
- $s&t$: doorsnede van s en t
- $s-t$: verschil van de verzamelingen s en t

```
In [11]: IP={"Jan","An","Illias","Rosa"}
        GAS={"Jan","Joris","Greet","Illias"}
        CSA={"Jan","Rosa"}

        # Geef alle studenten die IP of GAS volgen en niet CSA:
        print((IP|GAS)-CSA)

{'Greet', 'Illias', 'Joris', 'An'}
```

Set vergelijkingen

Je kan ook snel en makkelijk vergelijkingen tussen verzamelingen maken:

- $s \leq t$: waar als s deelverzameling is van t
- $s < t$: waar als s strikte deelverzameling is van t
- $s \geq t$, $s > t$: omgekeerde richting
- $s == t$: s en t hebben dezelfde elementen

```
In [12]: print("Does every student that takes CSA also follow IP and GAS?")
        if CSA<=IP&GAS:
            print("Yes they do!")
        else:
            print("No; the following students follow CSA without following IP and GAS:", CSA-(IP&GAS))
```

Does every student that takes CSA also follow IP and GAS?

No; the following students follow CSA without following IP and GAS: {'Rosa'}

Set omzetten in een lijst en vice versa

Soms is het handig om een set om te zetten in een lijst (bijvoorbeeld als we moeten sorteren) of omgekeerd. Dit doen we via respectievelijk de functies `list(Set)` en `set(List)`:

```
In [13]: next=True
S=set()
while next:
    student=input("Geef volgende student:")
    if student=="":
        next=False
    else:
        S.add(student)

print("Alle studenten: ",S)
L=list(S)
L.sort()
print("Alle studenten nu gesorteerd:",L)
```

```
Alle studenten: {'toon', 'ahmed', 'klaas', 'klaartje', 'kees', 'boris', 'sien'}
Alle studenten nu gesorteerd: ['ahmed', 'boris', 'kees', 'klaartje', 'klaas', 'sien', 'toon']
```

Hier een voorbeeldje van de omgekeerde richting waarbij we een lijst omzetten in een verzameling om dubbels weg te halen zodat we kunnen tellen hoeveel *verschillende* elementen er in een lijst zitten:

```
In [14]: # worp is een lijst met de dobbelsteen worpen
# bijvoorbeeld: [1,2,3,4,5] of [6,6,6,6,6]
# de functie kijkt na of de worp Yahtzee is (5 dezelfde)
def yahtzee(worp):
    if len(set(worp))==1:
        return True
    else:
        return False

print(yahtzee([1,2,2,2,2]), yahtzee([3,3,3,3,3]))
```

```
False True
```

```
In [ ]:
```