loganalyser Documentation

Release

Author

May 24, 2016

Contents

CHAPTER 1

Description

loganalyser est un analyseur de journal d'activité de serveur web Apache, réalisé dans le cadre du projet développement informatique en première année du cursus ingénieur de Télécom SudParis.

CHAPTER 2

Table des matières

2.1 Installation

Le projet reprend la structure standard d'un package Python, et peut donc facilement être installé via le `setup.py`. Conçu pour `python3.X`, il convient avant de procéder à l'installation de s'assurer de la version de l'environnement Python courant.

2.1.1 Installation directe

Cette procedure installera le package au sein de Python comme module. Le projet a été conçu pour que les étapes à suivre soient les même que pour tout autre module.

Il est possible de se placer, avant d'initier la procedure, dans un environnement Python virtuelle `venv` si l'environnement courant n'est pas approprié.

```
$ git clone https://github.com/Gingerbreadz/ProjetInfo2016
$ cd ./ProjetInfo2016
$ python setup.py install
```

2.1.2 Installation dans pip

Une procédure alternative permet d'installer le projet comme package `pip` et ainsi de plus facilement le désinstaller. Le projet n'étant pas disponible sur les dépots PyPi, l'installation se déroule comme suit.

```
$ git clone https://github.com/Gingerbreadz/ProjetInfo2016
$ cd ./ProjetInfo2016
$ python setup.py sdist
$ pip install ./dist/loganalyser-0.0.1.tar.gz
```

La désinstallation peut alors être faite avec un `pip uninstall` comme pour tout autre package pip. En cas de difficultés à identifier le nom du package, `pip freeze` permet de lister tout les package pip installés dans l'environnement Python courant.

2.2 Usage

L'utilisation du programme nécessite la possession du fichier de log que l'on souhaite analyser. Une fois le paquet installé, se fait directement depuis la ligne de commande de la façon :

```
$ loganalyser f [n] [o]
```

Paramètres

- f : Chemin fichier de logs.
- n : (Optionel) Nombre de ligne à afficher
- o : (Optionel) Fichier de sortie.

Exemple

```
$ loganalyser "/chemin/vers/fichier/de/log" 6 "/chemin/fichier/output"
```

2.3 Tests

Les tests sont disponible directement sous `/tests/` et executable comme un simple script Python une fois le programme installé. Les tests utilisent des fichiers de log disponibles sous `/res/`. Pour exectuer un test depuis le dossier parent du projet :

```
$ python ./tests/fichierdetest.py
```

2.4 Loganalyser package

2.4.1 Diagnostique module

Implementation du Diagnostique.

class `loganalyser.diagnostique.`**`Diagnostique`**(*token_dictionary*, *regexp_dictionary*, *n=5*, *nomatchcount=0*)

Bases: `object`

Classe instanciant le diagnostique, qui contient les résultats d'analyse et les fait.

`__init__`(*token_dictionary*, *regexp_dictionary*, *n=5*, *nomatchcount=0*)

Constructeur de classe. Un diagnostique est initialisé à partir de tokens.

Parameters

- **`token_dictionary`** (*dict*) – dictionnaire de token
- **`regexp_dictionary`** (*dict*) – dictionnaire de regexp
- **`n`** (*int*) – nombre de ligne à afficher (5 par défaut)
- **`nomatchcount`** (*int*) – nombre de ligne n'ayant pas matchés (0 par défaut)

`get_indices_top`(*liste*)

Permet de trier une liste nous permettant de récupérer des valeurs triées de nos données à l'affichage.

Parameters **`liste`** (*list*) – une liste

Returns indices des valeurs que l'on souhaite afficher dans l'ordre de leurs futur affichage

Return type list

`get_topfiles`(*stat*)

Ordonne la liste des top files et s'assure de sa configuration afin d'obtenir un affichage lisible des résultats.

Parameters **`stat`** (*dict*) – dictionnaire de statistiques

Returns Liste des strings organisées.

Return type list

get_topreferrers(*stat*)
Ordonne la liste des top referrers et s'assure de sa configuration afin d'obtenir un affichage lisible des résultats.

Parameters **stat** (*dict*) – dictionnaire de statistiques

Returns Liste des strings organisées.

Return type list

get_topvisitors(*stat*)
Ordonne la liste des top visitors et s'assure de sa configuration afin d'obtenir un affichage lisible des résultats

Parameters **stat** (*dict*) – dictionnaire de statistiques

Returns Liste des strings organisées.

Return type list

get_topuniqueresponses(*stat*)
Ordonne la liste des top unique responses et s'assure de sa configuration afin d'obtenir un affichage lisible des résultats.

Parameters **stat** (*dict*) – dictionnaire de statistiques

Returns Liste des strings organisées.

Return type list

get_attack(*attack*)
Ordonne la liste des potentiels attaques et s'assure de sa configuration afin d'obtenir un affichage lisible des résultats.

Parameters **attack** (*dict*) – dictionnaire d'attaque.

Returns Liste des strings organisées.

Return type list

get_report()
Ordonne les donnée issues des statistiques et des analyses, prépare pour l'affichage finale.

Returns tableau des lignes de résultats à partir des dictionnaires

Return type list

_Diagnostique__analyse()
Analyse les tokens par groupe selon certains motifs.

Parameters

- **self.token_dict** (*dict*) – dictionnaire de token
- **self.regexp_dict** (*dict*) – dictionnaire d'expression régulière

Returns Dictionnaire contenant le rapport des attaques subit

Return type dict

_Diagnostique__statistique()
Effectue des calculs statistiques sur les token.

Parameters **self.token_dict** (*dict*) – dictionnaire de token

Returns Dictionnaire contenant les statistiques

Return type dict

2.4.2 Fichier module

Sert à intéragir avec les fichiers.

class loganalyser.fichier.**Fichier**(*filepath*)
Bases: object

Classe abstraite interface pour fichier caractérisé par :

- son nombre de ligne
- son contenu
- son chemin d'accès
- si il est read-only ou non

__init__(*filepath*)
Constructeur de classe. Un fichier est initialisé à partir de son chemin d'accès

Parameters **filepath** (*str*) – chemin d'accès du fichier

lireligne(*noligne*)
Retourne la ligne n d'un fichier

Parameters **noligne** (*int*) – numero de la ligne voulu

Returns ligne n du fichier instancié

Return type str

fermerfichier()
Ferme le fichier pour libérer des ressources

class loganalyser.fichier.**FichierDeLog**(*filepath*)
Bases: *loganalyser.fichier.Fichier*

Classe instanciant des fichiers de log caractérisé par :

- son nombre de ligne
- son contenu
- son chemin d'accès
- si il est read-only ou non

decouperligne(*noligne*)
Decoupage syntaxique de la n-ieme ligne pour séparer les différents token

Parameters **noligne** (*int*) – Numéro de ligne

Returns Liste contenant les différents champs découpés.

Return type list

class loganalyser.fichier.**FichierRegExp**(*filepath*)
Bases: *loganalyser.fichier.Fichier*

Classe instanciant des fichiers d'expressions régulières caractérisé par :

- son nombre de ligne

- son contenu
- son chemin d'accès
- si il est read-only ou non

decouperligne(*noligne*)
Decoupage syntaxique de la n-ieme ligne pour récupérer les regExp

Parameters **noligne** (*int*) – Numéro de ligne

Returns Liste contenant les différents champs découpés.

Return type list

class loganalyser.fichier.**FichierRapportTextuel**(*filepath*)
Bases: *loganalyser.fichier.Fichier*

Classe instanciant le rapport textuel caractérisé par :

- son nombre de ligne
- son contenu
- son chemin d'accès
- si il est read-only ou non

__init__(*filepath*)
Constructeur de classe. Un fichier est initialisé à partir de son chemin d'accès

Parameters **filepath** (*str*) – chemin d'accès du fichier

ecriretexte(*data*)
Ecrit les lignes en entrée à la fin du fichier

Parameters **data** (*list*) – numero de la ligne voulu

2.4.3 Token module

Module token Ce sont les classes qui sont utilisées pour caractériser les différents champs de log. A l'instanciation de chacune des classes correspondant à un champ, la vérification du type de la donnée est effectuée et lève une erreur si le type n'est pas le bon.

class loganalyser.token.**Token**(*value*, *istypeok*)
Bases: object

Classe abstraite interface pour token caractérisé par : - sa donnée - sa sévérité

__init__(*value*, *istypeok*)
Constructeur de classe. Un fichier est initialisé à partir de son chemin d'accès

Parameters

- **value** (*str*) – donnee du token e.g. "127.0.0.1", "404".
- **istypeok** (*bool*) – booléen rendant autorisant la création du token.

_Token__analyse()
Analyse la donnee contenue dans le token pour obtenir la sévérité de cette donnee. Non implémenté car non-utile.

Returns Retourne la sévérité de la donnee de ce token

Return type int

_Token__verifier_type(*value*)
Vérifie si la donnee peut bien être instanciée sous cette classe de Token.

Parameters **value** (*str*) – valeur de création du token

Returns Retourne la réponse de la vérification

Return type bool

class loganalyser.token.**IP**(*value*)
Bases: *loganalyser.token.Token*

Classe concrète instanciant les token IP, le format attendu étant une adresse ipv4 ou ipv6

_Token__analyse()
Analyse la donnee contenue dans le token pour obtenir la sévérité de cette donnee. Non implémenté car non-utile.

Returns Retourne la sévérité de la donnee de ce token

Return type int

_Token__verifier_type(*value*)
Vérifie si la donnee peut bien être instanciée sous cette classe de Token.

Parameters **value** (*str*) – valeur de création du token

Returns Retourne la réponse de la vérification

Return type bool

class loganalyser.token.**Name**(*value*)
Bases: *loganalyser.token.Token*

Classe concrète instanciant les token Nom, le format attendu étant une chaine de caractères

_Token__analyse()
Analyse la donnee contenue dans le token pour obtenir la sévérité de cette donnee. Non implémenté car non-utile.

Returns Retourne la sévérité de la donnee de ce token

Return type int

_Token__verifier_type(*value*)
Vérifie si la donnee peut bien être instanciée sous cette classe de Token.

Parameters **value** (*str*) – valeur de création du token

Returns Retourne la réponse de la vérification

Return type bool

class loganalyser.token.**Date**(*value*)
Bases: *loganalyser.token.Token*

Classe concrète instanciant les token Date, le format attendu étant JJ/MM/YYYY:HH:MM:SS

_Token__analyse()
Analyse la donnee contenue dans le token pour obtenir la sévérité de cette donnee. Non implémenté car non-utile.

Returns Retourne la sévérité de la donnee de ce token

Return type int

_Token__verifier_type(*value*)
Vérifie si la donnee peut bien être instanciée sous cette classe de Token.

Parameters **value** (*str*) – valeur de création du token

Returns Retourne la réponse de la vérification

Return type bool

class loganalyser.token.**EXT**(*value*)
Bases: *loganalyser.token.Token*

Classe concrète instanciant les token Ext, le format attendu étant un entier

_Token__analyse()
Analyse la donnee contenue dans le token pour obtenir la sévérité de cette donnee. Non implémenté car non-utile.

Returns Retourne la sévérité de la donnee de ce token

Return type int

_Token__verifier_type(*value*)
Vérifie si la donnee peut bien être instanciée sous cette classe de Token.

Parameters **value** (*str*) – valeur de création du token

Returns Retourne la réponse de la vérification

Return type bool

class loganalyser.token.**Method**(*value*)
Bases: *loganalyser.token.Token*

Classe concrète instanciant les token Methode, le format attendu étant l'une des chaines de caractères suivante : GET, HEAD, POST, OPTIONS, CONNECT, TRACE, PUT, DELETE

_Token__analyse()
Analyse la donnee contenue dans le token pour obtenir la sévérité de cette donnee. Non implémenté car non-utile.

Returns Retourne la sévérité de la donnee de ce token

Return type int

_Token__verifier_type(*value*)
Vérifie si la donnee peut bien être instanciée sous cette classe de Token.

Parameters **value** (*str*) – valeur de création du token

Returns Retourne la réponse de la vérification

Return type bool

class loganalyser.token.**URL**(*value*)
Bases: *loganalyser.token.Token*

Classe concrète instanciant les token URL

_Token__analyse()
Analyse la donnee contenue dans le token pour obtenir la sévérité de cette donnee. Non implémenté car non-utile.

Returns Retourne la sévérité de la donnee de ce token

Return type int

_Token__verifier_type(*value*)
Vérifie si la donnee peut bien être instanciée sous cette classe de Token.

Parameters **value** (*str*) – valeur de création du token

Returns Retourne la réponse de la vérification

Return type bool

class loganalyser.token.**Response**(*value*)
Bases: *loganalyser.token.Token*

Classe concrète instanciant les token Réponse, le format attendu étant un entier entre 100 et 599 (compris)

_Token__analyse()
Analyse la donnee contenue dans le token pour obtenir la sévérité de cette donnee. Non implémenté car non-utile.

Returns Retourne la sévérité de la donnee de ce token

Return type int

_Token__verifier_type(*value*)
Vérifie si la donnee peut bien être instanciée sous cette classe de Token.

Parameters **value** (*str*) – valeur de création du token

Returns Retourne la réponse de la vérification

Return type bool

class loganalyser.token.**Byte**(*value*)
Bases: *loganalyser.token.Token*

Classe concrète instanciant les token Octet, le fomat attendu étant un entier

_Token__analyse()
Analyse la donnee contenue dans le token pour obtenir la sévérité de cette donnee. Non implémenté car non-utile.

Returns Retourne la sévérité de la donnee de ce token

Return type int

_Token__verifier_type(*value*)
Vérifie si la donnee peut bien être instanciée sous cette classe de Token.

Parameters **value** (*str*) – valeur de création du token

Returns Retourne la réponse de la vérification

Return type bool

class loganalyser.token.**Referer**(*value*)
Bases: *loganalyser.token.Token*

Classe concrète instanciant les token Referer

_Token__analyse()
Analyse la donnee contenue dans le token pour obtenir la sévérité de cette donnee. Non implémenté car non-utile.

Returns Retourne la sévérité de la donnee de ce token

Return type int

_Token__verifier_type(*value*)
Vérifie si la donnee peut bien être instanciée sous cette classe de Token.

Parameters **value** (*str*) – valeur de création du token

Returns Retourne la réponse de la vérification

Return type bool

2.4.4 Outils module

Sert à l'implémentation de notre classe Dictionnaire, qui étend la classe dict de Python, et y ajoute les opérations qui nous sont utiles sur les dictionnaires.

class loganalyser.outils.**Dictionary**(*keylist*)
Bases: dict

Extension de la classe dictionnaire. Cette classe possède comme attributs supplémentaires: - La liste des clefs du dictionnaire

__init__(*keylist*)
Constructeur de classe. Un dictionnaire est initialisé vide à partir de la liste des clefs

Parameters **keylist** (*list*) – Liste des clefs du dictionnaire.

keys()
Retourne les clefs du dictionnaire.

Returns Liste contenant les clefs du dictionnaire.

Return type list

addentry(*entry*)
Ajoute au dicitonnaire une nouvelle valeur dans chacunes de ses clefs à partir d'une liste.

Parameters **entry** (*list*) – Liste contenant les valeurs pour chacune des clefs

getentry(*entrynumber*)
Retourne la liste contenant les valeurs de chaques clefs pour un index donné.

Parameters **entrynumber** (*int*) – index de l'entrée.

Returns Liste contenant les valeurs de chaques clefs pour le même index.

Return type list

itemtoentrynumbers(*item*)
Retourne l'index d'une valeur dans le dictionnaire.

Parameters **item** (*str*) – valeur recherchée.

Returns Liste contenant les index associés à la valeur d'entrée.

Return type list

CHAPTER 3

Crédits

Projet conduit par Jeremy Venin, Clément Aubry, Antoine Tadros, Anatole Lefort dans le cadre du programme d'enseignement de Telecom SudParis.

l

Symbols

A

B

D

E

F

G

I

K

L

M

N

R

T

U