Descriptif du dossier Scripts-Python

Rignak

29 juin 2017
v1.0

Table des matières

1 Introduction

Ce document liste les différents programmes présents dans le dossier. Il décrit leurs fonctionnalités, leurs limites ainsi que les formats des données en entrée et en sortie. Il indique les conditions nécessaires à leurs exécution. Enfin, il apporte des indications quand aux évolutions futures des fichiers.

Première partie
Danbooru

Dans ce dossier se trouvent des programmes ayant pour dénominateur commun l'utilisation du site internet Danbooru. Cinq sous-dossiers sont présents :

- AddTags : permet d'afficher à l'écran des images correspondant à des tags particuliers et d'y entrer des additions ou des soustractions ;
- NotDan : permet de chercher des images n'étant probablement pas présentes sur Danbooru ;
- PicturesUtil : permet de travailler sur des images à l'aide de l'application Waifu2x, de détecter la présence de bruit JPEG. Danbooru est utilisé dans un but de téléchargement d'images.
- Post : permet la génération de fichier aidant à l'upload d'images. De part la mise en forme des tuples (url, tags) et la correction d'erreur de frappe.
- StatsDan : permet l'obtention de données d'upload journalier.

Dans un cadre général, nous utilisons l'hypothèse suivante : un fichier contenant les identifiants du compte danbooru est présent à la racine. Ce fichier, nommé Danboory_Codes.txt contient les informations sous la forme :

```
api_key xxx
login yyy
```

où « xxx » désigne la clef api et « yyy » le nom d'utilisateur. Sauf mention contraire, ce fichier est nécessaire pour l'exécution des programmes.

2 AddTags

2.1 AddTags

2.1.1 Hyphothèses

Ce programme fonctionne sous l'environnement Windows 7 mais n'a pas de raison sur un système autre que Windows. Les 10000 tags les plus populaires de Danbooru sont listés dans le fichier « tags.txt ». Une ligne y représente un tag.

2.1.2 Comportement

Lors de l'exécution, le programme utilise Danbooru pour télécharger des images dans la mémoire vive. Il commence par recherche les urls des images puis télécharge leur contenu. L'application affiche à l'utilisateur la progression de cette étape et une estimation de sa terminaison.

Une fois cette étape achevée, les images s'affichent une par une dans la visionneuse de photos Windows (notés ci-après VPW). L'utilisateur est alors invité à rentrer une chaîne de caractère représentant les tags à ajouter ou à supprimer de l'image. La validation de cette chaîne de caractère provoque la fermeture de la VPW puis l'ouverture de l'image suivante. Les informations renvoyées à l'utilisateur sont le numéro de l'image dans la série, l'identifiant sur Danbooru, ainsi qu'une estimation de la date de terminaison de la série. L'utilisateur peut terminer la série en validant la chaîne « return ». L'utilisateur peut passer à l'image suivante en validant la chaîne « pass ». Si un tag correspond à une clef dans le dictionnaire known_tags, ce tag sera remplacé par la valeur de l'entrée. Cela permet de minimiser le temps d'écriture des tags par l'utilisateur. Les tags entrés par l'utilisateur sont comparés à la liste du « tags.txt » et supprimés s'il n'y a pas de correspondance. La possibilité d'ajouter un tag par erreur est ainsi réduite.

Nous conseillons de paramétrer une souris avec les macros correspondant aux entrer dans le know_tags. La vitesse atteinte pour une série de 1000 images avant sept tags différents à rajouter fut de près de deux secondes par images.

2.1.3 Évolutions futures

— Passer le dictionnaire known_tags dans un fichier .json et non hard-codé dans le programme.
— Donner à l'utilisateur la possibilité de changer la plage d'identifiant, les tags et le nombre d'image. Cela n'a pas été fait en raison de l'utilisation générique (sur une seule série de tags) du programme.
— Utiliser les threads afin d'accroître la vitesse de récupération des informations préalables à l'entrée des tags. Les threads ne sont en revanche pas nécessaire lors de l'application des tags, du fait de la limitation du nombre de requête « post » par Danbooru. On pourrait cependant imaginer que la validation d'un tag entraîne son ajout simultané par le biais d'un thread.
— Ouvrir Danbooru_Codes.txt au sein d'un « with » à des fins purement esthétiques.

2.1.4 Implémentation

2.1.4.1 Pré-traitement

L'application récupére les informations de Danbooru_Codes.txt et créé le dictionnaire known_tags. La fonction main est lancée.

2.1.4.2 class Sample

Cette classe permet la description d'une image et des informations correspondantes nécessaire.

def __init__ Création de l'instance de Sample. Ses attributs sont :
— _Id : une chaîne de caractère décimaux indiquant l'identifiant de l'image sur Danbooru ;
— _data : les données binaires représentants l'image ;
— _tags : une chaîne de caractères indiquant les tags présents sur l'image ;
— _adds : une chaîne de caractères indiquant les tags à rajouter sur Danbooru.

def InputTags Affiche l'image dans la VPW et demande à l'utilisateur de rentrer des tags. Si l'affichage de l'image échoue (il peut s'agit d'un .webm par exemple) le tag renvoyé est « pass ». La VPW est fermée à l'issue de la fonction.

def InputTags Recherche sur Danbooru les tags actuellement présents sur l'image et y ajoute les tags entrés par l'utilsateur. Si l'utilisateur a rentré le tag « pass », ces actions ne sont pas effectuées. La fonction renvoie le status_code de la requête. Il est égale à 200 si la requête a réussi.

def GetTag Recherche sur Danbooru les tags actuellement présents sur l'image.

def VerifyTags Vérifie si les mots contenus dans la chaîne de caractère sont présents dans la liste des tags ou dans le dictionnaire known_tag. Les mots inconnus sont supprimés de la chaîne de sortie.

2.1.4.3 def ListUrl

Effectue une requête sur Danbooru pour obtenir les urls d'images correspondant aux tags d'entrée.

2.1.4.4 def ListImgs

Créé une liste d'instance de Sample à partir des données fourni par ListUrl. Indique la date de terminaison prévue.

2.1.4.5 def main

Initialise les variables de la recherche : plage d'identifiant, tags et nombre d'image.
Créé la liste des urls (sous la forme d'une concaténation de requêtes).
Utilise ces urls pour obtenir le contenu des image et créer les instances de Sample.
Invite l'utilisateur à rentrer les tags à ajouter ou supprimer.
Effectue les requêtes pour modifier les tags sur Danbooru.

2.2 DelFav

2.2.1 Hyphothèses

Il n'y a pas d'autres hypothèse que l'existence du Danbooru_Codes.txt.

2.2.2 Comportement

Ce programme utilise AddTags pour afficher des images. L'utilisateur est invité à rentrer le caractère « - », décidant alors que l'image sera supprimée de ses favoris. L'url de l'image est alors ajoutée dans un fichier texte, permettant alors de rapidement y accéder (via une extension comme « Multilink » sur le navigateur Chrome) et de supprimer l'image de ses favoris. Le fichier .txt ainsi créé est nommé « to_del.txt »

2.2.3 Traitement

Traitement similaire à celui d'AddTags.py, sans utiliser la partie mise-à-jour des tags.

2.2.4 Évolutions futures

Les favoris auraient pu être mis-à-jour directement via l'api. Un traitement manuel a été choisi pour pouvoir avec une deuxième validation des modifications. Ce programme est très rarement utilisé (développé pour une utilisation temporellement unique) et ne présente pas un intérêt justifiant un travail d'amélioration.

2.3 tags

2.3.1 Hypothèses

Aucune hypothèse n'est faite. L'existence du Danbooru_Codes.txt n'est pas nécessaire.

2.3.2 Comportement et traitement

Le programme effectue, sur les 500 premières pages du wiki, la recherche des tags. Chacune de ces pages est censée recueillir 20 tags, permettant ainsi de lister les 10000 tags les plus utilisés. Ces tags sont écrit dans le fichier tags.txt à raison d'un tag par ligne.

2.3.3 Évolutions futures

Aucune évolution n'est prévue. Il fut utilisé avec succès une fois. La conservation de la liste ainsi créé

3 NotDan

3.1 PixivNotDan

3.1.1 Hypothèses

Ce programme utilise, outre les bibliothèques usuelles, la bibliothèque AppPixivApi afin de pouvoir utiliser l'api de Pixiv.

Une limitation importante existe lors de la vérification de la présence sur Danbooru. En effet, pour que le résultat soit positif, il ne faut pas uniquement que l'image soit

effectivement sur le site, mais qu'elle ait été psotée depuis Pixiv. Cela signifie que si une image identique a été postée par le biais de Twitter ou de NicoSeiga, l'image sera considérée comme n'étant pas présente sur Danbooru. Notons que la compression avec perte imposée par Twitter change le md5 de l'image et ne permet pas d'utiliser de tels méthodes pour vérifier la présence de l'image. Cette méthode semble en revanche pouvoir être mise en œuvre pour détecter les faux-positifs provenant de NicoSeiga. Le calcul du md5 étant nécessaire (il ne fait pas parti des méta-données de pixiv), cette piste n'a pas été explorée.

3.1.2 Comportement

Lors de l'exécution, cinq modes sont proposés à l'utilisateur :

— 0 : requêtes sur pixiv et écriture d'un fichier .json ;
— 1 : lecture d'un fichier .json provenant du mode 1 et vérification sur danbooru ;
— 2 : fusion ou séparation de plusieurs fichiers .json ;

Le but de ces modes est d'aboutir à l'extraction d'informations concernant les images stockées sur pixiv et de vérifier leur présence sur Danbooru. Les informations stockées dans les fichiers .json sont les suivantes :

— t : les tags caractérisant l'image sur Pixiv.
— u : le lien direct vers la première image à une taille moyenne. Dans le cas d'une image contenant du contenu explicite, ce lien est remplacé par une censure R18. Dans le cas ou l'image ait été supprimé, le lien est également différent.
— n : le nombre d'images sur l'œuvre.
— d : la date de création, une chaîne de caractère de la forme "2000-01-30T20 :00 :00+09 :00"
— s : le nombre de fois où cette image a été placée en favori.
— r : le ratio taille/largeur de l'image.

3.1.3 Évolutions futures

Le nom de l'artiste pourrait être recherché afin de le coupler avec la liste des artistes retirés de Danbooru. Ce nombre est relativement faible et est considéré comme un moindre mal face à une nouvelle extraction des données.

Une ré-extraction des données correspondant à la plage 43M à 46M sera exécutée afin de supprimer certaines vérifications sur la présence du ratio h/w dans les données.

3.1.4 Implémentation

3.1.4.1 Pré-traitement

Initialisation de la blacklist et récupération des identifiants pour Danbooru. Déclaration des données nécessaire aux requêtes afin de les sortir des threads.

3.1.4.2 def PixIsOnDan

Prend pixivId en argument, un entier représentant l'identifiant sur Pixiv. Effectue la requête pixiv :pixivId sur Danbooru. Cette opération ne permet pas de s'asssurer de l'inexistence de l'image sur Danbooru (il faudrait utiliser IQDB pour s'approcher de cette affirmation) mais fournit un premier discriminant qui est d'autant plus fort que l'image

est ancienne. En cas d'échec de la requête, la manœuvre est tentée jusqu'à quatre fois supplémentaires.

En cas de réusite de la requête, le fichier de résultat est mis-à-jour (il est donc écrit simultanément à l'exécution).

3.1.4.3 def IndividualWritePixiv

Lancé par le mode 0.

Prend en argument un entier représentant l'identifiant sur Pixiv ainsi que le score minimum.

Met à jour le dictionnaire courant en rajoutant les informations relative à l'image i, via une requête à l'api de Pixiv. Les conditions de mise à jour sont :

— que l'image soit marquée comme une illustration, non un manga ou un ugoira ;
— que le nombre d'illustrations composant l'image soit inférieur à 10 (afin d'éviter les mangas) ;
— que le score de l'image soit supérieur à celui indiqué par l'utilisateur.

Les erreurs sont explicitement passées sous silence par le biais de la variable « find » qui les compte.

3.1.4.4 def IndividualFromDic

Lancé par le mode 1.

Prend en argument un entier représentant l'identifiant sur Pixiv ainsi que la plage de score.

Effectue la recherche sur Danbooru si les conditions suivantes sont réunies :

— le score de l'image est inclus dans la plage de valeur ;
— l'image n'est pas supprimée ou cachée par son auteur ;
— si le ratio de l'image est connu, il doit être inférieur à 2.5 ;
— l'image n'est pas décrite par une tag de la blacklist ;
— si des tags sont souhaités par l'utilisateur, l'image doit satisfaire à ces tags.

Le ratio n'est pas infaillible puisque des image d'un ratio très important sont curieusement indiquées comme ayant un ratio de 1.0.

3.1.4.5 def PixivNotDanbooru

Ce programme gère la création des threads ainsi que la demande à l'utilisateur des différents arguments. Le nombre de thread simultané est fixé par « limit_active » et est augmenté de « nb » jusqu'à atteindre la valeur maximum. Dans le cas d'une connexion filaire, ces nombres sont 200 et 50. Dans le cas d'une connexion 3G, ces nombres sont passés à 100 et 10.

Dans tous les modes, l'utilisateur est invité à choisir un score minimum.

La date de terminaison estimée est indiquée à chaque progression de 0.1

mode = 0 Ce mode permet l'extraction de Pixiv et l'écriture du .json. Le programme effectue donc la connexion à Pixiv en utilisant les codes indiqués dans le Pixiv_Codes.txt. Le login sera effectué toute les 3500 secondes.

L'utilisateur est invité à choisir un identifiant maximum et un nombre d'image à étudier.

Le thread est lancé pour la fonction IndividualWritePixiv.

mode = 1 Ce mode permet la vérification sur Danbooru.

L'utilisateur est invité à choisir un score maximum ainsi que d'éventuels tags à imposer à la recherche.

Le thread est lancé pour la fonction IndividualFromDic.

3.1.4.6 def ReadJSON

Permet de fusionner les dictionnaires contenus dans plusieurs .json et de lancer PixivNotDanbooru en mode 1.

3.1.4.7 def SplitJSON

Permet de lire les dictionnaires contenus dans plusieurs .json et de les placés dans un fichier avec un nom standardisé. Initialement prévu pour spliter des fichiers, il permet également de les fusionner.

Cette fonction est destructrice dans le sens où elle peut détruire les fichiers d'entrés s'ils ont les mêmes noms que les fichiers standardisés.

3.1.4.8 def getR18_URL

Cette fonction était utilisé au commencement du programme. Elle permet d'obtenir l'adresse d'une image non révisée à partir de la date de sa mise en ligne.

Elle n'est pas utilisée à l'heure actuelle mais est laissé en cas de développement ultérieurs.

3.2 YandereNotDan

3.2.1 Hypothèses

Le navigateur doit être lancé avant le lancement du programme. La version 6 et 7 fonctionnent sous Windows 7. Pour utiliser ce navigateur, la bibliothèque stem doit être installée.

Ce programme ne nécessite pas l'utilisation du Danbooru_Codes.txt.

3.2.2 Comportement

Ce programme permet de vérifier l'existence d'image sur Danbooru. Cette vérification est établie par l'application IQDB. Il s'agit donc d'une vérification plus fiable que celle de PixivNotDan, mais également plus lente. Le rythme est d'une seconde par image environ. IQDB bannissant l'adresse IP après 25 utilisations dans un cours laps de temps, nous utilison Tor afin de changer cette adresse après 24 vérifications. Les résultats sont écrit dans un fichier HTML contenant la liste des images dont on présume qu'elles ne sont pas sur Danbooru. Les images sont récupérées sur le site yande.re et correspondent à des tags fournis par l'utilisateur.

3.2.3 Évolutions futures

Un comportement inattendu est survenu lors du passage à Tor 7.0 et demande des investigations sur une connexion internet plus performante. La vitesse de l'application, bien que très faible par rapport aux résultats obtenus sur Pixiv, sont tout à fait convenables pour l'utilisation hebdomadaire qui en est faite.

On peut toutefois imaginer que l'obtention de l'url des samples soit effectuée en dehors des threads effectuant la requête IQDB. De même, il serait possible d'exécuter la recherche initiale sur des threads différents bien que cette dernière possiblité ne propose pas un gain important.

3.2.4 Implémentation

3.2.4.1 renew_tor

Permet de changer l'adresse IP contenue dans le header du paquet IP sortant. Le but premier de Tor est donc de permettre l'anonymat. Un sous-produit de ce logiciel est a possibilité de contourner les mécanismes de bannissement lors d'une utilisation trop intensive d'une application en ligne.

3.2.4.2 IsOnDan

Vérifie si IQDB renvoie un résultat positif. Si le résultat est une erreur différent d'un flood, ou qu'aucune image ne correspond, le programme renvoie False. Sinon, il renvoie True.

3.2.4.3 AlreadyFounds

Permet de vérifier si les résultats obtenus dans le traitement préalable on déjà été détectés. Auquel cas il n'est pas nécessaire d'exécuter quelque opération que ce soit pour l'image en question.

3.2.4.4 CreateListAllURL

Créé, a partir d'une chaîne de tags et d'un nombre maximum d'image, de créer la liste des urls des images qui seront vérifiées.

3.2.4.5 URLSample

Permet, à partir de l'adresse d'une image (page générale de l'image contenant, en autre, les tags), d'obtenir l'adresse du sample. Le sample est utilisé de sorte à vérifier les conditions de taille et de poids d'image imposées par IQDB.

3.2.4.6 YandereNotDanbooru

Demande à l'utilisateur des tags. S'il rentre plusieurs recherches séparées par des espaces, plusieurs recherches seront effectuées. L'utilisateur peut également rentrer des tags qui s'appliqueront à chaque recherches. Il peut choisir une limite du nombre d'image à rechercher.

La seconde étape est la recherche préliminaire des urls des images. Celles-ci peuvent être obtenues par série de 1000, il s'agit donc d'une étape rapide. Les threads sont ensuites lancées en série de 24. Chacun entraîne la recherche de l'url du sample et la requête IQDB. L'estimation de la date de terminaison est indiqué à la fin de chacune de ces séries, avec le nombre d'image repérées. De même, à la fin d'une série de 24, l'adresse IP est renouvelée.

3.2.4.7 IndividualYandereNotDanbooru

Effectue la requête IQDB sur une image particulière écrit une ligne dans le fichier de sortie si le résultat est négatif. Il s'agit de la fonction appelée par les threads.

4 PicturesUtil

4.1 DownloadDanbooru

4.1.1 Hypothèses

Il n'y a pas d'autres hypothèses que l'existence du Danbooru_Codes.txt.

4.1.2 Comportement

Ce programme permet de télécharger des images de danbooru dans le sous-dossier « result ».

4.1.3 Évolutions futures

Pas d'évolution prévue. Son utilisation est très annecdotique. On pourrait imaginer lui fournir des threads afin de télécharger de multiples image simultanément. La requête de ListPictureswithTag peut être effectuée avec l'api, plutôt que via une url.

4.1.4 Implémentation

4.1.4.1 ListPicturesWithTags

Prend en entrée une chaîne de caractère et un entier, représentants respectivmeent les tags de la requête ainsi que le nombre maximum d'image à rechercher. Retourne la liste urls menant aux données des images.

4.1.4.2 DownloadPictures

Prend en entrée une url, un nom et le chemin absolu d'un sous-dossier. Télécharge l'image de l'url avec le nom indiqué dans le sous-dossier.

4.1.4.3 Launch

Demande des tags et un nombre à l'utilisateur. Télécharge les images en donnant comme nom un entier.

4.2 GateDivine

4.2.1 Hypothèses

Présence des fichiers WhiteFunctions.py et WaifuFunction.py dans le dossier.

4.2.2 Comportement

Télécharge les images indiquées dans le fichier files.txt. Ces images sont censées provenir du site divine-gate.net. Il s'agit donc d'image PNG avec transparence, ayant subit une réduction du nombre de couleur. Le but est donc de rétablir une qualité correcte. On utilise pour cela Waifu2x.

4.2.3 Évolutions futures

Le programme est dans sa version finale.

4.2.4 Implémentation

4.2.4.1 Pré-traitement

Ouverture du files.txt et lecture de son contenu. Téléchargement de chaque image et traitement sur Waifu2x. Les paramètres de ce traitement sont : multiplication de la taille par un facteur 2 (sur chaque côté), donc surface par quatre, et réduction de bruit sur High. L'image est ensuite réduite à sa taille initiale, de 1024x1024.

4.2.4.2 Download

Télécharge l'image dans le sous-dossier « images ». Retourne le chemin complet de la dite image.

4.3 NoiseFunctions

4.3.1 Hypothèses

Aucune hypothèse

4.3.2 Comportement

Cette fonctions permet la détection de bruit JPEG par le calcul de l'intensité de la différences de deux images. Cette détection est améliorée par l'utilisation d'un filtre passe-haut qualibré de sorte à garder les contours et les bruits jpeg.

4.3.3 Évolutions futures

Le programme est en version finale et ne présente aucune raison d'être modifié.

4.3.4 Implémentation

4.3.4.1 np_from_img

Prend en argument une image (PIL.Image) et retourne la matrice correspondante. L'éventuelle transparence est remplacée par un fond blanc. La transparence du canal alpha est gérée.

4.3.4.2 img_from_np

Prend en argument une matrice et retourne l'image correspondante.

4.3.4.3 Grayscale

Prend en argument une matrice représentant une image RGB et retourne la matrice représentant cette même image en nuance de gris.

4.3.4.4 Intensity

Prend en argument une matrice et retourne l'intensité moyenne (c'est à dire la moyenne des valeurs contenues dans la matrice).

4.3.4.5 Filter

Prend en argument une matrice ainsi qu'un filtre (une matrice, généralement 9x9) et applique un filtrage par convolution.

4.3.4.6 MakeFilter

Prend en argument un nombre, flottant ou non. Retourne un filtre de détection de contours dont la puissance dépend de l'argument. Plus il est grand, plus le filtre sera fort.

Ce filtre est un filtre passe-haut.

4.3.4.7 DetectJPG

Prend en argument le chemin d'une image et le numéro du mode utilisé. Mode 1 correspond à test et est destiné au dessin d'un graphe. Mode 0 correspond à un fonctionnement normal.

L'image va d'abord être transformée en nuance de gris. On lui applique ensuite un filtre de détection de contour de facteur 6. L'image va ensuite être réenregistrée au format JPEG avec des qualités entre 74 et 96. Les mêmes opérations de transformation en nuance de gris et de filtrage sont effectuées. L'image résultante est soustraite à celle issue du premier filtrage et nous calculons l'intensité de cette différence. Nous postulons que cette intensité connaît deux minimums locaux : le qualité 100 pour des raisons évidentes, et la qualité de l'enregistrement initiale de l'image. Nous sommes donc en mesure de découvrir quelle était la qualité d'enregistrement de l'image avant son acquisition.

4.3.4.8 FirstLocalMinimum

Prend en argument le résultat du mode 0 de DetectJPG et cherche le minimum local parmi les valeurs de qualité jpeg 75, 80, 85, 90 et 95. Ces choix semblent être les valeurs privilégiées par les tiers. D'une part car une qualité inférieure à 70 est très mauvaise et provoque l'apparition d'artefacts particulièrement laids, mais aussi car il s'agit de multiple de 5. De plus, la recherche sur des valeurs inférieur est susceptible de produire des faux-positifs.

4.3.4.9 Test

Cette fonction effectue les opérations sur des fichiers pré-préparés afin de montrer la différence entre les graphes d'intensité sur une image de qualité 100 et son équivalent bruité. Ces graphes permettent de se rendre compte de la pertinence de la technique utilisée pour détecter le bruit JPEG.

4.4 WaifuFunctions

4.4.1 Hypothèses

La seule hypothèse est la présence de Noisefunctions.py dans le dossier.

4.4.2 Comportement

Ce programme permet de faire des appels à l'application Waifu2x, d'enlever la transparence d'une image (y compris celle du canal alpha) et de convertir l'image au format jpeg.

4.4.3 Évolutions futures

Le programme est en version finale, aucun changement n'est prévu.

4.4.4 Implémentation

4.4.4.1 LaunchWaifu

Prend en entrée le chemin d'une image et appelle l'application Waifu2x. L'opération est ré-effectuée en cas d'erreur.

4.4.4.2 Remove_Transparency

Copie l'image en argument sur un fond blanc en tenant compte du canal alpha.

4.4.4.3 Convert_to_jpg

Converti une image du format png au format jpeg avec une qualité de 100.

4.4.4.4 Unnoise

Utilise NoiseFunctions afin de détecter la qualité d'enregistrement de l'image. Si cette qualité est strictement inférieur à 95, Waifu2x est utilisé pour enlever ce bruit.

4.5 WhiteFunctions

4.5.1 Hypothèses

Nécessite la présence de DownloadDanbooru et de WaifuFunctions (implictiement de NoiseFunctions).

4.5.2 Comportement

Contient de nombreuses fonctions permettant de travailler sur des images. Ce programme fut utilisé pour créer des images de background pour Google et Wikipedia. Pour cela, le fond doit être blanc, les dimensions doivent respecter des dimensions précises. Dans le cas où l'image se prolonge en hors-champ, il faut déterminer quel est ce côté et l'orienter vers le côté de la fenêtre. Nous enlevons ensuite les blocs, c'est-à-dire les zones colorés séparées de la figure principale par des zones blanches. La transparence est également supprimée.

4.5.3 Évolutions futures

Ce programme est en version finale. Aucun évolution n'est prévue.

4.5.4 Implémentation

4.5.4.1 class bloc

Un bloc est une ensemble de coordonnées. Il s'agit des zones de pixel colorés contigües.

def __init__ Créé le bloc contenant la coordonnée indiquée.

def CreateBloc Pour déterminer le bloc nous utilisons l'algorithme suivant :
— On initialise la liste des pixels à explorer.
— On prend le premier élément de cette liste et on regarde les pixels adjacents. Si ces pixels ne sont pas blancs, on les rajoute à la liste des pixels à explorer s'ils ne sont pas déjà dans le bloc.

Dans le pire des cas, cette boucle se termine lorsque la totalité des pixels contenant l'image est explorée.

4.5.4.2 class IMG

def __init__ Créé l'instance de la classe IMG. Ses attributs sont :
— ._name : le chemin du fichier contenant l'image ;
— ._im : l'image (PIL.Image) contenue dans le fichier ;
— ._array : la représentation matricielle (matrice tri-dimensionnelle) de l'image ;

— ._height : la hauteur, en nombre de pixel, de l'image ;
— ._width : la largeur, en nombre de pixel, de l'image ;
— ._ratio : le rapport largeur/hauteur ;
— ._blocs : la liste des blocs contenus dans l'image
— ._border : une liste de quatre booléen, vrais si l'image sur prolonge en hors-champ sur le côté correspondant.

def GetBlocs Detecte tout les blocs composant l'image. Pour cela, nous regardons tous les pixels composant l'image. Si ce pixel est blanc ou fait partie d'un bloc précédemment visité, nous passons au pixel suivant. Sinon, nous déterminons l'étendu du bloc contenant le pixel. Il s'agit de la partie du programme la plus lente.

def TrimBloc Détecte le bloc contenant le bloc le plus grand. Tout les autres blocs sont remplacés par des pixels blancs.

def Reduce Supprime les limites de l'image sont avancées afin de ne pas tenir compte des lignes ou des colonnes entièrement blanches. Pour cela nous effectons quatre parcours de l'image :
— de haut en bas et de gauche à droite, afin de déterminer la limite à gauche ;
— de gauche à droite et de haut en bas, afin de déterminer la limite en haut ;
— de bas en haut et de gauche à droite, afin de déterminer la limite à droite ;
— de gauche à droite et de bas en haut, afin de déterminer la limite en bas.
L'image est ensuite recadrée en gardant une marge de 1 pixel.
Seule l'image est modifiée.

def Crop Le matrice d'entrée est recadréee selon les dimensions indiquées. Chaque case de la matrice bi-dimensionnelle est ensuite réduite à trois éléments, pour garder le mode RGB.
Seule la matrice est modifiée

def Remove_Transparency Copie l'image en argument sur un fond blanc en tenant compte du canal alpha.
Seule l'image est modifiée.

def Thumbnail Réduit l'image suivant la fonction de PIL.Image. Les dimensions indiquées par défaut sont celle correspondant à une image pour Google.
Seule l'image est modifiée.

def Sym_Y Effectue une symétrie verticale de l'image.
Seule la matrice est modifiée.

def ImToArray Met à jour les valeurs de hauteur, de largeur, de ratio et la matrice à partir de l'image.

def ArrayToIm Met à jour les valeurs de hauteur, de largeur, de ratio et l'image à partir de la matrice.

def Show Affiche l'image à l'écran en utilisant le WPV.

def Save Sauvegarde l'image au format indigué. La qualité est de 100 dans le cas d'un format jpeg.

def GetBorder Permet de savoir sur quel côté l'image se prolonge en hors-champs. La liste booléenne présente la configuration suivante : [gauche, droite, haut, bas] et vaut vrai si l'image se prolonge en hors-champs. Cette prolongation est considérée comme présente si le nombre de pixel non-blanc sur la colonne (respectivement la ligne) correspondante est plus grand que le dixième de la hauteur (respectivement la largeur) de l'image.

4.5.4.3 def Full

Lance l'application à partir du téléchargement des images depuis Danbooru, images correspondant à des tags hard-codés.

4.5.4.4 def onFolder

Lance l'application sur des images pré-téléchargées.

4.5.4.5 def Launch

Effectue la transformation des images, sans modification de celles d'origine puisque le résultat est dirigé vers un sous-dossier.

Si le mode choisi est « full », l'application va télécharger les images correspondant aux urls au fur et à mesure de l'exécution.

Le traitement se déroule de la façon suivante :

— Génération de l'instance de IMG à partir du fichier (téléchargée si besoin).
— Suppression e la transparence de l'image, puis mise-à-jour de la matrice.
— Suppression des lignes et colonnes exclusivement blanches, puis mise-à-jour de l'image.
— Sauvegarde de l'image dans le sous-dossier de résultat.
— Si l'image est d'une taille suffisamment faible pour pouvoir passer dans la version online de Waifu2x, nous estimons le bruit résultant d'un potentiel enregistrement jpeg.
— Si l'image est d'une taille importante (supérieur à 1500 pixel sur l'un de ses côtés), nous le réduisons.

  Ces deux étapes présentent un intérêt pour la réduction du nombre de blocs. En réduisant la taille de l'image (qui devra de toute façon être redimensionnée pour s'afficher entièrement sur Google ou Wikipédia) nous pouvons réduire le temps de calcul de la détection de blocs. De plus, la présence de bruit jpeg entraînent la détection de pixels initialement blancs dans les blocs.

— Détection des blocs.
— Suppressions des blocs à l'exception du plus grand.
— Détection des bordures.
— Décision de la cible de l'image (Wikipédia ou Google). La cible est Wikipédia si la hauteur est au moins plus de trois fois plus grande que la largeur. Sinon, ce la hauteur est plus grande que la largeur, l'image est destinée à Google. L'image est redimensionnée à 174x676 pour Wikipedia et à 549x706 pour Google. Mise-à-jour de la matrice.
— Changement du nom suivant le côté de la bordure. Si une bordure est à gauche et qu'il n'y en a pas à droite, l'image subit une symétrie verticale. En effet, dans le cas d'une image Google, elle sera placée sur le côté droit. Mise-à-jour de l'image dans le cas d'une symétrie.
— Changement de nom pour nommer la version finale de l'image. Affichage d'une date de terminaison estimée. Cette date est peu fiable du fait de l'hétérogénéité du temps de traitement.

Notons qu'un certain nombre d'images intermédiaires est créé.

5 Post

5.1 DanbooruPost

5.1.1 Hypothèses

Deux fichiers .txt doivent exister un le dossier « ../files » : error.txt et banned_artist.txt. Dans le cas d'un mode 3, le fichier trim.txt est également nécessaire.

5.1.2 Comportement

Ce programme permet la génération de fichiers facilitant l'upload d'images sur Danbooru. Le fichier final utilisé est un .html.

5.1.3 Évolutions futures

Ce programme a atteint sa version finale et n'a pas vocation a être modifié.

5.1.4 Implémentation

5.1.4.1 Pré-traitement

Définition des listes de fichiers .txt et .html qui seront utilisés.
Création du dictionnaire des erreurs et des artistes bannis.
Demande à l'utilisateur du mode utilisé. Ces modes sont les suivants :
— 1 : Génération des fichiers .html à partir des fichiers .txt ;
— 2 : Génération des fichiers .txt à partir des fichiers .html ;
— 3 : Délétion des images dont l'adresse est contenue dans le fichier trim.txt ;
— 4 : Demande du nombre d'image à supprimer de chaque fichier et délétion ;
— 5 : Comptage des images dans dans les fichiers .txt.

5.1.4.2 def CorrectorSample

Corrige les urls suivant des patterns connus :
— g.hitomi -> a.hitomi : il s'agit du changement d'anciennes urls ;
— i.hitomi -> a.hitomi : pour des raisons identiques ;
— sample -> jpeg : il s'agit d'adresses d'images sur yande.re ;
— :large -> :orig : il s'agit d'adresse d'images sur twitter.

5.1.4.3 def IsBanned

Si une chaine de tags contient un tags blacklisté, retourne True. Retourn False sinon. Le fichier banned_artist.txt contient les différents noms, tous sur une ligne différente. Cette fonction permet d'éviter d'uploader des images interdites sur Danbooru.

5.1.4.4 def ReplaceAll

Corrige les erreurs de frappes, ainsi que les incohérences de noms entre yande.re et Danbooru.

Les couples de correction se situent tous sur des lignes différentes. Le éléments de ces couples sont séparés par une tabulation.

5.1.4.5 def DelLine

Supprime les premières lignes de chaque fichier .html. Le nombre de lignes supprimées dépend des informations envoyées par l'utilisateur.

5.1.4.6 def TrimHTML

Supprime les lignes des .html dont les liens sont présents dans le trim.txt.

5.1.4.7 def GenerateTXT

Recréé les fichiers sources en .txt à partir des informations contenues dans les .html. Il existe deux fichiers .txt pour chaque .html. Le premier contient les urls des images et le deuxième les tags. Les urls sont écrite à la suite, sur des lignes différentes Les tags sont écrits à la suite, séparés par une ligne blanche ou contenant des caractères quelconques.

5.1.4.8 def GenerateHTML

Ecrit les fichiers .html. Le format est le suivant : « <A HREF="' + *url* + '"> ' *tags* + '
</br> » suivi d'un changement de line.

5.1.4.9 def CountLine

Affiche le nombre de ligne dans chacun des fichiers .txt contenant les liens ainsi que le nombre de ligne total.

6 stats_dan

6.1 stats_dan

6.1.1 Hypothèses

Nécéssite la présence du fichier Danbooru_Codes.txt

6.1.2 Comportement

Permet d'obtenir des données relatives au upload journaliers, sur une période de 30 jours. Ces données sont :

— le nombre d'images (uploadées et approuvées) ou (en pending) ;
— le nombre d'images supprimées ;
— le nombre d'images uploadées par l'utilisateur
— le nombre d'images uploadées par l'utilisateur mais supprimées par la suite.

6.1.3 Évolutions futures

Aucune

6.1.4 Implémentation

Les recherches correspondantes sont exécutées, il y a cinq requête par jour.

Deuxième partie
DownloadWebsite

6.2 DownloadWebsiteIMG

6.2.1 Hypothèses

Aucune

6.2.2 Comportement

Ce programme télécharge tout les images (format .png ou .jpg) du site internet indiqué par l'utilisateur.

6.2.3 Évolutions futures

Un sous-dossier Images doit se trouver à la racine.

6.2.4 Implémentation

6.2.4.1 def EndTime

Effectue les calculs permettant d'obtenir la date estimé de terminaison du programme.

6.2.4.2 def GetDataLink

Requiert les données d'une page internet, extrait tout les liens présents et les renvoie. Si la page ne renvoie rien, la tentative est refaite jusqu'à quatre fois supplémentaires.

6.2.4.3 def GetDataWebsite

Lance les opération de parcours de site. 100 threads sont utilisés simultanément. Les informations de progression sont affichées toutes les 10 secondes. La première boucle s'exécute tant qu'il reste des liens à visiter ou que le nombre de thread en cours est différent de l'initial. Cette deuxième condition permet d'assurer de ne pas sortir de la boucle dans l'éventualité ou des threads de parcours sont encore en cours, permettant d'augmenter le nombre de lien à visiter à leur issue. Une deuxième boucle permet de sortir lorsqu'il n'y a toujours aucun liens à visiter mais qu'après cinq minutes, le nombre de thread actif est toujours supérieur à celui initial. On considère en effet que les threads sont bloqués et que toute les images extrayables l'ont été. Les threads sont ensuite lancés un par un jusqu'à atteindre la limite de thread actif (fixée à 100 threads de parcours).

6.2.4.4 def IndividualGetData

Obtient la liste des href présents sur la page et l'analyse de sorte à savoir si les images doivent être téléchargées. Ces conditions varie selon le site extrait : le pattern est adapté. Lance un thread de téléchargement si nécéssaire.

6.2.4.5 def Download

Télécharge une image en effectuant des remplacements hard-codés spécifiques au site internet téléchargé.

Troisième partie
JapaneseApp

6.3 Questions

6.3.1 Hypothèses

Aucune

6.3.2 Comportement

Permet l'ouverture d'une fenêtre affichant un mot et invitant l'utilisateur à en rentrer un autre. Les correspondances sont choisis par l'utilisateur selon les en-têtes des fichiers de ressources. L'invite se nettoie à chaque pression sur le bouton « check » ou « pass ». Le bouton « break » permet de lancer la terminaison du programme.

6.3.3 Évolutions futures

Inconnues pour l'instant, elles seront évaluées en fonction des besoins futurs.

6.3.4 Implémentation

6.3.4.1 Pré-traitement

Demande à l'utilisateur le fichier de travail. La première ligne du fichier de travail doit contenir les en-têtes décrivant les informations contenue par chaque ligne. L'utilisateur choisi ensuite sur quels en-têtes il doit être interrogé. Le premier choix indique ce qui sera indiqué à l'écran (un idéogramme, une traduction français, ...). Le second indique la réponse attendue (transcription en romaji, traduction, ...). Les questions sont ensuite générées.

La fenêtre s'ouvre, le questionnaire peut commencer.

6.3.4.2 class Question

La classe Question possède trois attributs :
— .question : symbole qui devra être traduit/transcrit par l'utilisateur ;
— .answer : symbole attendu par l'utilisateur ;
— .all : toute les informations contenues dans la ligne du fichier.
Les symboles de fin de ligne sont systématiques détruits.

6.3.4.3 class Question_Canvas

def __init__ Génère la fenêtre. Celle-ci est composée de :
— le texte de la question ;
— l'invite où l'utilisateur doit rentrer sa réponse ;
— trois boutons : « Check », « Pass » et « Break ».
La pression de ces boutons lance la fonction correspondante.
La taille de la fenêtre est automatiquement choisie pour être la plus petite possible compte-tenu des élements qu'elle contient.

def Init Initialise le text affiché. Cette fonction est utilisée lors d'un changement de question.

def on_button_check Vérifie si la réponse entrée par l'utilisateur est correcte. La pression de la touche entrée provoque également l'appel de cette fonction. La question est ensuite passée (on peut rester sur la même question en commentant la ligne self._update = True.

def on_button_pass La question est passée. Le score n'est pas modifié.

def on_button_break Termine le programme après avoir fermé la fenêtre et affiché le score atteinte par l'utilisateur.

Quatrième partie
ToDoManager

6.4 ToDoManager

6.4.1 Hypothèses

La version portable de ffmpeg.exe doit être présente dans le dossier. De plus, les commandes étant envoyées au shell windows, ce programme peut ne fonctionner que sur W7.

Le fichier « To Do.txt » doit être présent dans le dossier « E :/Telechargements/Anime/to do », nommé par la suite « local ».

Les fichiers référencés dans le To Do.txt doivent être présents dans le sous-dossier « mp4 » du dossier local.

Les fichiers résultats sont écrits dans le sous-dossier « ok », qui doit donc exister.

Le traitement du fichier se fait à partir de la lecture de la ligne « video », qui doit donc être présent dans le fichier.

6.4.2 Comportement

Le programme lis le To Do.txt et effectue les opérations correspondantes.

Ces opérations se réduisent à l'extraction d'une partie de la vidéo en rajoutant des sous-titres si nécessaire.

6.4.3 Évolutions futures

Ce programme n'a été testé que de manière minimale et ne fonctionnera probablement pas du premier coup.

La question des sous-titres n'est pas encore bien comprise et est appelée à évolué. La gestion de vidéo contenant plusieurs sous-titre est ainsi inconnue.

6.4.4 Implémentation

6.4.4.1 def FromYoutube

Télécharge une vidéo de youtube, au formet mp4 avec la meilleure qualité possible. La ligne du fichier « To DO.txt »indique (dans l'ordre) le nom du fichier destination (sans l'extension), l'url de la vidéo à télécharger de youtube et l'extension (théoriquement mp4 ou mp3). Si l'extension est mp3, ToMP3 est lancé.

6.4.4.2 def CutMP4

Détermine les différents arguments de la ligne du fichier To « Do.txt ». Cette ligne est de la forme suivante : « new_name begin end episode source »avec

— new_name, le nom du nouveau fichier (auquel il faut rajouter l'extension .mp4 ;
— begin & end, les dates de début et de fin de la vidéo, selon le format HHMMSS ou MMSS selon la longueur de la vidéo ;
— episode, le numéro de l'épisode de la vidéo au sein de la série ; il n'est pas utilisé ;
— source, le nom de la vidéo au sein du dossier « ./mp4 ».

Ces arguments permettent l'élaboration d'une commande utilisant ffmpeg.

6.4.4.3 def ToMP3

Utilise ffmpeg pour transformer le .mp4 en .mp3 et le déplace dans le dossier « ok ».

6.4.4.4 def ConvertTime

Les date de début et de fin de clip, au format MMSS si la durée est inférieure à une heure, HHMMSS sinon, sont converties en insérant le signe « : » pour séparer les différentes unités. Cela permet de créer une date que ffmpeg comprend.

6.4.4.5 Traitement

Le programme se résume à la lecture de chaque ligne du To Do.txt et à la création d'une commande envoyée au shell de Windows.

Cinquième partie
WatermarkDengeki

6.5 WatermarkDengeki

6.5.1 Hypothèses

Le fichier files.txt doit exister à la racine.

On considère que chaque image téléchargée est marquée par le watermark produit par le DengekiOnline.

6.5.2 Comportement

Ce programme télécharge chaque image dont l'url est contenue dans le files.txt et enlève le watermark appliqué par le Dengeki Online.

6.5.3 Évolutions futures

6.5.4 Implémentation

6.5.4.1 def InverseFunction

Prend en entrée les arguments suivants :

— r, g et b : trois entiers entre 0 et 255. Ils représentent le contenu d'un pixel, respectivement les couleurs rouges, vertes et bleus.

— wtmk1_px et wtmk2_px : les pixels de deux watermarks, appliqués respectivement sur la luminosité et l'obscurité.

— l : un entier optionnel indiquant la nombre d'itération a effectuer.

Le principe est le suivant : le retro-ingénieuring du watermark nous a permis de déterminer que deux filtres étaient appliqués. L'un fonction de luminosité des pixels, l'autre de leur obscurité. Ils s'appliquent différent suivant les canaux lumineux et selon un facteur de luminosité normalisé entre 0 et 1.

Lorsque la valeur de la couleur est 0, seul le wtmk2 (appliqué à l'obscurité) est utilisé. A l'inverse, lorsque la couleur est 255, seul le wtmk1 (appliqué à la luminosité) est utilisé. Lorsque la couleur est 122, ils sont utilisé chacun avec un facteur 0.5. Nous utilisons la récurrence pour trouver le pixel initial, chaque itération provoquant une modification plus subtil. Cette modification finirait par être nulle si nous travaillions sur des réels, ce qui n'est pas le cas. L'usage d'entiers impose une limite à la récurrence, qui est fixée expérimentalement à 100. Une limite plus grande induit une altération inverse du résultat escompté.

Les pixels de watermark sont obtenus par la recherche d'une image entièrement noire, présentant donc uniquement le wtmk2 et d'un image entièrement blanche, présentant uniquement le wtmk1.

Pour le wtmk2, on utilise en réalité son négatif.

6.5.4.2 def Download

Téléchargement de l'image. S'il échoue, il est retenté jusqu'à sa réussite.

L'image est placée dans le sous-dossier image.

6.5.4.3 def RemoveWatermark

Recrée une image en inversant le watermark si nécessaire. Seule une petite zone, située dans la partie inférieure de l'image, est watermarkée.

6.5.4.4 def onDengeki

Effectue le téléchargement des images et l'application de RemoveWatermark sur chacune.

Sixième partie
Myanimelist

Regroupe deux programmes permettant de dessiner des graphes à partir d'informations extraites sur le site myanimelist.

De par sa récence, les évolutions futures ne sont pas encore déterminées.

7 MakeJSON

7.1 Hypothèses

Aucune.

7.2 Comportement

Ce programme extrait les informations pour chaque anime et les écrit dans un .json. Ces informations sont :

- le nom de l'anime ;
- le nombre d'épisodes composant l'anime (un entier) ;
- le type (TV, OVA, ONA, Movie, Music) ;
- la saison durant laquelle la diffusion a commencé (données incomplètes) ;
- les genres auxquels l'anime appartient (parmi Action, Adventure, Cars, Comedy, Dementia, Demons, Drama, Ecchi, Fantasy, Game, Harem, Hentai, Historical, Horror, Josei, Kids, Magic, Martial Arts, Mecha, Military, Music, Mystery, Parody, Police, Psychological, Romance, Samurai, School, Sci-Fi, Seinen, Shoujo, Shoujo Ai, Shounen, Shounen Ai, Slice of Life, Space, Sports, Super Power, Supernatural, Thriller et Vampire) ;
- les dates de début et de fin de diffusion, au format [DD/MM/AAAA, DD/MM/AAAA] ;
- les données concernant l'horaire et la chaîne de diffusion ;
- la liste des producteurs ;
- la liste des licenceurs ;
- la liste des studios ;
- la source (parmis Unknown, Manga, Game, Light novel, Visual novel, 4-koma manga, Novel, Original et Other) ;
- la durée des épisodes.

Ces informations sont contenues dans un dictionnaire, chacun de ces dictionnaire indexé par l'identifiant de l'anime.

7.3 Implémentation

7.3.1 def GetTuple

Prend en entré l'objet issu de BeautifulSoup et une clef. Returne les valeurs contenues dans la balise « span » d'une balise « div ». Pour trouver cette balise « span », nous utilisons le fait que sa chaîne de caractère est en égalité avec la clef passée en entrée.

Cette balise « span »peut contenir des balises « a »(comprenant dans ce cas des liens hypertextes). L'information a extraire est alors dans la chaîne de caractère du « a ». Sinon, elle est dans la chaîne de caractère du « div » contenant le « span ».

Ces informations sont renvoyées sous forme d'une liste si elles sont mutliples, d'une chaîne de caractère sinon.

7.3.2 def Month2Int

Prend en entré une chaîne de caractère et renvoie une chaîne de caractère. Si la chaîne est le diminutif d'un mois, renvoie le nombre du mois.

7.3.3 def FormatD

Supprime les virgules contenues dans la chaîne et rajoute un zéro si la chaîne ne contient qu'un caractère.

7.3.4 def Date

Transforme la date contenue dans les données de myanimelist à un format unique. Le format de myanimelist diffère en effet suivant les informations connues ou le nombre d'épisodes

7.3.5 def GetTitle

Extrait le titre de l'anime.

7.3.6 def IndividualNb

Extrait les informations correspondants aux clefs passées en entrée et à l'anime correspondant à l'identifiant indiqué. Créé un dictionnaire et la rajoute au dictionnaire global.

7.3.7 def CreateDic

Lance IndividualNb sur les identifiants de 1 à 36000 avec une fenêtre de 20 threads.

8 MakeGraph

8.1 Hypothèses

Aucune.

8.2 Comportement

Génère des graphes basées sur les demandes de l'utilisateur.

L'utilisateur peut rentrer des les conditions sous le format key :value (par exemple « Genres :Mecha »). Des conditions multiples peuvent être effectuées en séparant les conditions individuelles par le caractère « & » (par exemple « Genres :Mecha&Source :Light novel »).

Il peut également choisir le mode de présentation des données : en valeurs absolues ou en pourcentages.

Enfin, il peut choisir d'étudier les anime regardés par une certain usagé de Myanimelist.

8.3 Implémentation

8.3.1 def UserListCompleted

Extrait les identifiants des anime contenus dans la liste « Watched » de l'utilisateur référencé par son nom sur Myanimelist.

8.3.2 def LoadDic

Lis le fichier .json de la base de donnée de Myanimelist.

8.3.3 def ReduceDic

Réduit le dictionnaire d'anime aux seuls identifiants renvoyés par UserListCompleted.

8.3.4 def CheckCondition

Prend en entrée un tuple et une entrée du dictionnaire (correspondant donc à un anime). Ce tuple est composé d'une clef et d'une valeur. Si ce couple correspond à ce qui est contenu dans l'entrée, renvoie True : l'anime satisfait la conditions. Sinon, renvoie False.

8.3.5 def ReduceOnConditions

Prend en entrée le dictionnaire global et un ensemble de conditions. Réduit le dictionnaire aux seules entrées satisfaisant toutes les conditions.

8.3.6 def CountYear

Compte le nombre d'entrées du dictionnaire passé en entrée et les différencies selon l'année de début de diffusion.

8.3.7 def GetData

Demande à l'utilisateur les différentes variables de création du graphe et génère les données et les légendes. Les opérations de générations de données sont :

- lecture du dictionnaire correspondant à la base de donnée et génération du dictionnaire global ;
- réduction du dictionnaire sur les conditions indiquées par l'utilisateur ;
- réduction du dictionnaire pour un utilisateur particulier ;
- comptage des anime satisfaisant aux conditions ;
- génération des ratios si l'utilisateur demande une représentation en pourcentages ;
- génération du graphe.

8.3.8 def GetProportion

Prend en entrée deux listes d'entiers. Retourne une liste des rapports entre les entiers contenus dans ces listes.

8.3.9 def Plt

Génère le graphe à partir des informations extraites par GetData.