

# Attaque par force brute

La première partie du projet consiste à déterminer des mots de passe en utilisant une *attaque par force brute*. Les mots de passe protègent des fichiers `pdf` et il faudra utiliser Python pour générer des mots de passe et les tester jusqu'à trouver le bon. Une seconde partie du projet est la création d'un programme Python permettant de générer automatiquement des mots de passe forts.

On utilise dans ce projet le module `pymupdf` de Python permettant d'interagir avec des fichiers au format `pdf`. Pour installer ce module, écrire en ligne de commande :

## ❖ Texte

```
pip install pymupdf
```

On donne ci-dessous une fonction Python `test_password` qui prend en argument une chaîne de caractères `password` et renvoie `True` lorsque `password` est le mot de passe permettant d'ouvrir le fichier `FILE` déclaré en constante au début du programme :

## 🐍 Script Python

```
1 import fitz
2
3 FILE = "protege1.pdf"
4 DOC = fitz.Document(FILE)
5
6
7 def test_password(password):
8     '''Renvoie True lorsque password permet d'ouvrir le fichier FILE déclaré ci-dessus'''
9     return DOC.authenticate(password)
```

Par exemple pour tester si le mot de passe du fichier `protege1` est "1789", on peut par exemple faire :

## 🐍 Script Python

```
if test_password("1789"):
    print("Le mot de passe est 1789")
```

Attention, le fichier sur lequel on effectue le test est donné à la ligne 3 dans la variable `FILE`.

## 1. Etape 1 : Avec des chiffres

Ces deux premiers fichiers ne devraient pas poser problème : les mots de passe ne contiennent que des chiffres !

### 1.1. Fichier 1

- Fichier à télécharger :

[Fichier1 ↓](#)

- Informations sur le mot de passe:

Vous savez que ce mot de passe contient 4 caractères qui sont tous des chiffres. C'est à dire que ce mot de passe est semblable à un code de carte bancaire, comme par exemple 1492 ou 0141.

## 1.2. Fichier 2

- Fichier à télécharger :

[Fichier2 ↓](#)

- Informations sur le mot de passe:

Vous savez que ce mot de passe est une date de naissance au format jjmmaaaa et que l'année de naissance est supérieure à 1900.



### Aide

Afin d'optimiser votre programme et de ne pas tester inutilement des mots de passe, attention à bien générer des dates de naissances valides !

## 2. Etape 2 : avec des lettres aussi !

### 2.1. Fichier 3

- Fichier à télécharger :

[Fichier3 ↓](#)

- Informations sur le mot de passe:

Vous savez que ce mot de passe est un mot de la langue française écrit en minuscule. Pour vous aider, un dictionnaire peut être téléchargé ci-dessous :

[Dictionnaire ↓](#)

### 2.2. Fichier 4

- Fichier à télécharger :

[Fichier4 ↓](#)

- Informations sur le mot de passe:

Les mots de passe les plus courants ont été répertoriés sur ce [site](#). Le quatrième mot de passe figure dans cette

liste de mots de passe couramment utilisée.

### 3. Etape 3 : à peine plus compliqué !

#### 3.1. Fichier 5

- Fichier à télécharger :

[Fichier5 ↓](#)

- Informations sur le mot de passe: De nombreuses personnes utilisent leur prénom (écrit avec une majuscule) suivi de leur numéro de département comme mot de passe comme par exemple `Kevin974` pour un habitant de la Réunion se prénommant Kevin. C'est le cas pour ce mot de passe ou vous savez de plus que le prénom est courant (il figure dans le top100 des prénoms les plus attribués en France ces dernières années)

### 4. Etape 4 : générateur de mots de passe

Ecrire un programme Python permettant de générer des mots de passe aléatoires résistants à une attaque par force brute. L'utilisateur pourra préciser les caractères à utiliser (chiffres, majuscules, minuscules, caractères spéciaux) et la longueur du mot de passe souhaité.

### 5. Etape 5 : pour aller plus loin

Ecrire un programme Python permettant de déterminer le temps approximatif nécessaire pour trouver un mot de passe par une attaque par force brute en fonction de la longueur du mot de passe et du type de caractère utilisé.

#### Bibliographie :

- Ce cours s'inspire largement de celui de Fabrice Nativel.