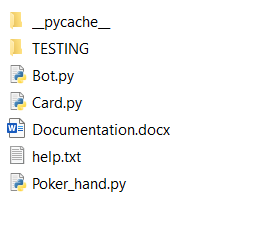
***Projet «Bot croupier»***

La joue de cartes Poker est extrêmement populaire et dispose de plusieurs variations. Celle la plus connu s’appelle classique où quatre joueurs dont chacun a 7 cartes. L’objectif est composer la combination la plus haut et faire le pari. Cette variation a été incarné dans ce projet conçu sur la plateforme de la réseau sociale Discord à l’air de bot. Ici nous allons nous interroger sur 3 aspects :

1. La structure du code source
2. L’interrogation sur le fonctionnement
3. Les chemins d’amélioration

**La structure du code source**

Le dossier racine comprends 4 fichiers dont 3 sont de type python et ce sont eux qui s’exécutent et 1 et le fichiers de type .txt qui comprends le petit interrogation sur le fonctionnement du projet.

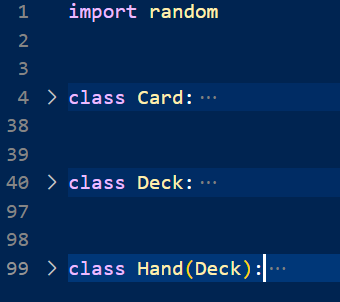


Img.1 – la structure du dossier racine

Étudions chaque fichier

**Card.py**

Le fichier est principal car il initialise trois figures principaux : la carte, la deck et la main de joueur, c’est ce qu’on peut voir en structure des classes



Img.2 – la structure de classes dans le fichier Card.py

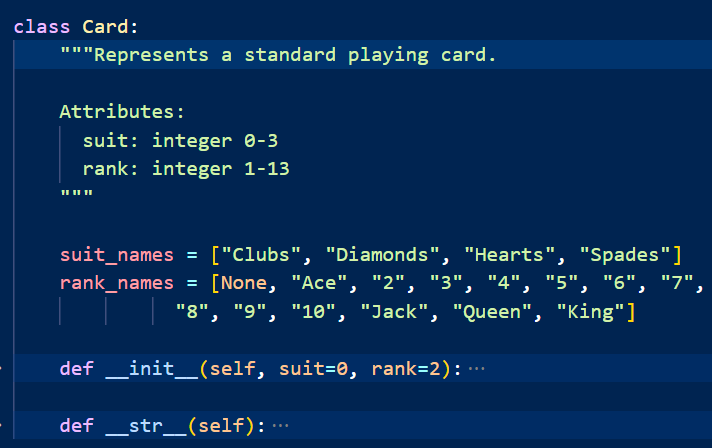
*Classe Card* initialise la représentation de la carte à jouer.

Elle n’a que 2 attributs :

* *Card .suit* – le nombre entier
* *Card.rank* – le nombre entier aussi.

Hormis le méthode de l’initialisation, classe a celui d’imprimer :

* *Card.\_\_str\_\_()* - imprimer la carte en mode lisible par l’homme. Rende la chaine des caractères



Img.3 – classe Card

*Classe Deck* est la représentation de la Deck de 56 cartes.

Elle a un attribut :

* *Deck.cards* – le liste des cartes à jouer

Les méthodes de classes sont suivantes :

* *Deck.\_\_str\_\_() -* permettent d’imprimer la deck rends la chaīne de caractères
* *Deck.add\_card()* - ajouter la carte*,* nerende rien
* *Deck.remove\_card() -* enlever la cartenerende rien
* *Deck.pop\_card() -* retirer la carde, nerende rien
* *Deck.shuffle() -* mélanger la deck nerende rien
* *Deck.sort() -* trier la deck, nerende rien
* *Deck.move\_cards()* - distribuer les cartes aux joueursnerende rien



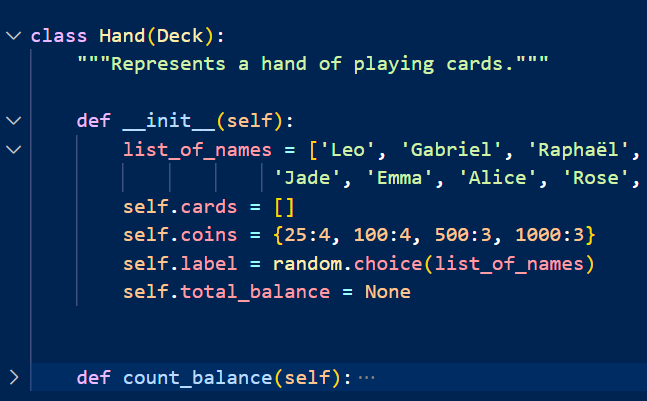
Img.4 – classe Deck

*Classe Hand* hérite celle Deck et représente la main de joueur. Elle a des attributs :

* *Hand.cards* – les cartes dans la main
* *Hand.coins* – les jetons de joueur
* *Hand.label* – le nom de joueur
* *Hand.total\_balance* – le solde total de joueur.

Classe dispose d’un méthode :

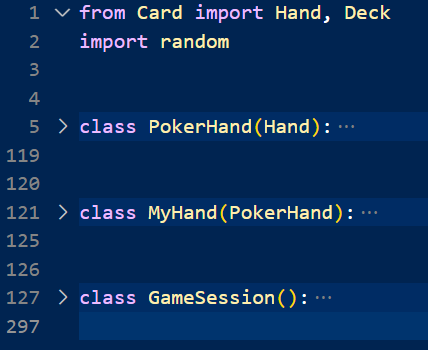
* *Hand.count\_balance() -* calcule le solde et rends le nombre entierre.



Img.5 – classe Hand

**Poker\_hand.py**

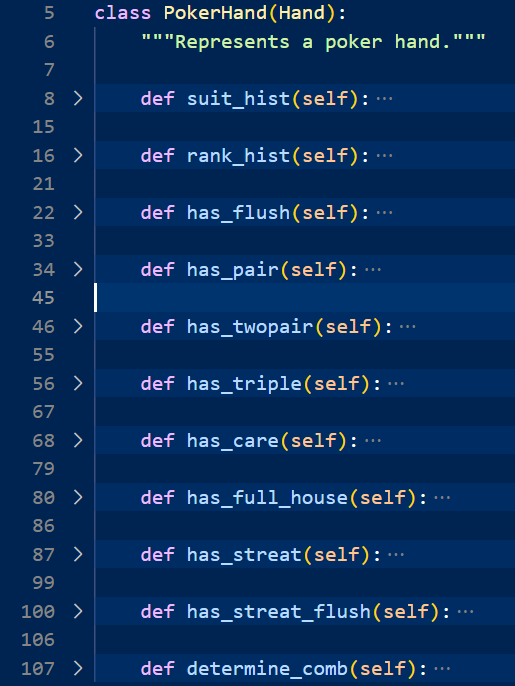
Dans ce fichier tout la logique de la joue est réalisé. Il utilise les méthodes les classes de fichier Card.py et il est aussi décomposé en 3 classes : *PokerHand*, *MyHand* et *GameSesion*



Img.6 – La structure du fichier Poker\_hand.py

*Classe PokerHand* hérite celle Hand et latransforme en main qui a déjà des cartes et se trouve en train de la joue. La classe dispose du même attribut que Hand – *cards.* Les méthodes de la classe sont suivantes :

* *PokerHand .suit\_hist(self) –* le méthode complémentaire qui construit l’histogramme des attributs *Card.suit.* Rends dictionnaire
* *PokerHand .rank\_hist(self) –* méthode complémentaire qui construit l’histogramme des attributs *Card.rank.* Rends dictionnaire
* *PokerHand .has\_pair(self) –* vérifie si la main a pair dans ses cartes. Rends True si oui, False sinon
* *PokerHand .has\_twopair() -* vérifie si la main a deux pairs dans ses cartes. Rends True si oui, False sinon
* *PokerHand .has\_triplef) -* vérifie si la main a triple dans ses cartes. Rends True si oui, False sinon
* *PokerHand .has\_care() –* vérifie si la main a care dans ses cartes. Rends True si oui, False sinon
* *PokerHand .has\_streat() -* vérifie si la main a streat dans ses cartes. Rends True si oui, False sinon
* *PokerHand .has\_full\_house() –* vérifie si la main a full house dans ses cartes. Rends True si oui, False sinon
* *PokerHand .has\_flush() –* vérifie si la main a flush dans ses cartes. Rends True si oui, False sinon
* *PokerHand .has\_flush\_royal() –* vérifie si la main a flush royal dans ses cartes. Rends True si oui, False sinon
* *PokerHand .determine\_comb() –* détermine la combination la plus haut parmi les autres. Rends le tuple où le premier élément est le nom de la combination et le deuxième – le rang ou le poids de la combination.



Img.7 – classe PokerHand

*Classe MyHand* hérite complètement la classe *PokerHand.* L’objectif de le créer c’est distinguer la main d’utilisateur parmi celles des autres. Alors il a la même structure que *PokerHand* et donc le même fonctionnement

*Classe GameSession* lance et gère la session de la joue. Les attributs sont suivantes :

* *GameSession.deck* – list : la deck utilisé dans la joue
* *GameSession*.*coefs* – dict : les coefficients des combinations
* *GameSession*.*bank* – int : le solde de la joue qui collecte tout les paris et puis son valeur est transmit au solde d’un/une vainqueur
* *GameSession.players –* list : le liste des joueurs
* *GameSession*.*players\_names* – list : le liste des prénoms des joueurs
* *GameSession.players\_combos* – dict : la combination correspondant à chaque joueur
* *GameSession.wins* – dict : les victoires aux tours de chaque joueur

Classe dispose des méthodes suivantes :

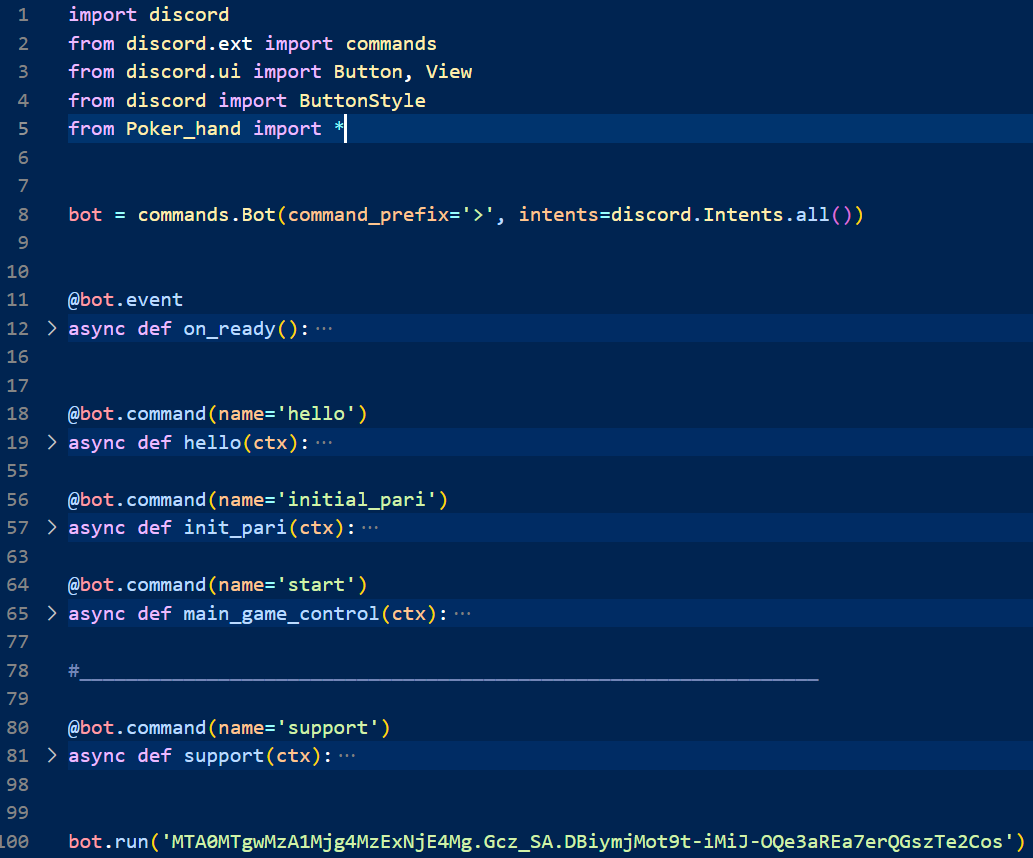
* *GameSession.prepare\_game\_session()* – prépare la session de la joue en exécutant les méthodes *self.create\_deck()* et *self.create\_players(n=3)*
* *GameSession.create\_deck() –* crée la deck
* *GameSession.create\_players(n=3) –* crée n nombre des joueurs (défaut - 3)
* *GameSession.give\_cards(number=7) –* donne *number* nombre des cartes aux joueurs (défaut - 7)
* *GameSession.make\_init\_pari(pari=25) –* fait le pari initiale (défaut - 25)
* *GameSession.find\_combs() –* trouve la combinations la plus haute dans la main de chaque joueur
* *GameSession.make\_paris() –* fait le pari de chaque joueur. Retourne le méthode *self.show\_paris()* et lui transmet comme argument le liste de pari
* *GameSession.show\_paris(paris\_list) –* imprime les paris de chaque joueur. Retourne la chaine de caractères
* *GameSession.show\_combs() –* imprime le pari correspondant à chaque joueur
* *GameSession.determine\_winners() –* détermine les vainqueurs du tour. Retourne la chaine des caractères
* *GameSession.determine\_losers\_and\_final\_winner() –* détermine les perdants et puis le gaigneur absolue

**Bot.py**

Fichier principal parc qu’il contient les commandes de bot pour marcher. Chaque commande est la fonction coroutine et précédé par le décorateur *@Bot.command()* ou *@Bot.event* qui sont les méthodes du classe *Bot* de Discord API.

Les fonctions – commandes présenté ici sont suivantes :

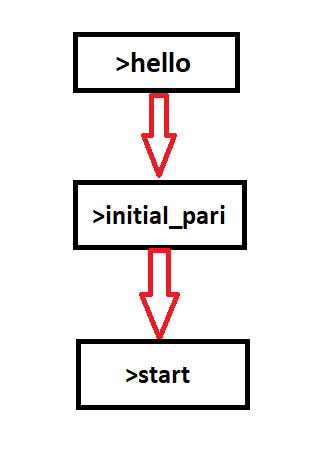
* *async def hello(ctx) –* la fonction crée la bouton avec le choix «Jouer » ou « «Ne pas jouer ». Si Jouer la session de joue s’exécute
* *async def init\_pari(ctx) –* en soi cette fonction fait le même que *GameSession.make\_init\_pari(pari=25)*
* *async def main\_game\_control(ctx) –* lance le tour de la joue dont en cours elle demande à l’utilisateur de déterminer son combination et faire le pari
* *async def support(ctx)* – la fonctioncrée deux boutons dont la premier envoie la petit instruction d’utilisation et la deuxième envoie cette gros document que vous lissez



Img.8 – la structure du fichiers Bot.py

**L’interrogation sur le fonctionnement**

Prenant en compte que la structure du code source est assez complexe, le fonctionnement du bot est super facile ! Elle est donc rendu de façon où on vous demande de saisir la commande après laquelle on fait qch. Tout d’abord il faut mentionner que chaque commande doit être précédé par le préfixe **‘>’,** par example « >hello ». Cette petit chose est fait due aux spécification de la bibliothèque utilisé. Sur la schéma ci-dessus l’ordre des commandes à saisir est montré.



img.9 – la schéma de l’ordre des commandes

**Les chemins d’amélioration**

Notre projet est à la stade initial maintenant et beaucoup des aspects doivent être amélioré. Celui le plus évidant est l’interface d’utilisateur, il peut être échangé en graphique car il est beaucoup plus compréhensible que celui textuel. Ensuite la système de calcul de pari fait par les joueurs, maintenant c’est la système très simple basé sur les coefficients déjà connu en constantes.

Nous espérons que vous allez aimer notre projet !

Les auteurs : Anton HLADYSHEV, Leo Moutinho Marta