Documentation

Documentation ImportExportParametres.py

Le but de ce fichier est de fournir des fonctions permettant de lire et d'écrire les fichiers nécessaires au bon fonctionnement de l'application, à savoir le fichier de sauvegarde des paramètres et le fichier de sauvegardes des formats.

Liste et description des fonctions :

exportInCSV(parametres) :

Paramètres : un dictionnaire de paramètres.
Écrit le dictionnaire dans le fichier "parametres.csv"
Ne retourne rien.

importFromCSV() :

Ne prend pas de paramètre.
Ouvre le fichier "parametres.csv" et renvoie son contenu dans un dictionnaire.

importFormat() :

Ne prend pas de paramètre.
Lit le fichier "formats.csv" et renvoie ses informations dans un dictionnaire.

addFormat(format_infos) :

Format_infos : dictionnaire contenant les informations d'un nouveau format.
Enregistre le nouveau format dans le fichier format.csv.
Ne retourne rien

createFormatFile() :

Ne prend pas de paramètre.
Crée le fichier de sauvegarde des formats s'il n'existe pas.
Ne retourne rien

Documentation TraitementFichiers.py

Le but de ce fichier est de fournir une interface pour toutes les vues souhaitant utiliser le modèle. Ce fichier peut gérer le chargement, la sauvegarde, l'entraînement d'un modèle, la génération à partir du modèle ainsi que la création de statistiques et de graphiques sur l'entraînement et les données.

Liste et description des fonctions :

saveModel(save_path) :

Save_path : chemin complet de fichier (string)
Sauvegarde les paramètres du modèle à ce chemin.
Ne retourne rien.

loadModel(load_path, user_param) :

Load_path : chemin complet de fichier (string)
User_param : dictionnaire de paramètres
Charge le modèle à partir des paramètres lus dans le fichier et crée un nouveau modèle avec ces paramètres et le dictionnaire.
Retourne les paramètres lus dans le fichier

getModelParametres() :

Ne prend pas de paramètre.
Renvoie les paramètres du modèle sous forme de dictionnaire.

lire_fichier(nom) :

Nom : chemin complet de fichier (string)
Renvoie le contenu du fichier.

ecrire_fichier(nom, donnees) :

Nom : chemin complet de fichier (string)
Écrit les données dans le fichier.
Ne retourne rien.

get_rnn_parameters(parametres) :

Paramètres : dictionnaire de paramètres
Renvoie une liste contenant certains des paramètres passés en entrée.

conversion_en_csv(writeAddr, readAddr, name_in) :

writeAddr : chemin de dossier d'écriture (string)
readAddr : chemin de dossier de lecture (string)
Name_in : nom de fichier (chaîne de caractère)
Convertit le fichier MIDI du dossier de lecteur en fichier CSV et l'écrit dans le dossier d'écriture.
Ne renvoie rien.

conversion_en_mid(name_out, csv_content) :

Name_out : nom complet de fichier (string).
Csv_content : liste de chaînes de caractères.
Écrit le contenu de la liste dans le fichier.
Ne renvoie rien.

check_conversions(parametres, format_infos) :

Paramètres : dictionnaire de paramètres
Format_infos : dictionnaire d'informations sur les formats.
Crée les dossiers de stockage nécessaires et vérifie que toutes les conversions mid => csv et csv => format on été faites, effectue les conversions dans le cas contraire.
Ne renvoie rien.

get_input_liste(parametres) :

Paramètres : dictionnaire de paramètres.
Lit tous les fichiers formats existants et affiche optionnellement des statistiques dessus.
Renvoie une partie de leur contenu sous forme de liste de chaînes de caractères.

train(parametres, is_model, queue, finQueue, format_infos) :

Paramètres : dictionnaire de paramètres
Queue : objet de type Queue
finQueue : objet de type Queue
Format_infos : dictionnaire de formats
Appelle la fonction d'entraînement du modèle.
Crée optionnellement des graphiques sur les données récoltées pendant l'entraînement.
Ne retourne rien.

genereMorceaux(parametres) :

Paramètres : dictionnaire de paramètres.
Appelle la fonction de création de morceaux du modèle et enregistre les morceaux générés.
Ne retourne rien.

getDate() :

Ne prend pas de paramètres.
Renvoie la date sous un format pratique.

saveGraph(graph_path, list_losses, list_acc, parametres) :

Graph_path : chemin du dossier sauvegarde des modèles (string)
List_losses : liste de flottants
List_acc : listes de flottants
Parametres : dictionnaire de paramètre
Crée et enregistre deux graphiques à partir des paramètres d'entrée et les enregistre dans le dossier de sauvegarde des paramètres.
Ne renvoie rien.

Documentation Morceau.py

Le but de cette classe est de parser un fichier au format CSV et d'en extraire les informations relatives aux notes. Cette classe fournit également des méthodes permettant les conversions depuis CSV dans des formats personnalisables et vice-versa.

Liste et description des méthodes de la classe :

__init__(self, path):

Path : chemin du fichier (string)
Déclare tous les paramètres de classe et lance la parsing en appelant la méthode get_info
Ne renvoie rien.

get_info(self):

Ne prend pas de paramètre.
Appelle la méthode de récupération du header et la méthode de parsing.
Ne renvoie rien.

get_header(self, lines):

Lines : liste de string
Lit la première ligne de la liste et récupère les informations
Renvoie toutes les lignes sauf la première

get_tracks(self, lines):

Lines : liste de string
Parcourt chaque ligne et appelle une méthode en fonction des informations présentes sur une ligne.
Ne renvoie rien

get_time_signature(self, line):

Line : string
Récupère les informations relatives à la signature temporelle présentes dans la ligne.
Ne renvoie rien.

get_key_signature(self, line):

Line : string
Récupère les informations relatives à la key signature présentes dans la ligne.
Ne renvoie rien.

get_smpte_offset(self, line):

Line : string
Récupère les informations relatives au SMPTE offset présentes dans la ligne.
Ne renvoie rien.

get_track(self, numero):

Numero : int
Renvoie une liste contenant toutes les notes présentes dans la piste correspondant au numéro.

arrondi_note(self, time):

Time : int
Renvoie la valeur la plus proche de Time présente dans le dictionnaire des temps des notes.

positionMesure(self, piste):

Piste : liste de string
Parcourt la liste et pour chaque couple d'événements correspondant à une note, ajoute la position de cette note dans sa mesure.
Renvoie la piste modifiée.

getFormat(self, format_infos):

Format_infos : dictionnaire contenant les informations sur un format
Fusionne toutes les pistes du morceau si cela est spécifié dans le dictionnaire et récupère les noms et les contenus des fichiers format à écrire.
Renvoie la liste des noms de fichiers (liste de string) et la liste des contenu (liste de string)

csv_to_format_select(self, numero=None):

Numero : None ou entier
Fusionne toutes les pistes si Numero = None et sélectionne une piste sinon. Applique la numérotation des mesures si nécessaire. Crée les séquences au bon format.
Renvoie une liste de séquences de notes (liste de string) et une liste d'ensemble de valeurs pour chaque élément de note (liste d'ensemble de string).

csv_to_format(self, L):

L : liste de string
Parcourt la liste d'événements de notes et les transforme dans le bon format.
Renvoie une liste de séquences (liste de string) et une liste d'ensemble de valeurs pour chaque élément de note (liste d'ensemble de string).

createNote(self, is_silence, time1, time2, touche, last_touche, velocite1,

velocite2, position):

Is_silence : Booléen précisant si la note est un silence
Time1 : entier codant l'instant de début de la note
Time2 : entier codant l'instant de fin de la note
Touche : entier codant numéro de touche de la note (hauteur de la note)
Velocite1 : entier codant la force appliquée sur la touche lors de l'appui de la touche
Velocite2 : entier codant la force appliquée sur la touche lors du relâchement de la touche
Position : position de la note dans sa mesure
Encode la note selon avec les informations nécessaires précisées dans le format.
Renvoie la note encodée (string)

format_to_csv(self, values):

Values : string contenant des notes au format voulu.
Reconstruit le contenu d'un fichier csv où toutes les notes ont été converties depuis le format choisi en évènements midi-csv.
Renvoie une liste contenant tous les événements midi-csv à écrire (liste de string)

Documentation RNN.py

Le but de cette classe est de créer un modèle de RNN-LSTM de Pytorch, de gérer son entraînement et la génération de morceaux.

Liste et description des méthodes de la classe :

__init__(self, input_list, param_list, ensemble_list, load_param=None):

Input_list : liste de string contenant les séquences sur d'entraînement et de test
Param_list : liste de paramètres
Ensemble_list : liste d'ensemble contenant chacun tous les éléments possibles par élément de note
Load_param : None ou liste de paramètres
Initialise les paramètres de classe avec les données passées en paramètres.
Appelle la méthode prepare().
Ne renvoie rien.

getParametres(self):

Ne prend pas de paramètre.
Renvoie un dictionnaire contenant les paramètres du modèle.

setParametres(self, param):

Param : dictionnaire de paramètres.
Initialise le modèle basé sur les paramètres dans le dictionnaire.
Ne renvoie rien.

prepare(self):

Ne prend pas de paramètre.
Crée les embeddings et initialise le modèle, la couche linéaire et l'optimiseur.
Ne renvoie rien.

distribution_pick(self, vecteur):

Vecteur : liste de flottants codant une concaténation de vecteurs de probabilités.
Crée un vecteur contenant les indices d'éléments choisis en fonction de leurs probabilités.
Retourne ce vecteur (liste d'int)

predict(self, indexes, parametres):

Indexes : liste d'entiers codant les index de chaque élément de la note dans leurs dictionnaires respectifs.
Parametres : tuple de deux tenseurs
Crée une nouvelle liste d'index basée sur les prédictions du modèle.
Renvoie cette liste (liste d'entiers)

sample(self, max_len):

Max_len : int
Crée une séquence de longueur maximale max_len basée sur les prédictions du modèle.
Renvoie cette séquence (string)

pick_batch(self):

Ne prend pas de paramètre.
Renvoie une liste sélection de séquences prises dans les séquences d'entraînements (liste de string).

train(self, nb_epochs, batch_size, queue, finQueue):

Nb_epoch : int codant le nombre d'entraînement sur toutes les séquences
Batch_size : int codant le nombre de séquences dans un batch
Queue : objet de type queue permettant de communiquer avec l'interface
finQueue : objet de type queue permettant de communiquer avec l'interface
Méthode responsable de l'entraînement du modèle en fonction des paramètres. Envoie des informations par la queue et reçoit des informations à travers finQueue.
Retourne la liste des loss (liste de listes de float) et la liste des précisions (liste de float) à chaque epoch de l'entraînement.

generate(self, nombre, duree):

Nombre : int
Duree : int
Produit Nombre séquences de longueur maximale Durée.
Renvoie la liste des séquences générées (liste de string)

split_input(self):

Ne prend pas de paramètre.
Découpe les séquences en listes de séquences d'entraînement et séquences de test.
Renvoie une liste contenant la liste des séquences d'entraînement et la liste des séquences de test (liste de listes de string)

Liste et description des fonctions :

training_file_number_choice(total):

Total : int
Renvoie la valeur entière de 90% de total (int) correspondant au nombre de séquences d'entraînement parmi toutes les séquences.

training_file_choice(liste, nb):

Liste : liste de string
Nb : int
Renvoie une liste composée d'une liste de nb valeurs tirée de liste et d'une liste correspondant au reste des valeurs (liste de listes)

device_choice():

Ne prend pas de paramètre.
Renvoie une string correspondant à l'appareil le plus approprié pour le modèle (GPU si disponible, sinon CPU)

Documentation Application.py

Le but de cette classe est de créer un objet héritant du type Tkinter.Tk servant de conteneur aux différentes vues de l'application. Cette classe permet de gérer les vues de l'application, le modèle utilisé ainsi que la fermeture de la fenêtre.

Liste et description des méthodes de la classe :

__init__(self):

Crée le conteneur qui va afficher les différentes vues de l'application et initialise les paramètres de classe. Crée et initialise un objet par vue existante.
Ne retourne rien.

switch_frame(self,frame):

Frame : string
Permet de changer la vue affichée en fonction de la valeur de frame.
Ne retourne rien.

askWhenClose(self):

Ne prend pas de paramètre.
Vérifie si un modèle si un modèle est en cours d'entraînement et/ou est en cours d'utilisation et n'a pas été sauvegardé par l'intermédiaire d'interactions avec l'utilisateur.
Ne retourne rien.

newModel(self):

Ne prend pas de paramètre.
Crée un nouveau modèle en restaurant quelques paramètres si besoin et en prévenant l'utilisateur que son modèle n'a pas été sauvegardé au besoin.
Ne retourne rien.

saveModel(self):

Ne prend pas de paramètre.
Sauvegarde le modèle en cours d'utilisation (s'il y en a un) à l'adresse choisie par l'utilisateur.
Ne renvoie rien.

loadModel(self):

Ne prend pas de paramètre.
Avertit l'utilisateur s'il n'a pas sauvegardé son modèle et charge le modèle présent à l'adresse rentrée par l'utilisateur.
Ne renvoie rien.

saveParametres(self):

Ne prend pas de paramètre.
Enregistre les paramètres de l'interface/vue Menu et les enregistre dans les paramètres de la classe.
Ne retourne rien.

getParametres(self):

Ne prend pas de paramètre.
Appelle saveParametres().
Renvoie la valeur des paramètres sauvegardés (dictionnaire).

Liste et description des fonctions :

credits_fen():

Ne prend pas de paramètre.
Affiche une simple fenêtre pop-up contenant les crédits des créateurs de l'application.
Ne retourne rien.

about_fen():

Ne prend pas de paramètre.
Affiche une simple fenêtre pop-up contenant quelques informations sur l'application.
Ne retourne rien.

github():

Ne prend pas de paramètre.
Ouvre une fenêtre du navigateur internet sur le repository où est stocké le code.
Ne retourne rien.

getDate():

Ne prend pas de paramètre.
Renvoie la date actuelle sous un format pratique.
Ne retourne rien.

centrefenetre(fen):

Fen : objet de type Tkinter.Tk.
Place la fenêtre sur l'écran au milieu de la partie gauche de l'écran, centrée.
Ne retourne rien.

start():

Ne prend pas de paramètre.
Crée une instance de la classe Application.
Ne retourne rien.

Documentation FormatCreator.py

Le but de cette classe est de créer une vue permettant à l'utilisateur de créer un format personnalisé afin de l'utiliser dans un entraînement.

Liste et description des méthodes de la classe :

__init__(self, master):

Master : Objet de type Tkinter.Tk
Crée et initialise les paramètres de classe.
Ne retourne rien.

fromApp(self):

Ne prend pas de paramètre.
Efface le contenu des champs de la vue.
Ne retourne rien.

maj_nbElementListe(self, *args):

Ne prend pas de paramètre.
Met à jour le nombre d'éléments dans la liste des numéros d'éléments et met à jour la liste des types par éléments.
Ne retourne rien.

selectionType(self):

Ne prend pas de paramètre.
Sauvegarde le type choisi pour le numéro d'élément sélectionné.
Ne retourne rien.

updateFormatDisplay(self):

Ne prend pas de paramètre.
Met à jour l'aperçu du format lors d'une modification d'un champ de l'interface.
Ne retourne rien.

updateSaveButton(self, *args):

Ne prend pas de paramètre.
Args : arguments jamais utilisés.
Active ou désactive le bouton d'enregistrement des paramètres du nouveau format en fonction de l'état de complétion des champs de l'interface.
Ne retourne rien.

selectionElement(self):

Ne prend pas de paramètre.
Affiche le type correspondant au numéro d'élément sélectionné.
Ne retourne rien.

save(self):

Ne prend pas de paramètre.
Ne retourne rien.

retourMenu(self):

Ne prend pas de paramètre.
Switche la fenêtre sur la vue Menu.
Ne retourne rien.

Liste et description des fonctions :

getDate():

Ne prend pas de paramètre.
Renvoie la date actuelle sous une forme pratique (string).

Documentation GraphDisplay.py :

Le but de cette classe est de permettre l'affichage des graphiques sur les données récupérées pendant l'entraînement du modèle et des graphiques sauvegardés.

Liste et description des méthodes de la classe :

__init__(self, master):

Master : objet de type Tkinter.Tk
Crée et initialise les paramètres de classe.
Ne retourne rien.

majRepertoire(self):

Ne prend pas de paramètre.
Liste le contenu du dossier de sauvegarde des graphiques et met à jour la liste des graphes disponibles.
Ne retourne rien.

displayFromApp(self, parametres):

Ne prend pas de paramètre.
Sélectionne le premier graphique dans la liste (s'il existe) et affiche les deux graphiques.
Ne retourne rien.

displayGraph(self):

Ne prend pas de paramètre.
Affiche les graphiques s'il le dossier des graphiques n'est pas vide, sinon affiche un texte d'erreur.
Ne retourne rien.

retour(self):

Ne prend pas de paramètre.
Switche l'interface sur la vue Menu.
Ne retourne rien.

Documentation Info.py

Le but de cette classe est de créer une vue permettant d'afficher l'avancée de l'entraînement du modèle, une estimation du temps restant. La vue affichée permet également d'arrêter l'entraînement.

Liste et description des méthodes de la classe :

__init__(self, master):

Master : objet de type tkinter.TK
Crée et initialise les paramètres de la classe.
Ne renvoie rien.

lanceTrain(self, parametres, is_model, format_choix):

Parametres : Dictionnaire de paramètres
Is_model : booléen codant le fait que le model en cours ait déjà été entraîné ou pas
Format_choix : dictionnaire contenant les informations sur le format d'entraînement choisi
Rafraîchit l'interface et lance l'entraînement dans un thread séparé. Lance la méthode de mise à jour de l'interface.
Ne retourne rien.

stopTrain(self):

Ne prend pas de paramètre.
Envoie le signal de fin d'entraînement au modèle par l'intermédiaire du paramètre de classe finQueue.
Ne renvoie rien.

updateEpoch(self):

Ne prend pas de paramètre.
Met à jour le nombre d'epoch effectuées et le temps restant en se basant sur les données reçues par l'intermédiaire du paramètre de classe queue.
Si le thread d'entraînement n'est pas fini et que la queue n'est pas vide, cette fonction se rappelle par l'intermédiaire de la méthode self.after (méthode de Tkinter).
Ne renvoie rien.

Liste et description des fonctions :

timeConversion(N):

N : int codant un nombre de secondes
Renvoie le la conversion de N en secondes, minutes et secondes ou heures, minutes et secondes selon la valeur de N (string).

Documentation Lecteur.py

Le but de cette classe est de fournir une interface simple pour écouter les fichiers midi générés par l'application.

Liste et description des méthodes de la classe :

__init__(self, master, parametres):

Master : objet de type Tkinter.Tk
Crée et initialise les paramètres de la classe.
Ne retourne rien.

supprimerFichiers(self):

Ne prend pas de paramètre.
Affiche une fenêtre demandant la confirmation à l'utilisateur de la suppression des fichiers générés. Les supprime si l'utilisateur le confirme.
Ne retourne rien.

miseAJourRepertoire(self, parametres):

Parametres : dictionnaire de paramètres
Liste les fichiers midi présents dans le chemin spécifié dans parametres et met à jour la liste des morceaux disponibles sur l'interface en conséquence.
Ne retourne rien.

play(self):

Ne prend pas de paramètre.
Permet de jouer le morceau sélectionné si aucun morceau n'est en train de jouer ou de reprendre la lecture si le morceau est en pause.
Ne retourne rien.

pause(self):

Ne prend pas de paramètre.
Met en pause la lecture d'un morceau si un morceau est en train d'être joué.
Ne retourne rien.

stop(self):

Ne prend pas de paramètre.
Arrête complètement la lecture d'un morceau (la lecture d'un morceau ne pourra pas être reprise).
Ne retourne rien.

nextSong(self):

Ne prend pas de paramètre.
Joue le morceau suivant dans la liste.
Ne retourne rien.

previousSong(self):

Ne prend pas de paramètre.
Joue le morceau précédent dans la liste.
Ne retourne rien.

selectionMusique(self, event):

Event : paramètre non utilisé.
Joue le morceau sélectionné depuis la liste des morceaux.
Ne retourne rien.

opendir(self):

Ne prend pas de paramètre.
Ouvre le dossier où sont stockés les morceaux de musique générés.
Ne retourne rien.

Liste et description des fonctions :

initLecteur():

Ne prend pas de paramètre.
Initialise le module de lecture de fichiers audio de Pygame.
Ne retourne rien.

Documentation Menu.py :

Le but de cette classe est de fournir une interface graphique à base de vues.

Liste et description des méthodes de la classe :

__init__(self, master, parametres, dico_formats):

Master : objet de type Tkinter.Tk
Parametres : dictionnaire de paramètres
Dico_formats : dictionnaire d'informations sur les formats
Crée et initialise les attributs de la classe.
Ne retourne rien.

fromApp(self):

Ne prend pas de paramètre.
Récupère les informations sur les formats depuis le fichier de sauvegarde des formats et met à jour la liste des formats disponibles.
Ne retourne rien.

genereNewMorceau(self):

Ne prend pas de paramètre.
Si les paramètres de génération sont corrects, appelle la fonction de génération de morceaux.
Ne retourne rien.

exportParametres(self):

Ne prend pas de paramètre.
Récupère les paramètres entrés par l'utilisateur et les enregistre dans le fichier de sauvegarde des paramètres du modèle et de génération.
Ne retourne rien.

changeInterface(self):

Ne prend pas de paramètre.
Active et désactive les paramètres avancés sur l'interface.
Ne retourne rien.

valide(self):

Ne prend pas de paramètre.
Vérifie que les paramètres entrés par l'utilisateur sont corrects (ne dépassent pas de leurs limites, etc). Alerte l'utilisateur si au moins un des paramètres est incorrect.
Retourne un booléen codant la validité des paramètres.

charging(self):

Ne prend pas de paramètre.
Appelle la fonction d'entraînement du modèle si les paramètres entrés par l'utilisateur sont corrects.
Ne retourne rien.

Browser(self):

Ne prend pas de paramètre.
Ouvre un explorateur de fichier pour que l'utilisateur puisse choisir le répertoire de travail. Avertit l'utilisateur si le dossier ne contient pas ou contient beaucoup de fichiers midi.
Ne retourne rien.

Liste et description des fonctions :

muchFileWarning(path):

Path : string représentant le chemin d'un dossier
Avertit l'utilisateur si le dossier contient beaucoup de fichiers midi.
Ne retourne rien.

verifMIDI(path):

Path : string représentant le chemin d'un dossier
Renvoie un booléen codant la présence de fichiers midi dans le dossier.

Documentation LineClient.py :

Le but de cette classe est de fournir une interface non graphique en ligne de commande capable de charger un modèle, modifier des paramètres, lancer l'entraînement et sauvegarder un modèle en une seule ligne.

Liste et description des méthodes de la classe :

__init__(self, args):

Argos : String. liste de paramètres entrés par l'utilisateur en ligne de commande
Crée et initialise certains des paramètres.
Ne retourne rien.

initParams(self):

Ne prend pas de paramètre.
Initialise certains paramètres par défaut.
Ne retourne rien.

parse_param(self):

Ne prend pas de paramètre.
Parcours les paramètres (et leurs valeurs associés) entrés par l'utilisateur et s'ils correspondent à des paramètres existant, les enregistre.
Renvoie False si les paramètres ou valeurs ne sont pas valides/n'existent pas et True sinon.

verif_one_param(self, name, value):

Name : String. nom du paramètre
Value : String. valeur du paramètre
Vérifie que la valeur "value" associée au paramètre de nom "name" est correcte, c'est-à-dire qu'elle respecte les règles imposées par les informations données par l'attribut self.parametres_infos.
Renvoie un booléen codant la validité de la valeur et la valeur évaluée dans son type (Int, Float, String ou Booléen).

displayParams(self):

Ne prend pas de paramètre.
Affiche tous les paramètres utilisés pour l'entraînement du modèle et la génération du morceau.
Ne retourne rien.

lance_train(self):

Ne prend pas de paramètre.
Appelle la fonction de lancement de l'entraînement.
Ne retourne rien.

saveModel(self):

Ne prend pas de paramètre.
Appelle la fonction de sauvegarde du modèle.
Ne retourne rien.

loadModel(self):

Ne prend pas de paramètre.
Appelle la fonction de chargement du modèle.
Ne retourne rien.

genererMorceau(self):

Ne prend pas de paramètre.
Appelle la fonction de génération des morceaux.
Ne retourne rien.

getParametres(self):

Ne prend pas de paramètre.
Renvoie le dictionnaire de paramètres utilisés pour l'entraînement du modèle et la génération.

launch(self):

Ne prend pas de paramètre.
Effectue toutes les actions demandées par l'utilisateur. Dans l'ordre : vérification de l'installation (optionnel), affichage de l'aide (optionnel), affichage des paramètres (optionnel), chargement d'un modèle (optionnel), lancement de l'entraînement, sauvegarde du modèle (optionnel), génération de morceaux (optionnel).
Ne retourne rien.

update_formats(self):

Ne prend pas de paramètre.
Met à jour la liste des formats disponibles depuis le fichier de sauvegarde des modèles.
Ne renvoie rien.

display_help(self):

Ne prend pas de paramètre.
Affiche l'aide.
Ne renvoie rien.

Liste et description des fonctions :

verifMIDI(path):

Path : String codant un chemin de dossier.
Renvoie un booléen codant la présence de fichiers midi dans le dossier.

start(args):

Args : arguments passés depuis la ligne de commande
Instancie un client.
Ne renvoie rien.

Documentation MusicBox.py :

Le but de ce fichier est de vérifier la bonne installation des packages nécessaires au fonctionnement de l’application avant de lancer l’interface elle-même.

Documentation MusicBoxLine.py :

Le but de ce fichier est de récupérer les paramètres en ligne de commande que l’utilisateur va rentrer et de avant de les passer à la classe qui va gérer l’application sous sa forme en ligne de commande.