UNIVERSITÉ DE CAEN **Année universitaire 2024-2025**

IUT Grand Ouest Normandie **Département Réseaux et Télécommunications**

SAÉ 1.05 — Traiter des données — Sujet de projet

Martin Pépin

1 Généralités

1.1 Organisation

— Le projet est à réaliser en **binômes**. Vous devez renseigner vos groupes sur le pad disponible sur la page Ecampus de la SAÉ, au **plus tard** le vendredi 6 décembre 2024. S'il y a un nombre impair d'étudiant·es, certain·es devront travailler en autonomie (trinômes interdits).

— Une personne du binôme fera un rendu sur Ecampus. Il sera constitué de votre code et d'un très court rapport (1 ou 2 pages, voir plus bas). Date de fermeture du dépôt : **vendredi 17 janvier 2025** à 23h59. Retard → pénalité.

— Votre code doit à minima s'exécuter correctement sur la version de Python installée en salle de TP (version 3.11).

— (INIT) Vous avez 12h de projet en autonomie (sans enseignant·e). Si vous bloquez, ou que le sujet du projet n'est pas clair, contactez l'enseignant·e par mail.

— Enfin je rappelle que le plagiat est une **fraude** qui est traitée en conseil de discipline et passible d'interdiction d'examen. Par ailleurs, l'utilisation d'IA génératives, telles que ChatGPT ou Copilot est considérée comme une fraude et sera traitée de la même façon.
Faites preuve de bon sens : le but de la SAÉ est d'évaluer **votre** capacité à coder en Python sur un petit projet, pas celle de l'IA.

1.2 Contenu du rapport

J'attends un **très court** rapport (1 à 2 pages) au format **Markdown**. Le rapport contiendra obligatoirement les 4 parties suivantes :

— les choix que vous avez faits
  → choix sur les parties du sujet où je vous ai laissé de la liberté,
  → choix techniques qu'il vous semble pertinent de mentionner ;
— les difficultés que vous avez rencontrées et les solutions que vous avez trouvées ;
— ce que vous pensez avoir appris dans le projet ;
— l'état d'avancement du projet au moment du rendu s'il n'est pas fini.

1.3 Vue d'ensemble

Le but du projet est de travailler avec l'API RESTful Pokémon[1], en Python, afin de récupérer quelques informations utiles et produire des statistiques (plus ou moins utiles).

Vous produirez des fiches aux formats Markdown et HTML, afin de présenter vos résultats de façon agréable à lire.

On veut réaliser les deux outils suivants.

`pokefiche`. Le but de cet outil est de produire une fiche de synthèse sur un Pokémon, aux formats Markdown et HTML.

Cette fiche indiquera des informations disponibles sur le Pokémon : sa taille, son poids, ses types et ses statistiques. De plus, on affichera une image du Pokéman (sprite).

Notre objectif final est d'avoir script qui prend en entrée l'identifiant d'un Pokémon dans l'API et qui produit une page HTML.

`pokestats`. Le but de cet outil est de produire une page web affichant des statistiques sur plusieurs Pokémons. Par exemple, on pourra trier les Pokémons par taille et afficher leurs points de vie de base pour déterminer si les plus grands ont le plus de points de vie.

2 Préliminaire : conversion Markdown → HTML

Dans les deux parties du projet, vous aurez besoin de convertir des fichiers markdown en HTML.

En vous aidant de la bibliothèque `markdown`, vous écrirez une fonction `convert` dans un fichier `md_to_html.py`, prenant en argument un nom de fichier Markdown et un nom de fichier HTML en convertissant le premier en le second. Écrire cette fonction dans un fichier à part vous permet de l'importer depuis les fichiers des deux autres projets :

```
from md_to_html import convert
```

De plus, vous ferez en sorte que votre fichier `md_to_html.py` puisse aussi être utilisé comme script pour convertir un fichier en ligne de commande. Par exemple (d'autres options sont possibles), on pourrait vouloir invoquer le script de la façon suivante : `python3 md_to_html.py input.md output.html`

2.1 Attentes

— La fonction `convert` peut être importée et utilisée depuis un autre module.
— Le script `md_to_html.py` est fonctionnel.

1. https://pokeapi.co/

3 Fiche sur un Pokémon

L'objectif de cette partie est d'avoir un script Python qui génère automatiquement une fiche HTML décrivant un Pokémon.

Par exemple, pikachu a le numéro 25 dans la base de données PokeAPI. Je veux donc pouvoir générer une fiche sur ce Pokémon :

— en exécutant la commande `python3 pokefiche.py 25` ;
— et/ou via l'appel de fonction `fiche_pokemon(25)` depuis l'interpréteur, en ayant préalablement importé la fonction depuis votre fichier.

3.1 Fonctions à implémenter

J'attends de vous que vous implémentiez les fonctions suivantes :

`download_poke(id: int) -> dict` . Cette fonction prend un identifiant de Pokémon en argument et télécharge les données sur ce Pokémon via l'API REST https://pokeapi.co/api/v2.

La documentation de cette API se situe à l'adresse https://pokeapi.co/docs/v2.

Les données seront renvoyées sous la forme d'un dictionnaire.

`poke_to_md(data: dict, filename: str) -> None` . Cette fonction prend un dictionnaire représentant Pokémon et un nom de fichier, et génère un fichier Markdown représentant toutes les informations sur le Pokémon de façon structurée (titres, listes, etc). On devra notamment trouver son poids, sa taille, ses types, ses statistiques et une image.

Cette fonction ne renvoie rien.

`fiche_pokemon(id: int) -> None` . Cette fonction utilise les deux fonctions précédentes, ainsi que la fonction `convert` que vous aurez implémentée dans le module `md_to_html`, pour générer automatiquement une fiche Markdown et une fiche HTML à partir d'un identifiant PokeAPI.

Cette fonction de renvoie rien, mais pourra afficher des informations à l'utilisateurice, si ça vous semble utile, pour indiquer que tout s'est bien passé.

De plus, faites en sorte que votre fichier puisse être utilisé comme un script pour générer une page depuis le terminal. Par exemple, quelque chose comme `python3 pokefiche.py 25` devrait gérer les deux fichiers demandés.

3.2 Format du résultat

Le format du fichier Markdown et du fichier HTML est laissé assez libre. L'essentiel est de présenter les données de façon pertinente. En particulier, utilisez les listes, titres, etc. à bon escient.

Il n'est pas non plus nécessaire de fournir une feuille de style avec le fichier HTML. Ça n'est pas l'objectif de la ressource.

En revanche, il est attendu que la page HTML soit du HTML valide. Regardez bien le résultat produit par la bibliothèque `markdown` et ajoutez ce qui manque.

3.3 Attentes

— Vous faites correctement la requête sur l'API Pokémon.
— Les fonctions demandées sont toutes implémentées et fonctionnent.
— Le fichier peut être utilisé comme un script.
— Le résultat présente bien les données.

4 Statistiques de base

La deuxième partie du projet consiste à faire des statistiques sur les Pokémons ou d'autres éléments présents dans la base de donnée. Je vous laisse libres de choisir la ou les statistiques qui vous intéressent le plus. Je vous donne ci-dessous quelques idées en vrac, mais l'idée est le laisser parler votre imagination.

Par exemple :
— On pourrait trier tous les Pokémons par taille pour les organiser en trois catégories : grands, moyens et petits, puis comparer la moyenne des points de vie dans chaque catégorie. On saura ainsi si les grands Pokémons ont plus de points de vie ou pas.
— On pourrait faire des statistiques sur la proportion de Pokémons de chaque type dans les différentes régions de l'univers Pokémon.
— On pourrait faire un classement des attaques les plus répandues parmi tous les Pokémon de la base de données.
— On pourrait compter le nombre d'attaque de chaque type élémentaire.
— On pourrait faire un classement similaire à ma première suggestion mais en comparant l'expérience gagnée en battant un Pokémon avec son poids.
— On pourrait faire des statistiques sur les caractéristiques des baies.

4.1 Fonctions à implémenter

J'attends de vous que vous implémentiez les fonctions suivantes :

`get_dataset(...) -> ...` . Cette fonction télécharge, au format JSON, toutes les données que vous avez choisi de récupérer dans le cadre de votre projet sous la forme d'un dictionnaire ou d'une liste.

Comme vous êtes libres ici, soignez la documentation de la fonction !

`compute_statistics(...) -> ...` . Cette fonction calcule des statistiques « intéressantes » sur le jeu de données.

Comme vous êtes libres ici, soignez la documentation de la fonction !

`dataset_to_md(dataset: dict, filename: str) -> None` . Cette fonction prend le jeu de donnée au format JSON téléchargé depuis l'API Pékomon, et produit un fichier Markdown présentant les statistiques que vous avez calculées.

Comme précédemment, soignez la présentation (titre, sous-titres, etc). Cette fonction ne renvoie rien.

`infos_locales(...) -> None` . Cette fonction utilise toutes les fonctions précédentes pour télécharger un jeu de données et construire un fichier Markdown, et un fichier HTML présentant ces données et leur statistiques.

Cette fonction de renvoie rien, mais pourra afficher des informations à l'utilisateurice, si ça vous semble utile, pour indiquer que tout s'est bien passé.

4.2 Attentes

— Vous faites correctement vos requêtes sur l'API.
— Les fonctions demandées sont toutes implémentées et fonctionnent.
— Vous avez fait l'effort de calculer quelques stats intéressantes.
— Vous avez documenté vos fonctions, et en particulier la fonction qui calcule des statistiques.
— Le fichier peut être utilisé comme un script (`python3 stats.py` voire plus compliqué si vos fonctions sont paramétrables).
— Le résultat présente bien les données.

5 Pour les initiaux / pour aller plus loin

Vous avez plus de temps dédié pour le projet, il est normal que vous ayez un petit peu plus de travail à faire.

5.1 [Obligatoire pour les initiaux] Traductions

Pour beaucoup d'informations, l'API propose une traduction dans plusieurs langues. Dès que c'est possible, affichez les traductions en français à l'aide de l'API.

Par exemple, les statistiques de base des Pokémon dans la première partie son traduisibles en français.

5.2 [Optionnel pour les motivé·es] Cache

Télécharger des données depuis l'API est lent. Pour gagner du temps, le document d'accompagnement au projet propose d'implémenter un cache. Implémentez ce cache.

5.3 Attentes

— [Obligatoire] Tous ce qui peut être traduit l'est.
— [Optionnel] Toutes les requêtes utilisent le cache → la deuxième fois qu'on fait la requête, la réponse est instantanée.