



POLYTECHNIQUE
MONTRÉAL

LE GÉNIE
EN PREMIÈRE CLASSE

Guide TP2

Version Python

Technologies

Pour ces labs, nous utilisons:

- HTML / CSS
- Plotly + Dash

Souvent, les TP peuvent être réalisés de plusieurs façons différentes - nous attendons de vous que vous choisissiez l'option qui utilise ces technologies !

Technologies

HTML

“Hypertext Markup Language” est utilisé pour structurer du contenu web.

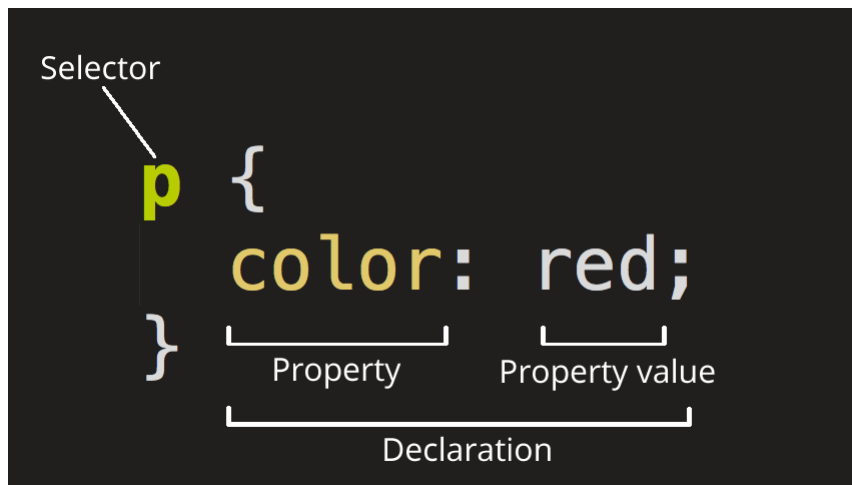
```
<!DOCTYPE html>
<html>
  <head>
    <title>Mon titre</title>
  </head>
  <body>
    <h1>Introduction</h1>
    <p>Ceci est un paragraphe intéressant..</p>
  </body>
</html>
```

- <...> ouvrant, </...> fermant
- DOM - Document Object Model (structure hiérarchique du HTML)
 - <...> </...> est un élément
 - Relations:
 - Enfants (h1 et p)
 - Parents (body)

Technologies

CSS

- Décrit la présentation visuelle d'une page HTML
- Utilise des **sélecteurs** (« *selectors* ») et **déclarations** (« *declarations* »)



Technologies

CSS : Sélecteurs et déclarations

- Exemples de sélecteurs

- Par type d'élément

```
p {  
  font-weight: bold;  
}
```

Tous les éléments <p> seront en gras (« bold »)

- Sur un élément : **par id**

```
#my-pretty-text {  
  font-family: 'Times New Roman';  
}
```

La police de l'élément "my-pretty-text" sera "Times New Roman"

```
<p id="my-pretty-text">Hello!</p>
```

Élément HTML correspondant

- Sur plusieurs éléments : **par classe**

```
.another-text {  
  font-size: 12px;  
}
```

La police des éléments avec la classe "another-text" sera de 12px

```
<p class="another-text">Goodbye!</p>
```

Élément HTML correspondant

Technologies

CSS : Sélecteurs et déclarations

- Déclarations

- Déterminent l'apparence visuelle du ou des éléments sélectionnés
- Exemples...

- **width, height** : La hauteur et la largeur du ou des éléments
- **fill** : couleur du ou des éléments
- **margin** : marge autour du ou des éléments

Technologies

Plotly et Dash

- Plotly est une librairie Python que permet de créer plusieurs types de graphiques.
- Dash est un cadre d'application web qui fournit une abstraction Python pure autour de HTML, CSS et JavaScript

Technologies

HTML

- Au lieu d'écrire du HTML ou d'utiliser un template HTML, vous composez votre mise en page en utilisant des structures Python avec la bibliothèque `dash-html-components`

```
import dash_html_components as html

html.Div([
    html.H1('Hello Dash'),
    html.Div([
        html.P('Dash converts Python classes into HTML'),
        html.P("This conversion happens behind the scenes by Dash's JavaScript front-end")
    ])
])
```

which gets converted (behind the scenes) into the following HTML in your web-app:

```
<div>
  <h1>Hello Dash</h1>
  <div>
    <p>Dash converts Python classes into HTML</p>
    <p>This conversion happens behind the scenes by Dash's JavaScript front-end</p>
  </div>
</div>
```

Vous pouvez trouver tous les composants HTML sur le site web ci-dessous

[Dash HTML Components](#) | [Dash for Python Documentation](#) | [Plotly](#)

Callbacks

Comment rendre votre graphique interactif

- Les fonctions Callback: Fonctions Python qui sont automatiquement appelées par Dash chaque fois que la propriété d'un composant d'entrée change.

▼ Dash Callbacks

[Basic Callbacks](#)

[Advanced Callbacks](#)

[Clientside Callbacks](#)

[Pattern-Matching Callbacks](#)

[Callback Gotchas](#)

Callbacks

Simple Interactive Dash App

- Vous aurez besoin d'un ID de composant et d'une **component property** pour informer l'application callback les actions sur votre *Input* et *Output*

```
import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output

external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash(__name__, external_stylesheets=external_stylesheets)

app.layout = html.Div([
    html.H6("Change the value in the text box to see callbacks in action!"),
    html.Div(["Input: ",
              dcc.Input(id='my-input', value='initial value', type='text')]),
    html.Br(),
    html.Div(id='my-output'),
])

@app.callback(
    Output(component_id='my-output', component_property='children'),
    Input(component_id='my-input', component_property='value')
)
def update_output_div(input_value):
    return 'Output: {}'.format(input_value)

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

Callbacks

Simple Interactive Dash App

Change the value in the text box to see callbacks in action!

Input:

Output: 123812u38

Lien pour le exemple: <https://dash.plotly.com/basic-callbacks>

Callbacks

Simple Interactive Dash App

- Dans Dash, les entrées et les sorties de notre application sont simplement les propriétés d'un composant particulier. Dans cet exemple, notre entrée est la propriété "value" du composant dont l'ID est "my-input". Notre sortie est la propriété "children" du composant avec l'ID "my-output".
- Les mots-clés `component_id` et `component_property` sont facultatifs (il n'y a que deux arguments pour chacun de ces objets). Ils sont inclus dans cet exemple pour des raisons de clarté mais seront omis dans le reste de la documentation pour des raisons de brièveté et de lisibilité.
- Remarquez que nous ne définissons pas de valeur pour la propriété `children` du composant `my-output` dans le layout. Lorsque l'application Dash démarre, elle appelle automatiquement tous les callbacks avec les valeurs initiales des composants d'entrée afin de remplir l'état initial des composants de sortie. Dans cet exemple, si vous aviez spécifié quelque chose comme `html.Div(id='my-output', children='Hello world')`, cette valeur serait écrasée au démarrage de l'application.

Callbacks

Dash app with State

- Dans certains cas, vous pouvez avoir un modèle de type "form"-type dans votre application. Dans ce cas, vous pouvez souhaiter lire la valeur du composant de saisie, mais uniquement lorsque l'utilisateur a fini de saisir toutes ses informations dans le formulaire.
- **State** vous permet de transmettre des valeurs supplémentaires sans déclencher les callbacks

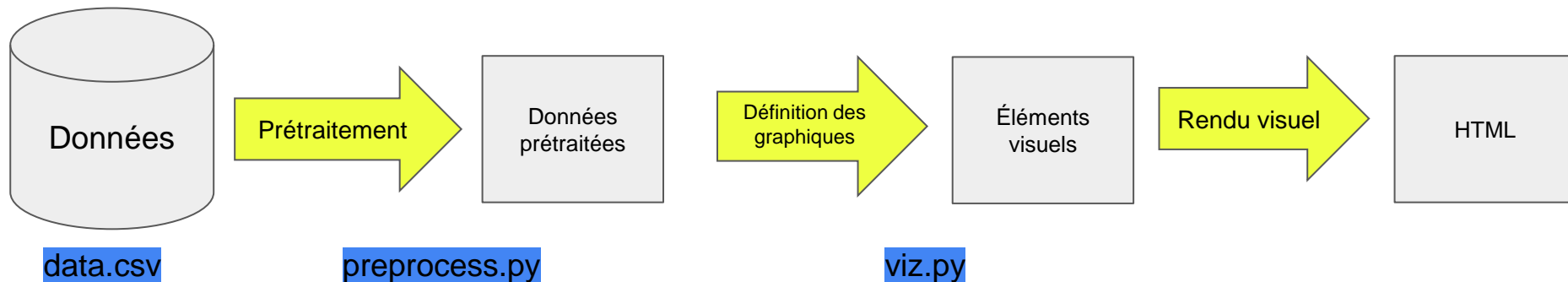
Callbacks

Simple Interactive Dash App

- Vérifiez pour plus d'informations :
 - <https://dash.plotly.com/basic-callbacks>
 - <https://www.youtube.com/watch?v=mTsZL-VmRVE>
 - <https://medium.com/@benshentist/dash-callbacks-where-the-magic-happens-ab19260dbc7e>

Plotly

Étapes typiques dans les TPs



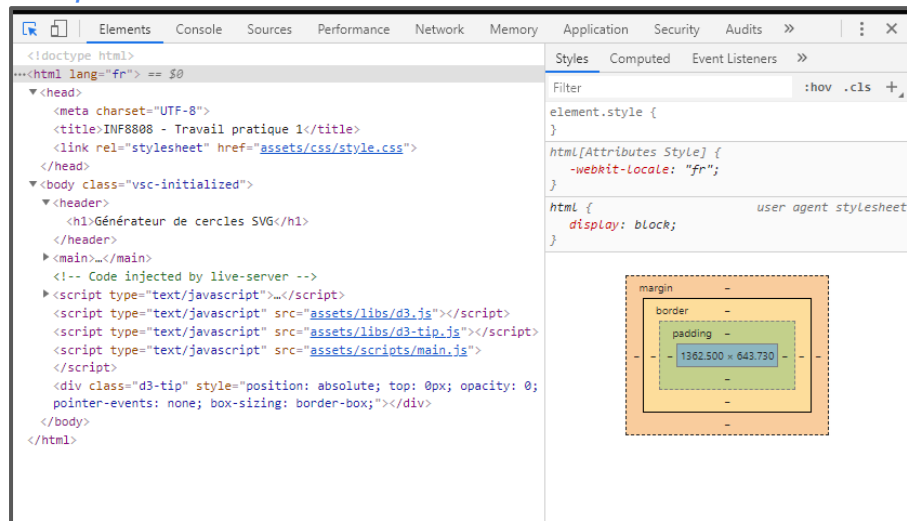
Conseils généraux de débogage

Débogage avec Chrome

Notez que d'autres fureteurs offrent des outils semblables.

1. Cliquez avec le bouton droit n'importe où sur la page et sélectionnez «Inspect» OU Ctrl + Maj + I
2. L'inspecteur s'ouvre, ce qui est utile pour le débogage
3. Les onglets «Éléments», «Console» et «Sources» seront les plus utiles pour ces TP (voir les diapositives suivantes)

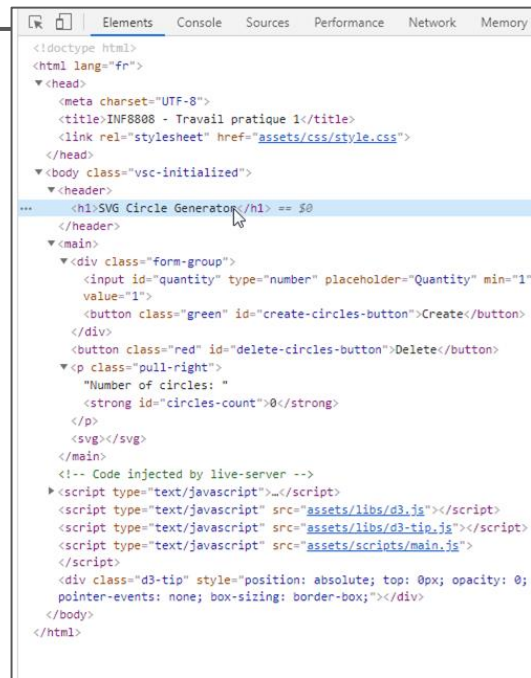
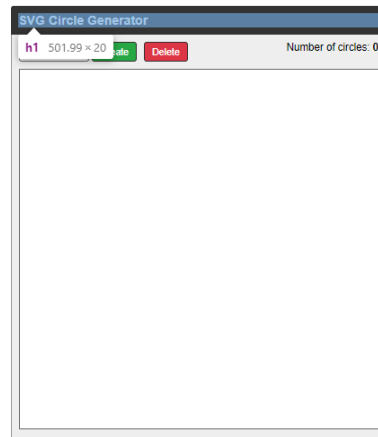
Inspecteur Chrome



Débogage avec Chrome

Inspection d'élément

- Affiche la structure HTML (DOM) de votre code
- Met en évidence l'élément actuellement survolé sur la page
- Utilisez-le pour vérifier que votre code Python génère correctement des éléments HTML
- Vous permet de tester directement des modifications au HTML et CSS



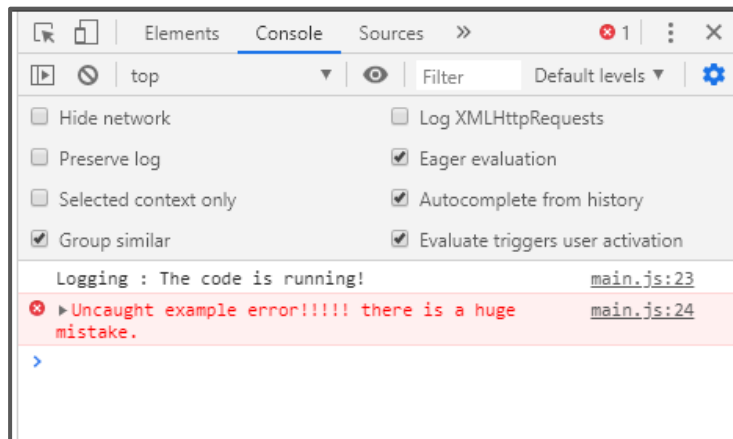
Inspection d'un élément h1

Débogage avec Chrome

Console

- Dans la console, vous verrez les sorties de votre code
- Celles-ci peuvent inclure des messages d'erreur et des « logs »
- Si quelque chose ne fonctionne pas, c'est le premier endroit où vous devriez regarder

```
function update() {  
  console.log("Logging : The code is running!")  
  throw "example error!!!! there is a huge mistake."  
}
```

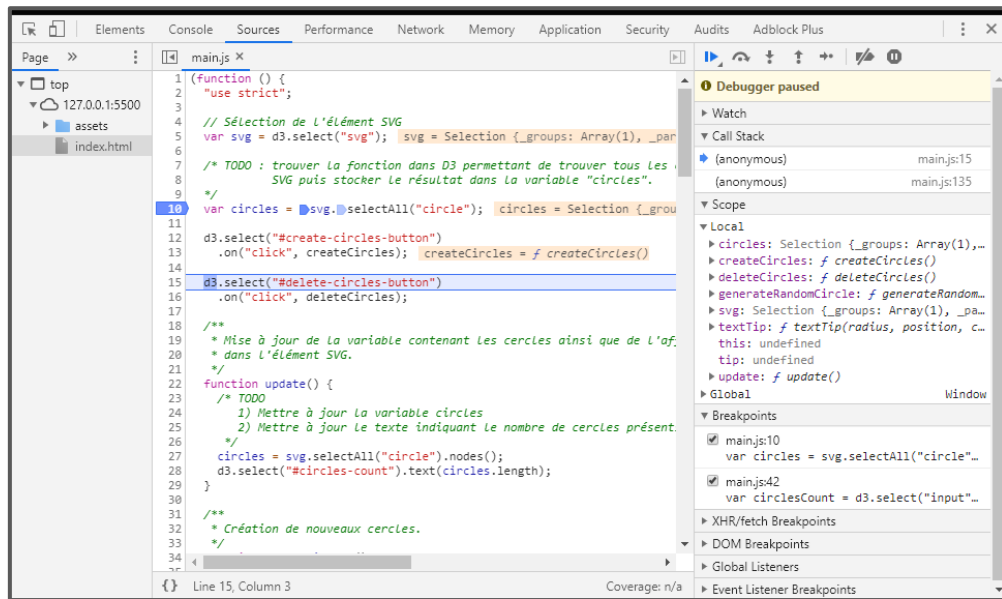


*Des messages d'erreur et de « logs »
apparaissent dans la console*

Débogage avec Chrome

Sources inspection tool

- Dans cet onglet, vous pouvez voir votre code source et tester sa modification
- Vous pouvez également ajouter des points d'arrêt, où l'exécution s'arrêtera
- À partir d'un point d'arrêt, vous pouvez voir la valeur de chaque variable et parcourir le code ligne par ligne



Parcours de code D3

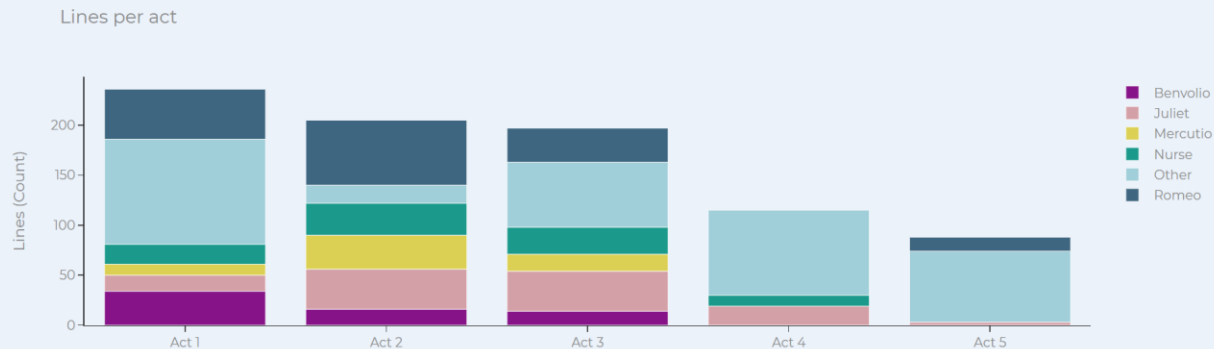
TP2

Introduction au TP2

Dans le TP2, vous allez créer un diagramme à bandes (« *bar chart* ») à partir de données tirées de la pièce de théâtre *Roméo et Juliette* de Shakespeare.

Who's Speaking?

An analysis of Shakespeare's Romeo and Juliet



Use the radio buttons to change the display.

The current mode is : **Count**

☒ Count ☐ Percent

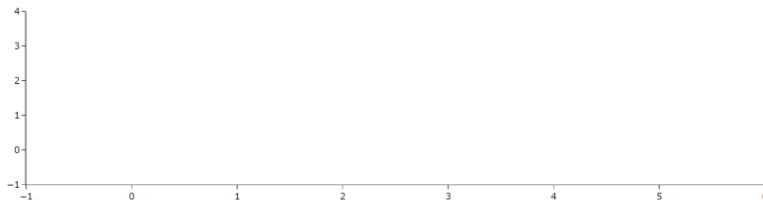
Rouler le code

- Dans un terminal, au même niveau que *app.py*:
 - `python -m virtualenv -p python3.8 venv`
 - `venv\Scripts\activate`
 - `python -m pip install -r requirements.txt`
 - `python server.py`
- Puis, consultez localhost:8050 dans votre navigateur

Résultat :

Who's Speaking?

An analysis of Shakespeare's Romeo and Juliet



Use the radio buttons to change the display.

The current mode is :

☒ Count ☐ Percent

Ensemble de données

Les données représentent le texte de la pièce de théâtre. Elles se trouvent dans le fichier `./src/assets/data/romeo_and_juliet.csv`

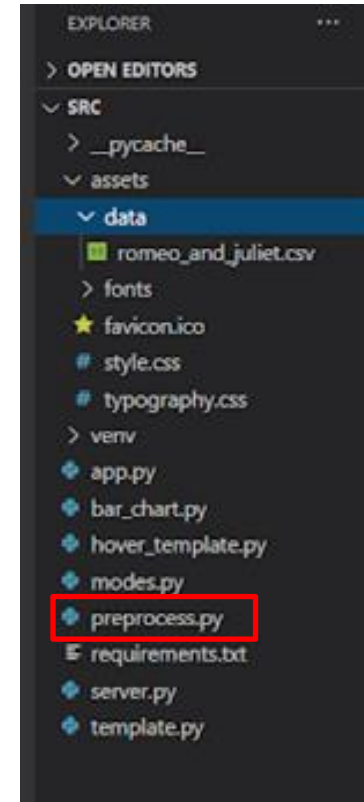
- **Act:** Cette colonne représente l'acte dans lequel la ligne est prononcée.
- **Scene:** Cette colonne représente la scène dans laquelle la ligne est prononcée.
- **Line:** Pour une n -ième ligne dans une scène donnée, cette colonne contient la valeur n .
- **Player:** Cette colonne contient le nom du joueur qui a prononcé la ligne.
- **PlayerLine:** Cette colonne contient le contenu prononcé par le joueur pour cette ligne.

```
Act,Scene,Line,Player,PlayerLine
1,0,1,RICHMOND,"Two households, both alike in dignity, / In fair
1,1,1,SAMPSON,"Gregory, o' my word, we'll not carry coals."
1,1,2,GREGORY,"No, for then we should be colliers."
1,1,3,SAMPSON,"I mean, an we be in choler, we'll draw."
1,1,4,GREGORY,"Ay, while you live, draw your neck out o' the coll
1,1,5,SAMPSON,"I strike quickly, being moved."
1,1,6,GREGORY,But thou art not quickly moved to strike.
1,1,7,SAMPSON,A dog of the house of Montague moves me.
1,1,8,GREGORY,"To move is to stir, and to be valiant is to stand:
```

Tâches à accomplir :

1. Prétraitement des données

- Fichier : `./src/preprocess.py`



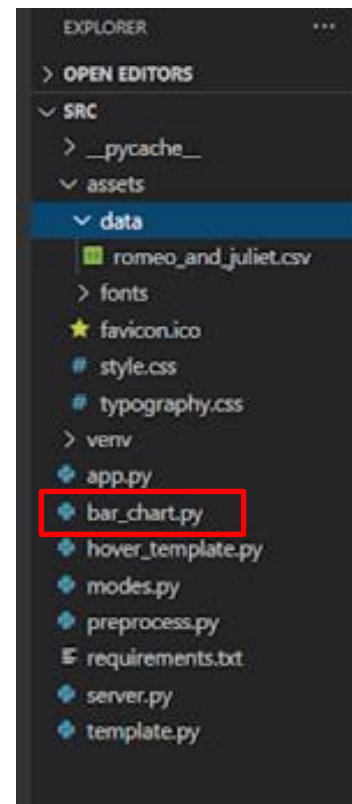
Tâches à accomplir :

1. Prétraitement des données

- Fichier : `./src/preprocess.py`

2. Création du diagramme à bandes

- Fichier : `./src/bar_chart.py`



Tâches à accomplir :

1. Prétraitement des données

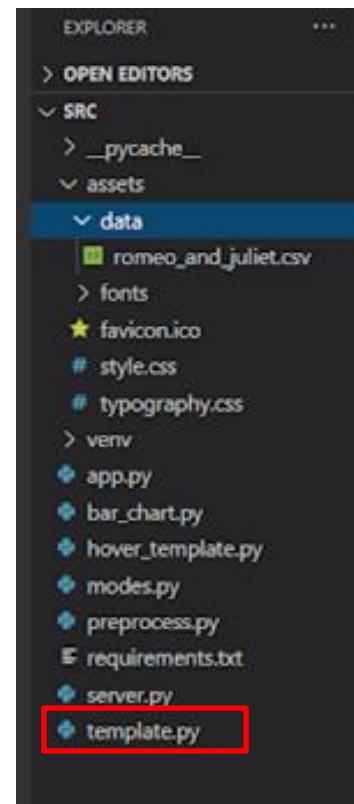
- Fichier : `./src/preprocess.py`

2. Création du diagramme à bandes

- Fichier : `./src/bar_chart.py`

3. Completion de un template

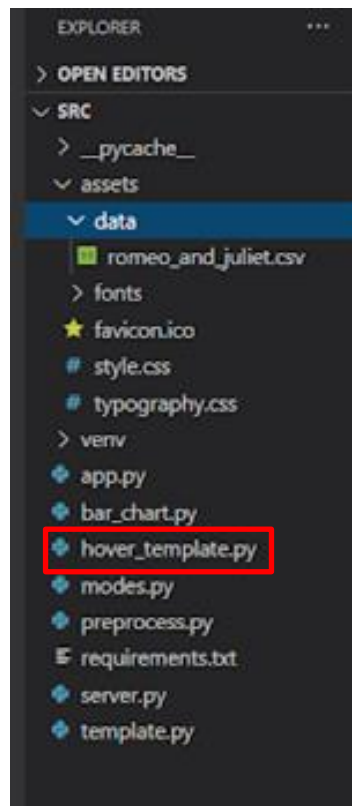
- Fichier : `./src/template.py`



Tâches à accomplir :

1. **Prétraitement des données**
 - Fichier : `./src/preprocess.py`
2. **Création du diagramme à bandes**
 - Fichier : `./src/bar_chart.py`
3. **Completion de un template**
 - Fichier : `./src/template.py`
4. **Ajout d'une info-bulle**
 - Fichier : `./src/hover_template.py`

Les autres fichiers ne doivent pas être modifiés.



1. Prétraitement des données

3 fonctions dans le fichier **preprocess.py** à remplir pour prétraiter les données :

1. `summarize_lines`
2. `replace_others`
3. `clean_names`

Le résultat sera ce type de structure de données :

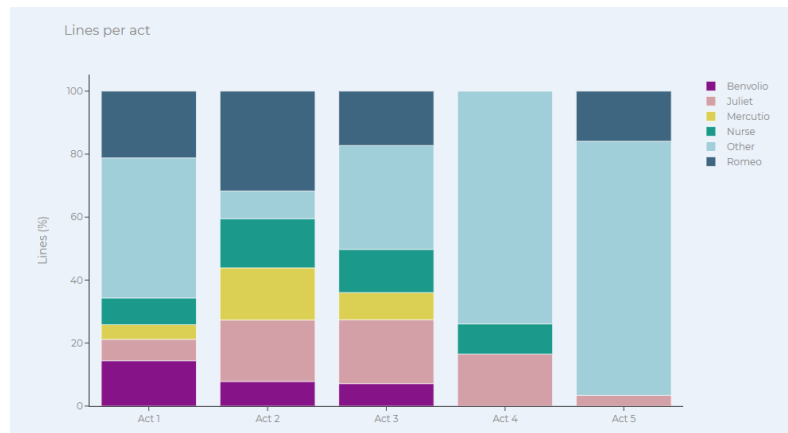
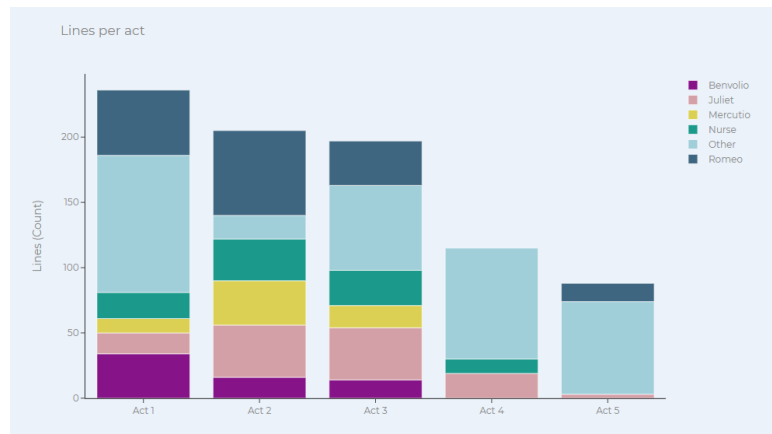
| Act | Player | LineCount | LinePercent |
|-----|----------|-----------|-------------|
| 1 | Benvolio | 34 | 14.406780 |
| 1 | Juliet | 16 | 6.779661 |
| 1 | Mercutio | 11 | 4.661017 |
| 1 | Nurse | 20 | 8.474576 |
| 1 | Other | 105 | 44.491525 |
| 1 | Romeo | 50 | 21.186441 |

2. Diagramme à bandes

3 fonctions dans `bar_chart.py` pour tracer les bandes :

1. `init_figure`
2. `draw`
3. `update_y_axis`

Le résultat de cette partie est le suivant à droite :



3. Template

Remplissez le code dans **template.py**.

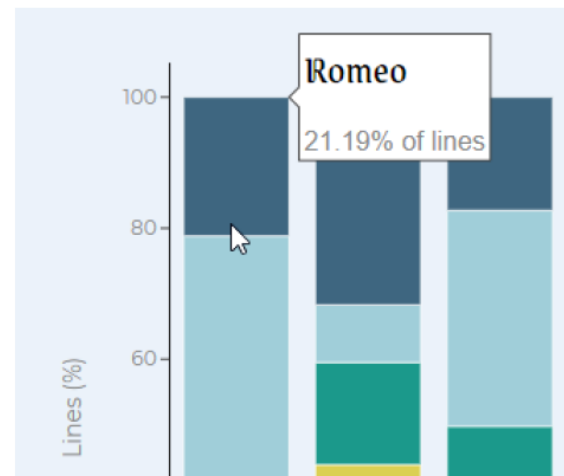
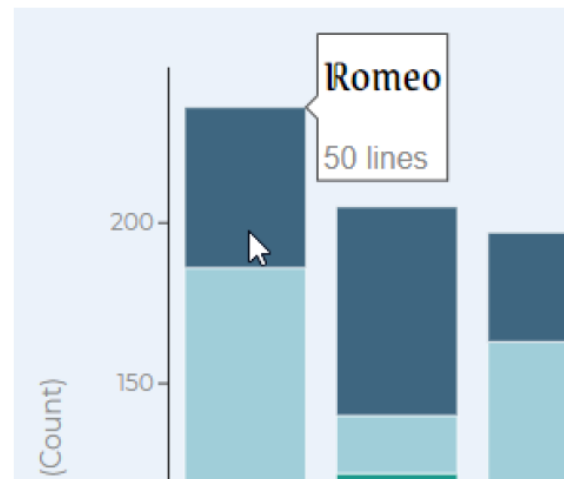
Portez une attention particulières aux classes CSS déjà fournis pour vous aider.

4. Info-bulle

Remplissez le code dans `hover_template.py` pour tracer l'info-bulle.

Conseil : Portez une attention particulière à l'apparence visuelle de la bulle.

Le résultat est tel qu'ici à droite :



Quelques conseils pour démarrer

Prétraitement des données

- Évitez de manipuler directement les indices de données lorsque cela est possible (c'est-à-dire un loop for index $i \rightarrow \text{data}[i]$)
 - Cette pratique améliorera la qualité et la maintenabilité du code tout en réduisant le risque d'erreurs
- Au lieu de cela, explorez les méthodes DataFrame de Pandas, telles que:
 - `.groupby()`
 - `.concat()`
 - `.replace()`
 - `.sort_values()`
 - `.sum()`

Qualité globale et clarté de la soumission

Plus de détails

Chaque TP sera aussi noté sur la **qualité globale et clarté de la soumission**

Quelques exemples de points à surveiller:

- Essayez de structurer le code clairement
- Ne modifiez pas les signatures des fonctions existantes
- Vous pouvez ajouter de nouvelles fonctions, mais cela ne devrait pas être nécessaire
- Alignez votre code de manière cohérente
- Ajoutez des commentaires pertinents au besoin, sans laisser de commentaires inutiles
- Ne laissez pas 'print' inutile
- Suivez attentivement les directives de soumission

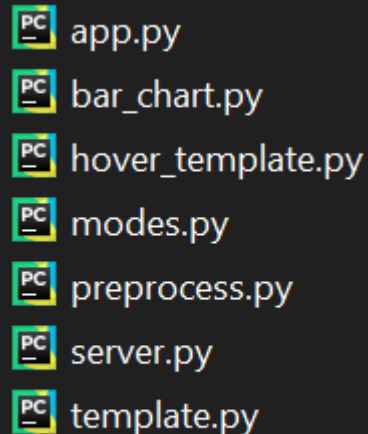
Etc.

Tous ces points peuvent faire perdre des points à votre note finale.

Submission

Plus de détails pour la soumission

- Vous allez soumettre un fichier .zip (pas .rar) avec les fichiers python affichés sur le côté
- La soumission se fera sur Moodle dans la boîte appropriée pour Python.



```
PC app.py
PC bar_chart.py
PC hover_template.py
PC modes.py
PC preprocess.py
PC server.py
PC template.py
```