

Onderzoek: Mogelijkheden van migratie

KAYLE BOERSEN

Contents

| | |
|---|----|
| Matlab | 3 |
| Wat is Matlab? | 3 |
| Enkele voordelen van MATLAB: | 3 |
| Nadelen van Matlab:..... | 3 |
| Migratie mogelijkheden | 4 |
| Voordelen:..... | 4 |
| Nadelen:..... | 4 |
| Libraries:..... | 4 |
| Voordelen:..... | 5 |
| Nadelen:..... | 5 |
| Voordelen:..... | 5 |
| Nadelen:..... | 5 |
| Julia: | 6 |
| Voordelen:..... | 6 |
| Nadelen:..... | 6 |
| Libraries:..... | 6 |
| Conclusie | 8 |
| Testen | 9 |
| Matlab | 9 |
| Python | 10 |
| Bokeh | 10 |
| Plotly | 10 |
| Dash | 10 |
| Conclusie | 10 |
| Matrix..... | 11 |
| Wat is een Matrix? | 11 |
| Waarom worden matrixen gebruikt?..... | 11 |
| Hoe word een matrix gebruik | 11 |
| Hoe kan een matrix gebruikt worden in Python? | 11 |
| Voorbeelden..... | 11 |
| Matrices definiëren | 11 |
| Optellen..... | 12 |

| | |
|--|----|
| Aftrekken | 12 |
| Vermenigvuldigen | 12 |
| Delen | 12 |
| Factor | 12 |
| Doorlopende berekening | 13 |
| Grafieken | 14 |
| Hoe word een grafiek gebruikt in de simulatie? | 14 |
| Hoe kan een Grafiek gebruikt worden in Python? | 14 |
| Hoe hebben wij het gebruikt | 14 |
| Imports | 14 |
| App starten | 15 |
| Hoe gebruik je dash in Python | 16 |
| Imports | 16 |
| Initialisatie | 17 |
| Layout | 17 |
| Callback-functie | 18 |
| Verduidelijking | 18 |
| App starten | 19 |
| Bibliografie | 20 |

Matlab

Wat is Matlab?

MATLAB (afkorting van **MAT**rix **LAB**oratory) is een softwareomgeving die gespecialiseerd is in wiskundige berekeningen en visualisatie. Het is ontwikkeld door MathWorks en wordt wereldwijd gebruikt door miljoenen engineers, wetenschappers en studenten.

Enkele voordelen van MATLAB:

Gebruiksvriendelijk: De interface van MATLAB is ontworpen met gebruiksgemak in gedachten. Dit maakt het toegankelijk voor zowel beginners als ervaren gebruikers. De omgeving biedt interactieve tools die helpen bij het uitvoeren van complexe berekeningen en het visualiseren van resultaten zonder diepgaande kennis van programmeertalen te vereisen.

Uitgebreide bibliotheek: MATLAB beschikt over een uitgebreide bibliotheek met kant-en-klare functies en toolboxes voor diverse toepassingen. Deze bibliotheken dekken een breed scala aan gebieden zoals signalenverwerking, beeldverwerking, optimalisatie, statistiek en machine learning. Dit zorgt ervoor dat gebruikers snel aan de slag kunnen met geavanceerde technieken zonder deze zelf te hoeven ontwikkelen.

Compatibel met andere software: MATLAB kan eenvoudig integreren met andere software en programmeertalen zoals C/C++, Python en Simulink. Dit maakt het mogelijk om bestaande code te hergebruiken, MATLAB te koppelen aan andere systemen, en de functionaliteit uit te breiden door gebruik te maken van bibliotheken en tools uit verschillende ecosystemen.

Nadelen van Matlab:

Kosten: MATLAB is een duur softwarepakket dat kan oplopen tot 900 euro per jaar. Aangezien dit een langdurige oplossing zou worden voor het trainen van artsen, zal dit aanzienlijke kosten met zich meebrengen.

Langzamer: MATLAB is een geïnterpreteerde taal. Hierdoor kan het programma langzamer werken dan gecompileerde talen, wat een probleem vormt voor het doel van het project. Artsen moeten namelijk snel, "on the fly", aanpassingen kunnen maken aan de berekeningen.

Migratie mogelijkheden

Python:

Voordelen:

- **Groot ecosysteem:** Bibliotheken zoals NumPy, SciPy en Matplotlib bieden vergelijkbare functionaliteit als MATLAB voor numeriek rekenen, lineaire algebra en data visualisatie.
- **Open-source en gratis:** Geen licentiekosten.
- **Over het algemeen sneller:** Python kan sneller zijn dan MATLAB, vooral voor taken met grote datasets.

Nadelen:

- **Kan minder snel zijn voor specifieke taken:** Hoewel Python over het algemeen sneller is, kan MATLAB in bepaalde gevallen (bijvoorbeeld sterk geoptimaliseerde code) nog steeds sneller zijn.
- **Leereffect:** Als je geen voorkennis hebt met Python, kan het even duren om de taal te leren.

(GAVRILOVA, 2023), (Basel, 2024), (Novotny, 2022)

Libraries:

Dash:

Voordelen:

- Gebruiksvriendelijk & snel te leren
- Plotly-integratie voor interactieve grafieken

Nadelen:

- Beperkte flexibiliteit
- Afhankelijk van Plotly

(Dash Enterprise, n.d.)

Plotly:

Voordelen:

- Interactieve grafieken
- Visueel aantrekkelijk
- JavaScript-bibliotheek voor webapps & notebooks

Nadelen:

- Moeilijkheidsgraad voor beginners
- Grote bestandsgrootte

(Data Apps for Production, n.d.)

Matplotlib:

Voordelen:

- Breed gebruik & stabiliteit
- Flexibiliteit
- Werkt met andere bibliotheken

Nadelen:

- Steile leercurve
- Snelheidsproblemen met grote datasets
- Niet bedoelt voor realtime data

(Matplotlib: Visualization with Python, n.d.)

Bokeh:

Voordelen:

- Real-time data visualisatie
- Interactieve grafieken
- Server-side rendering voor efficiënte verwerking

Nadelen:

- Moeilijkheidsgraad
- Beperkte grafiektypen

(Bokeh at a Glance, n.d.)

Julia:

Voordelen:

- **Snelheid:** Julia wordt vaak als sneller beschouwd dan zowel Python als MATLAB voor vergelijkbare taken.
- **Gratis en open-source:** Geen licentiekosten.

Nadelen:

- **Jonger en kleinere community:** Julia is een nieuwere taal dan Python, wat betekent dat de community kleiner is en er minder hulpbronnen beschikbaar zijn.
- **Steilere leercurve:** Julia kan een lastigere taal zijn om te leren dan Python, vooral voor beginners.

(Christianson, 2020), (Maas, 2023), (Sekan, 2023)

Libraries:

Plotly.jl:

Voordelen:

- Creëert interactieve grafieken.
- Ondersteunt een uitgebreid assortiment grafiektypen.

Nadelen:

- Vereist Plotly.js voor het renderen van grafieken.
- De leercurve kan steiler zijn in vergelijking met andere bibliotheken.

(Rackauckas, n.d.), (gvwilson, n.d.)

VegaLite.jl:

Voordelen:

- Maakt gebruik van een declaratieve syntaxis die eenvoudig te leren en te gebruiken is.
- Biedt flexibiliteit in het definiëren van de structuur en stijl van je grafieken.
- Kan grafieken genereren voor verschillende platforms.

Nadelen:

- Biedt mogelijk een kleiner scala aan grafiektypen in vergelijking met andere bibliotheken.
- De focus ligt op statische grafieken, met minder interactieve mogelijkheden dan andere opties.

(Rackauckas, n.d.), (VegaLite.jl, n.d.)

Gadfly.jl:

Voordelen:

- Ontworpen met een gebruiksvriendelijke interface die eenvoudig te leren is.
- Produceert visueel aantrekkelijke grafieken met publicatiewaardige kwaliteit.
- Maakt integratie met LaTeX-wiskundige notaties mogelijk.

Nadelen:

- Richt zich op statische grafieken, met beperkte interactieve mogelijkheden.
- Biedt mogelijk een kleiner scala aan grafiektypen in vergelijking met andere bibliotheken.

(Rackauckas, n.d.), (Gadfly.jl, n.d.)

Conclusie

Nadat ik Python en Julia heb vergeleken voor data-analyse en visualisatie, ligt mijn voorkeur bij Python. Hier zijn de belangrijkste redenen voor mijn keuze:
















Python biedt een grote selectie aan krachtige bibliotheken zoals NumPy, SciPy en Plotly, die essentieel zijn voor numerieke berekeningen, lineaire algebra en data visualisatie. Deze tools zijn vergelijkbaar met wat MATLAB biedt, maar zonder de kosten van een licentie. Python presteert over het algemeen goed bij taken met grote datasets, wat cruciaal is voor mijn studieprojecten en toekomstige werkzaamheden. Hoewel er een leercurve is, vooral bij complexere bibliotheken, is deze over het algemeen goed te overbruggen.

Julia heeft indruk op me gemaakt vanwege zijn snelheid, die vaak sneller wordt genoemd dan Python en MATLAB voor soortgelijke taken. Als open-source taal zonder licentiekosten biedt Julia zeker aantrekkelijke voordelen. Echter, de jongere gemeenschap en steilere leercurve kunnen beperkingen vormen voor mijn huidige studie- en projectbehoeften.

Daarnaast speelt populariteit een belangrijke rol in mijn keuze. Python is aanzienlijk populairder dan Julia volgens de [TIOBE Index - TIOBE](#), waar Python op de eerste plaats staat en MATLAB op de veertiende plaats. Dit betekent dat er meer bronnen, tutorials en gemeenschapsondersteuning beschikbaar zijn voor Python. Dit maakt het eenvoudiger om hulp te vinden en problemen op te lossen, wat van onschatbare waarde is bij het leren en gebruiken van een programmeertaal. Bovendien, als dit project wordt overgedragen en er nieuwe programmeurs worden gezocht om de implementatie voort te zetten, zal het vanwege de populariteit van Python gemakkelijker zijn om gekwalificeerde ontwikkelaars te vinden.

Hoewel Julia sneller is dan Python, zijn bibliotheken die we in dit project gebruiken, zoals NumPy, geschreven in C. Dit zorgt ervoor dat de performance voor complexe berekeningen vergelijkbaar is met die van Julia, waardoor het snelheidsvoordeel van Julia minder significant is in de context van dit project.

Ik heb daarom besloten om verder te gaan met Python vanwege zijn brede toepasbaarheid, ondersteuning vanuit de gemeenschap, vergelijkbare performance dankzij geoptimaliseerde bibliotheken, en de mogelijkheid om snel te leren en toe te passen in mijn studie en toekomstige carrière in de ICT.

| Jun 2024 | Jun 2023 | Change | Programming Language | Ratings | Change |
|----------|----------|--------|--|---------|--------|
| 1 | 1 | |  Python | 15.39% | +2.93% |
| 2 | 3 | ▲ |  C++ | 10.03% | -1.33% |
| 3 | 2 | ▼ |  C | 9.23% | -3.14% |
| 4 | 4 | |  Java | 8.40% | -2.88% |
| 5 | 5 | |  C# | 6.65% | -0.06% |
| 6 | 7 | ▲ |  JavaScript | 3.32% | +0.51% |
| 7 | 14 | ▲ |  Go | 1.93% | +0.93% |
| 8 | 9 | ▲ |  SQL | 1.75% | +0.28% |
| 9 | 6 | ▼ |  Visual Basic | 1.66% | -1.67% |
| 10 | 15 | ▲ |  Fortran | 1.53% | +0.53% |
| 11 | 11 | |  Delphi/Object Pascal | 1.52% | +0.27% |
| 12 | 19 | ▲ |  Swift | 1.27% | +0.33% |
| 13 | 10 | ▼ |  Assembly language | 1.26% | -0.03% |
| 14 | 12 | ▼ |  MATLAB | 1.26% | +0.14% |
| 15 | 8 | ▼ |  PHP | 1.22% | -0.52% |

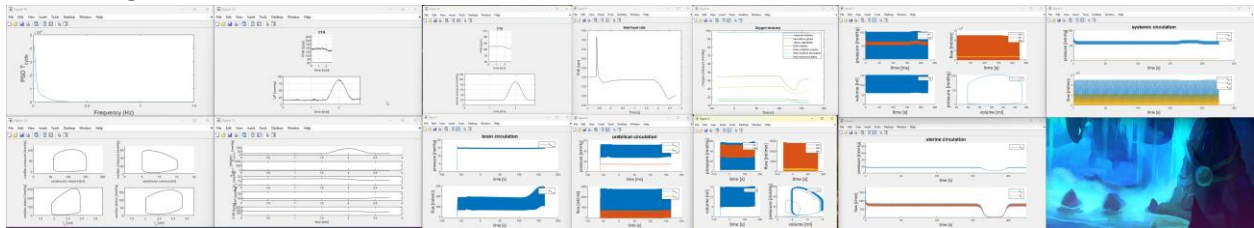
Testen

Matlab

Bij het testen van het bestaande MATLAB-project kwamen er meteen enkele problemen aan het licht. Om de MATLAB IDE te downloaden en gebruiken, moesten we ons eerst registreren. Als studenten konden we gelukkig gebruikmaken van een account via de school, waardoor er voor ons geen extra kosten aan verbonden waren. Voor anderen kan dit echter een drempel vormen.

Na de installatie van de IDE merkte ik dat het opstarten extreem lang duurde. De opstarttijd varieerde van 55 seconden tot wel 4 minuten. Toen de IDE eindelijk was opgestart, probeerde ik het programma zelf te starten. Dit duurde ook erg lang; in onze tests kostte het tussen de 35 en 40 seconden op een goede computer.

Toen het project volledig was opgestart, verschenen er 14 verschillende vensters, elk met andere data. Dit maakte het onmogelijk om een aanpassing te doen zonder de hele berekening opnieuw uit te voeren. In een trainingsscenario zou dit niet snel en effectief genoeg zijn om artsen in staat te stellen echte simulaties te maken. De gewenste performance is dat de simulatie snel kan worden gestart en dat er live updates aan de gegevens kunnen worden uitgevoerd, zonder 30 seconden te hoeven wachten tot de berekening klaar is.



Python

Om de mogelijkheden te testen van Python heb ik de verschillende libraries die eerder in dit document beschreven waren gepakt en deze tegen elkaar vergeleken in de praktijk.

Bokeh

Bokeh bleek een krachtige tool voor interactieve visualisaties. De mogelijkheid om real-time data te visualiseren met server-side rendering sprak me aan, vooral omdat het geschikt is voor dynamische toepassingen. Ik merkte echter op dat de programmeerstijl soms wat uitgebreider lijkt in vergelijking met andere bibliotheken, wat de leercurve iets kan verhogen.

Plotly

Plotly maakte indruk met zijn gebruiksvriendelijke API en de mogelijkheid om veel interactieve grafieken te creëren. De integratie met JavaScript voor rendering maakt het zeer geschikt voor webapplicaties. Ik vond het prettig om snel en effectief complexe grafieken te kunnen maken.

Dash

Dash, gebouwd bovenop Plotly, bood een uitstekende oplossing voor het ontwikkelen van interactieve visualisaties. De integratie met Plotly stelde me in staat om snel dynamische en aanpasbare visualisaties te realiseren met een minimale hoeveelheid code. De flexibiliteit en uitbreidingsmogelijkheden van Dash maken het een ideale keuze voor het project.

Conclusie

Na uitvoerig testen heb ik besloten om voor Dash te kiezen. Dash biedt de krachtige grafiekfunctionaliteiten van Plotly en combineert deze met uitgebreide mogelijkheden voor uitbreiding en aanpassing. Deze combinatie maakt Dash geschikt voor mijn project, waarbij ik complexe en interactieve visualisaties moet ontwikkelen.

Matrix

Wat is een Matrix?

Een matrix is een rechthoekig schema van getallen, ook wel elementen genoemd. De getallen zijn gerangschikt in rijen en kolommen. Het aantal rijen wordt aangeduid met m en het aantal kolommen met n . We noemen dit dan een $m \times n$ matrix.

Een matrix kan je noteren met haakjes en komma's, waarbij de getallen per rij gescheiden worden door komma's en de rijen door haakjes en puntkomma's.

Een voorbeeld van een 2×3 matrix:

$A = \begin{bmatrix} 1, & 2, & 3; \\ 4, & 5, & 6 \end{bmatrix}$

Hierbij is A de naam van de matrix, 1, 2, 3, 4, 5 en 6 zijn de elementen, en is $m = 2$ (aantal rijen) en $n = 3$ (aantal kolommen).

Waarom worden matrixen gebruikt?

Matrices worden veel gebruikt in de wetenschap, techniek, wiskunde en informatica omdat ze gemakkelijk kunnen worden gebruikt om lineaire vergelijkingen te representeren of om lineaire transformaties uit te voeren. Bijvoorbeeld, in computergraphics worden matrices gebruikt om rotatie, schaalverandering, vertaling, en andere transformaties te representeren.

Hoe word een matrix gebruik

Matrices worden gebruikt om CTG (cardiotocografie) data te simuleren tijdens de bevalling van een moeder door het vastleggen van de tijdreeksgegevens die de hartslag van de foetus en de weeënactiviteit van de moeder weergeven.

Hoe kan een matrix gebruikt worden in Python?

Python biedt een module genaamd NumPy om efficiënt met matrices te werken. NumPy biedt verschillende functies om matrices te maken, te manipuleren en te analyseren.

Voorbeelden

Matrices definiëren

```
import numpy as np

# Maak een 2 x 3 matrix met NumPy arrays
matrix_a = np.array([[1, 2, 3], [4, 5, 6]])
matrix_b = np.array([[7, 8, 9], [10, 11, 12]])
```

Optellen

```
# Tel de twee matrices op
som_matrix = matrix_a + matrix_b
print("Som van de matrices:")
print(som_matrix)
```

Aftrekken

```
# Trek matrix_b af van matrix_a
verschil_matrix = matrix_a - matrix_b
print("Verschil van de matrices:")
print(verschil_matrix)
```

Vermenigvuldigen

```
# Vermenigvuldig matrix_a met matrix_b
product_matrix = matrix_a.dot(matrix_b)
print("Product van de matrices:")
print(product_matrix)
```

Delen

```
# Deling van de matrices
deel_matrix = matrix_a / matrix_b
print("Deling van de matrices:")
print(deel_matrix)
```

Factor

```
# Vermenigvuldig matrix_a met matrix_b
product_matrix = matrix_a.dot(matrix_b)
print("Product van de matrices:")
print(product_matrix)
```

Dit zijn slechts enkele voorbeelden van de vele mogelijkheden die NumPy biedt voor het werken met matrices in Python. Naast NumPy zijn er nog andere bibliotheken beschikbaar voor matrixbewerkingen in Python, zoals SciPy en TensorFlow. SciPy wordt gebruikt voor geavanceerde statistische analyse, signaalverwerking of optimalisatieproblemen (SciPy, n.d.), terwijl TensorFlow meer gebruikt wordt voor het bouwen en trainen van complexe machine learning-modellen, met name deep learning-modellen (Tensorflow, n.d.).

Doorlopende berekening

Als we gebruik gaan maken van Matrix calculaties om continue berekeningen te maken moet het mogelijk zijn om de calculatie dus op langertermijn uit te voeren. Daarvoor kunnen we gebruik maken van een “While True” script. In deze “While true” zal alles steeds opnieuw gerund worden.

```
while True:
    # Preset data
    min_value = 10
    max_value = 21

    sleep_timer = 0.1

    # Genereer matrices met willekeurige waarden tussen 10 en 20
    matrix_a = np.random.randint(min_value, max_value, size=(2, 2))
    matrix_b = np.random.randint(min_value, max_value, size=(2, 2))

    # Optellen van de matrices
    plus_matrix = matrix_a + matrix_b

    # Print de gegenereerde matrices en hun som
    print("\nMatrix A:")
    print(matrix_a)
    print("\nMatrix B:")
    print(matrix_b)
    print("\nSom van de matrices opgeteld:")
    print(plus_matrix)

    # Wacht een seconde voordat de loop opnieuw wordt uitgevoerd
    time.sleep(sleep_timer)
```

Om ervoor te zorgen dat we niet steeds dezelfde data genereren heb ik erin gezet dat Matrix A en Matrix B steeds verschillende waarden hebben.

```
# Preset data
min_value = 10
max_value = 21

sleep_timer = 0.1

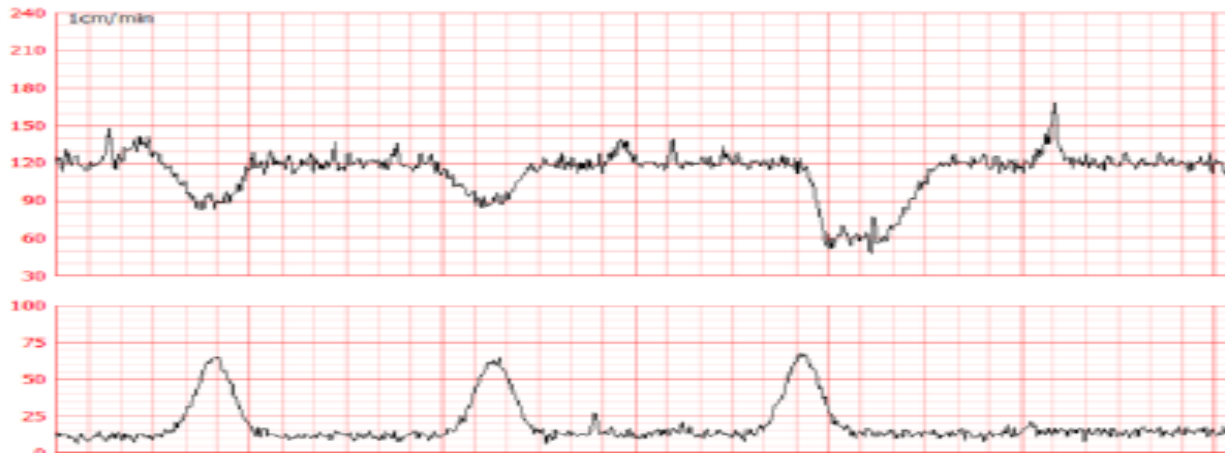
# Genereer matrices met willekeurige waarden tussen 10 en 20
matrix_a = np.random.randint(min_value, max_value, size=(2, 2))
matrix_b = np.random.randint(min_value, max_value, size=(2, 2))
```

Om de snelheid van de calculaties aan te passen kunnen we het getal bij “sleep_timer” aanpassen. Op deze manier kunnen de dus vragen aan het script om sneller of juist langzamer de calculaties te maken. Vanuit Beatrijs was de vraag om minimaal 1 datapunt te genereren per seconde. Hier zien we dat het mogelijk is vanuit de Matrix kant. Later zullen we onderzoeken of het ook mogelijk is vanuit de grafiek kant.

Grafieken

Hoe word een grafiek gebruikt in de simulatie?

Een grafiek is een visuele weergave van data die helpt om patronen, trends en relaties te begrijpen. Door data in grafieken te plotten, kunnen de artsen complexe gegevens op een eenvoudige en begrijpelijke manier terugzien.



Hoe kan een Grafiek gebruikt worden in Python?

In Python zijn er verschillende bibliotheken beschikbaar om grafieken te maken en te manipuleren. De meest gebruikte bibliotheek is Matplotlib, die krachtige functies biedt voor het maken van allerlei soorten grafieken.

Voor dit project zullen we gebruik maken van Dash, aangezien deze speciaal ontworpen is voor het bouwen van interactieve webapplicaties met grafieken. Dash maakt gebruik van Plotly om dynamische en responsieve grafieken te creëren die gemakkelijk te integreren zijn in webapplicaties.

Hoe hebben wij het gebruikt

Imports

We beginnen met het importeren van de benodigde bibliotheken voor het ontwikkelen van onze Dash-applicatie.

```
import dash
import dash_core_components as dcc
import dash_html_components as html
import plotly.graph_objs as go
```

Als eerste importeren we Dash zelf met “import dash”. Dash is een bibliotheek in Python die wordt gebruikt voor het bouwen van interactieve webapplicaties. Het stelt ons in staat om data-analyses en visualisaties te integreren in een webgebaseerde omgeving met behulp van Python.

Daarna importeren we “dash_core_components” en “dash_html_components” als “dcc” en “html”. Deze modules zijn essentieel voor het definiëren van de gebruikersinterface binnen een Dash-applicatie. dcc biedt componenten zoals grafieken, dropdowns, sliders en meer, terwijl html wordt gebruikt voor het

opzetten van de structuur en lay-out van de webpagina's met HTML-elementen zoals divs, headers en paragrafen.

Vervolgens importeren we “plotly.graph_objs” als “go”. Plotly is een bibliotheek voor interactieve datavisualisatie en biedt grafische objecten (graph objects) waarmee we complexe grafieken en visualisaties kunnen maken binnen onze Dash-applicatie. Door go te importeren, hebben we toegang tot functies en objecten waarmee we grafieken kunnen aanpassen en interactieve elementen kunnen toevoegen aan onze visualisaties.

Tot slot, om onze Dash-applicatie te initialiseren, maken we een object genaamd “app” aan met “dash.Dash(__name__)”. Hierbij is “__name__” een speciale variabele die de naam van het huidige Python-script bevat. Door app te initialiseren met Dash, leggen we de basis voor het opbouwen van een interactieve webapplicatie waarin we gebruik kunnen maken van alle functionaliteiten van Dash, inclusief interactieve datavisualisaties met behulp van Plotly.

App starten

Om de Dash-applicatie te laten draaien op een lokale server, maken we gebruik van een specifieke Python-conventie. We beginnen met het controleren van de speciale variabele __name__. Deze variabele bevat de naam van het huidige module of script dat wordt uitgevoerd. Als __name__ gelijk is aan __main__, betekent dit dat het script direct wordt uitgevoerd als het hoofdprogramma.

In ons geval gebruiken we deze conditie om ervoor te zorgen dat de Dash-applicatie wordt gestart wanneer het script rechtstreeks wordt uitgevoerd. Dit doen we door het app.run_server(debug=True) commando uit te voeren. Hierbij verwijst app naar het Dash-applicatie-object dat we eerder hebben geconfigureerd en opgebouwd.

De parameter debug=True die we doorgeven aan run_server() activeert de debug-modus van Dash. Deze modus zorgt ervoor dat er gedetailleerde foutmeldingen en nuttige informatie worden weergegeven in de console tijdens het draaien van de applicatie. Dit is vooral handig tijdens het ontwikkelen en testen van de Dash-applicatie.

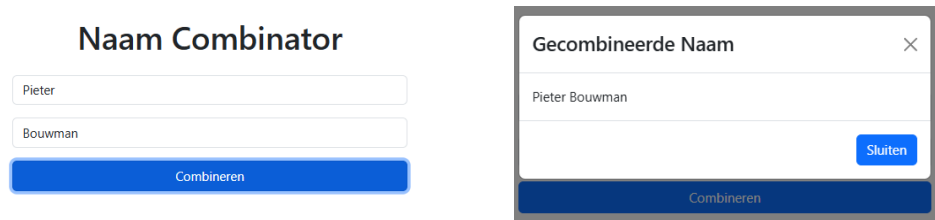
Samengevat, door deze conventie en configuratie te gebruiken, kunnen we onze Dash-applicatie effectief laten draaien op een lokale server. Dit stelt ons in staat om interactieve webapplicaties te ontwikkelen met geavanceerde datavisualisaties en interactieve gebruikerscomponenten, zoals grafieken en dropdowns, binnen een Python-omgeving.

```
# Run the app
if __name__ == '__main__':
    app.run_server(debug=True)
```


Dash GUI

Om onze grafieken op een nette en effectieve manier te visualiseren en andere website-interface mogelijkheden toe te voegen, maken we gebruik van Dash. Om een basisbegrip van Dash te krijgen, heb ik een klein project gemaakt waarin we twee inputs combineren in een modale pop-up. Hiermee demonstreer ik hoe je elementen op een pagina kunt genereren en hoe je data vanuit deze elementen kunt verzamelen en manipuleren.

In de onderstaande afbeelding zie je een eenvoudige pagina met twee invoervelden en een knop. Wanneer op deze knop wordt gedrukt, worden de waarden van de twee invoervelden gecombineerd en weergegeven in een pop-up. Er zijn verschillende stappen nodig om dit te realiseren.



Hoe gebruik je dash in Python

Imports

We beginnen met het importeren van de benodigde bibliotheken voor het ontwikkelen van onze Dash-applicatie.

```
import dash
from dash import html, dcc
from dash.dependencies import Input, Output, State
import dash_bootstrap_components as dbc
```

Als eerste importeren we Dash zelf met `“import dash”`. Dash is een library in Python die wordt gebruikt voor het bouwen van interactieve webapplicaties. Het stelt ons in staat om data-analyses en visualisaties te integreren in een webgebaseerde omgeving met behulp van Python.

Daarna importeren we `dash_core_components` en `dash_html_components` als `“dcc”` en `“html”`. Deze modules zijn essentieel voor het definiëren van de gebruikersinterface binnen een Dash-applicatie. dcc biedt componenten zoals grafieken, dropdowns, sliders en meer, terwijl html wordt gebruikt voor het opzetten van de structuur en lay-out van de webpagina's met HTML-elementen zoals divs, headers en paragrafen.

Vervolgens importeren we `“Input”`, `“Output”`, en `“State”` van `“dash.dependencies”`. Deze modules worden gebruikt voor het definiëren van callback-functies die de interactie tussen de verschillende componenten van de gebruikersinterface beheren.

Tot slot importeren we `“dash_bootstrap_components”` (dbc). Dit is een bibliotheek die Bootstrap-componenten integreert in Dash-applicaties, waardoor we toegang hebben tot gestileerde componenten zoals modalen, knoppen, en invoervelden. We gebruiken Bootstrap omdat het snel en gratis een mooi stylingpakket biedt, waardoor we onze Dash-applicatie gemakkelijk een professionele uitstraling kunnen geven.

Initialisatie

Om onze Dash-applicatie te initialiseren, maken we een object genaamd `app` aan met `"dash.Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP])"`. Hierbij is `__name__` een speciale variabele die de naam van het huidige Python-script bevat. Door `app` te initialiseren met Dash en een externe Bootstrap-stylesheet toe te voegen, leggen we de basis voor het opbouwen van een interactieve en gestileerde webapplicatie.

```
# Initialiseer de Dash-app met Bootstrap stylesheet
app = dash.Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP])
```

Layout

De layout van de applicatie wordt samengesteld met behulp van verschillende componenten:

dbc.Container: Deze component wordt gebruikt om de inhoud van de applicatie te structureren.

dbc.Row en **dbc.Col:** Deze componenten worden gebruikt om rijen en kolommen te creëren.

html.H1: Hiermee maken we de titel van de pagina, in dit geval "Naam Combinator".

dcc.Input: Deze component maakt twee invoervelden: één voor de voornaam (`id='first-name'`) en één voor de achternaam (`id='last-name'`).

html.Button: Deze component maakt een knop met de tekst "Combineren" (`id='combine-button'`).

dbc.Modal: Deze component maakt een modaal (pop-up) voor het tonen van de gecombineerde naam. Het bestaat uit een header, een body (die wordt bijgewerkt door de callback-functie), en een footer met een sluitknop.

```
# Definieer de layout van de app
app.layout = dbc.Container(
    [
        dbc.Row(
            dbc.Col(
                html.H1("Naam Combinator", className="text-center mt-4 mb-4"),
                width=12
            )
        ),
        dbc.Row(
            dbc.Col(
                [
                    dcc.Input(
                        id='first-name',
                        type='text',
                        placeholder='Voornaam',
                        className='form-control mb-3',
                        style={'width': '80%', 'margin': '0 auto'}
                    ),
                    dcc.Input(
                        id='last-name',
                        type='text',
                        placeholder='Achternaam',
                        className='form-control mb-3',
                        style={'width': '80%', 'margin': '0 auto'}
                    ),
                    html.Button(
                        'Combineren',
                        id='combine-button',
                        className='btn btn-primary',
                        style={'width': '80%', 'margin': '0 auto', 'display': 'block'}
                    ),
                ],
                width=6,
                className="mx-auto text-center"
            ),
            dbc.Modal(
                [
                    dbc.ModalHeader(dbc.ModalTitle("Gecombineerde Naam")),
                    dbc.ModalBody(id='modal-body'),
                    dbc.ModalFooter(
                        dbc.Button("Sluiten", id="close", className="ms-auto", n_clicks=0)
                    ),
                ],
                id="modal",
                is_open=False,
            ),
        ],
        className="p-4"
    )
)
```

Callback-functie

De callback-functie regelt de interactie tussen de verschillende componenten van de Dash-applicatie:

Output: Hiermee geeft de callback aan welke onderdelen van de gebruikersinterface (UI) moeten worden bijgewerkt. Bijvoorbeeld, de callback kan de status van een modale popup (`is_open`) en de inhoud van de modale body (`children`) bijwerken.

Input: Dit specificeert de gebeurtenissen of interacties die de callback activeren. Bijvoorbeeld, de callback kan worden geactiveerd wanneer een gebruiker op een combinatieknop klikt of wanneer de sluitknop van een modale popup wordt ingedrukt.

State: Hiermee haalt de callback de huidige status en waarden op van verschillende componenten in de applicatie. Bijvoorbeeld, de callback kan de huidige waarden ophalen die zijn ingevoerd in invoervelden zoals `first-name` en `last-name`, evenals de huidige status van de modale popup (`is_open`).

```
# Definieer de callback om de modal bij te werken
@app.callback(
    Output('modal', 'is_open'),
    Output('modal-body', 'children'),
    Input('combine-button', 'n_clicks'),
    Input('close', 'n_clicks'),
    State('first-name', 'value'),
    State('last-name', 'value'),
    State('modal', 'is_open'),
)
```

De functie `combine_names` binnen de callback controleert welke knop is ingedrukt (de combinatieknop of de sluitknop). Als de combinatieknop is ingedrukt en beide invoervelden zijn ingevuld, combineert de functie de voornaam en achternaam tot een volledige naam. Vervolgens opent de functie de modale popup om de gecombineerde naam weer te geven. Als alleen de sluitknop is ingedrukt, sluit de functie de weergave van de modale popup.

Verduidelijking

Dit proces van callback-functies stelt Dash in staat om dynamisch te reageren op gebruikersinteracties, gegevens te verwerken en de gebruikersinterface dienovereenkomstig bij te werken. Door dit mechanisme kunnen Dash-applicaties soepel en responsief werken, wat essentieel is voor het bouwen van moderne en interactieve webapplicaties met Python.

```
def combine_names(n_clicks_combine, n_clicks_close, first_name, last_name, is_open):
    if n_clicks_combine or n_clicks_close:
        if n_clicks_combine > 0 and first_name and last_name:
            combined_name = f"{first_name} {last_name}"
            return not is_open, combined_name
        return not is_open, ''
    return is_open, ''
```

App starten

Om de Dash-applicatie op een lokale server te laten draaien, volgen we een specifieke Python-conventie. Raadpleeg pagina 15 voor een gedetailleerde uitleg.

```
# Run the app
if __name__ == '__main__':
    app.run_server(debug=True)
```

Bibliografie

- Basel, K. (2024, May 25). *Python Pros and Cons in 2024*. Retrieved from netguru:
<https://www.netguru.com/blog/python-pros-and-cons>
- Bokeh at a Glance*. (n.d.). Retrieved from bokeh: <https://bokeh.org/>
- Christianson, A. (2020). *What are the pros and cons of the Julia programming language?* Retrieved from
Quora: <https://www.quora.com/What-are-the-pros-and-cons-of-the-Julia-programming-language>
- Dash Enterprise*. (n.d.). Retrieved from plotly: <https://plotly.com/dash/>
- Data Apps for Production*. (n.d.). Retrieved from plotly: <https://plotly.com/>
- Gadfly.jl*. (n.d.). Retrieved from gadflyjl: <https://gadflyjl.org/stable/>
- GAVRILOVA, Y. (2023, October 31). *Pros and Cons of Python Programming Language*. Retrieved from
serokell: <https://serokell.io/blog/python-pros-and-cons>
- gvwilson. (n.d.). *Plotly.jl*. Retrieved from github: <https://github.com/plotly/Plotly.jl>
- Maas, M. D. (2023, June 21). *Is the Julia Language Worth Learning? (Pros and Cons)*. Retrieved from
matecdev: <https://www.matecdev.com/posts/julia-worth-learning.html>
- Matplotlib: Visualization with Python*. (n.d.). Retrieved from matplotlib: <https://matplotlib.org/>
- Novotny, J. (2022, March 23). *The Pros and Cons of Python Programming*. Retrieved from linode:
<https://www.linode.com/docs/guides/pros-and-cons-of-python/>
- Rackauckas, C. (n.d.). *Summary of Julia Plotting Packages*. Retrieved from juliabloggers:
<https://www.juliabloggers.com/summary-of-julia-plotting-packages/>
- SciPy. (n.d.). *SciPy User Guide*. Retrieved from docs.scipy.org:
<https://docs.scipy.org/doc/scipy/tutorial/index.html#user-guide>
- Sekan, F. (2023, January 20). *Pros and Cons of Julia in Data Science*. Retrieved from medium:
<https://medium.com/@filip.sekan/pros-and-cons-of-julia-in-data-science-3f386cb71757>
- Tensorflow. (n.d.). *An end-to-end platform for machine learning*. Retrieved from Tensorflow:
<https://www.tensorflow.org/>
- VegaLite.jl*. (n.d.). Retrieved from queryverse: <https://www.queryverse.org/VegaLite.jl/stable/>