Projet

A. Introduction

Comme présenté lors du premier cours, l'examen consiste à apporter une amélioration au projet développé au cours du premier quadrimestre pendant les labos. Ce projet sert de base à l'examen. Cette base est nécessaire mais pas suffisante pour réussir.

Le projet consiste en l'analyse de fichier de log à l'aide de script python dans le but de déceler d'éventuels problèmes. Vous trouverez à la suite de ce document la première partie de l'énoncé de ce projet. Deux autres parties seront ajoutées dans les prochaines semaines.

Concernant la présence au labo, elle n'est pas prise en compte pour les points de l'examen CEPENDANT **aucune aide ne vous sera accordée en dehors des heures de labo** (et de théorie prévue à cet effet). De plus, l'aide sera prioritairement donnée aux étudiants régulièrement présents.

Bon travail !

B. Examen

Modalités

Les modalités complètes vous seront transmises à la fin du quadrimestre avant la session d'examen. Voici déjà un aperçu :

- L'examen est un examen écrit de 2h sur machine (de l'école sur une session spéciale examen).
- Vous devez vous munir d'une clé USB contenant votre projet de base (et rien d'autre).
- Votre code peut être commenté.
- Au plus tard à la fin des 2h d'examen, vous devrez déposer votre projet complété dans le dossier indiqué.
- Vous pourrez vous munir d'une (et une seule) feuille de note manuscrite (recto-verso) ainsi que de feuilles de brouillon (vierges).

Lors de l'examen, un ordinateur de l'école avec session exam vous est fourni. Voici quelques infos utiles concernant ces sessions examen :

- Aucun accès à internet
- Python 3.10
- Les éditeurs suivants sont disponibles : VS Code, PyCharme, IDLE, Anaconda (Spyder)

Vous seront également fournis (en version électronique sur ces PC) :

- L'énoncé d'examen
- La doc officielle et complète de Python (en anglais)
- Tous les tests unitaires nécessaires à tester le projet (ceux de base + ceux concernant les nouvelles fonctionnalités à développer)

Remarques :

Seuls les projets respectant le nommage demandé et remis en temps et en heure seront corrigés. Les autres seront considérés comme PP (pas présenté).

La correction du projet se fait via les tests unitaires et le linter. Il est donc impératif que le projet respecte les règles de clean code et fonctionne entièrement.

Le projet de base seul ne suffit pas à réussir l'examen mais il est nécessaire. C'est-à-dire que si le projet de base ne passe pas les tests unitaires correspondant, le projet est automatiquement en échec.

Pour qu'une fonctionnalité soit validée, touts les tests unitaires correspondants doivent être réussis.

C. Projet [Partie 1]

Qu'est-ce qu'un fichier de log ?

« En informatique, un fichier log permet de stocker un historique des événements survenus sur un serveur, un ordinateur ou une application. Ce "journal" présenté sous la forme d'un fichier, ou équivalent, liste et horodate tout ce qui se passe. Les informations contenues permettront ensuite de mieux comprendre les usages et résoudre les erreurs. »
source : https://blog.hubspot.fr/marketing/fichier-log#:~:text=En%20informatique%2C%20un%20fichier%20log,tout%20ce%20qui%20se%20passe.

Il existe un nombre quasi infini de type de fichier de log. Certains peuvent être très courts mais la plupart font plusieurs centaines de millier de lignes. Donc même si certains peuvent être analysés à la main, il devient vite impensable de demander à une personne de les feuilleter à la main. Dès lors, il est très intéressant de créer des scripts capables d'analyser les choses pour nous ou d'au moins nous aiguiller sur la bonne direction à prendre.

Pour le moment, nous allons travailler avec un fichier nommé « syslog». Il s'agit d'un fichier qui log tous les processus qui sont lancés sur un système (dans notre cas, une machine virtuelle). Nous allons créer plusieurs fonctions qui nous permettrons de ressortir des informations que nous considérons comme pertinentes pour surveiller ce système.

Voici à quoi peut ressembler une ligne de ce fichier :

Oct 28 10:31:27 kali systemd[1]: Starting Login Service...

Plusieurs informations sont présentes sur cette ligne :

- La date (avec le mois écrit en 3 lettres (en anglais) et le jour en 2 chiffres)
- L'heure,
- Le hostname,
- Le programme,
- L'id du processus (pas toujours présent notamment dans le cas d'un programme kernel)
- Un message explicatif (peut être composé de sous messages séparés par des « : »)

Pour la suite, nous parlerons parfois de ligne de log « bien formée », il s'agit d'une ligne respectant cette syntaxe.

Découpe du projet en plusieurs fichiers

Pour plus de clarté, il vous est demandé de ne pas mettre toutes les fonctions dans un seul et même fichier mais plutôt de les séparer dans différents fichiers et d'avoir un fichier principal qui importe les fonctions depuis ces autres fichiers.

Commencez par créer un dossier qui contiendra tous les fichiers de votre projet. Renommez ce dossier comme suit : « TIR121_SSBG_NomPrenom » où :

- SS sont les 2 lettres de votre section (IR ou TI)
- B est le chiffre du bloc auquel vous êtes rattaché (1 2 ou 3)
- G est la lettre de votre groupe (A, B, C, D, E, F, G, H ou I, inutile de préciser le demi-groupe)
- NomPrenom est votre nom et prénom.

Attention : un dossier mal nommé risque de ne pas être pris en compte le jour de l'examen !

Dans ce dossier, créez un fichier nommé « main.py ». C'est lui qui contiendra votre code principal c'est-à-dire tous les imports de fonction et un bloc « main » dans lequel se trouvera votre script principal. Créez également un dossier appelé « log ». C'est dans celui-ci qu'il faudra mettre les fichiers de log.

Pour ce qui est des autres fichiers, vous êtes libres de les appeler et de les répartir comme vous le voulez tant qu'ils sont dans le dossier à votre nom et aux formats suivant : .py, .txt ou .log.

Fonctions fournies

La première chose à pouvoir faire est de récupérer les lignes écrites dans le fichier. La lecture dans les fichiers n'est pas matière Q1 donc nous vous fournissons la fonction permettant de le faire. Celle-ci prend en paramètre le chemin du fichier (à partir du dossier courant) à analyser et retourne la liste des lignes qu'il contient.

```
def lines_from_file(path):
    """ Pre : path (str) est le chemin menant au fichier à lire
                (à partir du dossier courant)
        Post : Retourne une liste où chaque élément est une ligne du fichier.
                En cas d'erreur, retourne une liste vide
    """
    try :
        with open(path) as file:
            log = file.readlines()
            return log
    except Exception as e :
        print(e)
        return []
```

Prenez le temps de tester cette fonction

Fonction split

La fonction `split` permet de séparer une chaine de caractères en plusieurs éléments distincts. Elle nous sera très utile pour analyser les lignes de log.

`str.split(sep=None, maxsplit=- 1)` renvoie une liste des mots de la chaîne « str », en utilisant `sep` comme chaîne de délimitation. `maxsplit` correspond au maximum de split effectués (ainsi, la liste aura au plus `maxsplit`+1 éléments). Si `maxsplit` n'est pas spécifié ou -1, alors il n'y a pas de limite sur le nombre de divisions (toutes les divisions possibles sont effectuées).

Par exemple :

```
>>> "Hello World !".split()
['Hello', 'World', '!']

>>> "Hello, I'm Alice, I'm 20".split(',')
['Hello', " I'm Alice", " I'm 20"]

>>> "Hello, I'm Alice, I'm 20".split(',', 1)
['Hello', " I'm Alice, I'm 20"]

>>> "Alice#Bob#Chris#David".split('#')
['Alice', 'Bob', 'Chris', 'David']
```

Documentation : https://docs.python.org/3/library/stdtypes.html#str.split

On démarre

Pour commencer, écrivons quelques fonctions nous permettant d'extraire des informations basiques d'une ligne de log comme l'utilisateur ou le programme.

Implémentez les fonctions suivantes :

```
def get_host(line):
    """ Pre : line est une ligne de log bien formée (str)
        Post : Retourne le hostname
    """
```

```
def get_complete_date(line):
    """ Pre : line est une ligne de log bien formée (str)
        Post : Retourne la date et l'heure sous forme de chaine de caractère
sans changer le format.
    """
```

```
def get_message(line):
    """ Pre : line est une ligne de log bien formée (str)
        Post : Retourne le message de la ligne
          !!! le message peut être composé de sous messages (séparés pas d'autres « : »),
              dans ce cas, il faut tout
    """
```

Remarque : pour les deux fonctions suivantes, prenez en compte le fait qu'il n'y a pas toujours de process_id dans une ligne de log. C'est le cas lorsque le programme est « kernel » mais pas uniquement… Essayez de rester le plus général possible.

```
def get_program(line):
    """ Pre : line est une ligne de log bien formée (str)
        Post : Retourne le nom du programme
    """
```

```
def get_process_id(line):
    """ Pre : line est une ligne de log bien formée (str)
        Post : Retourne le numéro du processus. Si aucun id n'est disponible
(dans le cas d'un kernel par exemple), -1 est retourné.
    """
```

Tests Unitaires

Les tests unitaires pour tester chacune des fonctions demandées vous sont fournis et se basent sur les prés et posts conditions de chaque fonction. Télécharger le fichier « test_main.py » disponible sur moodle et mettez le dans votre dossier principal auprès de votre fichier « main.py ».

Vous pouvez maintenant tester vos fonctions en lançant la commande « pytest » depuis votre terminal (attention d'être dans le bon dossier). Tous les tests sont-ils réussis ? Y a-t-il eu des erreurs ? Si oui, lesquelles ? Corrigez-les s'il y en a !

Attention aux petits malins :

Dans notre cas, les tests unitaires sont assez basiques. En effet, le test de get_host vérifie simplement que l'appel à cette fonction avec « Moi JJ HH:MM:SS host program[process_id]: message » en paramètre retourne bien « host ». Dès lors, si votre fonction ne fait que retourner « host » dans tous les cas, elle passerait ce test. C'est « drôle » mais ça n'est pas très intelligent !

Sachez que l'examen est corrigé par des tests unitaires dont une partie vous est fournie. Cette partie vous permet amplement de tester votre code correctement si vous jouez le jeu. Pour le calcul final des points, des tests supplémentaires seront passés pour être sûr que vous n'avez pas essayé de contourner le système.

A bon entendeur 😉