Débrief séance 5

UE12 P24 - Python

Les principes

- Pour progresser : pas de par cœur, mais de la pratique
  - L'important n'est pas de se souvenir de tous les détails, mais de savoir ce qui existe et comment chercher la doc
  - Prenez du temps pour revenir sur ce qui vous a bloqué
- Votre code est fait pour être lu par des humains, pas par des machines
  - Bien le structurer
  - Commenter et documenter
  - Le faire évoluer régulièrement (refactorer)

Les classes

Le cœur d'un programme = les données

> Bad programmers worry about the code. Good programmers worry about data structures and their relationships.

Linus Torvalds, créateur de Linux (et Git)

Les classes

En Python, on structure les données en classes.

- Elles mettent les données au centre
- On peut leur rattacher des fonctions

Les classes

Par exemple, si on veut représenter un horodatage :

```python
class Datetime:
    def __init__(self,
        year: int, month: int, day: int,
        hour: int = 0, minute: int = 0, second: int = 0,
        timezone: str = 'UTC'
    ):
        self.year = year, self.month = month, self.day = day
        self.hour = hour, self.minute = minute, self.second = second
        self.timezone = timezone

    @classmethod
    def from_str(cls, date_as_str: str) -> 'Datetime':
        pass

    def change_timezone(self, new_timezone: str):
        pass
```

Une fois les données bien définies, on peut écrire toutes les fonctions sereinement.

La ligne de commande (CLI)

Vous utilisez souvent la ligne de commande :

```
1 git commit -m 'Remplacement de dictionnaires par des classes'
```

1. Votre shell parse la commande
2. Il trouve le programme (ici, `git`)
3. Il l'appelle avec ses arguments
4. La suite est gérée par le programme

La ligne de commande (CLI)

Donc si vous voulez pouvoir écrire :

```
1 messenger.py --url 'https://eleves.mines-paris.eu'
```

1. Votre shell parsera la commande
2. Il trouvera le programme (ici, `messenger.py`)
3. Il l'appellera avec ses arguments
4. C'est à vous d'écrire dans votre programme quoi faire de ces arguments
   - En Python, utilisez `argparse.ArgumentParser` ou `click`

Git et GitHub

- Dès que vous ajoutez une fonctionnalité, faites un commit
- Poussez votre code sur GitHub à la fin de chaque cours
- Lorsque vous travaillez à plusieurs, faites des branches et des Pull Request (PR)

La suite du cours

- Il nous reste 3 séances de 3h (7, 14, 21 janvier)
  - Préparez bien toutes vos questions pour la séance du 21
  - N’hésitez pas à me contacter par mail
- Vous aurez un hackathon d’1 journée le 31 janvier
- L’évaluation sera basée sur le projet “Messenger” :
  - Implémentation des fonctionnalités de base
  - Bonne structure de code