Documentation PyPoker

exécuter le code

il faut installer certaines bibliothèques:

sur linux

```
sudo apt-get install python3-tk
```

pour linux et windows

```
pip install Pillow
pip install playsound
pip install tkinter-tooltip
pip install requests
```

exécuter le fichier Accueil.py depuis le dossier source

Accueil.py

close_all()

ferme le jeu et les requêtes en cours

btnInteract(event, name, action)

gère l’appui des boutons

generate_accueil(accueil, images)

génère interface utilisateur de la page d’accueil avec tkinter

generate_choix(frame_choix, images)

génère interface utilisateur de la page de mode de jeu avec tkinter

Carte.py (fini)

class Carte:

__init__(self, couleur, valeur)

Construit la carte avec une couleur et une valeur

obtenir_figure(self)

Renvoie la figure (valet, dame, roi ou As) de la carte si elle en a

obtenir_features(self)

Renvoie la valeur et la couleur de la carte

obtenir_all_features(self)

Renvoie la valeur, la couleur et la figure (si elle existe) de la carte

obtenir_valeur(self)

Renvoie la valeur de la carte

obtenir_couleur(self)

Renvoie la couleur de la carte

Generer_Packet()

Fonction qui génère un packet de carte 52 cartes différentes en utilisant la class Carte

Fonction qui génère un packet de 52 cartes sous forme de pile de carte en utilisant en utilisant la class Carte et les fonctions de pile.py

Melanger_Packet(Packet)

Fonction qui Mélange le packet donné en paramètre en utilisant la librairie random

Chat.py

class ScrollWindow():

Classe permettant l'affichage du tchat ainsi que des invitations

chat(self,player, message='', date="2023")

affiche un message dans le chat

invite(self,player,sens= True)

affiche une invitation allant de l'utilisateur vers player si sens est Vrai sinon de player vers l'utilisateur

updateinvites(self, reponse)

permet de mettre à jour les anciennes invitations

scroll_linux(self,event) -> None

permet de faire défiler le chat sous linux

scroll_windows(self,event) -> None

permet de faire défiler le chat sous windows

getmessages(self, boucle = True)

envoie une requête à la base de donnée pour recevoir les nouveaux messages depuis le dernier message reçu ce qui limite le nombre de données transférées

getinvites(self, boucle = True)

envoie une requête à la base de donnée qui renvoie toutes les invitations en lien avec le joueur

si il y a des invitations on appelle la fonction **updateinvites** pour actualiser les invitations déjà existantes, ajouter les nouvelles et supprimer les invitations supprimées.

focusOut(self,event,name)

quitte l'entrée de texte et remet la valeur par défaut

OnKeyPress(self,event, name)

détecte l'écriture dans une entrée de texte pour détecter les caractère interdits

pre_chat(self, event)

prépare la requête asynchrone pour écrire un nouveau message dans la base de données du chat global

pre_chat_game(self, event)

prépare la requête asynchrone pour écrire un nouveau message dans la base de données du chat d'une partie

pre_invite(self, event)

prépare la requête asynchrone pour écrire l'invitation dans la base de données

respond_invitation(self,event,action)

prépare la requête asynchrone qui répond à l'invitation(soit accepte, soit refuse, soit supprime)

check_partie_existe(self)

si une requête est supprimée par update_invite() alors il est possible que ce soit car une invitation ait été acceptée alors on regarde si il existe une partie avec le joueur et si oui on lance la partie en ligne Game Online

Duel.py (fini)

Ici hand1 et hand2 réfère aux mains des deux joueurs composées de 2 instances de Cartes, de même river correspond aux 5 cartes communes aux deux joueurs

HighestCard(hand1,hand2)

Fonction comparant la valeur de la carte la plus haute des deux mains en utilisant High, renvoie une valeur différente en fonction du gagnant ou de l'éventuelle égalité

duel(hand1,hand2,river)

Fonction déterminant quel joueur gagne une manche, premièrement la fonction fusionne chaque main avec la rivière, puis elle compare la valeur de victoire des deux joueurs, renvoie une valeur différente en fonction de quel joueur gagne ou en cas d'égalité, utilise la plupart des fonctions de Winning.py

Encrypt.py (fini)

Encrypt(text)

Fonction qui sert à crypter les mots de passe. Utilise la librairie ***hashlib*** pour crypter le texte fourni en paramètre dans la fonction en **sha256**.

Game.py (fini)

__init__(self, app, images, skins, utilisateur)

Cette fonction a pour but de créer et d'initialiser toutes les variables nécessaires au fonctionnement de la partie.

paramètres:

app est la frame Tkinter dans laquelle sera affiché la partie
images est un dictionnaire contenant les images ouvertes par images_handler.py
skins est une liste contenant les skins ouverts par images_handler.py
utilisateur est une instance de la Classe Utilisateur

On génère les deux joueurs comme instances de la classe Player: avec leur montant d'argent ils sont stockés dans une liste

On crée les différent éléments de l'interface utilisateur, la plupart sont des attributs de la classe Game pour pouvoir les modifier ensuite

initialise(self)

Cette méthode de la classe Game a pour but de réinitialiser les deux joueurs et leur argent pour recommencer une partie.

on appelle ensuite newound

newround(self)

Cette méthode de la classe Game réinitialise chaque tour c'est à dire quand après que les gains aient été distribués (et ce même si la partie n'est pas finie) . Ainsi cette méthode met à jour l'interface utilisateur, distribue de nouvelles cartes et définit que joueur doit jouer d'abord.

allin(self,player)

paramètres:
player est l'indice du joueur dans la liste self.joueurs

cette méthode de la classe Game est exécutée quand le joueur “player” met tapis
On vérifie que ce soit le tour du joueur en question
on appelle la méthode allin de la classe Player elle renvoie l’argent que le joueur a à disposition
on met l’argent du joueur dans le pot et s' il est inférieur à la mise précédente on exécute endriver ce qui affiche les cartes de la rivière sinon le jeu continue.

mise(self,player, valeur)

paramètres:
player est l’indice du joueur dans la liste self.joueurs
valeur est le montant de la mise

cette méthode de la classe Game est exécutée quand le joueur “player” mise un montant valeur
On vérifie que ce soit le tour du joueur en question et que le montant de la mise est supérieur ou égal à la mise précédente

on appelle la méthode miser de la classe Player elle vérifie le joueur ait assez d’argent
puis si le montant de la mise est égal à la mise précédente alors les joueurs ont trouvé un accord et on retourne certaines cartes de la rivière avec la méthode finmise
sinon c’est au tour de l’autre joueur et le jeu continue

suivrecall(self, player)

paramètres:
player est l’indice du joueur dans la liste self.joueurs

cette méthode de la classe Game est exécutée si le joueur suit ou joue parole, c’est le même bouton dans l’interface utilisateur
On vérifie que ce soit le tour du joueur en question
on appelle la méthode miser de la classe Player qui vérifie que le joueur ait assez d’argent puis on retourne certaines cartes de la rivière avec la méthode finmise

secouche(self,player)

paramètres:

player est l'indice du joueur dans la liste self.joueurs
cette méthode de la classe Game est exécutée quand le joueur décide de se coucher
On vérifie que ce soit le tour du joueur en question
on donne le montant du pot au joueur adverse avec la méthode distribuergain
on recommence un tour avec la méthode newround

endriver(self)

cette méthode de la classe Game a pour but d'afficher l'entièreté de la rivière en cas de tapis tant que la longueur de la rivière n'est pas 5 on tire une carte du paquet avec la fonction depiler
on génère l'image de la carte avec la méthode generer_images_riviere puis on affiche cette image

on appelle ensuite la méthode pour déterminer le gagnant

finmise(self)

cette méthode de la classe Game a pour but d'afficher les cartes de la rivière au fur et à mesure (d'abord 3, puis 1, puis 1)
pour cela tire une (ou 3) carte du paquet avec la fonction depiler
on génère l'image de la carte avec la méthode generer_images_riviere puis on affiche cette image

generer_images_riviere(self,carte)

paramètres:
carte est une instance de la classe Carte
on récupère les informations de la carte grâce à la méthode obtenir_features de la classe Carte
ensuite on ouvre l'image correspondante dans le dossier Images/, on recadre l'image et on la donne a Tkinter

gagnantjoueur(self)

cette méthode de la classe Game a pour but de déterminer lequel des deux joueur à la main gagnante pour cela on utilise la fonction duel on distribue ensuite l'argent au gagnant avec la méthode distribuergain

Puis on montre les cartes des deux joueurs pendant 5 secondes avec la méthode interactCarte

si un des deux joueurs n'a plus d'argent on affiche la page de fin
sinon on commence une nouveau tour avec la méthode newround

release(self)

cette méthode force les cartes des joueurs à être cachées après avoir étées montrées à la fin de la partie

distribuergain(self,player, egalite=False)

paramètres:
player est l'indice du joueur dans la liste self.joueurs
egalite est un booléen Vrai en cas d'égalité et Faux par défaut

cette méthode de la classe Game a pour but de donner l'argent du pot au gagnant
si il y a égalité, le pot est partagé de manière égale entre les deux joueurs
sinon le joueur player gagne le pot

generate_end_game(self)

cette méthode de la classe Game a pour but de générer la page de fin de partie qui affiche le gagnant de la partie et propose de revenir à l'Accueil ou rejouer grâce à deux boutons qui font appel à la méthode end_game_bouton

end_game_bouton(self,event, action, name)

paramètres:
event est donné par Tkinter quand un bind est exécuté il contient les informations du widget (ici un bouton) qui a été cliqué
action correspond à l'état du bouton appuyé ou relaché
name est ce que le bouton doit faire ("rejouer", "retourner à l'accueil")

cette méthode de la classe Game a pour but d'écouter les boutons de la page de fin
on change l'image en fonction de si il est appuyé ou non

si le joueur veut rejouer on appelle la méthode initialise
si il veut revenir à l'accueil on affiche l'accueil et on cache la partie

boutonInteract(self,event, player, button, action)

paramètres:
event est donné par Tkinter quand un bind est exécuté il contient les informations du widget (ici un bouton) qui a été cliqué
action correspond à l'état du bouton appuyé ou relaché
player est l'indice du joueur dans la liste self.joueurs
button est ce que le bouton doit faire ("miser", "tapis", "se coucher", "suivre")

cette méthode de la classe Game a pour but d'écouter les boutons de la page de jeu
on change l'image en fonction de si il est appuyé ou non

on appelle les méthodes miser , allin, secouche, suivrecall en fonction de name

update_mise_labels(self,player)

paramètres:
player est l'indice du joueur dans la liste self.joueurs

cette méthode de la classe Game a pour but de mettre à jour le texte des mises des joueurs

update_parole_relance(self)

cette méthode de la classe Game a pour but de mettre à jour les images des boutons de mise et de suivi car le bouton "mise" et "relance" est le même de même que "parole" et "suivre"

interactCarte(self,player)

paramètres:
player est l'indice du joueur dans la liste self.joueurs

cette méthode de la classe Game a pour but de retourner la carte de joueur lorsqu'il clique dessus en changeant l'image affichée (soit le skin soit la carte face visible)

focus_in(self,event)

paramètres:
event est donné par Tkinter quand un bind est exécuté il contient les informations du widget (ici une entrée de texte) qui a été cliquée

on change alors la couleur du texte en noir et on retire le texte par défaut

focus_out(self,event)

paramètres:
event est donné par Tkinter quand un bind est exécuté il contient les informations du widget (ici une entrée de texte) qui a été cliquée

on change alors la couleur du texte en gris et on insère le texte par défaut

OnKeyPress(self,event, player)

paramètres:
event est donné par Tkinter quand un bind est exécuté il contient les informations du widget (ici un bouton) qui a été cliqué
player est l'indice du joueur dans la liste self.joueurs

cette méthode de la classe Game a pour but de seulement autoriser l'écriture chiffres pour ne miser que des nombres sinon on supprime les caractères interdits

showtour(self)

cette méthode de la classe Game a pour but d'entourer les cartes du joueur dont c'est le tour afin qu'il sache que c'est à lui de jouer

Game Online.py

même chose que Game

prepare_game(self,partie, pseudo2)

initialise les attributs en lien avec le joueur adverse et la partie
puis lance l'écoute de la partie sur un nouveau Thread en envoyant des requêtes au serveur toutes les deux secondes. grâce à la méthode getactions

getactions(self, loop = True)

envoie une requête asynchrone au serveur pour avoir les nouvelles actions qui ont eu lieu dans la partie. Puis en fonction du résultat on modifie le cours de la partie.
cette fonction est appelée en boucle toutes les deux secondes

send_actions(self, action, mise=0, text="")

envoie les nouvelles action du joueur au serveur afin de synchroniser la partie

set_seed(self,seed)

initialise la fonction random avec un nombre commun aux deux joueurs donné par le serveur pour que les actions aléatoires soient les mêmes pour les deux joueurs.

Pile.py (fini)

creer_pile(valeurs=[])

Créer une pile vide par défaut mais pouvant prendre des valeurs placés sous formes de liste en paramètre

taille(Pile)

Renvoie la taille de la pile placée en paramètre

empiler(Pile, value)

Prends en paramètre une valeur et une pile. La fonction ajoute la valeur à la pile.

peek(Pile)

Renvoie le sommet de la pile placée en paramètre

depiler(Pile)

Renvoie le sommet de la pile placée en paramètre en le supprimant

is_empty(Pile)

Renvoie oui si la pile placée en paramètre est vide, non sinon

Player.py (fini)

class Player:

Classe simulant un joueur durant une partie (main,argent,mise et images des cartes)

givecards(self, carte)

Fonction ajoutant à la main du joueur une instance de la classe Cartes
De plus, s'occupe de charger l'image correspondant à la carte

miser(self, mise)

Fonction utilisé pour miser une somme (mise) si le joueur a assez d'argent sur lui, sinon renvoie un False pour signaler que la somme n'est pas valide

allin(self)

Fonction ajoutant toute l'argent du joueur à la mise puis mets son argent à 0

getcards(self)

Renvoie la main du joueur

clearcards(self)

Vide la main du joueur

gains(self, pot)

Ajoute l'argent du pot à celle du joueur

get_mise(self)

Renvoie la mise actuelle du joueur

reset_mise(self)

Mets la mise à 0

get_argent(self)

Renvoie l'argent du joueur

get_cartes_images(self)

Renvoie les images des cartes du joueur

Skin.py

class skinPage():

update_text(self)

met à jour les textes sous les images afin que le joueur sacoche si le skin est en sa possession, si il est sélectionné ou si il peut l'acheter

__select_skin(self, id_skin)

permet de sélectionner un skin s'il a été acheté ou d'acheter un skin s'il ne l'est pas.

Son.py (fini)

Jouer_Son(Type_de_son)

Cette fonction nécessite d'avoir installé la librairie ***playsound***, pour cela, utilisez la ligne de commande suivante : pip install playsound

La fonction ***Jouer_Son*** prend une argument un ***Type_de_son***. Ensuite la fonction va ouvrir le fichier json ***sons.json*** et va choisir aléatoirement parmi le type de son demandé, un son.

Winning.py (fini)

Victoire : dico

Dictionnaire servant à déterminer le classement de chaque combinaisons ainsi que leur nom

Ici **pack** réfère à une liste de 7 instance de la classe Carte correspondant aux 2 cartes d'une main plus les 5 de la rivière

High(pack)

Fonction renvoyant la les features de la carte avec la valeur la plus haute du pack en utilisant l'index de la liste

Pair(pack)

Fonction renvoyant une liste contenant les features de 2 cartes ayant une valeur identique du pack, en utilisant une boucle pour comparer ces valeurs 2 à 2

ThreeOfKind(pack)

Fonction renvoyant une liste contenant les features de 3 cartes ayant une valeur identique du pack, en utilisant une boucle pour comparer ces valeurs 3 à 3

FourOfKind(pack)

Fonction renvoyant une liste contenant les features de 4 cartes ayant une valeur identique du pack, en utilisant une boucle pour comparer ces valeurs 4 à 4

Straight(pack)

Fonction renvoyant une liste contenant les features de 5 cartes ayant des valeurs qui se suivent (ex. 3, 4, 5, 6, 7), en créant une liste annexe des valeurs du pack pour pouvoir les ordonner, puis de

Color(pack)

Fonction renvoyant des listes d'instances de la classe Cartes triée en fonction de leur symbole seulement s'il y en a plus de 5, les trie en fonction de leurs couleurs dans différentes listes puis les trie en fonction de leur valeurs.

WinColor(pack)

Fonction appelant Color(pack), renvoyant la liste d'instances de Cartes puis la couleur associée

FullHouse(pack)

Fonction renvoyant les Cartes s'il y a un ThreeOfKind et une Pair dans le pack, appelle ThreeOfKind et puis créé une liste avec toutes les autres cartes à part ces trois là pour appeler Pair. Si les deux fonctions renvoyent des instances, alors la fonction renvoie le résultat des deux précédente

StrFlush(pack)

Fonction renvoyant une liste de Cartes de même couleur formant une suite de 5 cartes en utilisant Color et Straight

RoyalFlush(pack)

Fonction renvoyant une liste de Cartes de même couleur formant une suite de 5 cartes avec la valeur la plus haute de 14 (As), en utilisant StrFlush

WinCondition(pack)

Fonction renvoyant la valeur de la combinaison la plus haute du joueur en partant du plus haut et en appelant toute les autres fonctions du fichier

connexion.py

class PageAuthentification():

focusIn(self,event, name)

quand le joueur clique dans une entrée de texte cela cache le texte par défaut

focusOut(self,event,name)

quand le joueur sort d'une entrée de texte cela affiche le texte par défaut

OnKeyPress(self,event, name)

détecte l'écriture dans une entrée de texte afin d'empêcher les caractères interdits

visible(self,event)

montre ou non le mot de passe

pre_process(self,event, name)

prépare la requête au serveur qui va permettre la connexion ou la création d'un compte

fonctions_verif.py (fini)

password_valide(word)

S’assure que le word donné en paramètre corresponde à différents critères pour constituer un mot de passe solide.

Ces critères sont :

- Une longueur d’au moins 8 caractères
- Avoir au moins un caractère “spécial” appartenant à un ensemble prédéfini
- Ne doit pas contenir de caractère qui ne soit ni une lettre ni un chiffre ni un caractère “spécial”

pseudo_valide(word)

S’assure que le word donné en paramètre corresponde à différents critères pour constituer un pseudo correct.

Ces critères sont :

- Une longueur d’au plus 20 caractères
- Ne comporte que des caractères alphanumériques ou des underscores “ _ “

images_handler.py

import_images()

importe les images depuis le dossier Images et les recadre d’après un dictionnaire contenant leurs informations

import_skins()

importe les skins depuis le dossier Skins et les ouvre en deux tailles différentes.

send_request.py (fini)

partie_online(Joueur1, Partie, Action, Mise, Texte):

Envoie une requête contenant les différents paramètres au fichier `partie_online.php` . Si la requête échoue la fonction renvoie un erreur sinon elle renvoie la réponse sous forme de dictionnaire.

get_partie_online(pseudo1,partie, Date_et_heure):

Envoie une requête contenant les différents paramètres au fichier `getpartie_online.php`. Si la requête échoue la fonction renvoie un erreur sinon elle renvoie la réponse sous forme de dictionnaire.

create_invitation(pseudo1, pseudo2):

Envoie une requête contenant les différents paramètres au fichier `createinvitation.php`. Si la requête échoue la fonction renvoie un erreur sinon elle renvoie la réponse sous forme de dictionnaire.

connexion(pseudo1, mdp):

Envoie une requête contenant les différents paramètres au fichier `connexion.php`. Si la requête échoue la fonction renvoie un erreur sinon elle renvoie la réponse sous forme de dictionnaire.

ecrire_chat_global(pseudo1, text):

Envoie une requête contenant les différents paramètres au fichier `ecrirechatglobal.php`. Si la requête échoue la fonction renvoie un erreur sinon elle renvoie la réponse sous forme de dictionnaire.

get_chat_global(pseudo1, Date_et_heure):

Envoie une requête contenant les différents paramètres au fichier `getchatglobal.php`. Si la requête échoue la fonction renvoie un erreur sinon elle renvoie la réponse sous forme de dictionnaire.

get_invitations(pseudo1,):

Envoie une requête contenant les différents paramètres au fichier `getinvitations.php`. Si la requête échoue la fonction renvoie un erreur sinon elle renvoie la réponse sous forme de dictionnaire.

inscription(pseudo1, mdp):

Envoie une requête contenant les différents paramètres au fichier `inscription.php`. Si la requête échoue la fonction renvoie un erreur sinon elle renvoie la réponse sous forme de dictionnaire.

respond_invite(pseudo1,pseudo2,action):

Envoie une requête contenant les différents paramètres au fichier `respond_invite.php`. Si la requête échoue la fonction renvoie un erreur sinon elle renvoie la réponse sous forme de dictionnaire.

get_features(pseudo1):

Envoie une requête contenant les différents paramètres au fichier `getfeatures.php`. Si la requête échoue la fonction renvoie un erreur sinon elle renvoie la réponse sous forme de dictionnaire.

update_features(pseudo1, gemmes, skins, skin_selectionne):

Envoie une requête contenant les différents paramètres au fichier `update_features.php`. Si la requête échoue la fonction renvoie un erreur sinon elle renvoie la réponse sous forme de dictionnaire.

partie_existe(pseudo1):

Envoie une requête contenant les différents paramètres au fichier `partie_existe.php`. Si la requête échoue la fonction renvoie un erreur sinon elle renvoie la réponse sous forme de dictionnaire.

delete_partie(pseudo1):

Envoie une requête contenant les différents paramètres au fichier `delete_partie.php`. Si la requête échoue la fonction renvoie un erreur sinon elle renvoie la réponse sous forme de dictionnaire.

utilisateur.py

class Utilisateur:

get_all(self)

envoie une requête au serveur afin de connaître les skins et les gemmes d’un utilisateur enregistrés dans une base de données.

update_features(self)

envoie une requête au serveur afin de mettre à jour les skins et les gemmes d’un utilisateur.