

12/09/2018

Python pour l'ingénieur



Exercice : Poker

Module de formation

Introduction

- ❑ Le cours propose plusieurs exercices permettant de comprendre le fonctionnement et le paradigme propre du langage Python.
- ❑ Ce cahier propose un exercice d'un autre genre : suite d'étapes permettant de créer un programme complet, du moteur de jeu jusqu'à l'IHM graphique.
- ❑ Cet exercice « géant » est donc très proche du projet personnel réalisé en binôme.
- ❑ Thème : Poker Texas Hold'em !

Rappel des règles

- ❑ Jeu de 52 cartes
- ❑ Couleurs : pique, cœur, carreau, trèfle
 - Identifiants : P C K T
- ❑ Rangs : 2, 3, 4, 5, 6, 7, 8, 9, 10, Valet, Dame, Roi, As
 - Identifiants : 2 3 4 5 6 7 8 9 X V D R A
- ❑ Combinaisons
 - Paire, Deux paires, Brelan, Suite, Full, Carré, Quinte flush
- ❑ But du jeu
 - S'emparer du stock de jetons de chacun des adversaires !
- ❑ Déroulement du jeu
 - Partie de poker = succession de coups indépendants
 - Coup = donne + tours d'enchères + abattage + attribution du pot à la meilleure main
 - Texas Hold'em = deux cartes privatives, tour d'enchères, trois cartes communes (flop), tour d'enchères, quatrième carte commune (turn), tour d'enchères, cinquième carte commune (river), dernier tour d'enchères

Classes

- ❑ classe **Carte**
 - Une carte
- ❑ classe **Joueur**
 - Un joueur
- ❑ classe **Croupier**
 - Gestion des cartes
- ❑ classe **Coup**
 - Gestion d'un coup
- ❑ classe **Partie**
 - Déroulement de la partie et des coups



Classes génériques valables pour toutes les variantes de Poker

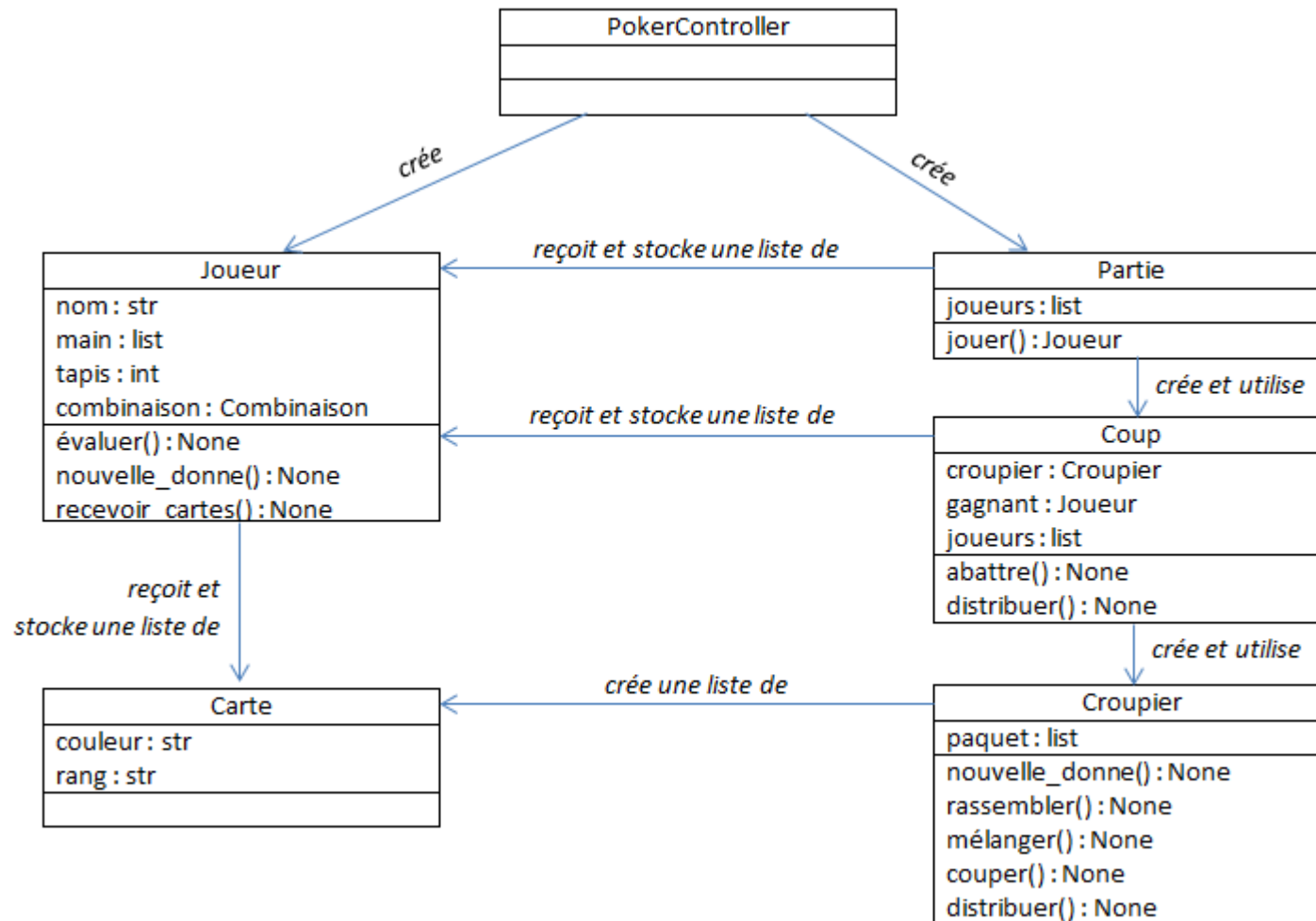


Classe adaptée à une variante particulière de Poker.

Responsabilités

<i>Contrôleur</i>	<i>Partie</i>	<i>Coup</i>
	Liste des joueurs	Liste des joueurs Croupier
Création des joueurs Création d'une partie		
	Création des coups	
		Création d'un croupier <i>distribuer, miser, échanger, abattre...</i>

Diagramme de classes



Modules

Créer les fichiers suivants :

- poker1.py : moteur du jeu Supop Poker
- poker1t.py : interface utilisateur en mode texte
- poker1g.py : interface utilisateur en mode graphique

Ces fichiers seront mis à jour à chaque étape. Pour l'exercice n°2 par exemple, il sera nécessaire de copier et de renommer ces 3 fichiers, afin de travailler avec poker2.py, poker2t.py et poker2g.py.

Les fichiers poker*g.py sont mis de côté en attendant le début du cours sur les interfaces graphiques.

Moteur de jeu (poker1)

poker1.py

Dans ce fichier, coder et tester les classes suivantes :

- une classe Carte : une carte
- une classe Joueur : un joueur
- une classe Croupier : gestion des cartes
- une classe Coup : gestion d'un coup
- une classe Partie : déroulement de la partie et des coups

Pour l'instant :

- une carte est définie par un rang et une couleur
- un joueur est défini par un nom
- Croupier, Coup et Partie sont "vides" (utiliser le mot-clé pass !)

Définir les rangs et les couleurs :

(identifiant de type caractère = chaîne de caractères de taille 1)

RANGS : 2 3 4 5 6 7 8 9 X V D R A

COULEURS : P C K T

Implémenter `__init__()` et `__repr__()` pour chacune des classes.

Ne pas oublier de coder le test directement dans le module !

Interface utilisateur (poker1t)

poker1t.py (interface texte)

Réutiliser poker1.py (import poker1 as poker).

Le programme a les fonctions suivantes :

- choix d'une couleur
- tirage au hasard d'une carte de la couleur choisie
- affichage d'un menu (une option permet de quitter le programme)

Moteur de jeu (poker2)

poker2.py

Coder et tester un coup de poker avec les règles suivantes :

- distribution de 5 cartes privatives

Il faut modifier et compléter les classes suivantes :

- Classe Joueur :

- main : liste des Cartes possédées par le joueur
 - nouvelle_donne() : réinitialise la main du joueur (ie plus de carte en main)
 - recevoir() : donne de nouvelles Cartes au joueur (fournies par Croupier)

- Classe Croupier :

- paquet : liste des Cartes
 - rassembler() : créer un paquet de 52 cartes, non mélangé
 - mélanger() : mélanger aléatoirement le paquet de cartes (utiliser le module random)
 - couper() : couper le paquet, autour d'une position aléatoire
 - nouvelle_donne() : rassembler(), mélanger() puis couper() le paquet
 - distribuer() : distribuer N cartes à une liste de Joueurs

- Classe Coup :

- liste de joueurs : liste de Joueurs participant au coup
 - croupier
 - __init__() : faire une nouvelle donne (Joueurs et Croupier)
 - distribuer() : demander au Croupier de distribuer N cartes à la liste des joueurs

Ne pas oublier de modifier la fonction `__repr__()` des classes !

Conserver le test précédent dans une fonction `test1()` et créer un nouveau test `test2()`.

Interface utilisateur (poker2t)

poker2t.py (interface texte)

Utiliser poker2.py (import poker2 as poker).

Le programme a les fonctions suivantes :

- création d'un joueur, option pour créer directement 3 joueurs prédéfinis (Alice, Bob et Carl)
- exécution d'un coup (uniquement la distribution !)
- affichage d'un menu

Moteur de jeu (poker3)

poker3.py

Coder et tester une partie de poker avec les règles suivantes :

- distribution de 5 cartes privatives
- abattage et désignation du gagnant
- tapis initial de 50 unités
- 10 unités mises en jeu par coup

Il faut modifier et compléter les classes suivantes :

- Classe Joueur :
 - tapis
 - combinaison
 - évaluer() : calcule la combinaison du joueur
(en utilisant le module pokerlib disponible sur learnpython.ovh)
- Classe Coup :
 - résultats
 - gagnant
 - abattre() : compare les mains des joueurs et détermine le joueur gagnant
- Classe Partie :
 - joueurs : joueurs participant à la partie
 - __init__() : initialise le tapis des joueurs
 - jouer() : détermine les joueurs pouvant participer à un Coup (tapis >= mise), puis réalise un Coup avec ces joueurs, puis répartit les gains et les pertes en fonction du gagnant du Coup

Ne pas oublier de modifier la fonction `__repr__()` des classes !

Conserver le test précédent dans une fonction `test2()` et créer un nouveau test `test3()`

pokerlib

Module fourni sur le site learnpython.ovh (<http://learnpython.ovh/src/pokerlib.py>)
Définit une classe Combinaison qui permet de comparer la valeur de mains de 5 cartes.

Exemple d'utilisation :

```
import pokerlib

main1 = [Carte(), Carte(), Carte(), Carte(), Carte()]
main2 = [Carte(), Carte(), Carte(), Carte(), Carte()]
combinaison1 = pokerlib.Combinaison(main1)
combinaison2 = pokerlib.Combinaison(main2)
print(combinaison1.name(), combinaison2.name())

if combinaison1 > combinaison2:
    print('main1 est plus forte que main2')
elif combinaison1 == combinaison2 :
    print('main1 et main2 ont la même valeur')
else:
    print('main2 est plus forte que main1')

# ou encore
combinaison_max = max([combinaison1, combinaison2])
```

Interface utilisateur (poker3t)

poker3t.py (interface texte)

Réutiliser poker3.py (import poker3 as poker).

Le programme a les fonctions suivantes :

- création d'un joueur, option pour créer directement 3 joueurs prédéfinis (Alice, Bob et Carl)
- démarrage d'une partie
- exécution d'un coup
- affichage d'un menu (une option permet de démarrer la partie, une autre de jouer un coup)

Interface utilisateur (poker1g)

poker1g.py (interface graphique)

Réutiliser poker1.py (import poker1 as poker).

Le programme a les fonctions suivantes :

- choix d'une couleur
- tirage au hasard d'une carte de la couleur choisie
- affichage d'un bouton par couleur

Créer les classes suivantes :

- Classe CouleurWidget : hérite de la classe QWidget
 - Affiche une carte dans une couleur donnée
 - Affiche un bouton sous la carte pour tirer une nouvelle carte
 - Utiliser les fichiers images fournies sur le site learnpython.ovh
o http://learnpython.ovh/png/cartes_png.zip
 - Utiliser un layout pour réaliser la disposition
- Classe PokerWidget : hérite de la classe QWidget
 - Affiche horizontalement un widget CouleurWidget par couleur
 - Utiliser un layout pour réaliser la disposition
- Classe PokerWindow : hérite de la classe QMainWindow
 - Fenêtre du programme, affiche un PokerWidget

Interface utilisateur (poker1g)



Interface utilisateur (poker2g)

poker2g.py (interface graphique)

Utiliser poker2.py (import poker2 as poker).

Le programme a les fonctions suivantes :

- création d'un joueur (jusqu'à 4 joueurs maximum)
- exécution d'un coup (uniquement la distribution !)
- affichage d'un menu (une option permet de réunir Alice, Bob et Carl)
- affichage des mains de chaque joueur (main de 5 cartes)

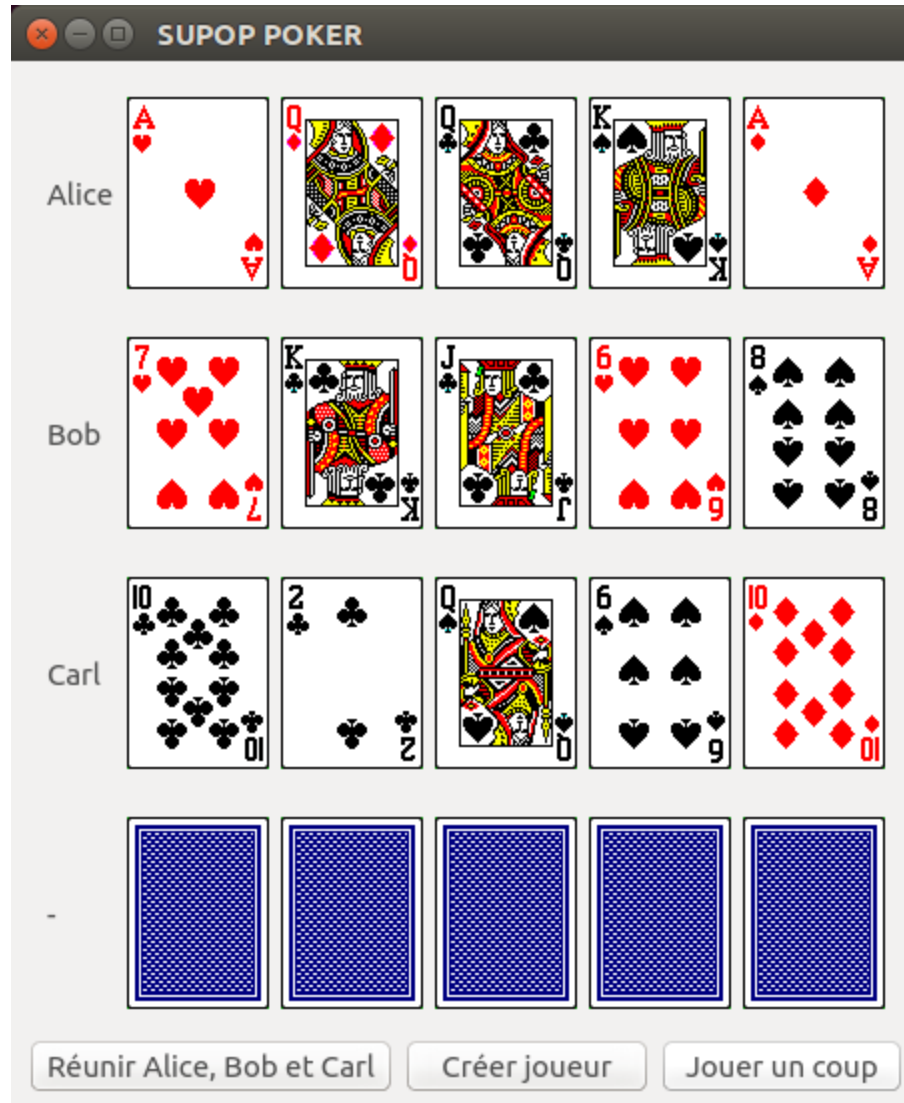
Modifier ou créer les classes suivantes :

- Classe JoueurWidget (hérite de QWidget)
 - `__init__()` : met en place l'affichage (avec un layout) du nom et des 5 cartes d'un joueur identifié par son numéro (0 à 3). A ce stade le dos des cartes est affiché, car soit le joueur n'existe pas, soit il n'a pas encore de cartes en mains.
 - `rafraichir()` : cette fonction met à jour l'affichage pour afficher le nom et les cartes du joueur associé au widget (si le joueur existe).
- Classe PokerWidget
 - Affiche verticalement les 4 widgets JoueurWidget correspondant à chaque joueur
 - Affiche les boutons de commande
 - Créer la callback associée à chaque bouton (pour le moment vide)

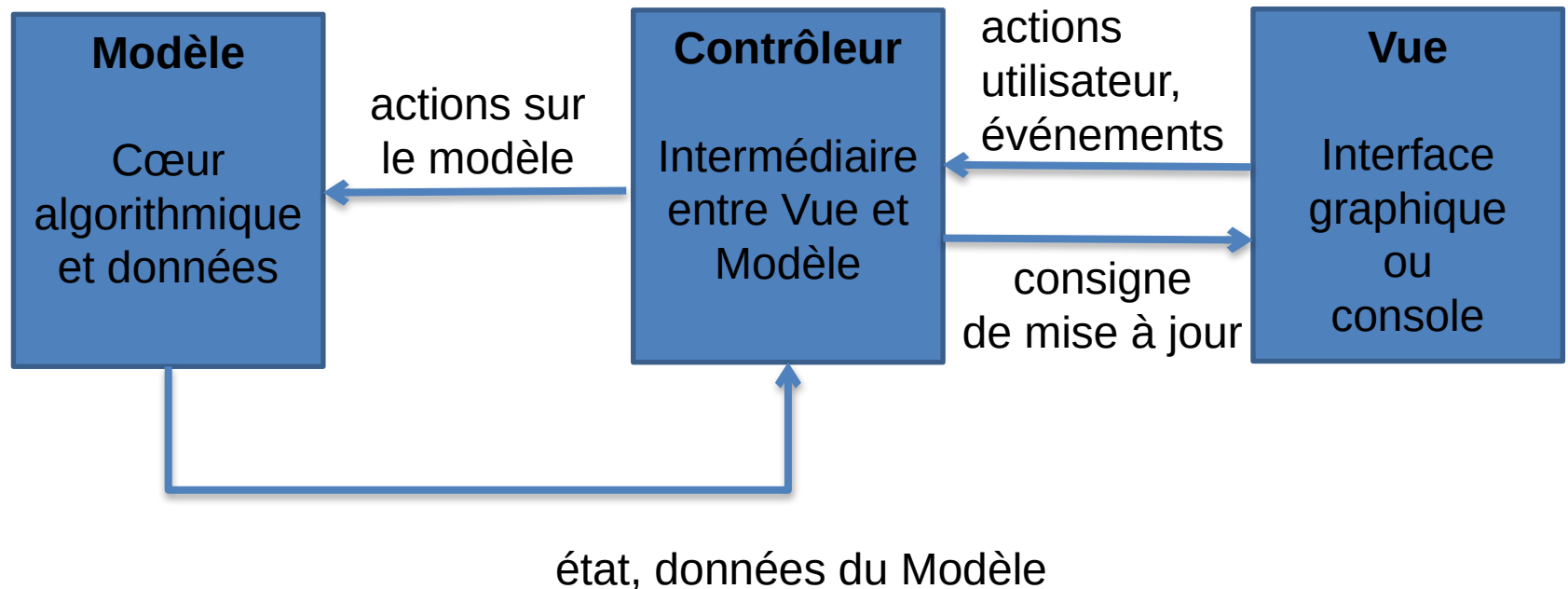
Connecter ensuite l'interface graphique au moteur de jeu (poker2) en créant un contrôleur :

- Développer la classe ControllerBase (suivant le modèle du cours)
- Créer une instance de ControllerBase au niveau de l'application PyQt
- Fournir cet objet à toutes les instances de classe PyQt (en argument des constructeurs)
- Dans le constructeur de JoueurWidget, appeler la méthode inscrire() de l'objet ControllerBase
- Développer la méthode rafraichir de JoueurWidget (par exemple afficher cinq As de Coeur)
- Dans le callback d'un des boutons, appeler la méthode avertir() de l'objet ControllerBase
- Tester le bon fonctionnement du mécanisme inscrire() / avertir() / rafraichir()
- Importer poker2, puis développer et utiliser la classe PokerController
- Coder le contenu des callbacks de PokerWidget

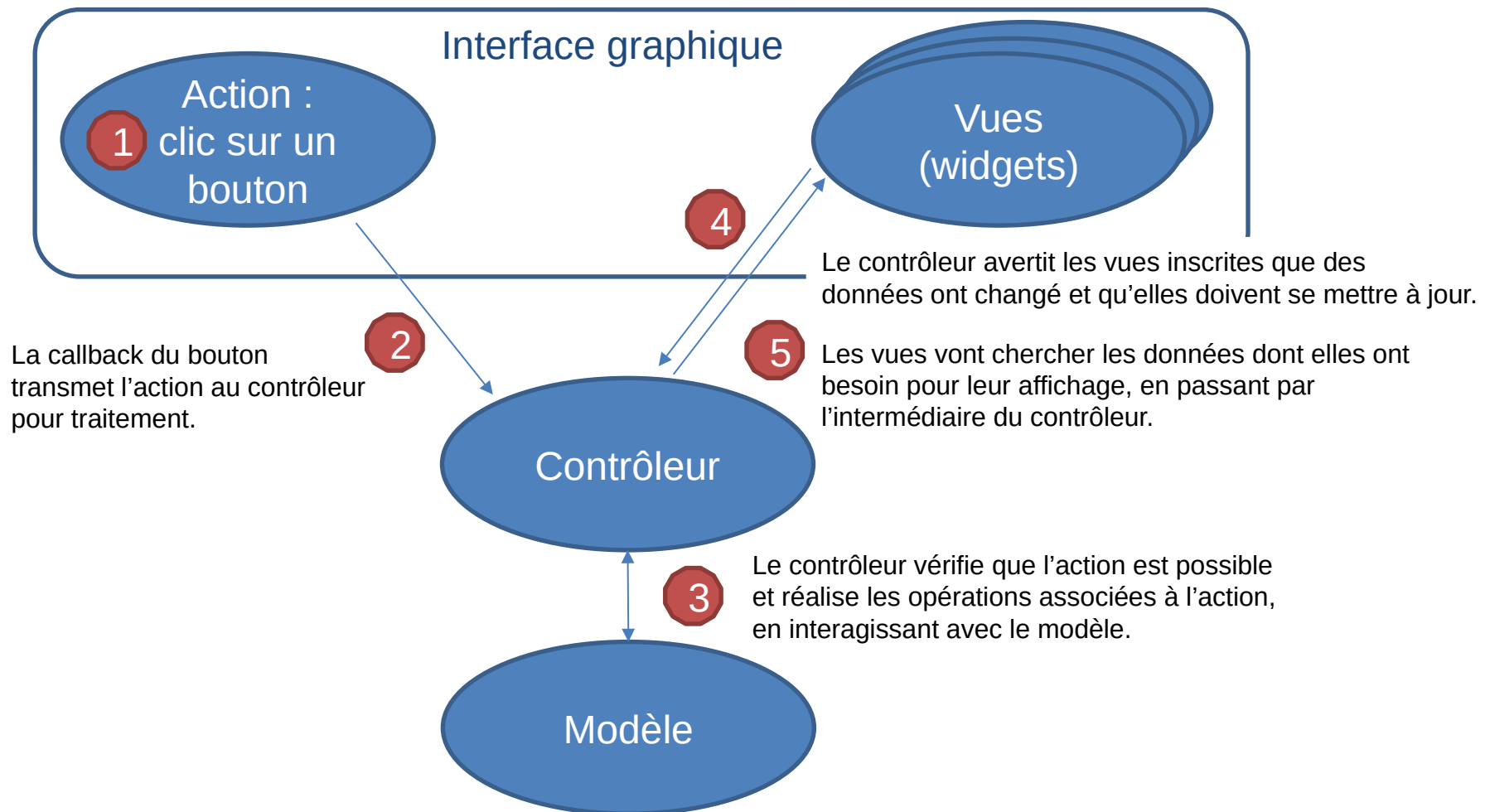
Interface utilisateur (poker2g)



Architecture MVC : principe



Architecture MVC : exemple



Architecture MVC : code

```
class ControllerBase:
    def __init__(self):
        self.clients = []

    def inscrire(self, client):
        self.clients.append(client)

    def avertir(self):
        for client in self.clients:
            client.rafraichir()
```

➔ À dériver en une classe *Controller* spécifique au projet

```
class ElementVue(QWidget):
    def __init__(self, parent, controller):
        super().__init__(parent)
        self.controller = controller
        self.controller.inscrire(self)

    def rafraichir(self):
```

...



Accès au contrôleur via self.controller

Interface utilisateur (poker3g)

poker3g.py (interface graphique)

Le programme a les fonctions suivantes :

- création d'un joueur
- démarrage d'une partie
- exécution d'un coup
- affichage du tapis de chaque joueur
- utilisation d'un contrôleur

Modifier les classes suivantes :

- JoueurWidget
 - Afficher le tapis et la combinaison du joueur
- PokerWidget
 - Ajouter un widget QTextEdit servant à afficher des messages (fenêtre de log)
 - Ajouter un bouton permettant de démarrer une partie
- Contrôleur
 - Développer le code permettant de dérouler une partie

Contrôleur commun

Déplacer le contrôleur dans un module distinct (poker3c).

Copier et modifier poker3g de manière à utiliser ce contrôleur (poker3g1).

Copier et modifier poker3t de manière à utiliser ce contrôleur (poker3t1).

Interface utilisateur (poker3g)

