Streamlit 0

Installation et test

```
pip install streamlit
```

Exemple d'utilisation :

```
streamlit run sample.py
```

1. Titre :

```
# import module
import streamlit as st

# Title
st.title("Hello World !!!")
```

2. En-tête et sous-en-tête :

```
# Header
st.header("This is a header")

# Subheader
st.subheader("This is a subheader")
```

3. Texte :

```
# import module
import streamlit as st
```

```
# Text
st.text("Hello world !!!")
```

4. Démarquage :

```
# import module
import streamlit as st

# Markdown
st.markdown("### This is a markdown")
```

5. Succès, informations, avertissement, erreur, exception :

```
# success
st.success("Success")

# success
st.info("Information")

# success
st.warning("Warning")

# success
st.error("Error")

# Exception - This has been added later
exp = ZeroDivisionError("Trying to divide by Zero")
st.exception(exp)
```

6. Écrivez :

Grâce à la fonction d'écriture, nous pouvons également afficher le code au format de codage. Ceci n'est pas possible en utilisant st.text(").

```
# Write text
st.write("Text with write")

# Writing python inbuilt function range()
st.write(range(10))
```

7. Afficher les images :

```
# Display Images

# import Image from pillow to open images
from PIL import Image
img = Image.open("streamlit.png")

# display image using streamlit
# width is used to set the width of an image
st.image(img, width=200)
```

8. Case à cocher :

Une case à cocher renvoie une **valeur booléenne** . Lorsque la case est cochée, elle renvoie une valeur **True** , sinon elle renvoie une valeur **False .**

```
# checkbox
# check if the checkbox is checked
# title of the checkbox is 'Show/Hide'
if st.checkbox("Show/Hide"):

    # display the text if the checkbox returns True value
    st.text("Showing the widget")
```

9. Bouton radio :

```
# radio button
# first argument is the title of the radio button
```

```
# second argument is the options for the radio button
status = st.radio("Select Gender: ", ('Male', 'Female'))

# conditional statement to print
# Male if male is selected else print female
# show the result using the success function
if (status == 'Male'):
    st.success("Male")
else:
    st.success("Female")
```

10. Boîte de sélection :

Vous pouvez sélectionner n’importe quelle option dans la zone de sélection.

```
# Selection box

# first argument takes the titleof the selectionbox
# second argument takes options
hobby = st.selectbox("Hobbies: ",
                     ['Dancing', 'Reading', 'Sports'])

# print the selected hobby
st.write("Your hobby is: ", hobby)
```

11. Boîte de sélection multiple :

La zone de sélection multiple renvoie la sortie sous la forme d'une liste. Vous pouvez sélectionner plusieurs options.

```
# multi select box

# first argument takes the box title
# second argument takes the options to show
hobbies = st.multiselect("Hobbies: ",
                         ['Dancing', 'Reading', 'Sports'])
```

```
# write the selected options
st.write("You selected", len(hobbies), 'hobbies')
```

12. Bouton :

st.button() renvoie une **valeur booléenne** . Il renvoie une valeur **True** lorsque vous cliquez dessus, sinon renvoie **False** .

```
# Create a simple button that does nothing
st.button("Click me for no reason")

# Create a button, that when clicked, shows a text
if(st.button("About")):
    st.text("Welcome To US!!!")
```

13. Saisie de texte :

```
# Text Input

# save the input text in the variable 'name'
# first argument shows the title of the text input box
# second argument displays a default text inside the text input
area
name = st.text_input("Enter Your name", "Type Here ...")

# display the name when the submit button is clicked
# .title() is used to get the input text string
if(st.button('Submit')):
    result = name.title()
    st.success(result)
```

14. Curseur :

```
# slider

# first argument takes the title of the slider
# second argument takes the starting of the slider
# last argument takes the end number
level = st.slider("Select the level", 1, 5)

# print the level
# format() is used to print value
# of a variable at a specific position
st.text('Selected: {}'.format(level))
```

Autres possibilités

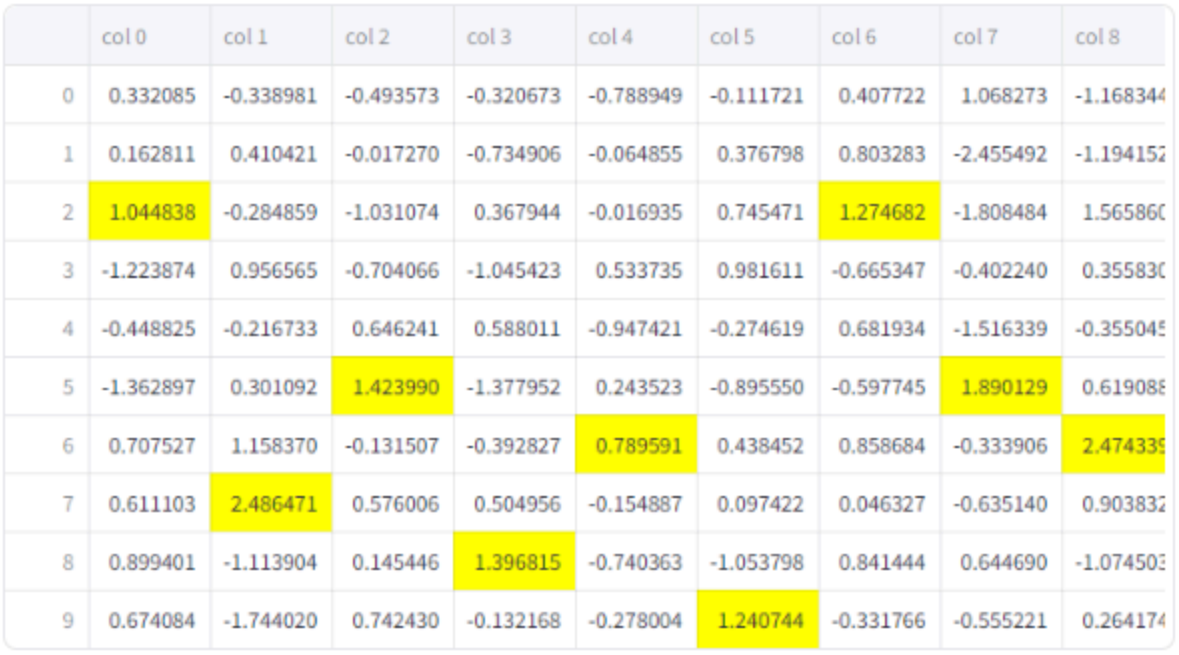
Ecrire un dataframe

```
import streamlit as st
import pandas as pd

st.write("Here's our first attempt at using data to create a
table:")
st.write(pd.DataFrame({
    'first column': [1, 2, 3, 4],
    'second column': [10, 20, 30, 40]
}))
```

Avec style

```
import streamlit as st
import numpy as np
import pandas as pd

dataframe = pd.DataFrame(
    np.random.randn(10, 20),
    columns=('col %d' % i for i in range(20)))
```

```
st.dataframe(dataframe.style.highlight_max(axis=0))
```



	col 0	col 1	col 2	col 3	col 4	col 5	col 6	col 7	col 8
0	0.332085	-0.338981	-0.493573	-0.320673	-0.788949	-0.111721	0.407722	1.068273	-1.168344
1	0.162811	0.410421	-0.017270	-0.734906	-0.064855	0.376798	0.803283	-2.455492	-1.194157
2	1.044838	-0.284859	-1.031074	0.367944	-0.016935	0.745471	1.274682	-1.808484	1.565860
3	-1.223874	0.956565	-0.704066	-1.045423	0.533735	0.981611	-0.665347	-0.402240	0.355830
4	-0.448825	-0.216733	0.646241	0.588011	-0.947421	-0.274619	0.681934	-1.516339	-0.355045
5	-1.362897	0.301092	1.423990	-1.377952	0.243523	-0.895550	-0.597745	1.890129	0.619088
6	0.707527	1.158370	-0.131507	-0.392827	0.789591	0.438452	0.858684	-0.333906	2.474339
7	0.611103	2.486471	0.576006	0.504956	-0.154887	0.097422	0.046327	-0.635140	0.903832
8	0.899401	-1.113904	0.145446	1.396815	-0.740363	-1.053798	0.841444	0.644690	-1.074503
9	0.674084	-1.744020	0.742430	-0.132168	-0.278004	1.240744	-0.331766	-0.555221	0.264174

Dessiner des graphiques et des cartes

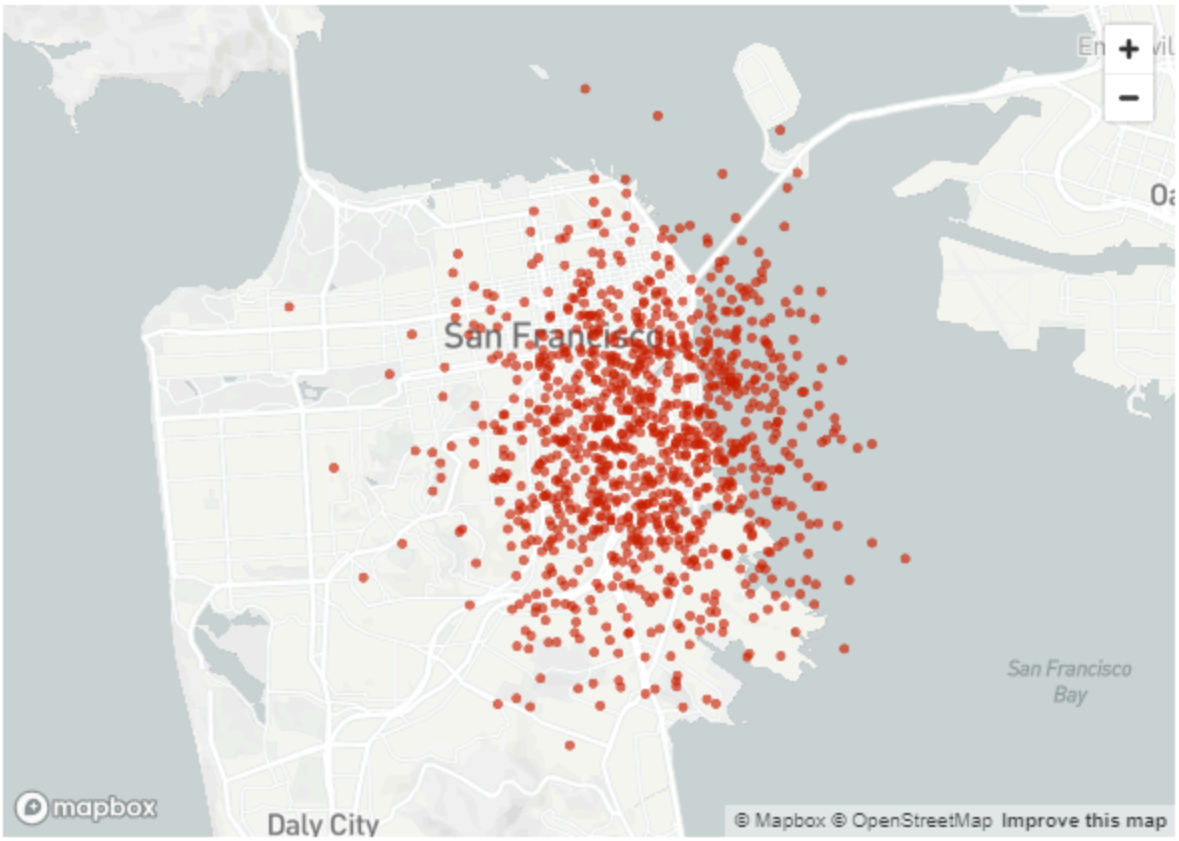
Streamlit prend en charge plusieurs bibliothèques populaires de création de graphiques de données telles que Matplotlib, bokeh, folium etc

Dessiner un graphique linéaire

```
import streamlit as st
import numpy as np
import pandas as pd

chart_data = pd.DataFrame(
     np.random.randn(20, 3),
     columns=['a', 'b', 'c'])

st.line_chart(chart_data)
```

Ajouter une map

```
import streamlit as st
import numpy as np
import pandas as pd

map_data = pd.DataFrame(
    np.random.randn(1000, 2) / [50, 50] + [37.76, -122.4],
    columns=['lat', 'lon'])

st.map(map_data)
```



Afficher une progression

```
import streamlit as st
import time


'Starting a long computation...'
```

```python
# Add a placeholder
latest_iteration = st.empty()
bar = st.progress(0)

for i in range(100):
  # Update the progress bar with each iteration.
  latest_iteration.text(f'Iteration {i+1}')
  bar.progress(i + 1)
  time.sleep(0.1)

'...and now we\'re done!'
```

Caching

Le caching permet à votre application de rester performante même lors du chargement de données depuis le web, de la manipulation de grands ensembles de données, ou de l'exécution de calculs coûteux.

L'idée de base derrière le caching est de stocker les résultats des appels de fonction coûteux et de retourner le résultat mis en cache lorsque les mêmes entrées se produisent à nouveau. Cela évite l'exécution répétée d'une fonction avec les mêmes valeurs d'entrée.

Pour mettre en cache une fonction dans Streamlit, vous devez lui appliquer un décorateur de caching. Vous avez deux choix :

- **st.cache_data** est la manière recommandée de mettre en cache des calculs qui renvoient des données. Utilisez st.cache_data lorsque vous utilisez une fonction qui renvoie un objet de données sérialisable (par exemple, str, int, float, DataFrame, dict, list). Il crée une nouvelle copie des données à chaque appel de fonction, ce qui le rend sûr contre les mutations et les conditions de course. Le comportement de st.cache_data est ce que vous voulez dans la plupart des cas - donc si vous n'êtes pas sûr, commencez avec st.cache_data et voyez si ça fonctionne !
- **st.cache_resource** est la manière recommandée de mettre en cache des ressources globales comme les modèles ML ou les connexions de base de données. Utilisez st.cache_resource lorsque votre fonction renvoie des objets non sérialisables que vous ne voulez pas charger plusieurs fois. Il renvoie l'objet mis en cache lui-même, qui est partagé entre toutes les réexécutions et sessions sans copie ni duplication. Si vous mutez un objet qui est mis en cache en utilisant st.cache_resource, cette mutation existera à travers toutes les réexécutions et sessions.

Streamlit's two caching decorators and their use cases.

```
import pandas as pd
import streamlit as st
```

```
# Cache the dataframe so it's only loaded once
@st.cache_data
def load_data():
    return pd.DataFrame(
        {
            "first column": [1, 2, 3, 4],
            "second column": [10, 20, 30, 40],
        }
    )

# Boolean to resize the dataframe, stored as a session state
variable
st.checkbox("Use container width", value=False,
key="use_container_width")

df = load_data()

# Display the dataframe and allow the user to stretch the
dataframe
# across the full width of the container, based on the checkbox
value
st.dataframe(df,
use_container_width=st.session_state.use_container_width)
```

Connections

Comme indiqué ci-dessus, vous pouvez utiliser @st.cache_resource pour mettre en cache des connexions. Il s'agit de la solution la plus générale qui vous permet d'utiliser presque n'importe quelle connexion à partir de n'importe quelle bibliothèque Python. Cependant, Streamlit offre également un moyen pratique de gérer certaines des connexions les plus populaires, comme SQL ! st.connection se charge du caching pour vous afin que vous puissiez profiter de moins de lignes de code. Récupérer des données depuis votre base de données peut être aussi simple que :

```python
import streamlit as st

conn = st.connection("my_database")
df = conn.query("select * from my_table")

st.dataframe(df)
```

Vous vous demandez peut-être où vont votre nom d'utilisateur et votre mot de passe. Streamlit dispose d'un mécanisme pratique de gestion des secrets. Pour l'instant, voyons simplement comment st.connection fonctionne très bien avec les secrets. Dans votre répertoire de projet local, vous pouvez enregistrer un fichier .streamlit/secrets.toml. Vous enregistrez vos secrets dans le fichier toml et st.connection les utilise simplement ! Par exemple, si vous avez un fichier d'application streamlit_app.py, votre répertoire de projet peut ressembler à ceci :

```
your-LOCAL-repository/
├── .streamlit/
│   └── secrets.toml # Make sure to gitignore this!
└── streamlit_app.py
```

Votre fichier secrets.toml devra ressembler à ça :

```toml
[connections.my_database]
    type="sql"
    dialect="mysql"
    username="xxx"
    password="xxx"
    host="example.com" # IP or URL
    port=3306 # Port number
    database="mydb" # Database name
```