

Exercice Commun : Py-Nigma

Il faut une **application** (pour espions) qui aide à chiffrer et déchiffrer des messages.

Un **repo github** existe. Le squelette du projet s'y trouve.

Les **tests unitaires** sont écrits pour valider le projet. (réduits pour être abordables)

Un **Github Project** est prêt pour aider à gérer les tâches.

Un **pipe Github Actions** est prêt pour valider les Pull Requests.

Arriverez-vous à **coder** ensemble ce projet, le **soumettre** et le **lancer** ?

```
=====
🕵️ PY-NIGMA - Outil de Cryptographie 🕵️
=====
--- 🏰 CODE CÉSAR ---
1. Chiffrer      : Décaler les lettres (A + 1 -> B)
2. Déchiffrer   : Retrouver le message original

--- 📧 SUBSTITUTION ---
3. Leet Speak   : Remplacer les lettres par des chiffres (E->3, A->4)
4. Miroir       : Inverser l'alphabet (A->Z, B->Y)

--- 📊 ANALYSE ---
5. Fréquence    : Trouver la lettre qui apparait le plus souvent
6. Palindrome   : Vérifier si le mot se lit dans les deux sens

--- ✂️ UTILITAIRES ---
7. Générer MDP  : Créer un mot de passe fort et aléatoire
8. Masquer      : Cacher un secret avec des étoiles (****ok)
-----
Q. Quitter
-----
votre choix > 
```

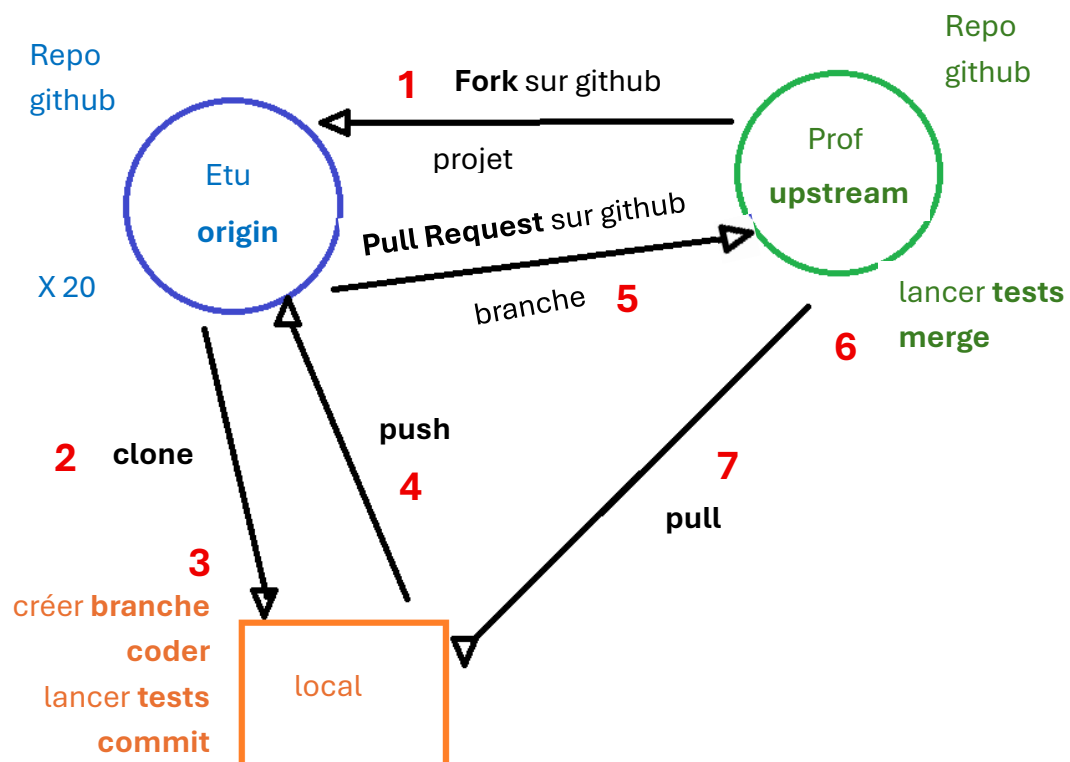
Guide de Contribution au Projet Py-Nigma

Votre mission : Contribuer au développement de la librairie de cryptographie.

🔧 Étape 0 : L'installation

1. Fork :

- Allez sur le lien GitHub du projet (upstream) fourni par le prof :
[https://github.com/\[PROF\]/\[PROJET\]](https://github.com/[PROF]/[PROJET])
- Cliquez sur le bouton **Fork** (en haut à droite)
Cela crée une copie sur *votre* compte github (origin).
Un **Fork** est une copie personnelle d'un projet, permettant d'y travailler, puis de soumettre les changements d'une branche au projet initial (Pull Request)



2. Clone (Rapatrier sur PC) :

- `git clone https://github.com/[VOTRE_PSEUDO]/[PROJET].git`
- `cd [PROJET]`

3. La Liaison Upstream :

- Pour récupérer le code validé des autres sur le repo du prof, il faut spécifier l'upstream pour ensuite l'utiliser :

git remote add upstream [https://github.com/\[PROF\]/\[PROJET\].git](https://github.com/[PROF]/[PROJET].git)

4. .venv (environnement virtuel)

(Windows) **python -m venv .venv**
ou
Mac/Linux) **python3 -m venv .venv**

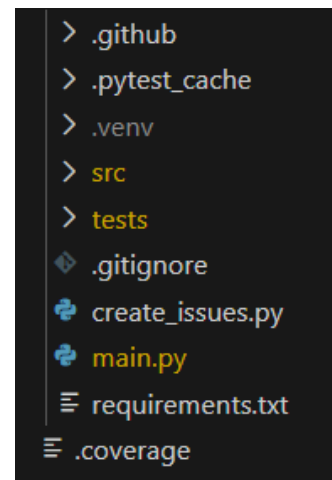
5. Installer les requirements du projet.

Soit à la main :

- **pip install pytest** (*pour tests unitaires*)
- **pip install pytest-cov** (*pour test coverage*)

Soit sur base du fichier requirements.txt :

- activer le .venv
(Windows) **.\.venv\Scripts\activate**
ou
(Mac/Linux) **source .venv bin/activate**
- Installer les requirements
pip install -r requirements.txt
- quitter le .venv
deactivate



Comprendre, après votre fork, clone, upstream :

- **origin** : c'est votre repo en ligne
- **upstream** : c'est le repo du prof

Pour vérifier à quoi est lié votre remote et votre upstream !

git remote -v

🎯 Étape 1 : Choisir une mission

Il y a **20 fonctions** à coder, réparties en **20 tâches**, visibles dans **issues** sur

[https://github.com/\[PROF\]/\[PROJET\]](https://github.com/[PROF]/[PROJET])

Pour ne pas s'embrouiller, chacun doit indiquer sur quelle tâche il travaille **avant** de commencer (!)



lvdEphéc 20 minutes ago

/assign



github-actions assigned lvdEphéc 20 minutes ago

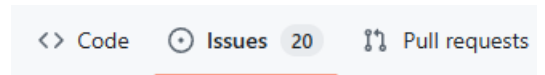


lvdEphéc moved this from Backlog to In progress in

Allez dans

l'onglet **Issues** du repo du prof.

1. Choisissez une tâche, et commentez **/assign** dessus.
2. Notez le numéro (ex: **#12**).



is:issue state:open

☐ **Open** 20 ☐ Closed 1

☐ **implémenter un_pour_prof**

#21 · lvdEphéc opened 2 hours ago

☐ **Implémenter masquer_texte**

#20 · github-actions opened 6 hours ago

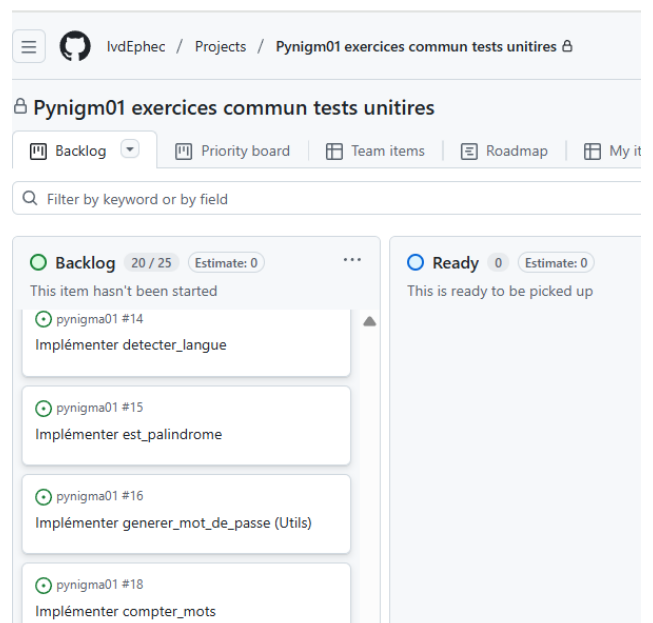
☐ **Implémenter est_mot_de_passe_f**

#19 · github-actions opened 6 hours ago

Le prof montrera l'état du Github Project au fur et à mesure de l'avancement.

Les **colonnes** :

- **Backlog** : tâches à réaliser du projet
- **Ready** : tâches validées et analysées qui peuvent être codées
- **In progress** : on travaille dessus
- **In Review** : lorsque c'est terminé, en attente de validation
- **Done** : lorsque la Pull Request est validée et le code mergé dans main



Étape 2 : Coder

1. Mettez-vous à jour localement :

- **git switch main**
(On retourne sur la base)
- **git pull upstream main**
(On télécharge les nouveautés depuis le repo du prof)
- **git push origin main**
(On met à jour notre fork en ligne, notre repo perso)

2. Créez votre branche de travail :

- **git switch -c prenom/ma-fonction**


3. Codez votre fonction dans src/.

4. Testez :

- Lancez dans le terminal :
pytest -v -k mot-clef

-v : verbose, plus d'infos

-k : seuls les tests contenant ***mot-clef*** seront lancés
(Pour ne pas lancer les 20 tests. Les autres vont « deselected ». A tester.)

○ Si c'est VERT  (et les autres « skip »), c'est bon !

```
tests/test_utilitaires.py::test_masquer_texte_long SKIPPED (Fonction à implémenter) [ 95%]
tests/test_utilitaires.py::test_masquer_texte_limite SKIPPED (Fonction à implémenter) [ 97%]
tests/test_utilitaires.py::test_masquer_texte_court SKIPPED (Fonction à implémenter) [ 98%]
tests/test_utilitaires.py::test_un_pour_prof PASSED [100%]
===== 1 passed, 68 skipped in 0.31s =====
```

```
PS C:\bureau\c2\2T-Python\2025-2026\Projets\enigma-solution\PROJET_PY-NIGMA> pytest -v -k prof
on313\python.exe
cachedir: .pytest_cache
rootdir: C:\bureau\c2\2T-Python\2025-2026\Projets\enigma-solution\PROJET_PY-NIGMA
plugins: anyio-4.10.0
collected 69 items / 68 deselected / 1 selected

tests/test_utilitaires.py::test_un_pour_prof PASSED [100%]
===== 1 passed, 68 deselected in 0.09s =====
```

Étape 3 : Envoyer

1. Sauvegardez :

- **git add .**
- **git commit -m "Fini fonction X (closes #12)"**
*Mettre "closes #numéro" fermera le ticket **automatiquement** !*
*Mettre #numéro dans le **comit** génère les liens entre tâche / comit !*

2. Envoyez sur votre compte :

- **git push origin prenom/ma-fonction**

3. Proposez (Pull Request) vers le repo du prof (upstream) :

- Allez sur GitHub. Cliquez sur "**Compare & pull request**".
- Validez et Créez le **pull request** (*main du repo prof ← branche repo perso*)
- Demander au prof d'accepter le PR.

- Le **pipe** du repo du prof va s'activer et lancer les **tests unitaires**.
(se relancent automatiquement lors de chaque push)

Cas 1, le cas normal ! : Pas d'erreurs de tests unitaires ?

Votre code est **mergé** dans le main !

Se lancer dans le codage d'une prochaine tâche, s'il en reste.

Cas 2 : Sinon, si **erreur** (avez-vous bien lancé les tests en local ?) :

corriger, puis mettre à jour votre repo perso

git add .

git commit -m "Correction du bug sur la fonction X"

git push origin prenom/ma-fonction

et le PR précédemment créé se met automatiquement à jour.

Cas 3 : Sinon, si **conflit** (avez-vous bien travaillé sur votre fonction ?) :

On rapatrie le code du repo du prof pour résoudre la situation.

git switch main (*pour mettre à jour main local*)

git pull upstream main (*récupérer le projet du repo du prof*)

git switch prenom/ma-fonction (*on se remet dans sa branche de travail*)

git merge main (*on merge le contenu du main dans sa branche*)

=> les conflits à résoudre apparaissent

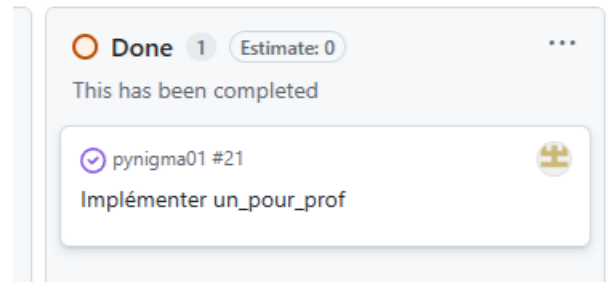
Corriger, puis **idem Cas 2** pour enregistrer les changements et les envoyer vers son repo perso (et mettre automatiquement à jour la PR).

Étape 4 : Récupérer le projet et l'utiliser

Lorsque toutes les tâches du Github Project sont à Done (ou même avant),

Vous pouvez récupérer le code du repo du prof et l'essayer.

1. Revenir sur la branche principale
git switch main
2. Télécharger la version finale du prof (contenant toutes les fonctions)
git pull upstream main
3. Lancer (les tests unitaires puis) le **main**



```
=====
👤 PY-NIGMA - Outil de Cryptographie 👤
=====
--- 🏠 CODE CÉSAR ---
1. Chiffrer      : Décaler les lettres (A + 1 -> B)
2. Déchiffrer   : Retrouver le message original

--- 🔄 SUBSTITUTION ---
3. Leet Speak   : Remplacer les lettres par des chiffres (E->3, A->4)
4. Miroir       : Inverser l'alphabet (A->Z, B->Y)

--- 📊 ANALYSE ---
5. Fréquence    : Trouver la lettre qui apparait le plus souvent
6. Palindrome   : Vérifier si le mot se lit dans les deux sens

--- ✂ UTILITAIRES ---
7. Générer MDP  : Créer un mot de passe fort et aléatoire
8. Masquer      : Cacher un secret avec des étoiles (***ok)

-----
Q. Quitter
-----
Votre choix > 
```