Tutoriel sur les variables locales et globales

Quelle est la distinction entre une variable locale et globale : pourquoi et comment les utiliser ?

Par : Laurie Croteau

7 février 2024

Table des matières

1. Décoder le Monde des Données avec Python

La manipulation de données est une tâche courante en programmation. Elle consiste à transformer, analyser, filtrer ou visualiser des données provenant de différentes sources. Pour effectuer ces opérations, il est souvent nécessaire de créer et d'utiliser des variables, qui sont des noms associés à des valeurs. Cependant, toutes les variables ne sont pas accessibles ou modifiables de la même manière, selon où elles sont définies. Il existe donc deux types principaux de variables : les variables locales et les variables globales.

La compréhension de la différence entre les variables locales et globales est essentielle pour éviter des erreurs de programmation, des conflits de noms ou des effets indésirables sur les données. Par exemple, si vous utilisez une variable globale dans une fonction qui modifie sa valeur, vous risquez de changer la valeur de la variable globale pour tout le programme, ce qui peut entraîner des résultats inattendus ou incorrects. Il est donc important de savoir quand et comment utiliser les variables locales et globales, et comment gérer leur portée.

2. Les Fondements Indispensables

Pour suivre ce tutoriel, vous devez avoir des connaissances de base en python sur les notions suivantes :

- Les types de données (nombres, chaînes de caractères, listes, dictionnaires, etc.)
- Les structures de contrôle (conditions, boucles, etc.)
- Les fonctions (définition, appel, paramètres, retour, etc.)

3. Mise en Pratique : Explorer les Variables dans un Contexte Réel

Supposons que vous ayez un fichier CSV (Voir l'Annexe 1 : Le fichier CSV) contenant des données sur les ventes de différents produits dans plusieurs pays. Vous voulez créer un programme python qui lit ce fichier, calcule le total des ventes par produit et par pays, et affiche un graphique avec ces informations. Dans cette situation, vous allez utiliser la bibliothèque pandas. Pandas est une collection de modules Python qui facilite le traitement et l'exploration des données. Pandas renforce Python en y donnant la capacité de travailler avec des données de type feuille de calcul permettant un chargement, un alignement et une manipulation, en plus d'autres fonctions clés.

4. Les Acteurs Principaux : Variables Locales vs Variables Globales

Une **variable locale** est une variable qui est définie et utilisée dans un bloc de code spécifique, comme une fonction, une boucle ou une condition. Une variable globale, définie en dehors de tout bloc de code, peut être utilisée dans l'ensemble du programme, sauf si une variable locale du même nom la masque.

La portée d'une variable est l'ensemble des endroits où elle peut être accédée ou modifiée. La portée d'une variable locale est limitée au bloc de code où elle est définie. La portée d'une variable globale s'étend à tout le programme, sauf si elle est masquée par une variable locale du même nom. Par exemple, si vous définissez une variable globale nommée ventes_canada au début de votre programme et que vous définissez une variable locale nommée ventes_canada dans une fonction, la variable locale ventes_canada sera utilisée dans la fonction et la variable globale ventes_canada sera utilisée en dehors de la fonction. (Voir l'Annexe 2)

5. Mise en Pratique : Utilisation de Variables Locales et Globales

Exemple 1 : Code démontrant l'utilisation d'une variable locale.

```
# Définition d'une fonction qui calcule le pourcentage des ventes d'un produit
par rapport au total
def calculer_pourcentage(ventes_produit):
  # Définition d'une variable locale nommée pourcentage
  pourcentage = (ventes_produit / ventes_totales) * 100
  # Retour de la valeur de pourcentage
  return pourcentage

# On définit des variables globales pour les ventes de chaque produit et le total
ventes_a = 100
ventes_b = 150
ventes_c = 200
ventes_totales = 450

# Appel de la fonction calculer_pourcentage avec l'argument ventes_a
pourcentage_a = calculer_pourcentage(ventes_a)

# Affichage de la valeur de pourcentage_a
print(pourcentage_a) # Affiche 22.22

# Affichage de la valeur de pourcentage
print(pourcentage) # Erreur : pourcentage n'est pas défini
```

Dans cet exemple, la variable pourcentage est une variable locale qui est définie et utilisée dans la fonction calculer_pourcentage. Elle n'est pas accessible en dehors de la fonction, et donc l'instruction print(pourcentage) provoque une erreur.

Exemple 2 : Code démontrant l'utilisation d'une variable globale.

```
# Une variable globale nommée ventes_etats_unis
ventes_etats_unis = 200

# Définition d'une fonction qui affiche la valeur de ventes_etats_unis
def afficher_ventes_etats_unis():
  # Utilisation de la variable globale ventes_etats_unis
  print(ventes_etats_unis)

# Appel de la fonction afficher_ventes_etats_unis
afficher_ventes_etats_unis() # Affiche 200

# Affichage de la valeur de ventes_etats_unis
print(ventes_etats_unis) # Affiche 200
```

Dans cet exemple, la variable ventes_etats_unis est une variable globale qui est définie en dehors de toute fonction, et qui peut être utilisée partout dans le programme, y compris dans la fonction afficher_ventes_etats_unis.

Exemple 3 : Code illustrant les implications de la portée des variables dans la manipulation de données.

```
# Importation de la bibliothèque pandas
import pandas as pd

# Définition d'une variable globale nommée donnees
donnees = pd.read_csv("ventes.csv") # Lecture du fichier CSV contenant les
données sur les ventes

# Définition d'une fonction qui calcule le total des ventes par produit
def calculer_total_par_produit():
  # Définition d'une variable locale nommée total
  donnees = donnees.groupby("produit").sum() # Regroupement des données par
produit et somme des ventes
  # Retour de la valeur de total
  return donnees

# Appel de la fonction calculer_total_par_produit
resultat = calculer_total_par_produit()

# Affichage du résultat
print(resultat)

# Affichage de la valeur de donnees
print(donnees) # Erreur : référence circulaire
```

Dans cet exemple, la variable donnees est une variable globale qui contient les données brutes sur les ventes. La fonction calculer_total_par_produit définit une variable locale nommée donnees qui contient le résultat du regroupement des données par produit. Cependant, cette variable locale donnees fait référence à la variable globale donnees, ce qui crée une référence circulaire. En effet, pour définir la variable locale donnees, il faut utiliser la variable globale donnees, mais pour utiliser la variable globale donnees, il faut définir la variable locale donnees. Cela provoque une erreur lors de l'appel de la fonction calculer_total_par_produit. Pour éviter cette erreur, il faut utiliser un nom différent pour la variable locale, par exemple donnees_produit.

6. Les Clés du Succès : Bonnes Pratiques en Manipulation de Données

Voici quelques conseils pour utiliser les variables locales et globales de manière efficace et sécurisée dans la manipulation de données :

- Privilégiez l'utilisation de variables locales plutôt que de variables globales, car elles réduisent les risques de conflits de noms, de modifications involontaires ou de fuites de mémoire.
- Utilisez des noms de variables explicites et distincts qui reflètent le contenu et la portée des variables.
- Évitez de créer une variable locale du même nom qu'une variable globale
- Si vous voulez utiliser une variable globale dans une fonction et la modifier dans cette fonction, vous devez utiliser le mot-clé « global » avant le nom de la variable. Par exemple, global ventes_france permet d'utiliser et de modifier la variable globale ventes_france dans la fonction.

```
# Une variable globale nommée ventes_france
ventes_france = 100

def augmenter_ventes_france():
  # On utilise le mot-clé global pour accéder à la variable globale
  global ventes_france
  # On modifie la valeur de la variable globale ventes_france
  ventes_france = ventes_france + 10
  print(ventes_france) # Affiche 110

# On appelle la fonction augmenter_ventes_france
augmenter_ventes_france()

# On vérifie la valeur de la variable ventes_france après la fonction
print(ventes_france) # Affiche 110
```

- Si vous voulez utiliser une variable locale dans une fonction et la rendre accessible en dehors de cette fonction, vous devez utiliser le mot-clé « return » pour renvoyer la valeur de la variable. Par exemple, return ventes_totales permet de renvoyer la valeur de la variable locale ventes_totales à la fin de la fonction.

```
def calculer_ventes_totales():
  # Une variable locale nommée ventes_totales
  ventes_totales = ventes_france + ventes_canada + ventes_etats_unis
  # On renvoie la valeur de la variable locale ventes_totales
  return ventes_totales

# On définit des variables globales pour les ventes de chaque pays
ventes_france = 100
ventes_canada = 150
ventes_etats_unis = 200

# On entrepose la valeur de retour de la fonction calculer_ventes_totales
resultat = calculer_ventes_totales()

# On affiche la valeur de la variable resultat
print(resultat) # Affiche 450
```

7. Prérequis Essentiels : Garantir la Reproductibilité des Exemples

Pour que le lecteur puisse reproduire les exemples présentés dans le tutoriel, il doit disposer des éléments suivants :

- Un environnement de programmation python, comme VS Code, PyCharm ou Spider.
- La bibliothèque pandas, qui peut être installée avec la commande pip install pandas ou conda install pandas.
- Le fichier CSV contenant les données sur les ventes, qui peut être téléchargé à partir du lien fourni dans la section 14.

8. Introduction Générale: Application Avancée des Variables Locales et Globales

La compréhension des variables locales et globales est cruciale dans de nombreuses situations où la manipulation de données est impliquée. Voici quelques exemples de cas d'utilisation :

- Analyse de données : Si vous voulez analyser des données provenant de différentes sources, vous devez utiliser des variables locales pour stocker les résultats intermédiaires de vos opérations et des variables globales pour stocker les données brutes ou les paramètres généraux. Par exemple, si vous voulez calculer la moyenne, la médiane et l'écart-type d'une colonne de données, vous pouvez utiliser des variables locales pour stocker ces valeurs et une variable globale pour stocker le nom de la colonne. (Voir l'Annexe 3)
- Visualisation de données : Si vous voulez créer des graphiques à partir de vos données, vous devez utiliser des variables locales pour stocker les objets graphiques et des variables globales pour stocker les options de style ou de formatage. Par exemple, si vous voulez créer un histogramme avec la bibliothèque matplotlib, vous pouvez utiliser une variable locale pour stocker l'objet figure et une variable globale pour stocker la couleur ou la taille des barres. (Voir l'Annexe 4)
- Automatisation de tâches : Si vous voulez automatiser des tâches répétitives sur vos données, vous devez utiliser des variables locales pour stocker les données temporaires et des variables globales pour stocker les données persistantes ou les fonctions utiles. Par exemple, si vous voulez créer un programme qui envoie un courriel personnalisé à chaque client en fonction de ses achats, vous pouvez utiliser une variable locale pour stocker le message à envoyer, et une variable globale pour stocker la fonction qui envoie le courriel. (Voir l'Annexe 5)
- Système de gestion de base de données : Explorez comment les variables locales et globales sont utilisées dans un projet de gestion de base de données. Apprenez à maintenir l'intégrité des données, à gérer les transactions et à optimiser les performances grâce à une manipulation judicieuse des variables. (Voir l'**Error! Reference source not found.**)

9. Limites et Horizons : Comprendre les Défis et les Opportunités Futures

Ce tutoriel a pour but de vous initier aux concepts de base des variables locales et globales en python, et de vous montrer comment les utiliser dans la manipulation de données. Il ne couvre pas tous les aspects ou les subtilités de ces concepts, ni toutes les situations possibles où ils sont utiles. Par exemple, il ne traite pas des cas où il y a plusieurs niveaux de portée, comme les variables locales à une classe ou à un module, ni des cas où il y a des interactions entre les variables locales et globales, comme les closures ou les décorateurs.

Si vous voulez approfondir vos connaissances sur les variables locales et globales en python, et sur la manipulation de données en général, voici quelques ressources que vous pouvez consulter :

- Le site officiel de python, qui contient la documentation complète du langage, ainsi que des tutoriels et des exemples : https://www.python.org/
- Le site officiel de pandas, qui contient la documentation complète de la bibliothèque, ainsi que des tutoriels et des exemples : https://pandas.pydata.org/
- Le livre "Python for Data Analysis" de Wes McKinney, qui est une référence sur l'utilisation de python et de pandas pour la manipulation de données
- Le cours en ligne "Introduction to Data Science in Python" de l'Université du Michigan, qui est un cours gratuit et interactif sur les bases de la science des données en python

10. Comprendre l'Essentiel : Synthèse des Variables Locales et Globales en Python

Dans ce tutoriel, vous avez appris ce que sont les variables locales et globales en python, quelle est leur portée et comment les utiliser dans la manipulation de données. Vous avez vu que les variables locales sont définies et utilisées dans un bloc de code spécifique, comme une fonction, une boucle ou une condition, et que leur portée est limitée à ce bloc de code. Vous avez vu que les variables globales sont définies en dehors de tout bloc de code et qu'elles peuvent être utilisées partout dans le programme, sauf si une variable locale du même nom masque la variable globale. Vous avez vu que les variables locales et globales ont des avantages et des inconvénients, et qu'il faut les utiliser avec précaution et discernement. Vous avez vu des exemples pratiques de l'utilisation des variables locales et globales dans la manipulation de données, avec la bibliothèque pandas, et des conseils pour les utiliser efficacement et de manière sécuritaire.

En comprenant la différence entre les variables locales et globales, vous êtes capable de gérer plus facilement et plus efficacement les données que vous manipulez, et d'éviter des erreurs de programmation, des conflits de noms ou des effets indésirables sur les données. Vous êtes également capable de créer des programmes plus lisibles, plus modulaires et plus réutilisables qui répondent à vos besoins et à vos objectifs.

11. Au-delà du Code : Impact Personnel de la Maîtrise des Variables Locales et Globales

En maîtrisant les concepts de variables locales et globales en python, vous gagnez en compétence et en confiance dans la manipulation de données. Vous pouvez réaliser des analyses plus poussées, des visualisations plus attrayantes, et des automatisations plus performantes sur vos données. Vous pouvez également partager vos résultats avec d'autres personnes, et contribuer à la résolution de problèmes réels ou à la création de valeur ajoutée.

12. Où Aller à Partir d'Ici : Perspectives Futures dans la Manipulation de Données

Si vous avez aimé ce tutoriel, et que vous voulez continuer à apprendre et à progresser dans le domaine de la manipulation de données en python, voici quelques notions ou étapes que vous pouvez explorer :

- Les structures de données avancées, comme les tuples, les ensembles ou les collections, qui offrent des fonctionnalités supplémentaires, pour stocker et manipuler des données.
- Les expressions régulières, qui permettent de rechercher, de filtrer ou de remplacer des motifs de texte dans des données.
- Les bases de données, qui permettent de stocker, de gérer ou de récupérer des données de manière structurée et sécurisée.
- Les tests unitaires, qui permettent de vérifier le bon fonctionnement de votre code, et de détecter les erreurs ou les bugs.

13. Fin d'un Chapitre : Les Réalisations et les Perspectives

Félicitations, vous avez terminé ce tutoriel sur les variables locales et globales en python en manipulation de données. Vous avez appris ce que sont ces concepts, comment les utiliser, et pourquoi ils sont importants. Vous avez vu des exemples concrets et des conseils pratiques pour les mettre en œuvre. Vous avez également découvert des ressources et des perspectives pour approfondir vos connaissances et vos compétences. J'espère que ce tutoriel vous a été utile et intéressant, et que vous avez pris plaisir à le suivre. N'hésitez pas à me faire part de vos commentaires, de vos questions ou de vos suggestions. Merci de votre attention et à bientôt !

14. Le Savoir à Votre Portée : Liens et Ressources GITHUB

Vous pouvez trouver le tutoriel complet, le code source python, et le fichier CSV contenant les données sur les ventes sur le dépôt GitHub suivant : https://github.com/LaurieUDES/Tutoriel-variables-locales-et-globales/. Vous pouvez télécharger, cloner ou fork le dépôt, et exécuter le code sur votre propre environnement. Vous pouvez également modifier, améliorer ou étendre le code selon vos besoins ou vos envies.

I. Annexe 1 : Le fichier CSV

Le fichier ventes.csv a le contenu suivant :

```
produit,pays,ventes
A,France,100
A,Canada,150
A,États-Unis,200
B,France,50
B,Canada,75
B,États-Unis,100
C,France,25
C,Canada,40
C,États-Unis,60
```

II. Annexe 2 : Définition de l'ombrage de variable

Une variable globale est une variable qui peut être accédée et modifiée par n'importe quelle fonction ou instruction dans le programme. Par exemple, en Python, on peut définir une variable globale comme ceci:

```
# Une variable globale nommée ventes_canada
ventes_canada = 150
```

Une variable locale est une variable qui est définie à l'intérieur d'un bloc de code, comme une fonction ou une boucle, et qui n'existe que dans ce bloc. Par exemple, en Python, on peut définir une variable locale comme ceci:

```
def modifier_ventes_france ():
  # Une variable locale nommée ventes_france
  ventes_france = 160
```

Le nom d'une variable locale peut être le même que celui d'une variable globale, mais elles sont considérées comme deux variables différentes. Par exemple, en Python, on peut avoir:

```
# Une variable globale nommée ventes_canada
ventes_canada = 150

def modifier_ventes_canada():
  # Une variable locale nommée ventes_canada
  ventes_canada = 160
  print(ventes_canada) # Affiche 160

# Appel de la fonction modifier_ventes_canada
modifier_ventes_canada()

# Affichage de la valeur de ventes_canada
print(ventes_canada) # Affiche 150
```

Dans ce cas, la variable locale ventes_canada masque la variable globale ventes_canada, c'est-à-dire qu'elle empêche l'accès à la variable globale dans le bloc où elle est définie. Ainsi, si on essaie d'utiliser la variable ventes_canada à l'intérieur de la fonction modifier_ventes_canada, on obtient la valeur de la variable locale (160) et non celle de la variable globale (150).

Masquer une variable globale par une variable locale du même nom peut être une source de confusion et d'erreurs, car on peut perdre la référence à la variable globale et modifier involontairement la variable locale. Il est donc recommandé d'éviter de donner le même nom à une variable globale et à une variable locale.

III. Annexe 3 : Exemple d'une analyse de données

Supposons que vous avez un fichier CSV nommé donnees.csv qui contient une colonne de données numériques appelée valeur. Vous voulez calculer la moyenne, la médiane et l'écart-type de cette colonne, et afficher les résultats. Vous pouvez utiliser le module pandas pour lire le fichier CSV et le module statistics pour calculer les statistiques. Par exemple, en Python, on peut écrire:

```
# On importe les modules pandas et statistics
import pandas as pd
import statistics as st

# On définit une variable globale pour le nom de la colonne
colonne = "ventes"

# On lit le fichier CSV avec pandas et on stocke le résultat dans une variable df
df = pd.read_csv("ventes.csv")

# On extrait la colonne de ventes avec pandas et on la convertit en une liste
ventes = df[colonne].tolist()

# On calcule la moyenne, la médiane et l'écart-type des ventes avec statistics et
on stocke les résultats dans des variables locales
moyenne = st.mean(ventes)
mediane = st.median(ventes)
ecart_type = st.stdev(ventes)

# On affiche les résultats avec des chaînes de caractères formatées
print(f"La moyenne des ventes est {moyenne:.2f} euros")
print(f"La médiane des ventes est {mediane:.2f} euros")
print(f"L'écart-type des ventes est {ecart_type:.2f} euros")
```

Dans cet exemple, on utilise une variable globale pour le nom de la colonne, car on peut vouloir l'utiliser dans d'autres parties du programme. On utilise des variables locales pour les valeurs calculées, car on n'en a pas besoin en dehors de la fonction qui les calcule. On utilise le module pandas pour lire et manipuler les données du fichier CSV, et le module statistics pour calculer les statistiques descriptives. Vous pouvez trouver plus d'informations sur ces modules dans les liens suivants: pandas et statistics.

IV. Annexe 4 : Exemple de visualisation de données

La bibliothèque matplotlib est une bibliothèque graphique qui permet de tracer des figures en 2D et 3D avec Python. Un histogramme est un type de figure qui représente la distribution d'une série de données numériques en utilisant des barres verticales ou horizontales.

Pour créer un histogramme avec matplotlib, on peut utiliser la fonction plt.hist() qui prend en argument une liste de données et d'autres paramètres optionnels. Par exemple, on peut spécifier le nombre ou les limites des intervalles (bins), la couleur ou la taille des barres, l'orientation de l'histogramme, etc.

On peut utiliser une variable locale pour stocker l'objet figure qui contient l'histogramme, en utilisant la fonction plt.subplots(). Cela permet de manipuler plus facilement les propriétés de la figure, comme son titre, ses axes, sa taille, etc.

On peut utiliser une variable globale pour stocker la couleur ou la taille des barres, si on veut les réutiliser dans d'autres parties du programme. Par exemple, on peut définir une variable globale couleur qui contient le nom d'une couleur, et l'utiliser comme argument de la fonction plt.hist().

Voici un exemple de code qui crée un histogramme avec matplotlib, en utilisant une variable locale pour l'objet figure et une variable globale pour la couleur des barres:

```
# On importe la bibliothèque matplotlib
import matplotlib.pyplot as plt

# On crée une liste de ventes aléatoires
ventes = [200, 400, 400, 500, 600, 700, 700, 700, 800, 800, 800, 1200, 1300]

# On définit une variable globale pour la couleur des barres
couleur = "orange"

# On crée un objet figure avec la fonction plt.subplots()
fig, ax = plt.subplots()

# On crée un histogramme avec la fonction plt.hist()
ax.hist(ventes, color=couleur, edgecolor="black")

# On ajoute un titre et des étiquettes aux axes
ax.set_title("Exemple d'histogramme des ventes avec matplotlib")
ax.set_xlabel("Ventes (en euros)")
ax.set_ylabel("Fréquences")

# On affiche la figure
plt.show()
```

V. Annexe 5 : Exemple d'automatisation de tâches

Ce programme est un exemple de comment créer un programme qui envoie un courriel personnalisé à chaque vendeurs en fonction de leurs ventes. Une variable locale (message) est utilisée pour stocker le message à envoyer et une variable globale (vendeurs) est utilisée pour envoyer le courriel.

```
# Une variable globale qui contient la fonction qui envoie le courriel
def envoyer_courriel(destinataire, message):
  # Ici, vous pouvez utiliser une bibliothèque comme smtplib pour envoyer le courriel
  print(f"Envoi du courriel à {destinataire} avec le message: {message}")

# Une liste de vendeurs avec leurs commissions
vendeurs = [
  {"nom": "Alice", "commissions": 120},
  {"nom": "Bob", "commissions": 240},
  {"nom": "Charlie", "commissions": 180}
]

# Une boucle qui parcourt les vendeurs et crée un message personnalisé pour chacun
for vendeur in vendeurs:
  # Une variable locale qui stocke le message à envoyer
  message = f"Bonjour {vendeur['nom']},\nFélicitations pour vos ventes du mois dernier. Vous avez gagné {vendeur['commissions']} euros de commissions.\nContinuez comme ça et vous atteindrez bientôt vos objectifs.\nCordialement,\nL'équipe Bing."
  # On appelle la fonction globale qui envoie le courriel
  envoyer_courriel(vendeur['nom'], message)
```

VI. Annexe 6 : Système de gestion de base de données

Dans ce projet, nous allons apprendre à créer et à manipuler une base de données à l'aide de la bibliothèque sqlite3. Nous verrons comment les variables locales et globales sont utilisées dans un projet de gestion de base de données. Nous apprendrons à maintenir l'intégrité des données, à gérer les transactions et à optimiser les performances grâce à une manipulation judicieuse des variables.

```
# Importation de la bibliothèque sqlite3
import sqlite3

# Définition d'une variable globale nommée connexion
connexion = sqlite3.connect('base_de_donnees.db') # Connexion à la base de
données

# Définition d'une fonction qui supprime une table
def supprimer_table(nom_table):
  # Création d'un objet Cursor qui permet d'exécuter des requêtes SQL
  curseur = connexion.cursor()
  # Définition d'une variable locale nommée requete
  requete = f"DROP TABLE IF EXISTS {nom_table}"
  # Exécution de la requête
  curseur.execute(requete)
  # Validation de la transaction
  connexion.commit()

# Appel de la fonction supprimer_table pour supprimer la table produits et ventes
supprimer_table("produits")
supprimer_table("ventes")


# Définition d'une fonction qui crée une table
def creer_table(nom_table, colonnes):
  # Création d'un objet Cursor qui permet d'exécuter des requêtes SQL
  curseur = connexion.cursor()
  # Définition d'une variable locale nommée requete
  requete = f"CREATE TABLE {nom_table} ({colonnes})"
  # Exécution de la requête
  curseur.execute(requete)
  # Validation de la transaction
  connexion.commit()

# Définition d'une fonction qui insère des données dans une table
def inserer_donnees(nom_table, valeurs):
  # Création d'un objet Cursor qui permet d'exécuter des requêtes SQL
  curseur = connexion.cursor()
  # Définition d'une variable locale nommée requete
  requete = f"INSERT INTO {nom_table} VALUES ({valeurs})"
  # Exécution de la requête
  curseur.execute(requete)
  # Validation de la transaction
  connexion.commit()
```

```
# Définition d'une fonction qui sélectionne des données d'une table
def selectionner_donnees(nom_table, condition):
  # Création d'un objet Cursor qui permet d'exécuter des requêtes SQL
  curseur = connexion.cursor()
  # Définition d'une variable locale nommée requete
  requete = f"SELECT * FROM {nom_table} WHERE {condition}"
  # Exécution de la requête
  curseur.execute(requete)
  # Récupération du résultat
  resultat = curseur.fetchall()
  # Affichage du résultat
  print(resultat)


# Appel de la fonction creer_table pour créer la table produits
creer_table("produits", "nom TEXT, prix REAL, categorie TEXT, couleur TEXT")

# Appel de la fonction creer_table pour créer la table ventes
creer_table("ventes", "produit TEXT, quantite INTEGER, date TEXT")

# Appel de la fonction inserer_donnees pour insérer des données dans la table
produits
inserer_donnees("produits", "'Livre', 10.0, 'Culture', 'rouge'")
inserer_donnees("produits", "'Stylo', 1.5, 'Papeterie', 'bleu'")
inserer_donnees("produits", "'T-shirt', 15.0, 'Mode', 'vert'")
inserer_donnees("produits", "'Casque', 50.0, 'High-tech', 'noir'")
inserer_donnees("produits", "'Chocolat', 2.0, 'Alimentation', 'marron'")

# Appel de la fonction inserer_donnees pour insérer des données dans la table
ventes
inserer_donnees("ventes", "'Livre', 3, '2024-01-01'")
inserer_donnees("ventes", "'Stylo', 5, '2024-01-02'")
inserer_donnees("ventes", "'T-shirt', 2, '2024-01-03'")
inserer_donnees("ventes", "'Casque', 1, '2024-01-04'")
inserer_donnees("ventes", "'Chocolat', 4, '2024-01-05'")


# Exemple 1 : Sélectionner les produits dont le prix est supérieur à 10 euros
selectionner_donnees("produits", "prix > 10")
# Exemple de sortie
[('B', 15, 'bleu'), ('C', 20, 'vert')] # Liste de tuples contenant le nom, le
prix, et la couleur des produits

# Exemple 2 : Sélectionner les ventes dont la quantité est inférieure à 5
selectionner_donnees("ventes", "quantite < 5")
# Exemple de sortie
[(1, 'A', 2, '2020-01-01'), (3, 'B', 3, '2020-01-03'), (4, 'C', 4, '2020-01-04')]
# Liste de tuples contenant l'id, le produit, la quantité, et la date des ventes

# Exemple 3 : Sélectionner les produits dont la couleur est rouge ou jaune
selectionner_donnees("produits", "couleur = 'rouge' OR couleur = 'jaune'")
# Exemple de sortie
[('A', 10, 'rouge'), ('D', 25, 'jaune')] # Liste de tuples contenant le nom, le
prix, et la couleur des produits
```

- Nous importons la bibliothèque sqlite3 qui permet d'interagir avec une base de données SQLite. • Nous définissons une variable globale connexion qui représente la connexion à notre base de données SQLite nommée base_de_donnees.db.
- Nous définissons une fonction supprimer_table(nom_table) qui supprime une table spécifique de notre base de données. Elle utilise la commande SQL DROP TABLE IF EXISTS pour supprimer la table si elle existe.
- Nous définissons une fonction creer_table(nom_table, colonnes) qui crée une nouvelle table dans notre base de données. Elle utilise la commande SQL CREATE TABLE pour créer la table.
- Nous définissons une fonction inserer_donnees(nom_table, valeurs) qui insère des données dans une table spécifique. Elle utilise la commande SQL INSERT INTO pour insérer les données.
- Nous définissons une fonction selectionner_donnees(nom_table, condition) qui sélectionne des données d'une table spécifique en fonction d'une condition donnée. Elle utilise la commande SQL SELECT * FROM pour sélectionner les données.

Dans ce projet, nous avons appris à créer et à manipuler une base de données à l'aide de la bibliothèque sqlite3. Nous avons vu comment les variables locales et globales sont utilisées dans un projet de gestion de base de données. Nous avons appris à maintenir l'intégrité des données, à gérer les transactions et à optimiser les performances grâce à une manipulation judicieuse des variables. Dans l'ensemble, ce script démontre comment utiliser Python et SQLite pour effectuer des opérations de base de données telles que la suppression de tables, la création de tables, l'insertion de données et la sélection de données.

VII. Références

Variable locale et globale | Python - WayToLearnX : Ce site explique la différence entre les variables locales et globales, et donne des exemples de code en python.

Variables globales et locales Python - Informatikka : Ce site donne également des exemples de code en python et montre comment utiliser le mot-clé global pour modifier une variable globale dans une fonction.

Différence entre la variable locale et globale - Guru99 : Ce site compare les variables locales et globales en termes de stockage, de passage de paramètres et d'avantages et d'inconvénients.

FAQ de programmation — Documentation Python 3.12.1 : Ce site contient la documentation officielle du langage python et répond à des questions fréquentes sur les variables locales et globales comme les closures, les décorateurs ou les variables non locales.

Variables Python : Comment définir/déclarer des types de … - Guru99 : Ce site donne une introduction générale aux variables en python et explique comment les déclarer, les concaténer, les supprimer, etc.