Lesbrief 5 Grafieken

[**Grafieken maken met matplotlib**](#_jkfg19dy2yg) **1**

[De staafdiagram (barchart)](#_lr6ymk56km73) 1

[Horizontaal staafdiagram](#_2ddfrtda37og) 3

[Functies](#_25r5xyvsr0bv) 3

[Groeperen](#_47vawm9j4ne1) 5

[Cirkeldiagrammen (piecharts)](#_3hx8raiarkac) 7

[Lijngrafieken](#_ypfqgte0jii5) 9

[Puntgrafieken (scatterplots)](#_cn0nszhmpsff) 9

# Grafieken maken met matplotlib

Voor het visualiseren van gegevens kunnen we grafieken maken. Hiervoor komt de Python library matplotlib erg goed van toepassing. We beginnen eerst met een staafdiagram.

## De staafdiagram (barchart)

Een staafdiagram is een schematische weergave van gegevens door met staafjes te werken. Hieronder zie je een voorbeeld:



We gaan nu een staafdiagram maken van de verkoopcijfers van [Neo Geo](https://en.wikipedia.org/wiki/Neo_Geo_(system))-spellen. Dat doen we als volgt:

* We selecteren eerst alle Neo Geo spellen (dfGames3Kolommen.Platform == "NG")
* Uiteindelijk maken we het staafdiagram door dfGamesNG.plot(kind="bar") in te geven.

**Code:**

|  |
| --- |
| **import** matplotlib.pyplot **as** plt **import** pandas **as** pd  dfGames = pd.read\_csv("games.csv") kolommen = ["Name", "Platform", "Global\_Sales"]  dfGames3Kolommen = dfGames[kolommen] dfGamesNG = dfGames3Kolommen[dfGames3Kolommen.Platform == "NG"] dfGamesNG.plot(kind="bar", x="Name", y="Global\_Sales")  plt.show() |

**Uitvoer:**

|  |
| --- |
| figure_1.png |

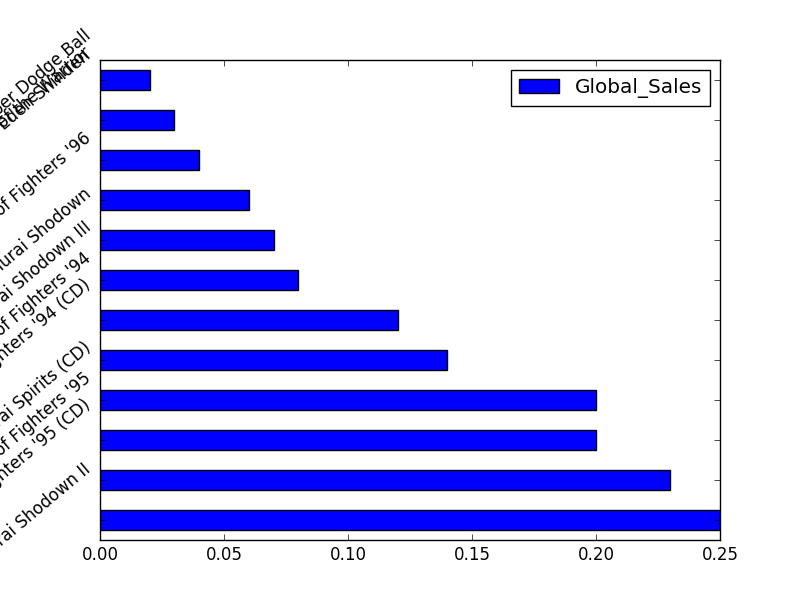
In de x-as hebben we de namen van de spellen. De staafjes geven de globale verkoopcijfers weer in miljoenen euro's.

**Opdracht**

Maak een staafdiagram waarbij je de verkoopcijfers van de best verkochte 20 PS3 spellen laat zien.

### Horizontaal staafdiagram

Je kunt de staafdiagram ook horizontaal maken. Dan verander je kind="bar" naar kind="barh". Zie: dfGamesNG.plot(kind="barh")



## Samenvattingsfuncties

Stel dat je wilt weten hoeveel geld alle PS3 spellen bij elkaar hebben opgeleverd in Europa. Hoe doe je dit dan?

* Stap 1: Eerst alle PS3-spellen selecteren
* Stap 2: Roep vervolgens de methode sum() op.

**Code:**

|  |
| --- |
| **import** pandas **as** pd dfGames = pd.read\_csv("games.csv") kolommen = ["Name", "Platform", "EU\_Sales"]  dfGames3Kolommen = dfGames[kolommen] dfGamesPS3 = dfGames3Kolommen[dfGames3Kolommen.Platform == "PS3"] **print** (dfGamesPS3.sum()) |

**Uitvoer:**

|  |
| --- |
| Name Grand Theft Auto VCall of Duty: Black Ops IICa...  Platform PS3PS3PS3PS3PS3PS3PS3PS3PS3PS3PS3PS3PS3PS3PS3P...  EU\_Sales 343.71 |

Wat is er gebeurd? De methode sum() telt alle waarden in alle kolommen bij elkaar op. We hadden drie kolommen en dus worden al die waarden van die drie kolommen per kolom bij elkaar opgeteld. Dus alle PS3 spelnamen zijn bij elkaar opgeteld, alle Platforms ("PS3") en alle EU\_Sales bedragen.

Omdat we alleen zijn geïnteresseerd in de optelling van de EU\_Sales hadden we ook dit kunnen doen:

**Code:**

|  |
| --- |
| **import** pandas **as** pd dfGames = pd.read\_csv("games.csv") kolommen = ["Name", "Platform", "EU\_Sales"]  dfGames3Kolommen = dfGames[kolommen] dfGamesPS3 = dfGames3Kolommen[dfGames3Kolommen.Platform == "PS3"] **print** (dfGamesPS3["EU\_Sales"].sum()) |

**Uitvoer:**

|  |
| --- |
| 343.71 |

**Opdracht**

1. Schrijf een programma dat een overzicht maakt van de verkoopcijfers in Japan, Europa en Noord Amerika voor de volgende Platforms: NES, SNES, Wii
2. Schrijf nu een programma dat de *gemiddelde* globale(Global\_Sales) verkoopcijfers voor de Platforms Wii en PS3 weergeeft.

Naast sum() en mean() kun je natuurlijk ook andere samenvattingsfuncties aanroepen. In Lesbrief 3 is dit ook al behandeld.

## Groeperen

Het lastige van de vorige opgave is dat je voor elk Platform de optelling moet gaan berekenen. Kan dat niet makkelijker? Ja, want je kunt *groeperen* op basis van een bepaalde kolom! Groeperen betekent dat je de rijen in de DataFrame in groepjes gaat verdelen op basis van de waarden in een bepaalde kolom. Vervolgens kunnen we de som van elke groep berekenen. Kijk maar eens:

**Code:**

|  |
| --- |
| **import** pandas **as** pd dfGames = pd.read\_csv("games.csv") kolommen = ["Platform", "EU\_Sales", "JP\_Sales", "NA\_Sales", "Global\_Sales"]  dfGamesSales = dfGames[kolommen]  print (dfGamesSales.groupby("Platform").sum()) |

**Uitvoer:**

|  |
| --- |
| EU\_Sales JP\_Sales NA\_Sales Global\_Sales  Platform  2600 5.47 0.00 90.60 97.08  3DO 0.00 0.10 0.00 0.10  3DS 58.52 97.35 78.87 247.46  DC 1.69 8.56 5.43 15.97  DS 194.65 175.57 390.71 822.49  GB 47.82 85.12 114.32 255.45  GBA 75.25 47.33 187.54 318.50  GC 38.71 21.58 133.46 199.36  GEN 5.52 2.67 19.27 28.36  GG 0.00 0.04 0.00 0.04  N64 41.06 34.22 139.02 218.88  NES 21.15 98.65 125.94 251.07  NG 0.00 1.44 0.00 1.44  PC 139.68 0.17 93.28 258.82  PCFX 0.00 0.03 0.00 0.03  PS 213.60 139.82 336.51 730.66  PS2 339.29 139.20 583.84 1255.64  PS3 343.71 79.99 392.26 957.84  PS4 123.70 14.30 96.80 278.10  PSP 68.25 76.79 108.99 296.28  PSV 16.33 20.96 16.20 61.93  SAT 0.54 32.26 0.72 33.59  SCD 0.36 0.45 1.00 1.87  SNES 19.04 116.55 61.23 200.05  TG16 0.00 0.16 0.00 0.16  WS 0.00 1.42 0.00 1.42  Wii 268.38 69.35 507.71 926.71  WiiU 24.23 12.79 38.32 81.86  X360 280.58 12.43 601.05 979.96  XB 60.95 1.38 186.69 258.26  XOne 45.65 0.34 83.19 141.06 |

Het eerste waar je op moet letten is dat de index is ingesteld op de kolom waarop je hebt gegroepeerd. In dit geval is dat dus de kolom "Platform". Per Platform zijn de waarden opgeteld voor EU\_Sales, JP\_Sales etc. We zouden er nu ook een staafdiagram van kunnen maken:

**Code:**

|  |
| --- |
| **import** pandas **as** pd **import** matplotlib.pyplot **as** plt  dfGames = pd.read\_csv("games.csv") kolommen = ["Platform", "EU\_Sales", "JP\_Sales", "NA\_Sales"]  dfGames3Kolommen = dfGames[kolommen] dfGamesPerPlatform = dfGames3Kolommen.groupby("Platform") dfGamesGrouped = dfGamesPerPlatform.sum()  dfGamesGrouped.plot(kind="bar", stacked=**True**) plt.show() |

**Uitvoer:**

|  |
| --- |
| figure_1.png |

**Uitleg:**

We hebben een nieuw argument meegegeven aan de plot functie. dfGamesGrouped.plot(kind="bar", stacked=True)

Met stacked=True zorg je dat de staafjes die behoren tot hetzelfde jaartal bij elkaar worden *opgestapeld*. Vergelijk het maar eens met de grafiek die je krijgt als je stacked=True weglaat.

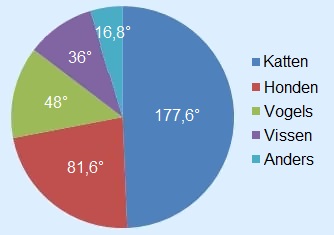
**Opgave**

1. Maak nu eens een grafiek waarmee je per genre aangeeft hoeveel spellen er zijn gemaakt. Je moet hiervoor de functie count() gebruiken.
2. Maak nu een andere grafiek waarin je per jaar aangeeft wat het gemiddelde aantal spellen zijn die zijn uitgebracht. Hiervoor gebruik je de functie mean()

## Cirkeldiagrammen (piecharts)

Een cirkeldiagram is een schematische weergave van gegevens in een cirkel.

Hieronder zie je een voorbeeld.



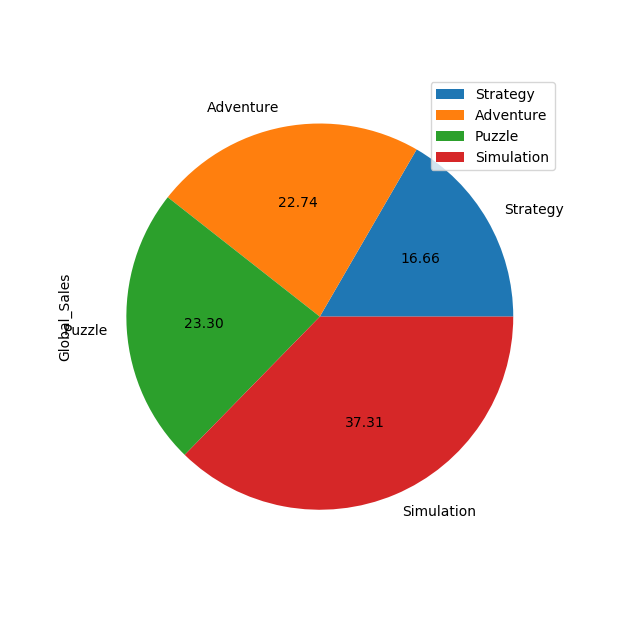
Cirkeldiagrammen zijn het meest geschikt wanneer de verschillende categorieën sterk in grootte verschillen. Wanneer de categorieën (ongeveer) even groot zijn is een staafdiagram een geschikter.

We gaan nu een cirkeldiagram maken waarbij we zien wat de omzet is van de vier meest opgeleverde games in een bepaalde *genre*.

**Code:**

|  |
| --- |
| **import** pandas **as** pd **import** matplotlib.pyplot **as** plt  dfGames = pd.read\_csv("games.csv") kolommen = ["Genre", "Global\_Sales"]  dfGamesKolommen = dfGames[kolommen] dfGamesPerGenre = dfGamesKolommen.groupby("Genre") dfGamesGrouped = dfGamesPerGenre.sum().sort\_values(by="Global\_Sales") # sorteren dfGamesGrouped = dfGamesGrouped[0:4] # de eerste vier pakken. zelfde als head(4) dfGamesGrouped.plot(kind="pie", y="Global\_Sales", autopct='%.2f', fontsize=10) plt.show() |

**Uitvoer:**



**Uitleg:**

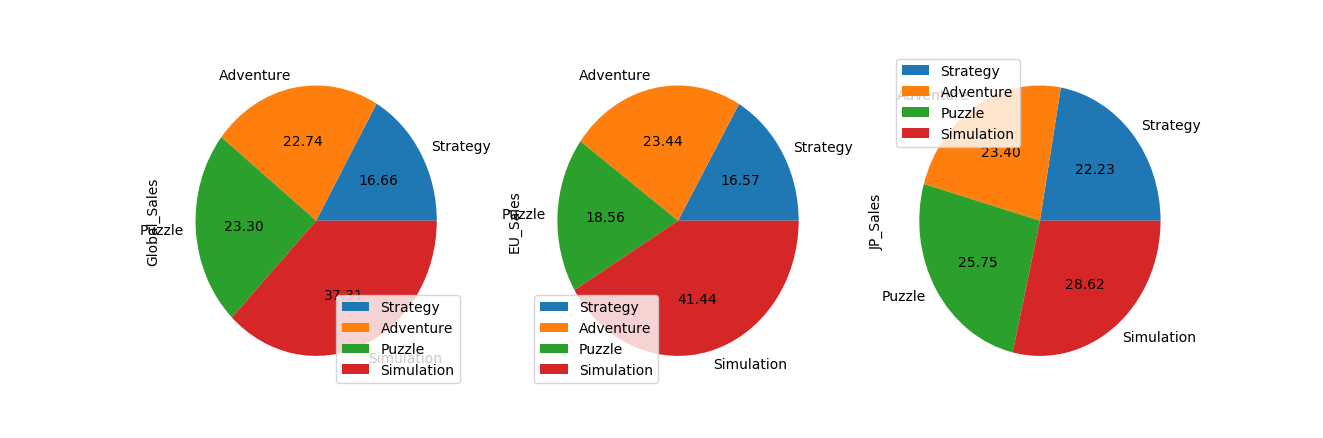
* We selecteren alleen de kolommen Genre en Global\_Sales
* Nadat we de sum() methode hebben aangeroepen sorteren we de waarden ook met de methode sort\_values(). Als argument moet je meegeven op welke kolom je de lijst wilt sorteren. dfGamesPerGenre.sum().sort\_values(by="Global\_Sales")
* Er zijn veel verschillende genres en ons cirkeldiagram zou heel onoverzichtelijk worden als we alle genres willen laten zien. Daarom pakken we met *slicing* de eerste vier met dfGamesGrouped = dfGamesGrouped[0:4]. We hadden dit ook met de methode head(4) kunnen doen overigens.
* Met de regel dfGamesGrouped.plot(kind="pie", y="Global\_Sales", autopct='%.2f', fontsize=10) zorgen we dat er een taartdiagram (piechart) wordt gemaakt op basis van Global\_Sales. Het percentage dat toegekend moet worden heeft twee cijfers achter de komma (autopct='%.2f') en de lettertypegrootte is 10px.

We kunnen ook meerdere kolommen selecteren en voor elke kolom een aparte cirkeldiagram laten maken. Dit is handig als je wilt vergelijken.

**Code:**

|  |
| --- |
| **import** pandas **as** pd **import** matplotlib.pyplot **as** plt  dfGames = pd.read\_csv("games.csv") kolommen = ["Genre", "Global\_Sales", "EU\_Sales", "JP\_Sales"]  dfGamesKolommen = dfGames[kolommen] dfGamesPerGenre = dfGamesKolommen.groupby("Genre") dfGamesGrouped = dfGamesPerGenre.sum().sort\_values(by="Global\_Sales") # sorteren dfGamesGrouped = dfGamesGrouped[0:4] # de eerste vier pakken (slicing) dfGamesGrouped.plot(kind="pie", subplots=**True**, autopct='%.2f', fontsize=10) plt.show() |

**Uitvoer:**



**Uitleg:**

De functie plot() heeft nu een nieuw argument gekregen: subplots=True. Dit zorgt ervoor dat voor elke kolom een aparte cirkeldiagram wordt gemaakt. Zoals je ziet zijn simulatiespellen in Europa veel populairder dan in Japan.

## Lijngrafieken

## Puntgrafieken (scatterplots)