



DASH BY PLOTLY

INTRODUCTION

But :

Apprendre à créer des dashboard (tableau de bord) en Python à l'aide de Dash.

Dash est un framework Python développé pour faciliter la construction d'applications Web. Il s'est construit autour de Flask, Plotly.js, React et React Js. Il vous permet de créer des dashboards en utilisant uniquement le langage python.

Dash est open source et ses applications s'exécutent sur un navigateur Web. Nous allons voir ensemble les principes de base de Dash.

Installation de Dash :

Pour commencer à utiliser Dash, nous devons installer plusieurs packages.

1. Le noyau principal du dashboard (core).
2. La partie interface de Dash (front-end).
3. Les composants HTML Dash.
4. Les composants principaux.
5. Plotly

Lignes de commandes :

```
pip install dash==0.21.1
pip install dash-renderer==0.13.0
pip install dash-html-components==0.11.0
pip install dash-core-components==0.23.0 # pour générer du contenu html
pip install plotly --upgrade
```

Disposition de l'application Dash :

Une application Dash est généralement composée de deux parties.

1. La première partie est la mise en page. Ca décrit à quoi ressemblera l'application.
2. La deuxième partie décrit l'interactivité de l'application. Dash fournit des classes HTML qui permettent de générer du contenu HTML avec Python. Pour utiliser ces classes, il est nécessaire d'importer `dash_core_components` et `dash_html_components`. Vous pouvez également créer vos propres composants personnalisés à l'aide de Javascript et React Js.

Pour commencer, nous allons créer un fichier appelé `tuto.py` :

```
import dash
import dash_core_components as dcc
import dash_html_components as html
```

Description :

Tout comme dans Flask, nous initialisons Dash en appelant la classe Dash de dash.

Une fois cela fait, nous pouvons créer la mise en page de notre application.

Nous utilisons la classe Div de `dash_html_components` pour créer une div HTML.

Nous utilisons ensuite les composants HTML pour générer des composants HTML tels que H1, H2 etc. `dash_html_components` possède toutes les balises HTML.

Pour créer un graphique sur la mise en page, nous utilisons la classe Graph de `dash_core_components`.

Le graphique rend les visualisations de données interactives à l'aide de plotly.js.

La classe Graph attend un objet figure avec les données à tracer et les détails de disposition.

Dash permet également de faire des styles tels que changer la couleur d'arrière-plan et la couleur du texte. Vous pouvez modifier l'arrière-plan en utilisant l'attribut style et en passant un objet avec votre couleur spécifique.

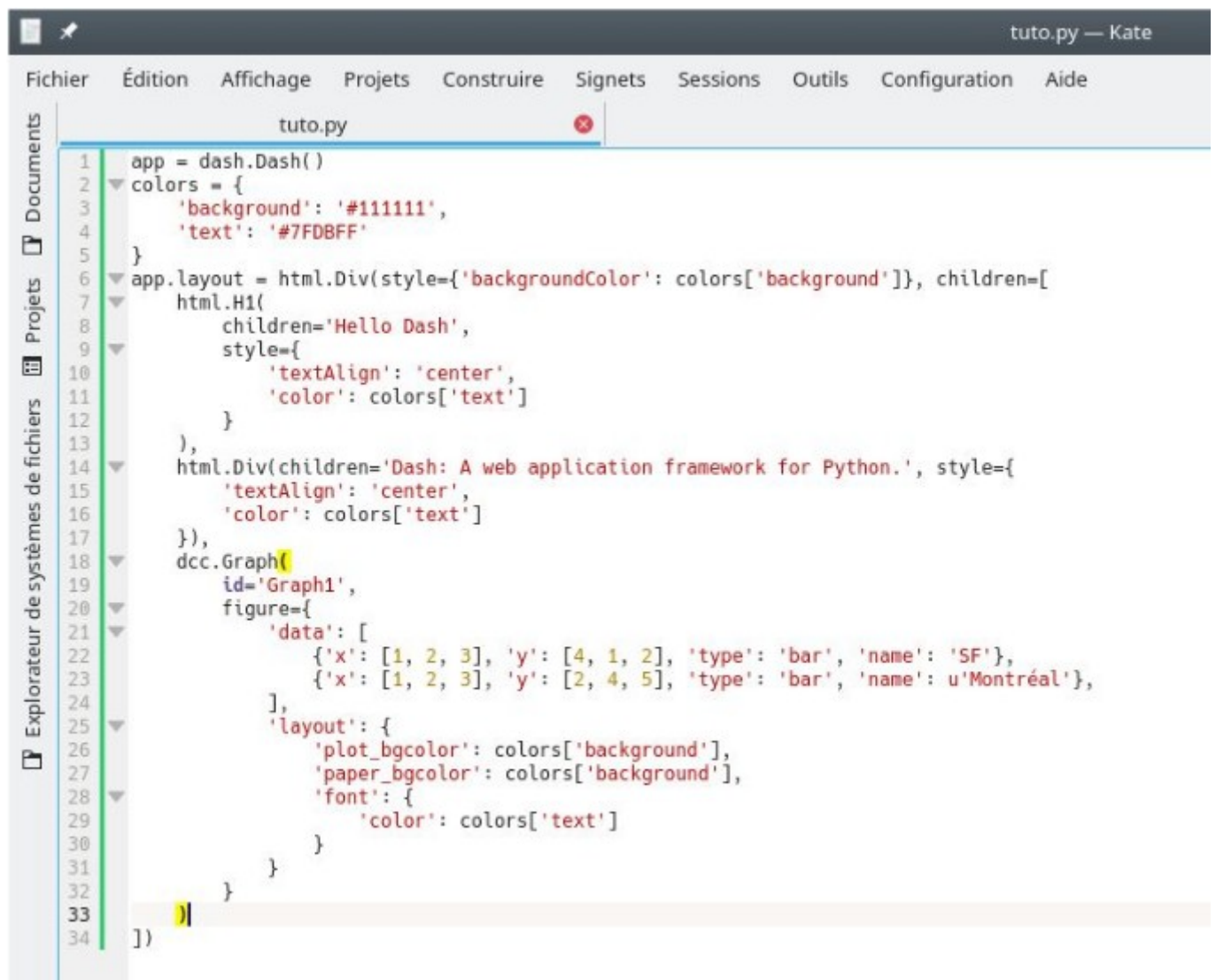
Dans notre cas, nous allons définir un dictionnaire de couleurs avec l'arrière-plan et la couleur du texte que nous souhaitons.

De même, nous pouvons changer l'arrière-plan de la mise en page en utilisant l'attribut plot_bgcolor.

En HTML, la propriété de style est spécifiée à l'aide d'un point-virgule, mais dans Dash, un dictionnaire est fourni.

Les mots clé du dictionnaire sont camelCased (par exemple text-align est textAlign).

Au lieu d'utiliser des classes comme en HTML, on utilise className.

A screenshot of a code editor window titled 'tuto.py — Kate'. The editor shows a Python script for a Dash application. The script defines a color dictionary, sets the page background and text color, adds a title and a paragraph, and includes a bar chart. The bar chart has two series: 'SF' with values [4, 1, 2] and 'Montréal' with values [2, 4, 5]. The chart's background and paper background are set to the same color as the page background, and the text color is set to the same color as the text in the paragraph. The script is as follows:

```
1 app = dash.Dash()
2 colors = {
3     'background': '#111111',
4     'text': '#7FDBFF'
5 }
6 app.layout = html.Div(style={'backgroundColor': colors['background']}, children=[
7     html.H1(
8         children='Hello Dash',
9         style={
10             'textAlign': 'center',
11             'color': colors['text']
12         }
13     ),
14     html.Div(children='Dash: A web application framework for Python.', style={
15         'textAlign': 'center',
16         'color': colors['text']
17     }),
18     dcc.Graph(
19         id='Graph1',
20         figure={
21             'data': [
22                 {'x': [1, 2, 3], 'y': [4, 1, 2], 'type': 'bar', 'name': 'SF'},
23                 {'x': [1, 2, 3], 'y': [2, 4, 5], 'type': 'bar', 'name': 'Montréal'},
24             ],
25             'layout': {
26                 'plot_bgcolor': colors['background'],
27                 'paper_bgcolor': colors['background'],
28                 'font': {
29                     'color': colors['text']
30                 }
31             }
32         }
33     )
34 ])
```

Visualisation de app.py :

A

fin de visualiser notre application, nous devons exécuter notre serveur Web comme dans Flask.

Rappelez-vous que Dash est construit au-dessus de Flask.

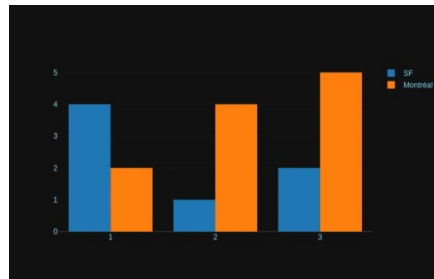
Nous allons également définir le débogage à 'true' pour nous assurer de ne pas avoir à actualiser le serveur à chaque fois que nous apportons des modifications.

Pour cela, rajouter ces deux lignes :

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

Exécution de l'application :

Exécuter dans un terminal, la commande : **python tuto.py**
Cela va démarrer un nouveau serveur Web à l'adresse **http://127.0.0.1:8050/**.
Vous devriez visualiser votre nouveau dashboard !!



Scatter plots : diagramme de dispersion :

Afin de tracer un nuage de points, nous importons les composants de dash standard comme dans l'exemple précédent.

Nous devons également importer graph_objs de Plotly afin de tracer le nuage de points.

Comme mentionné précédemment, nous utilisons la classe Div et les composants Graph de Dash pour y parvenir.

Le composant Graph prend un objet figure qui contient les données et la description de la disposition.

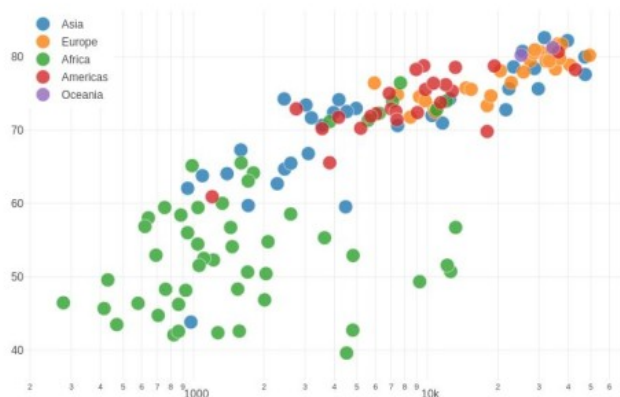
Nous traçons le nuage de points en utilisant la propriété de dispersion graph_objs.

Afin de nous assurer que le tracé est un nuage de points, nous passons un attribut de mode et le définissons comme marqueurs (markers).

Sinon, nous aurions des lignes sur le graphique.

Analyse du diagramme :

Vous venez de tracer le PIB par habitant en fonction de l'espérance de vie suivant les différents continents.



Exercice

Créer un dashboard Dash affichant un scatter plots avec le nombre d'étudiants total en fonction du Ratio étudiant féminin / étudiant masculin, pour les 20 premières université du classement.

IMPORTATION DES LIBRAIRIES

```
import dash
import dash_core_components as dcc
import dash_html_components as html
import pandas as pd
import plotly.graph_objs as go
print("tout est ok")
```

CREATION DE L'APPLICATION

```
app = dash.Dash()
```

RECUPERATION DES DONNEES

```
timesData = pd.read_csv("donnees/timesData.csv")
df = timesData.iloc[:20,:] # récupération des 20 premières lignes
```

```
# CREATION DU LAYOUT
```

```
app.layout = html.Div([
    dcc.Graph(
        id="Etudiants et ratios mâle femelle",
        figure = {
            "data": [
                go.Scatter(
                    x=df["female_male_ratio"], # récupération des données
                    y=df["num_students"],
                    text = df["university_name"],
                    mode="markers", # avec lines+markers : incompréhension
                    opacity=0.8,
                    marker={
                        "size": 25,
                        "color": "#0099FF",
                        "line": {"width": 5, "color": "black"}
                    }
                )
            ],
            "layout": go.Layout(
                xaxis={"title": "Ratio mâle femelle"},
                yaxis={"title": "nb total d'étudiants"},
                margin={'l': 40, 'b': 40, 't': 10, 'r': 10},
                hovermode='closest',
                width = 1000,
                height= 400
            )
        }
    )
])

if __name__ == "__main__":
    app.run_server(debug=True)
```