



Recherche d'information

Interface de recherche avec Streamlit

Enzo Doyen

2025 - LGC6KM43 - M2

Principe de Streamlit

Streamlit est un framework Python qui permet de créer simplement et rapidement des applications Web interactives en Python.

Ce framework est particulièrement adapté pour les applications de visualisation de données, de machine learning et de TAL en raison de l'intégration facile de code Python existant.

Principe de Streamlit

Streamlit est un framework Python qui permet de créer simplement et rapidement des applications Web interactives en Python.

Ce framework est particulièrement adapté pour les applications de visualisation de données, de machine learning et de TAL en raison de l'intégration facile de code Python existant.

En outre, le framework est conçu pour éviter (par défaut) l'utilisation de code HTML, CSS ou JavaScript en utilisant des modèles prédéfinis, ce qui simplifie le développement.

Exemple de page Streamlit (stefanrmmr-gpt3-email-generator-streamlit-app-ku3fbq.streamlit.app)

rephrase



Generate professional sounding emails based on your direct comments - powered by Artificial Intelligence (OpenAI GPT-3) Implemented by [stefanrmmr](#) - view project source code on [GitHub](#)

What is your email all about?

SECTION - Email Input

Enter email contents down below! (currently 2x seperate topics supported)

topic 1

topic 2 (optional)

Sender Name

[rephrase]

Recipient Name

[recipient]

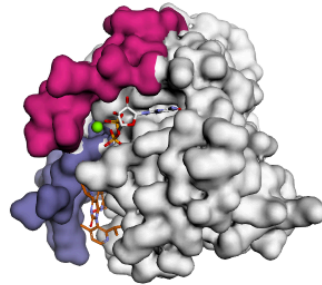
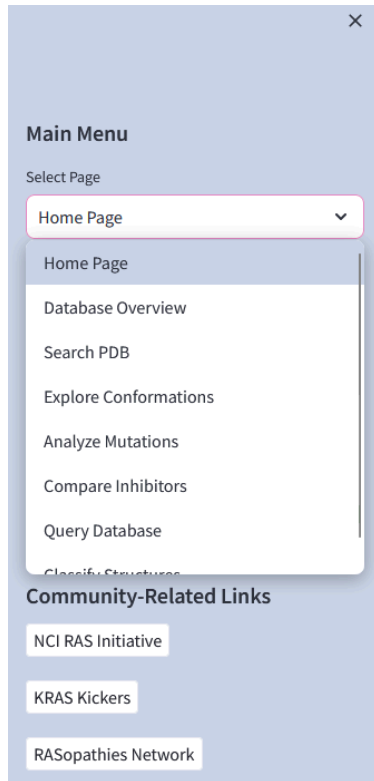
Writing Style

formal

▼

Generate Email

Exemple de page Streamlit (rascore.streamlit.app)



Powered by [Stmol](#) (PDB: [6OIM](#)).

Rascore

A tool for analyzing RAS protein structures

Created by Mitchell Parker and Roland Dunbrack

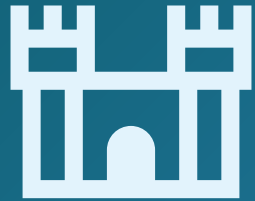
Fox Chase Cancer Center

Summary

Rascore is a tool for analyzing structures of the RAS protein family (KRAS, NRAS, and HRAS). The *Rascore* database presents a continually updated analysis of all available RAS structures in the PDB with their catalytic switch 1 (SW1) and switch 2 (SW2) loops conformationally classified and their molecular contents annotated (e.g., mutation status, nucleotide state, bound protein, inhibitor).

Details of our work are provided in the [Cancer Research](#) paper, [Delineating The RAS Conformational Landscape](#). We hope that researchers will use *Rascore* to gain novel insights into RAS biology.

..



I. Fondements de Streamlit

Exemple basique de page Streamlit

```
1 import streamlit as st
```

 Python

```
2
```

```
3 st.title("Exemple de page Streamlit")
```

```
4
```

```
5 st.markdown(
```

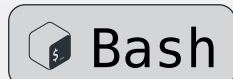
```
6     "Voici à quoi ressemble une page Streamlit, que  
vous devrez créer pour le projet de _Recherche  
d'information_ !"
```

```
7 )
```

Exemple basique de page Streamlit

On peut lancer un serveur local depuis la ligne de commande pour accéder à notre page Streamlit :

```
1 streamlit run app.py
```



Où `app.py` est le nom du fichier Python contenant le code Streamlit.

Exemple basique de page Streamlit

Exemple de page Streamlit

Voici à quoi ressemble une page Streamlit, que vous devrez créer pour le projet de *Recherche d'information* !

Affichage de texte

```
1 st.title("Titre de l'application")
2 st.header("En-tête de section")
3 st.text("Ceci est un texte normal, brut.")
4 st.write("Prise en charge de *Markdown* !")
```



Affichage de données tabulées

Il est possible d'utiliser soit `st.dataframe()` pour afficher les données sous forme de tableau interactif, soit `st.table()` pour afficher les données sous forme de tableau statique.

`st.dataframe()` offre plusieurs options de personnalisation, telles que le tri, le filtrage, le téléchargement des données ou encore le redimensionnement des colonnes.

```
1 st.dataframe({"A": [1, 2], "B": [3, 4]})  
2 st.table({"A": [1, 2], "B": [3, 4]})
```



Titre de l'application

En-tête de section

Ceci est un texte normal, brut.

Show/hide columns

Prise en charge de *Markdown* !



A	B
1	3
2	4

Saisie et variables

```
1 name = st.text_input("Nom")
2 age = st.number_input("Âge", min_value=0,
max_value=120, step=5)
3 color = st.selectbox("Choisissez une couleur",
["Rouge", "Vert", "Bleu"])
4
5 st.write(f"Bonjour {name} ! Tu as {age} ans et tu as
choisi la couleur {color}.")
```



Saisie et variables : conditions

```
1 name = st.text_input("Nom")
2 age = st.number_input("Âge", min_value=0,
max_value=120, step=5)
3 color = st.selectbox("Choisissez une couleur",
["Rouge", "Vert", "Bleu"])
4 agree = st.checkbox("Faire apparaître mes infos")
5 if agree:
6     st.write(f"Bonjour {name} ! Tu as {age} ans et tu
as choisi la couleur {color}.")
```



Agencement de la page : colonnes

```
1 col1, col2 = st.columns(2)
2 col1.write("Texte dans la colonne de gauche")
3 col2.write("Texte dans la colonne de droite")
```



Agencement de la page : barre latérale

```
1 with st.sidebar:
2     st.header("Paramètres")
3     st.selectbox("Choisissez une option", ["A", "B", "C"])
```



Agencement de la page : barre latérale

```
1 with st.sidebar:
2     st.header("Paramètres")
3     option = st.selectbox("Choisissez une option",
4                             ["A", "B", "C"])
5 st.write(f"Option sélectionnée : {option}")
```



Python

Agencement de la page : barre latérale et pages distinctes

```
1 page = st.sidebar.selectbox("Choisissez une  
page", ["Vue d'ensemble", "Analyse",  
"Paramètres"])  
2 if page == "Vue d'ensemble":  
3     st.write("Bienvenue sur la page de vue  
d'ensemble !")  
4 elif page == "Analyse":  
5     st.write("Page d'analyse des données.")  
6 elif page == "Paramètres":  
7     st.write("Page des paramètres.")
```



Agencement de la page : accordéon

```
1 with st.expander("Voir les détails"):  
2     st.write("Informations supplémentaires à  
   l'intérieur d'un accordéon.")
```



Agencement de la page : conteneurs

```
1 with st.container(width=40, border=True):  
2     st.write("Élément à l'intérieur d'un conteneur de  
   largeur 40 avec bordure")  
3     col1, col2 = st.columns([1, 2])  
4     with col1:  
5         st.write("Colonne 1")  
6     with col2:  
7         st.write("Colonne 2")
```



Intégration de code HTML

Utilisation du paramètre `unsafe_allow_html=True` pour autoriser l'utilisation de contenu HTML :

```
1 st.markdown( """  
2     <div style="color: red; font-size: 24px;">  
3         Texte rouge, de taille 24 pixels.  
4     </div>  
5 """ , unsafe_allow_html=True)
```



Intégration de code HTML

```
1  st.markdown("""
2      <style>
3          .class {
4              color: red;
5              font-size: 24px;
6          }
7      </style>
8      <div class="class">
9          Texte rouge, de taille 24 pixels.
10     </div>
11 """, unsafe_allow_html=True)
```

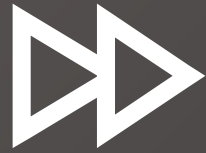


Intégration de code CSS

En utilisant un fichier CSS externe (p. ex., avec le nom `styles.css`) :

```
st.markdown('<style>' +  
1 open('styles.css').read() + '</style>',  
unsafe_allow_html=True)
```





II. Considérations avancées

États

Par défaut, Streamlit **exécute le code de l'application à chaque interaction** (clic de bouton, texte modifié...).

Ainsi, quand une variable est définie, sa valeur est réinitialisée à chaque interaction.

```
1 count = 0
2 increment = st.button("Augmenter")
3 if increment:
4     count += 1 # count reste à 1
```



États

Par défaut, Streamlit **exécute le code de l'application à chaque interaction** (clic de bouton, texte modifié...).

Ainsi, quand une variable est définie, sa valeur est réinitialisée à chaque interaction.

Pour conserver l'état d'une variable, on peut utiliser des **états**, avec `st.session_state`.

États

`st.session_state` agit de manière très similaire aux dictionnaires Python, avec des clés et des valeurs, qu'il faut initialiser.

```
1 if "count" not in st.session_state:
2     st.session_state.count = 0
3
4 increment = st.button("Augmenter")
5 if increment:
6     st.session_state.count += 1
7
8 st.write("Nombre = ", st.session_state.count)
```



États

La valeur d'une clé est accessible soit via `st.session_state.key`, soit via `st.session_state["key"]`.

```
1 a = st.session_state["count"]  
2 b = st.session_state.count
```



États

On peut obtenir le même résultat en utilisant des fonctions de rappel (*callbacks*), où une fonction est utilisée comme argument pour certains éléments spécifiques, par exemple `on_click` pour l'élément `button` :

```
1 def increment_counter():  
2     st.session_state.count += 1  
3  
4 increment = st.button("Augmenter",  
    on_click=increment_counter)
```



Mise en cache

Comme vu précédemment, Streamlit **exécute le code de l'application à chaque interaction** (clic de bouton, texte modifié...).

Pour certaines opérations coûteuses (comme le chargement de données ou l'entraînement de modèles), il est possible d'utiliser la mise en cache pour éviter de réexécuter le code à chaque fois quand la valeur attendue ne change pas.

Mise en cache

La mise en cache se fait par l'utilisation de **décorateurs** [?] dans les fonctions Python concernées. Il en existe deux types : `@st.cache_data` et `@st.cache_resource`.

`@st.cache_data` est privilégié pour les données pures fixes (nombres, listes, DataFrames...), par exemple les données textuelles. Le décorateur crée une copie de l'objet.

`@st.cache_resource` est utilisé pour les ressources qui doivent être créées une seule fois, comme les modèles de machine learning ou les objets lourds. Le décorateur sauvegarde l'objet original en mémoire ainsi que son état.

Mise en cache : exemples

```
1 @st.cache_data
2 def load_data(path):
3     return pd.read_csv(path)
```

 Python

```
1 @st.cache_resource
2 def load_vectorizer_and_matrix(docs):
3     vectorizer = TfidfVectorizer()
4     tfidf_matrix = vectorizer.fit_transform(docs)
5     return vectorizer, tfidf_matrix
```

 Python

Ressources complémentaires

La documentation de Streamlit est très complète et propose de nombreux exemples ; n'hésitez pas à la consulter : <https://docs.streamlit.io/>

- ♦ Documentation sur les états : <https://docs.streamlit.io/develop/concepts/architecture/session-state>
- ♦ Documentation sur la mise en cache : <https://docs.streamlit.io/develop/concepts/architecture/caching>
- ♦ Exemples de projets Streamlit : <https://github.com/jrieke/best-of-streamlit>

Bibliographie

Manning, C. D., Raghavan, P., et Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Mitra, B. (2018). *An Introduction to Neural Information Retrieval* (Numéro v.41). Now Publishers.

Zhai, C., et Massung, S. (juin 2016). *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. Association for Computing Machinery and Morgan & Claypool. [10.1145/2915031](#)