**Importation des bibliothèques**

python

Copier le code

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from tkinter import Tk, Label, Button, Listbox, Scrollbar, StringVar, END

from tkinter import ttk # Importer ttk pour la combobox

* **pandas** : Bibliothèque utilisée pour la manipulation et l'analyse des données, surtout avec des DataFrames.
* **matplotlib.pyplot** : Module de Matplotlib pour créer des visualisations graphiques (comme des graphiques en ligne, des barres, etc.).
* **seaborn** : Basé sur Matplotlib, il simplifie la création de graphiques statistiques plus élégants et informatifs.
* **tkinter** : Bibliothèque pour créer des interfaces graphiques en Python.
* **ttk** : Module de tkinter qui offre des widgets plus modernes et esthétiques, comme les combobox.

**Chargement des données**

python

Copier le code

df = pd.read\_csv('crypto\_data\_cleaned.csv') # Assurez-vous d'avoir le bon chemin

df['date'] = pd.to\_datetime(df['date'])

* **pd.read\_csv** : Charge le fichier CSV contenant les données des cryptomonnaies dans un DataFrame df.
* **pd.to\_datetime** : Convertit la colonne 'date' du DataFrame en type datetime pour faciliter les opérations de filtrage et de comparaison des dates.

**Fonction pour afficher les graphiques**

python

Copier le code

def afficher\_graphique():

selected\_date = date\_var.get()

selected\_cryptos = crypto\_listbox.curselection()

if not selected\_cryptos:

print("Veuillez sélectionner au moins une cryptomonnaie.")

return

* **def afficher\_graphique()** : Déclare une fonction pour afficher un graphique.
* **selected\_date = date\_var.get()** : Récupère la date sélectionnée dans la combobox.
* **selected\_cryptos = crypto\_listbox.curselection()** : Récupère les indices des cryptomonnaies sélectionnées dans la liste.
* **if not selected\_cryptos** : Vérifie si au moins une cryptomonnaie a été sélectionnée. Si non, un message est affiché et la fonction retourne sans rien faire.

**Filtrage des données**

python

Copier le code

date\_selectionnee = pd.to\_datetime(selected\_date)

# Filtrer les données pour la date sélectionnée et les cryptomonnaies sélectionnées

df\_selectionne = df[df['date'] == date\_selectionnee]

cryptos\_selectionnees = [crypto\_listbox.get(i) for i in selected\_cryptos]

df\_selectionne = df\_selectionne[df\_selectionne['Name'].isin(cryptos\_selectionnees)]

* **date\_selectionnee = pd.to\_datetime(selected\_date)** : Convertit la date sélectionnée en format datetime.
* **df\_selectionne = df[df['date'] == date\_selectionnee]** : Filtre le DataFrame pour ne garder que les lignes correspondant à la date sélectionnée.
* **cryptos\_selectionnees = [crypto\_listbox.get(i) for i in selected\_cryptos]** : Crée une liste des noms des cryptomonnaies sélectionnées.
* **df\_selectionne = df\_selectionne[df\_selectionne['Name'].isin(cryptos\_selectionnees)]** : Filtre encore les données pour ne garder que celles des cryptomonnaies sélectionnées.

**Vérification et visualisation des données**

python

Copier le code

# Vérification du contenu du DataFrame filtré

print("Données filtrées :", df\_selectionne)

if df\_selectionne.empty:

print("Aucune donnée trouvée pour cette date et les cryptomonnaies sélectionnées.")

return

* **print("Données filtrées :", df\_selectionne)** : Affiche les données filtrées dans la console pour le débogage.
* **if df\_selectionne.empty** : Vérifie si le DataFrame filtré est vide. Si c'est le cas, un message d'erreur est affiché et la fonction retourne.

**Visualisation avec un graphique**

python

Copier le code

# Visualisation en barplot pour des valeurs individuelles par cryptomonnaie

plt.figure(figsize=(12, 6))

sns.lineplot(data=df\_selectionne, x='Name', y='price', marker='o', palette='tab20')

plt.title(f"Prix standardisé des cryptomonnaies à la date : {date\_selectionnee}")

plt.xlabel("Cryptomonnaie")

plt.ylabel("Prix standardisé")

plt.xticks(rotation=90)

plt.show()

* **plt.figure(figsize=(12, 6))** : Crée une nouvelle figure de taille 12x6 pouces pour le graphique.
* **sns.lineplot(...)** : Crée un graphique en lignes avec les données filtrées. Les axes x et y représentent respectivement les noms des cryptomonnaies et leur prix.
* **plt.title(...)** : Définit le titre du graphique.
* **plt.xlabel(...)** et **plt.ylabel(...)** : Nomme les axes x et y.
* **plt.xticks(rotation=90)** : Fait pivoter les étiquettes des axes x de 90 degrés pour une meilleure lisibilité.
* **plt.show()** : Affiche le graphique.

**Création de la fenêtre principale**

python

Copier le code

def main\_window():

global date\_var

root = Tk()

root.title("Analyse des Cryptomonnaies")

* **def main\_window()** : Déclare une fonction pour créer la fenêtre principale de l'application.
* **global date\_var** : Déclare date\_var comme une variable globale pour pouvoir l'utiliser dans d'autres fonctions.
* **root = Tk()** : Crée une instance de la fenêtre principale de l'application.
* **root.title(...)** : Définit le titre de la fenêtre.

**Composants de l'interface graphique**

python

Copier le code

# Variable pour la date

date\_var = StringVar()

# Label pour la date

Label(root, text="Choisissez une date :").pack()

# Création d'une combobox pour sélectionner une date

date\_combobox = ttk.Combobox(root, textvariable=date\_var)

date\_combobox.pack()

# Remplir la combobox avec les dates uniques

date\_combobox['values'] = df['date'].dt.date.unique().tolist() # Obtenir les dates uniques

date\_combobox.current(0) # Sélectionner la première date par défaut

* **date\_var = StringVar()** : Crée une variable qui peut être utilisée pour stocker la date sélectionnée.
* **Label(...)** : Crée un label affichant le texte "Choisissez une date".
* **ttk.Combobox(...)** : Crée une combobox pour permettre à l'utilisateur de sélectionner une date.
* **date\_combobox['values']** : Remplit la combobox avec les dates uniques extraites du DataFrame.
* **date\_combobox.current(0)** : Sélectionne la première date par défaut dans la combobox.

**Liste des cryptomonnaies**

python

Copier le code

# Liste des cryptomonnaies

Label(root, text="Sélectionnez des cryptomonnaies :").pack()

# Création d'une fenêtre secondaire pour la liste

global crypto\_listbox

crypto\_listbox = Listbox(root, selectmode="multiple", width=50)

scrollbar = Scrollbar(root)

scrollbar.pack(side="right", fill="y")

crypto\_listbox.pack()

scrollbar.config(command=crypto\_listbox.yview)

crypto\_listbox.config(yscrollcommand=scrollbar.set)

# Remplir la liste avec les noms de cryptomonnaies

for crypto in df['Name'].unique():

crypto\_listbox.insert(END, crypto)

* **Label(...)** : Crée un label indiquant à l'utilisateur de sélectionner des cryptomonnaies.
* **crypto\_listbox = Listbox(...)** : Crée une liste qui permet de sélectionner plusieurs cryptomonnaies.
* **Scrollbar(...)** : Crée une barre de défilement pour la liste.
* **scrollbar.pack(...)** et **crypto\_listbox.pack()** : Affiche la barre de défilement et la liste dans la fenêtre.
* **scrollbar.config(...)** et **crypto\_listbox.config(...)** : Configure la barre de défilement pour qu'elle défile la liste.
* **for crypto in df['Name'].unique()** : Remplit la liste avec les noms uniques des cryptomonnaies présentes dans le DataFrame.

**Bouton pour afficher le graphique**

python

Copier le code

# Bouton pour afficher le graphique

Button(root, text="Afficher le graphique", command=afficher\_graphique).pack()

* **Button(...)** : Crée un bouton qui, lorsqu'il est cliqué, appelle la fonction afficher\_graphique().

**Boucle principale de l'application**

python

Copier le code

root.mainloop()

* **root.mainloop()** : Lance la boucle principale de l'application, permettant à l'interface graphique de rester ouverte et de répondre aux actions de l'utilisateur.

**Exécution de l'application**

python

Copier le code

if \_\_name\_\_ == "\_\_main\_\_":

main\_window()

* **if name == "main"** : Vérifie si le script est exécuté directement (et non importé comme module).
* **main\_window()** : Appelle la fonction pour créer et afficher la fenêtre principale de l'application.

**Conclusion**

Ce code permet à l'utilisateur de sélectionner une