

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 fourni par le prof :
[LIEN_UPSTREAM_PROJET]
- Cliquez sur le bouton **Fork** (en haut à droite)
Cela crée une copie sur *votre* compte github.
Un **Fork** est une copie personnelle d'un projet, permettant d'y collaborer.

2. Clone (Rapatrifier sur PC) :

- **git clone https://github.com/[VOTRE_PSEUDO]/[PROJET].git**
- **cd pynigma01**

3. La Liaison Upstream :

- Pour soumettre votre code repo du prof
et récupérer le code des autres validés par le projet du prof :
git remote add upstream 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. (Ici ça installe juste pytest)

- (Windows) **venv\Scripts\activate**
ou
(Mac/Linux) **source .venv/bin/activate**
- **pip install -r requirements.txt**

Comprendre, près votre fork, clone, upstream :

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

```
> .github
> .pytest_cache
> .venv
> src
> tests
❖ .gitignore
🔗 create_issues.py
🔗 main.py
📄 requirements.txt
📄 .coverage
```

🎯 Étape 1 : Choisir une mission

Il y a **20 fonctions** à coder, répartie en **20 tâches**.

Pour ne pas s'embrouiller, dans le gestionnaire de tâches **Github Project**, chacun doit indiquer sur quelle tâche il travaille **avant** de commencer (!) et indiquer lorsqu'il a terminé, en mettant les tâches dans les **bonnes colonnes**.

- **In progress** : pendant qu'il travaille dessus
- **In Review** : lorsque c'est terminé, en attente de validation
- **Done** : quand la Pull Request est validée et le code mergé dans main.

A faire :

1. Allez dans l'onglet **Issues** du repo du prof.
2. Choisissez une tâche, et commentez **/assign** dessus.
3. Notez le numéro (ex: **#12**).



lvdEphéc 20 minutes ago

/assign



github-actions assigned lvdEphéc 20 minutes ago



lvdEphéc moved this from Backlog to In progress in

The screenshot shows a GitHub Project board for the repository 'lvdEphéc / Projects / Pynigma01 exercices commun tests unitaires'. The board has a tab for 'Backlog' selected, showing a list of tasks. The 'Ready' column is also visible.

Column	Count	Estimate
Backlog	20 / 25	0
Ready	0	0

Backlog tasks:

- pynigma01 #14: Implémenter detecter_langue
- pynigma01 #15: Implémenter est_palindrome
- pynigma01 #16: Implémenter generer_mot_de_passe (Utils)
- pynigma01 #18: Implémenter compter_mots

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, optionnel mais conseillé)

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 « skiper ». A tester.)
 - Si c'est VERT  (et pas « 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 "Finit la fonction X (closes #12)"**
*Mettre "closes #numéro" fermera le ticket **automatiquement** !*
*Mettre #numéro dans le **commit** génère les liens entre tâche / commit !*

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.
- Le pipe va s'activer et lancer les tests unitaires. (se relance lors de push)
- Pas d'erreurs lors des tests unitaires ?
-> Demandez au prof d'accepter la PR !
Et se lancer dans le codage d'une prochaine tâche s'il en reste.

Sinon, si erreur (avez-vous bien lancer les test en local ??) :

corriger, puis :

git add .

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

git push origin prenom/ma-fonction

et le PR se met automatiquement à jour

Étape 4 : Récupérer le projet et l'utiliser

Lorsque toutes les tâches du Github Project sont à Done,

Vous pouvez le récupérer 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**

