

# UNIVERSITE DON BOSCO DE LUBUMBASHI

REPUBLIQUE DEEMOCRATIQUE DU CONGO

PROVINCE DE HAUT KATANGA

LUBUMBASHI

[WWW.ESSALAMA](http://WWW.ESSALAMA.ORG) .ORG

---



## COURS DE LANGAGE C

---

### Sujet : Rapport du projet sur la réservation de bus

---

*Réalisé par : RUMB IRUNG PASCAL SAM*

*OHUMA KATUNGU JONATHAN*

*MUSOMB TSHISUZ TSHIBAND*

*SPIRO BALASKAS STAVROS*

*Dirigé par : Mr. Ferdinand kayenga*

*Faculté : SCIENCES INFORMATIQUE*

*Promotion : Licence 2B Groupe 4*

Année Académique 2023 – 2024

## Introduction

Dans le cadre du cours d'algorithmique, il nous a été demandé de réaliser un programme en Python divisé en quatre modules distincts : le module principal (main), le module article, le module vente et le module mise en forme. Chacun de ces modules contient des fonctions spécifiques permettant de répondre aux différentes exigences du projet. Ce rapport a pour objectif de présenter en détail l'organisation et le fonctionnement de ce programme.

## Développement

Le programme est composé de quatre modules principaux :

### 1. Module principal (main) :

- Ce module sert de point d'entrée du programme.
- Il importe les modules article, vente et mise en forme.
- Il contient la fonction main() qui coordonne l'exécution des différentes fonctions des autres modules.

```
def main():
    """
    fonction principale
    """
    while True:
        choix = menu_principal()
        if os.path.exists("produits.json"):
            pass
        else:
            donne_debut()
            print(f"le fichier n'est pas trouvé, donc on vous crée le fichier POUR LE PRODUIT automatiquement")
            pause()
        if os.path.exists("fichier_vente.json"):
            pass
        else:
            donne_debut_vente()
            print(f"le fichier n'est pas trouvé, donc on vous crée le fichier POUR LA VENTE automatiquement")
            pause()
        if choix == '1':
            #interface_ajout_produit()
            ajout_article(donne_produit())
        elif choix == '2':
            #interface_affichage_produits()
            afficher_article()
        elif choix == '3':
            #interface_recherche_produit()
            echange_verifier()
        elif choix == '4':
            #interface_pour_supprimer_un_produit()
            supprimer_produit()
        elif choix == '5':
            #interface_enregistrement_vente()
            charger_historique_ventes()
            ajouter_vente()
        elif choix == '6':
            #interface_affichage_ventes()
            lister_ventes()
        elif choix == '7':
```

### 2. Module article :

- Ce module gère les informations relatives aux articles, telles que le nom, la description, le prix, etc.
- Il contient des fonctions pour créer, modifier et afficher les informations des articles.
- Par exemple, la fonction create\_article() permet de créer un nouvel article avec ses différentes propriétés.

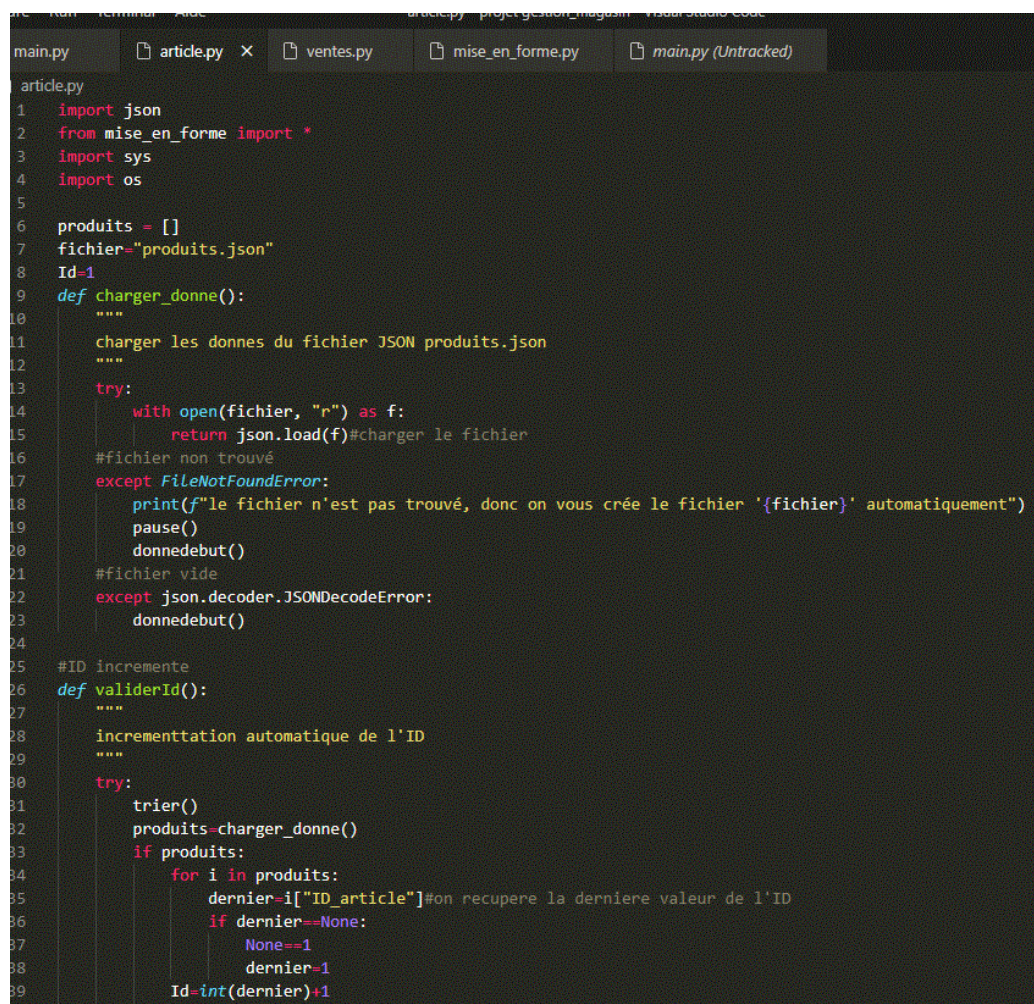


```
main.py | article.py | ventes.py x | mise_en_forme.py | main.py (Untracked)
ventes.py
1  import json
2  import csv
3  from mise_en_forme import *
4  from article import *
5  from datetime import date
6
7
8
9  fichiervente="fichiervente.json"
10 rapport="rapport.json"
11 rapcsv="rapcsv.csv"
12
13 #LIB
14 import json
15 from datetime import datetime
16
17 # Initialisation de la liste des ventes
18 ventes_liste = []
19
20 # Fonction pour générer un nouvel ID unique
21 def generer_nouveau_id():
22     try:
23         with open('fichiervente.json', 'r') as fichier:
24             historique = json.load(fichier)
25             if historique:
26                 derniers_id = [vente['id'] for vente in historique]
27                 nouveau_id = max(derniers_id) + 1
28                 return nouveau_id
29             else:
30                 return 1
31     except FileNotFoundError:
32         return 1
33
34 # Fonction pour vérifier si un produit existe dans l'inventaire
35 def verifier_produit(nom_produit):
36     try:
37         with open("produits.json", "r") as f:
38             produits = json.load(f)
39             for produit in produits:
40                 if nom_produit == produit["nom_produit"]:
```

### 3. Module vente :

- Ce module s'occupe de la gestion des ventes.
- Il contient des fonctions pour ajouter, supprimer et afficher les ventes.
- La fonction add\_sale() permet d'enregistrer une nouvelle vente avec les informations pertinentes.





```

main.py  article.py  X  ventes.py  mise_en_forme.py  main.py (Untracked)
article.py
1  import json
2  from mise_en_forme import *
3  import sys
4  import os
5
6  produits = []
7  fichier="produits.json"
8  Id=1
9  def charger_donne():
10     """
11     charger les donnees du fichier JSON produits.json
12     """
13     try:
14         with open(fichier, "r") as f:
15             return json.load(f)#charger le fichier
16     #fichier non trouvé
17     except FileNotFoundError:
18         print(f"le fichier n'est pas trouvé, donc on vous crée le fichier '{fichier}' automatiquement")
19         pause()
20         donne debut()
21     #fichier vide
22     except json.decoder.JSONDecodeError:
23         donne debut()
24
25     #ID incremente
26     def validerId():
27         """
28         incrementation automatique de l'ID
29         """
30         try:
31             trier()
32             produits=charger_donne()
33             if produits:
34                 for i in produits:
35                     dernier=i["ID_article"]#on recupere la derniere valeur de l'ID
36                     if dernier==None:
37                         None==1
38                         dernier=1
39                     Id=int(dernier)+1

```

#### 4. Module mise en forme :

- Ce module est responsable de la mise en forme du contenu affiché à l'utilisateur.
- Il contient des fonctions pour formater l'affichage des articles et des ventes.
- Par exemple, la fonction `format_article()` met en forme les informations d'un article pour une meilleure présentation.

Chaque module possède ses propres fonctions qui interagissent entre elles pour offrir une expérience utilisateur cohérente et fonctionnelle.

#### Conclusion

Ce programme en Python, divisé en quatre modules distincts, permet une gestion efficace des articles et des ventes. Grâce à cette architecture modulaire, le code est plus lisible, maintenable et évolutif. Chaque module étant responsable d'une tâche spécifique, il est plus facile de comprendre le fonctionnement global du programme et d'apporter des modifications si nécessaire. Cette approche modulaire est un excellent exemple de conception orientée objet et de bonne pratique en programmation.