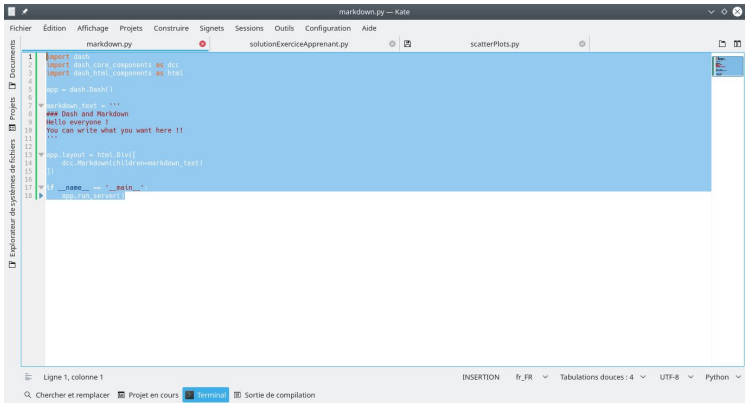
Dash et Heroku

Markdown

Parfois, vous devrez peut-être inclure beaucoup de texte dans vos tableaux de bord. Vous pouvez le faire en utilisant l'**attribut Mardown dash_core_components** comme dans l'exemple :



Composants dash_core_components

Vous pouvez générer une **liste déroulante.** Pour ce faire, appeler Dropdown appartenant à dash_core_components puis passer les options sous forme de liste de dictionnaires. Vous pouvez définir la valeur par défaut à l'aide de l'attribut ‘value’ puis passer l'option par défaut.

Cases à cocher

Pour générer des cases à cocher, vous appelez l'attribut ‘Checklist’ de dash_core_components.

Liste déroulante à sélection multiple

La génération d'une liste déroulante à sélection multiple est similaire. **Les seules modifications sont que vous définissez l'attribut ‘multi’ à ‘true’ car il est faux par défaut**. Vous pouvez ensuite spécifier les éléments que vous souhaitez être multi-sélectionnés par défaut en spécifiant l'attribut ‘value’.

Boutons Radio

Les boutons radio peuvent être générés à l'aide de l'attribut **RadioItems**. Vous passez ensuite les options sous forme de liste de dictionnaires. Vous pouvez également définir une valeur par défaut en spécifiant l'attribut values.

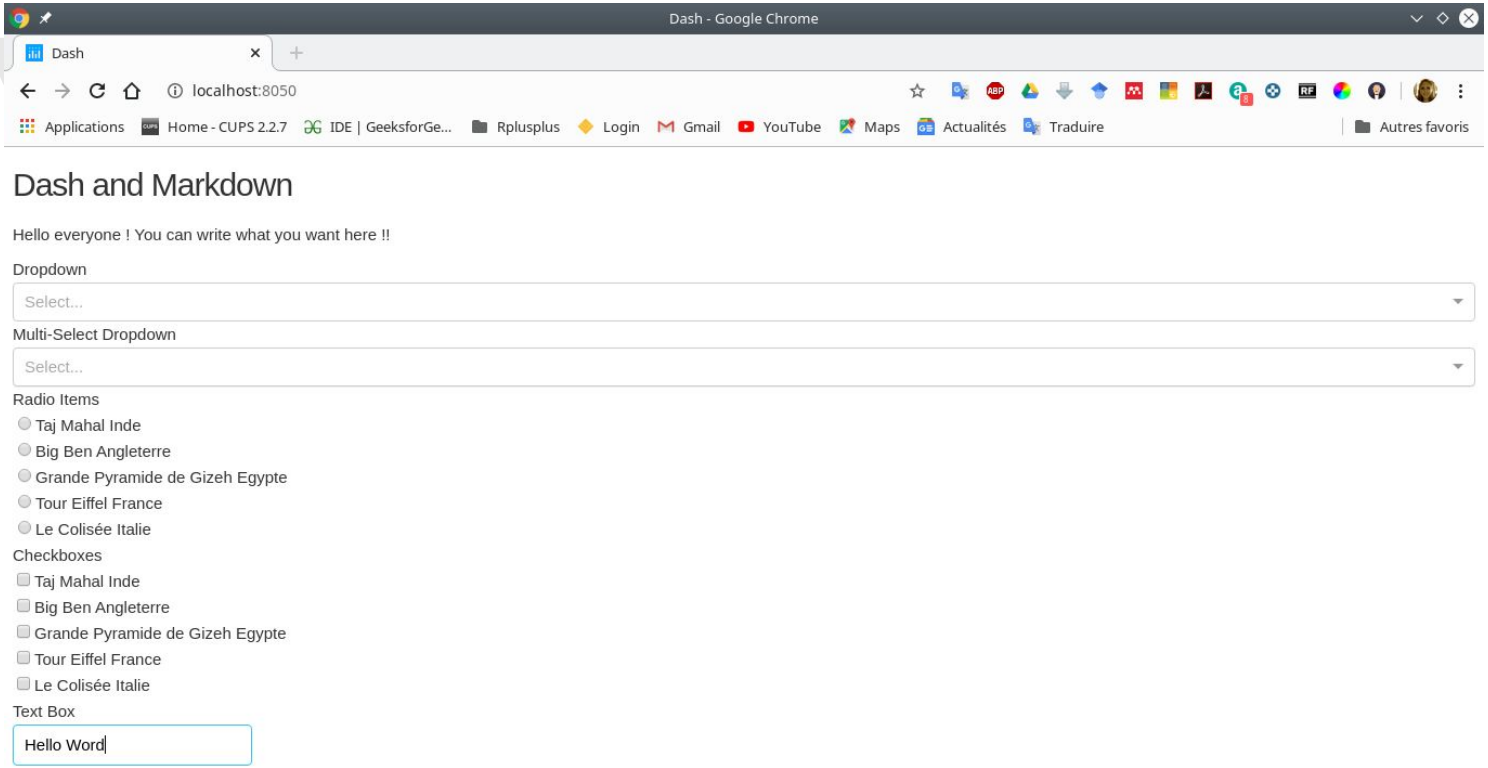
Zone de saisie de texte

Il est probable que vous ayez besoin de zones de saisie de textes dans votre application. Pour les générer, vous utiliserez l'attribut ‘Input’. À l'aide de la balise ‘Html Label’, vous pouvez créer une étiquette pour le fichier d'entrée. En utilisant l'attribut ‘value’, vous pouvez spécifier du texte dans le champ et spécifier le type de type d'utilisation. Vous pouvez également indiquer s'il s'agit d'un champ de texte, d'un numéro, etc.

Aide spécifique à un composant

Étant donné que les composants Dash sont déclaratifs, l'appel de la commande help sur l'un d'eux génèrera l'aide pour ce composant.

help(dcc.Input)


Dash - Google Chrome
Dash
localhost:8050
Applications
Home - CUPS 2.2.7
IDE | GeeksforGe...
Rplusplus
Login
Gmail
YouTube
Maps
Actualités
Traduire
Autres favoris
Dash and Markdown
Hello everyone ! You can write what you want here !!
Dropdown
Select...
Multi-Select Dropdown
Select...
Radio Items
Taj Mahal Inde
Big Ben Angleterre
Grande Pyramide de Gizeh Egypte
Tour Eiffel France
Le Colisée Italie
Checkboxes
Taj Mahal Inde
Big Ben Angleterre
Grande Pyramide de Gizeh Egypte
Tour Eiffel France
Le Colisée Italie
Text Box
Hello Word

Authentification avec Dash

Dash fournit une authentification via un package distinct appelé '**dash-auth**'. Il fournit deux modes d'authentification HTTP '**Basic Auth**' et '**Plotly OAuth**'. Dans 'Basic Auth', vous stockez en dur un ensemble de noms d'utilisateur et de mots de passe dans votre application. Cette méthode présente certaines contraintes, notamment le fait que les utilisateurs ne peuvent pas se déconnecter de l'application, ils ne peuvent pas non plus créer de comptes ou modifier les mots de passe, et le développeur est responsable de la sécurité du stockage des noms d'utilisateur et des mots de passe. 'Plotly OAuth' fournit une authentification via votre compte Plotly en ligne, mais il n'est pas gratuit.

dash-auth

pip install dash-auth

Définissez ensuite les paires nom d'utilisateur et mot de passe que vous souhaitez avoir dans votre application.

```
VALID_USERNAME_PASSWORD_PAIRS = [
    ['anne', 'laure']
]
```

BasicAuth

L'utilitaire dash_auth.BasicAuth de Dash se charge ensuite de l'authentification une fois que les paires de mots de passe sont définies. Tout ce que vous avez à faire est de transmettre les paires de mots de passe et le nom de l’application à dash_auth.BasicAuth.

Aller boire un café !!!!

Il va vous falloir des forces :-)

Application interactives

Pour utiliser le mode interactif de Dash, **nous devons importer les entrées et les sorties de dash.dependencies**. **Celles-ci sont différentes des entrées HTML**. L'entrée HTML est importée à partir des composants principaux du tableau de bord. Nous allons créer un texte d'entrée et lui lier un rappel de sorte que chaque fois que l'on tape quelque chose dans cette case, my-div sera mis à jour. **Dash fournit un décorateur @app** qui permet de lier une fonction de rappel à my-div et au champ de saisie HTML. **Notez que nous utilisons le décorateur avant de déclarer la fonction update_output_div.**

Heroku

Heroku est une plateforme cloud en tant que service (PaaS) prenant en charge plusieurs langages de programmation. C’est l'une des premières plateformes cloud. Heroku est en développement depuis juin 2007. Au départ, elle ne supportait que le langage de programmation Ruby, mais elle prend désormais en charge Java, Node.js, Scala, Clojure, Python, PHP et Go. Pour cette raison, Heroku est considéré comme une plateforme polyglotte car elle possède des fonctionnalités permettant à un développeur de créer, d'exécuter et de mettre à l'échelle des applications de manière similaire dans la plupart des langues.

Hébergement de tableaux de bord sur Heroku

L'hébergement de dashboards sur Heroku est assez facile.

1. Créez un répertoire qui contiendra tous vos fichiers de projets. Vous pouvez le faire sur Linux en utilisant la commande mkdir

   $ mkdir herokuDir
   $ cd herokuDir

Hébergement de tableaux de bord sur Heroku

2. Initialiser le dossier avec git et virtualenv (qui permet de créer un environnement virtuel pour contenir toutes les dépendances Python). Après avoir créé l'environnement, nous l'activons ce dernier à l'aide de la commande 'source'.

$ git init
$ virtualenv venv
$ source venv/bin/activate

Installation Dash dans venv

$ pip install dash

$ pip install dash-renderer

$ pip install dash-core-components

$ pip install dash-html-components

$ pip install plotly

Serveur de production

nous avons besoin d'un serveur Web Python. Nous n'utilisons jamais le serveur de développement de Flask en production. Nous utilisons le serveur Web Gunicorn pour cette fonction :

$ pip install gunicorn

Configugation de l’environnement

Ensuite, nous devons créer quelques fichiers dans notre dossier:

1. **app.py** où nous allons coder notre application dash.
2. Un **.gitignore** pour s'assurer que les fichiers inutiles ne sont pas poussés en production Exemple : gitignore
3. Un fichier **requirements.txt** qui contiendra toutes les dépendances Python et leurs versions.
4. Un **Procfile** pour le déploiement.

Ouvrez app.py. Ceci n'est qu'un exemple pour que vous puissiez utiliser votre propre dashboard.

Le server d’application

Il est important de se rappeler que son application Dash est également une application Flask. Pour le déploiement, nous devons accéder à l'instance d'application Flask. Dash nous permet de le faire en utilisant app.server :

```
app = dash.Dash(__name__)

server = app.server
```

Informations de déploiement

Précisons maintenant nos informations de déploiement dans le Procfile. Nous utilisons la variable web pour spécifier notre serveur (gunicorn). Nous spécifions également les détails de notre application en utilisant la variable d'application. app fait référence au nom de fichier app.py et server fait référence à la variable serveur à l'intérieur de ce fichier.

web: gunicorn app:server

Nous devons également écrire nos dépendances Python dans le fichier requirements. Heroku les installera pendant le déploiement.

pip freeze > requirements.txt

Créer une application Heroku

1. Créer un compte Heroku : https://signup.heroku.com/login

2. Installer Heroku CLI : https://devcenter.heroku.com/articles/heroku-cli

   curl https://cli-assets.heroku.com/install.sh | sh

3. L'étape suivante consiste à créer une application Heroku sur le terminal et à ajouter tous nos packages d'applications. Une fois que vous avez ‘commité’ les modifications, Faites un push de votre application à heroku master. La sortie de cette commande aura un lien vers votre application Dash ‘live’ sur Heroku.

Créer une application Heroku

$ heroku create annedashboard #adaptez le nom

$ git add .

$ git commit -m "add my first applications"

$ git push heroku master

Le résultat est pas très fameux :-)

https://annedashboard.herokuapp.com/