Atelier_6

N'oubliez pas d'utiliser des commentaires pour expliquer le fonctionnement de chaque partie de votre code.

Activité Pandas

Considérez un ensemble de données fictif représentant les ventes d'une entreprise. L'ensemble de données est stocké dans un fichier CSV nommé "ventes.csv" avec les colonnes suivantes :

Date : La date de la vente (format : AAAA-MM-JJ).

Produit : Le nom du produit vendu.

Quantité : La quantité vendue.

Prix unitaire : Le prix unitaire du produit.

Tâches à effectuer :

Chargez l'ensemble de données dans une structure DataFrame.

1. Affichez les 5 premières lignes du DataFrame.
2. Calculez le chiffre d'affaires total réalisé par l'entreprise avec sum()
3. Identifiez le produit le plus vendu. Utiliser groupby() et idxmax()
4. Trouvez la date à laquelle l'entreprise a réalisé le meilleur chiffre d'affaires journalier.
5. Tracez un graphique barres pour visualiser les ventes par produit. Utiliser le paramètre kind avec la fonction plot(kind='bar')

Correction :

```python
import pandas as pd
import matplotlib.pyplot as plt

# 1. Chargez l'ensemble de données dans une structure DataFrame.
df = pd.read_csv('ventes.csv')

# 2. Affichez les 5 premières lignes du DataFrame.
print(df.head())

# 3. Calculez le chiffre d'affaires total réalisé par l'entreprise.
df['Chiffre_d_affaires'] = df['Quantité'] * df['Prix unitaire']
chiffre_d_affaires_total = df['Chiffre_d_affaires'].sum()
print(f"Le chiffre d'affaires total est : {chiffre_d_affaires_total}")
```

```
# 4. Identifiez le produit le plus vendu.
produit_plus_vendu = df.groupby('Produit')['Quantité'].sum().idxmax()
print(f"Le produit le plus vendu est : {produit_plus_vendu}")

# 5. Trouvez la date à laquelle l'entreprise a réalisé le meilleur chiffre
d'affaires journalier.
meilleur_chiffre_date = df.groupby('Date')['Chiffre_d_affaires'].sum().idx-
max()
print(f"La date avec le meilleur chiffre d'affaires est : {meil-
leur_chiffre_date}")

# 6. Tracez un graphique barres pour visualiser les ventes par produit.
ventes_par_produit = df.groupby('Produit')['Chiffre_d_affaires'].sum()
ventes_par_produit.plot(kind='bar')
plt.title('Ventes par produit')
plt.xlabel('Produit')
plt.ylabel('Chiffre d\'affaires')
plt.show()
```

Activité test unitaire :

Test unitaire d'une fonction de calcul

Considérez la fonction suivante qui effectue une opération de multiplication :

calc.py

def multiply(a, b):

    return a * b

Écrire des tests unitaires :

Créez un fichier de test (par exemple, test_calc.py) pour tester la fonction multiply.

Écrivez des tests pour vérifier le bon fonctionnement de la fonction dans différentes situations (positif, négatif, zéro, etc.).

Correction

calc.py

def multiply(a, b):

    return a * b

test_calc.py

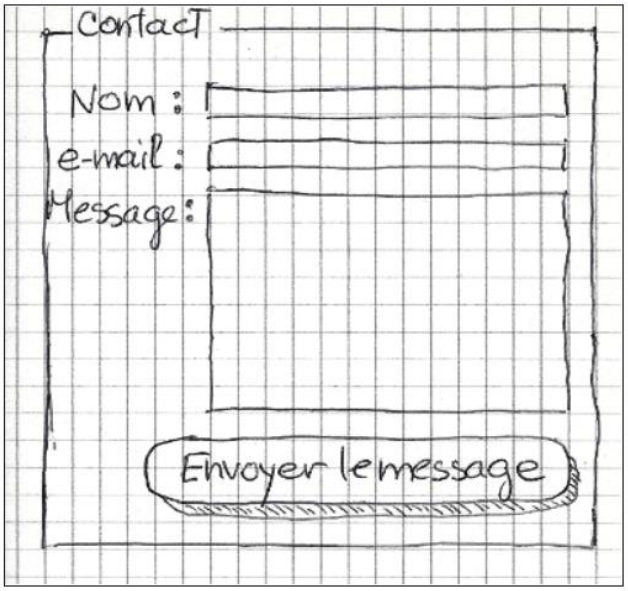
from calc import multiply

```
def test_multiply_positive_numbers():
    assert multiply(2, 3) == 6


def test_multiply_negative_numbers():
    assert multiply(-2, 3) == -6


def test_multiply_by_zero():
    assert multiply(2, 0) == 0


def test_multiply_by_one():
    assert multiply(5, 1) == 5
def test_multiply_invalid_input():
    with pytest.raises(TypeError):
        multiply('a', 3)
```

Activité Flask

Réaliser ce formulaire suivant dans une template projet flask

Le message écrit doit être affiche dans /message avec le nom et le mail



Correction

```
from flask import Flask, render_template, request, redirect, url_for

app = Flask(__name__)

@app.route('/')

def index():

    return render_template('index.html')

@app.route('/message', methods=['POST'])

def message():

    if request.method == 'POST':

        user_name = request.form['user_name']

        user_mail = request.form['user_mail']

        user_message = request.form['user_message']

        return render_template('message.html', user_name=user_name,
user_mail=user_mail, user_message=user_message)

    else:

        return redirect(url_for('index'))
```

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

Activité Pandas

Considérez un ensemble de données fictif représentant les ventes d'une entreprise. L'ensemble de données est stocké dans un fichier CSV nommé "ventes.csv" avec les colonnes suivantes :

Date : La date de la vente (format : AAAA-MM-JJ).

Produit : Le nom du produit vendu.

Quantité : La quantité vendue.

Prix unitaire : Le prix unitaire du produit.

Tâches à effectuer :

Chargez l'ensemble de données dans une structure DataFrame.

1. Affichez les 5 premières lignes du DataFrame.
2. Calculez le chiffre d'affaires total réalisé par l'entreprise avec sum()
3. Identifiez le produit le plus vendu. Utiliser groupby() et idxmax()
4. Trouvez la date à laquelle l'entreprise a réalisé le meilleur chiffre d'affaires journalier.
5. Tracez un graphique barres pour visualiser les ventes par produit:

```
ventes_par_produit = df.groupby('Produit').agg({'Quantité': 'sum', 'Prix unitaire': 'sum'})
ventes_par_produit.columns = ['Quantité totale', 'Chiffre d\'affaires total']
print("\nVentes par produit :\n"
, ventes_par_produit)
```

Utiliser le paramètre kind avec la fonction plot(kind='bar')

Correction

```
import pandas as pd
# . Chargez l'ensemble de données dans une structure DataFrame.
df = pd.read_csv("ventes.csv")
# 1. Affichez les 5 premières lignes du DataFrame.
print("Les 5 premières lignes du DataFrame :\n", df.head())
```

```
# 2. Calculez le chiffre d'affaires total réalisé par l'entreprise.

chiffre_affaires_total = (df['Quantité'] * df['Prix unitaire']).sum()

print("\nChiffre d'affaires total : ${:.2f}".format(chiffre_affaires_total))

# 3. Identifiez le produit le plus vendu.

produit_plus_vendu = df.groupby('Produit')['Quantité'].sum().idxmax()

print("\nProduit le plus vendu :", produit_plus_vendu)

# 4. Trouvez la date à laquelle l'entreprise a réalisé le meilleur chiffre d'affaires journalier.

meilleure_date = df.groupby('Date')['Quantité'].sum().idxmax()

print("\nMeilleure date en termes de chiffre d'affaires journalier :", meilleure_date)

# 5. Tracez un graphique barres pour visualiser les ventes par produit.

import matplotlib.pyplot as plt

ventes_par_produit['Quantité totale'].plot(kind='bar', rot=45, ylabel='Quantité totale',
title='Ventes par produit')

plt.show()
```

Activité: Gestion de fichiers avec subprocess

Créez une fonction Python appelée create_file qui prend un nom de fichier en paramètre et utilise la commande shell touch pour créer ce fichier.

Liste de fichiers :

Créez une fonction Python appelée list_files qui utilise la commande shell ls pour lister les fichiers dans le répertoire courant. La fonction devrait renvoyer la liste des noms de fichiers.

Suppression de fichiers :

Créez une fonction Python appelée delete_file qui prend un nom de fichier en paramètre et utilise la commande shell rm pour supprimer ce fichier.

Modification de fichiers :

Créez une fonction Python appelée append_to_file qui prend un nom de fichier et une chaîne de texte en paramètres, puis utilise la commande shell echo pour ajouter cette chaîne à la fin du fichier.

Correction

```
import subprocess
```

```
def create_file(file_name):
    try:
        subprocess.run(['touch', file_name], check=True)
        print(f"Le fichier {file_name} a été créé.")
    except subprocess.CalledProcessError:
        print(f"Erreur : Impossible de créer le fichier {file_name}.")

def list_files():
    try:
        result = subprocess.run(['ls'], stdout=subprocess.PIPE, text=True, check=True)
        file_list = result.stdout.split('\n')
        file_list = [file.strip() for file in file_list if file.strip()]
        print("Liste des fichiers dans le répertoire courant :")
        print(file_list)
        return file_list
    except subprocess.CalledProcessError:
        print("Erreur : Impossible de lister les fichiers.")

def delete_file(file_name):
    try:
        subprocess.run(['rm', file_name], check=True)
        print(f"Le fichier {file_name} a été supprimé.")
    except subprocess.CalledProcessError:
        print(f"Erreur : Impossible de supprimer le fichier {file_name}.")

def append_to_file(file_name, text):
    try:
        subprocess.run(['echo', text, '>>', file_name], shell=True, check=True)
        print(f"Le texte a été ajouté à {file_name}.")
    except subprocess.CalledProcessError:
        print(f"Erreur : Impossible d'ajouter du texte à {file_name}.")

# Tests
create_file('test_file.txt')
file_list = list_files()
if file_list:
    delete_file(file_list[0])
    append_to_file(file_list[0], "Nouveau texte ajouté.")
    list_files()
```

Activité Json

Créer un script Python qui utilise des fichiers JSON pour stocker des données.

Ajoutez une fonction create_json_file qui prend un nom de fichier et des données en paramètres, puis écrit ces données dans un fichier JSON.
Lecture d'un fichier JSON :
Ajoutez une fonction read_json_file qui prend un nom de fichier en paramètre et lit les données depuis un fichier JSON.
Modification d'un fichier JSON :
Ajoutez une fonction update_json_file qui prend un nom de fichier, une clé et une nouvelle valeur en paramètres. La fonction devrait mettre à jour la valeur associée à la clé dans le fichier JSON.
Affichage des données :
Ajoutez une fonction display_data qui prend un nom de fichier en paramètre et affiche le contenu du fichier JSON.

Tests :

Écrivez des tests pour chaque fonction que vous avez créée. Testez la création, la lecture, la mise à jour et l'affichage des données JSON.
Correction

```
from tinydb import TinyDB, Query

# Initialisation de la base de données
db = TinyDB('data.json')

def create_json_file(data):
    try:
        # Insertion des données dans la base de données
        db.insert(data)
        print("Les données ont été ajoutées avec succès à la base de données.")
    except Exception as e:
        print(f"Erreur : Impossible d'ajouter les données à la base de données. {e}")

def read_json_file():
    try:
        # Récupération de toutes les données de la base de données
        data = db.all()
        return data
    except Exception as e:
        print(f"Erreur inattendue lors de la lecture des données depuis la base de données. {e}")
        return None

def update_json_file(key, new_value):
    try:
        # Mise à jour des données dans la base de données
        Person = Query()
        db.update({'age': new_value}, Person.nom == key)
```

```
        print(f"La valeur associée à la clé {key} a été mise à jour avec succès.")
    except Exception as e:
        print(f"Erreur : Impossible de mettre à jour la valeur dans la base de données. {e}")

def display_data():
    try:
        # Affichage de toutes les données de la base de données
        data = read_json_file()
        if data:
            print("Contenu de la base de données :")
            for entry in data:
                print(entry)
    except Exception as e:
        print(f"Erreur inattendue lors de l'affichage des données. {e}")

# Test
sample_data = {"nom": "Alice", "age": 30}
create_json_file(sample_data)
display_data()
update_json_file("Alice", 31)
display_data()
```

Bon travail