

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
```

```
pip install dash-renderer
```

```
pip install dash-html-components
```

```
pip install dash-core-components
```

```
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.



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.**

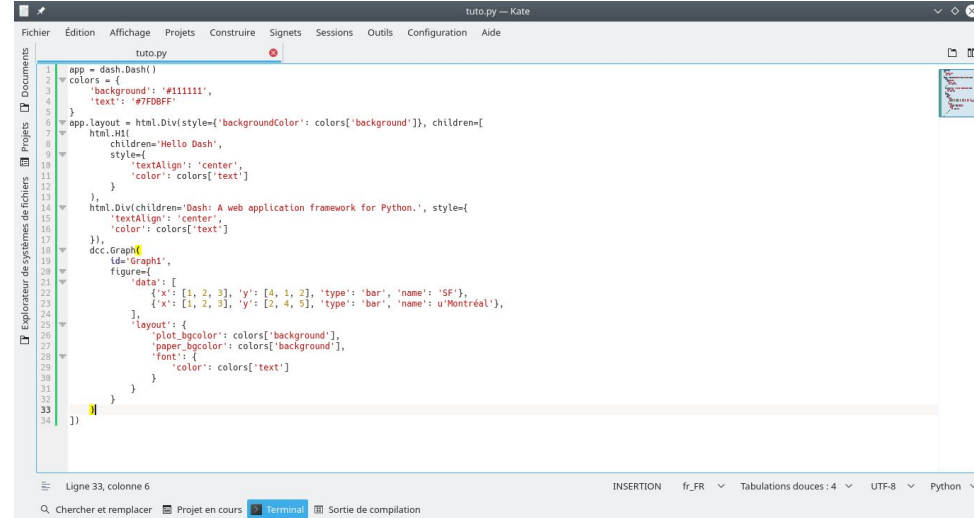


Description

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`.

Description

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 web application. The script defines a Dash app, sets a background color and text color, and creates a layout with a title, a description, and a graph. The graph displays two bar charts. The status bar at the bottom indicates 'Ligne 33, colonne 6' and 'Python' is selected as the interpreter.

```
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': 'u'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

Afin 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)
```

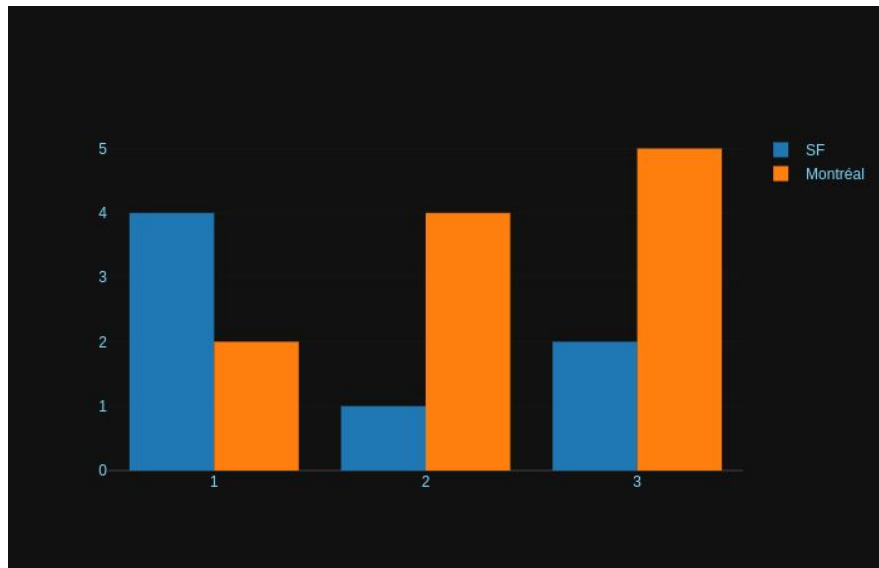


Exécution de l'application

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 !!





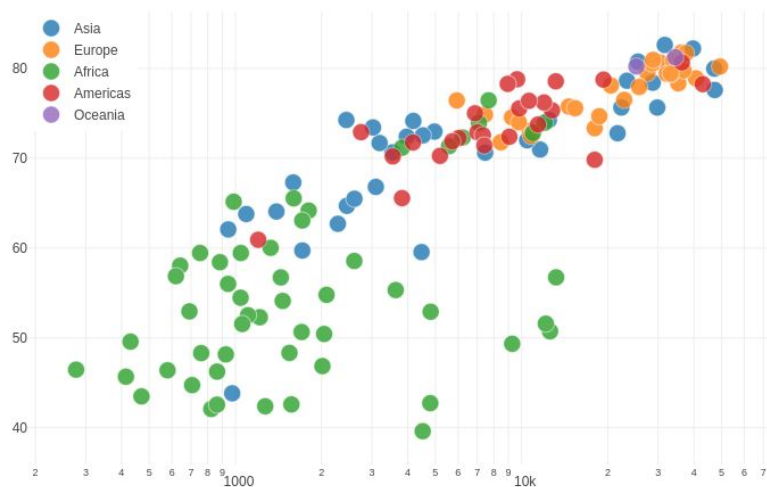
Scatter plots : diagramme de dispersion

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

En reprenant le fichier de données du classement des universités, `timesData.csv`

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.