Start to Program: Python, Les 10

Dictionaries

Een *dictionary* is een soort lijst waar je niet aan de hand van indexering data kan opvragen, maar eerder aan de hand van sleutelwaarden.

Bij een lijst of tuple werd er soortgelijke informatie opgeslagen. Een dictionary gaat verschillende informatie over 1 object of onderwerp opslaan.

Dit wordt vooral gebruikt informatie uit data-banken toegankelijk te maken. Informatie van bijvoorbeeld een gebruiker kan hierin worden gegroepeerd.

Om een dictionary aan te maken gebruik je nog een ander soort haakjes, de accolade.

Hierin ga je dan de informatie benoemen en er een waarde aan toekennen.

```
kristoff = {'gebruikersnaam' : 'Krzyzstoff', 'online' : True, 'volgers' : 356}
print(kristoff)
```

Gebruikersnaam, online en volgers zijn de sleutelwaarden. Deze kan je zelf benoemen en zijn dus vrij te kiezen.

Hetgeen daarop volgt: 'Krzyzstoff', True, 356 zijn de waarden die in de sleutelwaarden zijn opgeslagen. Terwijl een lijst of tuple waarschijnlijk uit dezelfde data-types. In een dictionary kunnen verschillende data-types worden bijgehouden. Deze kunnen dus een string, number of bool zijn.

In tegenstelling tot de lijsten en tuples is de volgorde hier niet belangrijk. Als er geprint wordt dan worden de waarden in een willekeurige volgorde weergegeven.

LET OP! De sleutelwaarden zijn strings dus je plaatst deze tussen aanhalingstekens.

De waarden hierin kunnen van data-type verschillen en aanhalingstekens moeten alleen gebruikt worden bij strings.

Data opvragen

Aangezien de volgorde willekeurig is kan je niet met indexering werken zoals bij lijsten en tuples.

Je gaat rechtstreeks het benodigde onderdeel van de dictionary opvragen.

```
print(kristoff['gebruikersnaam'])
=> Krzyzstoff
```

Deze manier van opvragen is echter beperkt tot een bepaalde sleutelwaarde.

Er zijn een aantal functies die speciaal voor het opvragen dan data uit dictionaries dienen.

Deze gaan een lijst of tuple weergeven van de opgevraagde waardes.

```
var_naam.keys()     => geeft een lijst van alle sleutelwaarden

var_naam.values()   => geeft een lijst van alle waardes in de sleutelwaardes

var_naam.items()    => geeft een lijst van tuple paartjes met zowel de sleutlwaarde als de opgeslagen waarde.
```

Er worden nu lijsten gemaakt waardoor je weer een telbaar iets krijgt, een lijst.

Deze lijsten kunnen dan worden gebruikt om vergelijkingen of loops mee te maken.

Zo kan je bijvoorbeeld data opvragen in gemeenschappelijke sleutelwaarden.

```
kristoff = {'gebruikersnaam' : 'Krzyzstoff', 'online' : True, 'volgers' : 356}
chantal = {'gebruikersnaam' : 'Chant1', 'online' : False, 'punten' : 451}

for common_key in set(kristoff.keys()) & set(chantal.keys()):
        print(kristoff[common_key],chantal[common_key])
```

Je kan een for-loop gebruiken om de data leesbaarder te maken.

```
for sleutel, waarde in kristoff.items():
    print(sleutel, 'is de sleutel voor de waarde:', waarde)
```

Data toevoegen en aanpassen

Je kan waarden en sleutelwaarden toevoegen aan een dictionary.

```
kristoff['punten'] = 5421
```

De dictionary wordt nu uitgebreid met een extra sleutelwaarde punten met waarde 5421

Op eenzelfde manier pas je de waarde van een bestaande sleutelwaarde aan.

```
kristoff['online'] = False
```

Je kan ook toevoegen en aanpassen door methodes te gebruiken.

```
var_naam.update({'sleutelwaarde' : waarde})
```

Deze mannier gaat duidelijker weergeven in je code dat er een bestaande dictionary aangepast of aangevuld wordt.

Leegmaken of verwijderen

Een specifieke sleutelwaarde verwijden:

```
del var_naam['sleutelwaarde']
```

Een hele dictionary leeg maken:

```
var_naam.clear()
```

Hier gaat de dictionary nog wel bestaan, alleen is hij helemaal leeggemaakt.