Lesbrief 4 Pandas DataFrame

[**Pandas DataFrame**](#_jkfg19dy2yg) **1**

[Kolommen selecteren](#_r4g52klf97bz) 2

[Inlezen van een bestand in een DataFrame](#_i2o8jdrvcflt) 3

[De eerste 5 rijen](#_8awdz238mstx) 4

[Sorteren](#_kgboomrfxzwc) 5

[Sorteren in omgekeerde richting](#_7c9empci476f) 6

[Selecteren](#_z2ufkkk1mxjy) 6

# Pandas DataFrame

Een DataFrame is meer dan een Series. Terwijl een Series een lijst is met allerlei waarden erin, is een DataFrame een tabel met meerdere kolommen.

Hier het verschil tussen een pandas Series object en een pandas DataFrame met een voorbeeld van gegevens van bezoekers uit een restaurant:



We gaan een DataFrame maken met gegevens van een voetbalcompetitie erin.

**Code:**

|  |
| --- |
| **import** pandas **as** pd  data = {'jaar': [2010, 2011, 2012, 2011, 2012, 2010, 2011, 2012],  'team': ['Ajax', 'Ajax', 'Ajax', 'PSV', 'PSV', 'AZ', 'AZ', 'AZ'],  'win': [11, 8, 10, 15, 11, 6, 10, 4],  'verlies': [5, 8, 6, 1, 5, 10, 6, 12]}  dfVoetbal = pd.DataFrame(data)  **print** (dfVoetbal) |

**Uitvoer:**

|  |
| --- |
| jaar team win verlies  0 2010 Ajax 11 5  1 2011 Ajax 8 8  2 2012 Ajax 10 6  3 2011 PSV 15 1  4 2012 PSV 11 5  5 2010 AZ 6 10  6 2011 AZ 10 6  7 2012 AZ 4 12 |

Je ziet hierboven inderdaad een tabel met de kolommen **jaar**, **team**, **win** en **verlies**. Merk op dat, zoals bij een Series, de eerste kolom de index is.

Eigenlijk mag je variabelenamen zelf bepalen. Maar het is een goed gebruik om variabelen voor een DataFrame te beginnen met de letter df. Zo weet iedereen die je code leest dat er een DataFrame in de variabele zit.

Het maken van een DataFrame ziet er wat moeilijk uit, maar met een beetje kopieer en plakwerk, moet het je wel lukken. We gaan er in dit stadium nog niet al te diep op in.

*Als je goed kijkt zul je zien dat elke rij en elke kolom in een DataFrame eigenlijk een Series is.*

## Kolommen selecteren

Je kunt ook specifieke kolommen selecteren uit een DataFrame object. Dat doe je door blokhaken [] te gebruiken.

**Code:**

|  |
| --- |
| **import** pandas **as** pd  data = {'jaar': [2010, 2011, 2012, 2011, 2012, 2010, 2011, 2012],  'team': ['Ajax', 'Ajax', 'Ajax', 'PSV', 'PSV', 'AZ', 'AZ', 'AZ'],  'win': [11, 8, 10, 15, 11, 6, 10, 4],  'verlies': [5, 8, 6, 1, 5, 10, 6, 12]}  dfVoetbal = pd.DataFrame(data)  **print** (dfVoetbal["jaar"]) |

**Uitvoer:**

|  |
| --- |
| 0 2010  1 2011  2 2012  3 2011  4 2012  5 2010  6 2011  7 2012 |

Je kunt ook meerdere kolommen selecteren. Voor de overzichtelijkheid maken we een aparte variabele met een lijst van de gewenste kolommen: kolommen = ["team", "win"]

**Code:**

|  |
| --- |
| **import** pandas **as** pd  data = {'jaar': [2010, 2011, 2012, 2011, 2012, 2010, 2011, 2012],  'team': ['Ajax', 'Ajax', 'Ajax', 'PSV', 'PSV', 'AZ', 'AZ', 'AZ'],  'win': [11, 8, 10, 15, 11, 6, 10, 4],  'verlies': [5, 8, 6, 1, 5, 10, 6, 12]}  dfVoetbal = pd.DataFrame(data) kolommen = ["team", "win"] **print** (dfVoetbal[kolommen]) |

**Uitvoer:**

|  |
| --- |
| team win  0 Ajax 11  1 Ajax 8  2 Ajax 10  3 PSV 15  4 PSV 11  5 AZ 6  6 AZ 10  7 AZ 4 |

## 

## Inlezen van een bestand in een DataFrame

Je kunt ook *bestanden* inlezen in een DataFrame. We gaan een CSV-bestand inlezen. CSV staat voor *Comma Separated Values*. Het zijn bestanden met veel gegevens in een tabel waarbij de waarden van elkaar worden gescheiden door een komma (of een ander teken). We hebben voor jullie een CSV-bestand met allerlei informatie over de verkoopcijfers van games. Hij is [hier](https://drive.google.com/file/d/0B-PCvKu5Uw4QTk9LdV9SWEE3cjg/view?usp=sharing) te downloaden.

Het is een joekel van een bestand met gegevens over ruim 16000 spellen!

Download het CSV-bestand en zet hem in dezelfde map als je Python-bestanden.

Laten we eerst het bestand inlezen en kijken welke kolommen het DataFrame bevat:

**Code:**

|  |
| --- |
| **import** pandas **as** pd  dfGames = pd.read\_csv("games.csv")  **print** (dfGames.columns) |

**Uitvoer:**

|  |
| --- |
| Index(['Name', 'Platform', 'Year\_of\_Release', 'Genre', 'Publisher', 'NA\_Sales', 'EU\_Sales', 'JP\_Sales', 'Other\_Sales', 'Global\_Sales', 'Critic\_Score', 'Critic\_Count', 'User\_Score', 'User\_Count', 'Developer', 'Rating'], dtype='object') |

Dat zijn veel kolommen! Wel 16! Laten we de volgende kolommen selecteren: Name, Platform, Year\_of\_release en Genre. En we gaan iets bijzonders doen .. We gaan namelijk de DataFrame met de geselecteerde kolommen (dfGames[kolommen]) even in een andere variabele zetten. Deze noemen we dfGamesKlein

**Code:**

|  |
| --- |
| **import** pandas **as** pd  dfGames = pd.read\_csv("games.csv") kolommen = ["Name", "Platform", "Year\_of\_Release", "Genre"]  dfGamesKlein = dfGames[kolommen]  **print** (dfGamesKlein) |

**Uitvoer:**

|  |
| --- |
| Dit gaan we niet laten zien :) Het zijn namelijk in totaal 16719 rijen! |

## De eerste 5 rijen

Gelukkig kunnen we ook de eerste 5 rijen selecteren. Dat doen met de methode head()

**Code:**

|  |
| --- |
| **import** pandas **as** pd  dfGames = pd.read\_csv("games.csv") kolommen = ["Name", "Platform", "Year\_of\_Release", "Genre"]  dfGamesKlein = dfGames[kolommen]  **print** (dfGamesKlein.head()) |

**Uitvoer:**

|  |
| --- |
| Name Platform Year\_of\_Release Genre  0 Wii Sports Wii 2006.0 Sports  1 Super Mario Bros. NES 1985.0 Platform  2 Mario Kart Wii Wii 2008.0 Racing  3 Wii Sports Resort Wii 2009.0 Sports  4 Pokemon Red/Pokemon Blue GB 1996.0 Role-Playing |

**Opdracht**

Print nu de eerste 15 games uit de oorspronkelijke tabel. Maar wel met de volgende kolommen: Name (naam van het spel) en EU\_Sales (hoeveel geld in miljoenen het spel in Europa heeft opgeleverd).

**TIP:** Je kunt de methode head() ook een getal meegeven. Dan geeft hij niet de eerste 5 rijen terug maar wordt het aantal beperkt met het getal dat jij hebt meegegeven.

## Sorteren

Stel dat je de eerste 10 oudste spellen in de tabel wilt zien. Dan is het handig als je kan sorteren op de kolom "Year\_of\_Release". Dit kan vrij eenvoudig in Pandas door de methode sort() te gebruiken.

**Code:**

|  |
| --- |
| **import** pandas **as** pd  dfGames = pd.read\_csv("games.csv") kolommen = ["Name", "Year\_of\_Release", "Global\_Sales"]  dfGamesKlein = dfGames[kolommen]  dfGamesKleinGesorteerd = dfGamesKlein.sort("Year\_of\_Release") **print** (dfGamesKleinGesorteerd.head(10)) |

**Uitvoer:**

|  |
| --- |
| Name Year\_of\_Release Global\_Sales  262 Asteroids 1980.0 4.31  5360 Freeway 1980.0 0.34  546 Missile Command 1980.0 2.76  2650 Boxing 1980.0 0.77  4019 Ice Hockey 1980.0 0.49  6301 Bridge 1980.0 0.27  1764 Kaboom! 1980.0 1.15  6876 Checkers 1980.0 0.24  1968 Defender 1980.0 1.05  4123 Grand Prix 1981.0 0.48 |

### Sorteren in omgekeerde richting

Je kunt ook in omgekeerde richting sorteren. Dat is handig als je bijvoorbeeld een top 5 wilt maken van de meest opgeleverde spellen aller tijden! Je geeft de methode sort()een tweede argument mee: ascending=False. Ascending betekent dat je van klein naar groot (oplopend) wilt sorteren. Het omgekeerde is descending (aflopend).

**Code:**

|  |
| --- |
| **import** pandas **as** pd dfGames = pd.read\_csv("games.csv") kolommen = ["Name", "Year\_of\_Release", "Global\_Sales"]  dfGamesKlein = dfGames[kolommen] dfGamesKleinSorted = dfGamesKlein.sort("Global\_Sales", ascending=**False**) **print** (dfGamesKleinSorted.head(5)) |

**Uitvoer[[1]](#footnote-0):**

|  |
| --- |
| Name Year\_of\_Release Global\_Sales  0 Wii Sports 2006.0 82.53  1 Super Mario Bros. 1985.0 40.24  2 Mario Kart Wii 2008.0 35.52  3 Wii Sports Resort 2009.0 32.77  4 Pokemon Red/Pokemon Blue 1996.0 31.37 |

## Selecteren

Met Pandas Series konden we met een voorwaarde bepaalde waarden selecteren door de selectievoorwaarde tussen blokhaken [] te plaatsen. Dat zag er zo uit:

**Code:**

|  |
| --- |
| **import** pandas **as** pd serLijst = pd.Series([23,43,102,76,8,98,90]) serGroterdan20 = serLijst[serLijst > 30] **print** (serGroterdan20) |

**Uitvoer:**

|  |
| --- |
| 1 43  2 102  3 76  5 98  6 90 |

Hetzelfde kunnen we ook met DataFrames doen. Stel dat we een top 5 van verkochte spellen voor de *NES* willen selecteren. Dat doen we zo (zie vooral ook de gemarkeerde stukken):

**Code:**

|  |
| --- |
| **import** pandas **as** pd dfGames = pd.read\_csv("games.csv") kolommen = ["Name", "Year\_of\_Release", "Platform", "Global\_Sales"]  dfGamesKlein = dfGames[kolommen]  dfGamesKleinSorted = dfGamesKlein.sort("Global\_Sales", ascending=**False**) dfNesGames = dfGamesKleinSorted[dfGamesKleinSorted.Platform == "NES"] **print** (dfNesGames.head(5)) |

**Uitvoer:**

|  |
| --- |
| Name Year\_of\_Release Platform Global\_Sales  1 Super Mario Bros. 1985.0 NES 40.24  9 Duck Hunt 1984.0 NES 28.31  22 Super Mario Bros. 3 1988.0 NES 17.28  98 Super Mario Bros. 2 1988.0 NES 7.46  124 The Legend of Zelda 1986.0 NES 6.51 |

*Merk op* dat we tussen de blokhaken de voorwaarde dfGamesKleinSorted.Platform == "NES" hebben geplaatst. Je mag dfGamesKleinSorted.Platform alleen gebruiken, omdat in de kolommenlijst ook het kopje "Platform" is geplaatst.

**Opdracht**

1. Maak een top 10 van de in Europa *minst* verkochte Playstationspellen.
2. Maak een top 5 van de nieuwste spellen voor de Wii.

1. Het klopt! Zie https://nl.wikipedia.org/wiki/Lijst\_van\_bestverkopende\_computerspellen [↑](#footnote-ref-0)