

Mémo Git – Projet « Numeric Bases Converter »

Ce mémo résume les commandes Git les plus utiles pour ton projet, avec l'organisation classique :

- branche **main** : version stable publiée sur GitHub
- branche **dev** : zone de travail, tests et nouvelles fonctionnalités

1. Mettre à jour la branche **main** avec la branche **dev**

1.1. Méthode simple via GitHub (Pull Request)

1. Aller sur ton dépôt GitHub → onglet **Pull requests**
2. Cliquer sur **New pull request**
3. Choisir :
 - **base** = **main**
 - **compare** = **dev**
4. Vérifier les modifications affichées
5. Cliquer sur **Create pull request**
6. Ajouter un titre, par exemple : **Fusion dev → main**
7. Cliquer sur **Merge pull request**
8. Confirmer avec **Confirm merge**

Résultat : la branche **main** contient maintenant tout ce qu'il y avait dans **dev**.

1.2. Même chose en ligne de commande (local)

```
# Se placer sur la branche main
git checkout main

# Récupérer la dernière version de main depuis GitHub
git pull origin main

# Fusionner la branche dev dans main
git merge dev

# Envoyer la branche main mise à jour vers GitHub
git push origin main
```

2. Mettre à jour **un seul fichier** de **main** à partir de **dev**

2.1. Méthode GitHub (copier-coller)

1. Sur GitHub, passer sur la branche **dev**
2. Ouvrir le fichier à transférer

3. Cliquer sur le crayon (**Edit this file**)
4. Copier tout le contenu du fichier
5. Revenir sur la branche **main**
6. Ouvrir le même fichier
7. Cliquer sur **Edit**, coller le contenu copié
8. Valider avec un commit (message du type : **Mise à jour depuis dev**)

Pratique pour corriger un seul fichier sans fusionner toute la branche.

2.2. Méthode ligne de commande : récupérer un fichier précis depuis **dev**

Ici, tu restes sur la branche **main**, mais tu prends **un seul fichier** tel qu'il existe dans **dev**.

```
# Se placer sur main
git checkout main

# Récupérer un fichier précis depuis dev
git checkout dev -- chemin/vers/le/fichier.py

# Vérifier dans ton éditeur que le fichier est bien mis à jour

# Préparer le commit
git add chemin/vers/le/fichier.py

# Créer le commit
git commit -m "Mise à jour du fichier depuis dev"

# Envoyer vers GitHub
git push origin main
```

Exemple adapté à ton projet :

```
git checkout main
git checkout dev -- src/convertisseur_bases.py
git add src/convertisseur_bases.py
git commit -m "Mise à jour convertisseur_bases depuis dev"
git push origin main
```

3. Mettre à jour ta branche **dev** depuis **main**

Parfois, tu veux repartir de la dernière version stable de **main** dans ta branche de développement **dev**.

```
# Se placer sur dev
git checkout dev
```

```
# Récupérer la dernière version de main  
git checkout main  
git pull origin main  
  
# Revenir sur dev  
git checkout dev  
  
# Fusionner main dans dev  
git merge main  
  
# (optionnel) Pousser dev si tu veux l'aligner sur GitHub aussi  
git push origin dev
```

Idée : fais ça quand tu as fait une release stable sur main et que tu veux continuer à coder dans dev à partir de cette base propre.

4. Workflow conseillé pour ton projet

1. **Coder et tester** dans la branche **dev**
2. Quand une fonctionnalité est **stable** (GUI OK, tests OK) :
 - soit tu **fais une Pull Request** **dev → main**
 - soit tu **fais un merge en local** puis un **git push** sur main
3. Pour un **petit correctif isolé** sur un fichier :
 - utiliser la méthode de **fichier unique** (section 2)
4. Garder **main toujours propre et fonctionnelle**, c'est la version pour :
 - GitHub public
 - futures releases
 - documentation

5. Quelques commandes Git utiles (rappel)

```
# Voir sur quelle branche tu es  
git branch  
  
# Lister les branches  
git branch -a  
  
# Créer une nouvelle branche  
git checkout -b nouvelle_branche  
  
# Voir l'état des fichiers (modifiés, suivis, etc.)  
git status  
  
# Voir l'historique des commits  
git log --oneline --graph --all
```

Mémo rédigé avec ton copilote Pylo – *Per Scientiam, ad Caelum* ✈