#### Министерство образования Республики Беларусь

Учреждение образования

«Гомельский государственный университет

имени Франциска Скорины»

Факультет математики и технологий программирования

Кафедра вычислительной математики и программирования

**Забыл название темы=)**

Курсовой проект

Исполнитель:

студент группы ПМ-32 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Б.В. Бондаренко

Ассистент кафедры  
вычислительной математики  
и программирования. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.В.Киргинцева

Гомель 2020

**Реферат**

Курсовой проект: 21 страница, 8 рисунков, 6 источников.

**Ключевые слова:** Flask, URL, Jinja2, Покер, вероятность, html.

**Объект исследования (разработки):** Вычисление вероятности выпадения комбинации в покере

**Цель работы:** Разработка веб-приложения.

**Инструментарий разработки:** Flask, python, jinja2, html, css.

**Результатом работы** является разработка веб-приложения с функциональным графическим интерфейсом и вычислительным процессом.

**Область применения:** Упрощение подсчета вероятности в покере.

Содержание

[Введение](#_Toc407142478) 4

[1 Обзор используемых технологий](file:///C:\Users\Екатерина\Downloads\КУРСОВАЯ%20РАБОТА%20ТИХОНОВИЧ.docx) 5

[1](#_Toc407142481)[.1 Flask](#_Toc407142482) 5

[1.2 Python](#_Toc407142485) 7

[1.3 Jinja2](#_Toc407142485) 8

[1.4 HTML](#_Toc407142485) 8

[2 Описание структуры веб-приложения](#_Toc407142492) 10

[2.1 Создание текстур](#_Toc407142492) 10

[2.2 Структура игрового интерфейса](#_Toc407142492) 12

[2.3 Скрипты игрового приложения](#_Toc407142492) 13

[3 Демонстрация работы приложения 1](#_Toc407142496)4

[3.1 Начало игры 1](#_Toc407142512)4

[3.2 Переход на другую лакацию 1](#_Toc407142516)6

[3.3 Главная задача игры 1](#_Toc407142516)8

[Заключение](#_Toc407142523) 19

[Список использованных источников](#_Toc407142524) 20

[Приложение. Текст скриптов](#_Toc407142524) 21

**Введение**

С развитием цифровых технологий компьютеры все больше вливаются в жизнь человека. Изобретение компьютеров послужило переломным моментом в развитии многих отраслей промышленности, на порядок повысило мощь и эффективность военной техники, внесло множество прогрессивных изменений в работу средств массовой информации, систем связи, качественно изменило принцип работы банков и административных учреждений.

Сегодня темпы компьютеризации превышают темпы развития всех других отраслей. Без компьютеров и компьютерных сетей не обходится ни одна средняя фирма, не говоря о крупных компаниях. Современный человек начинает взаимодействовать с компьютером постоянно на работе, дома, в машине и даже в самолете. Компьютеры стремительно внедряются в человеческую жизнь, занимая свое место в нашем сознании, а мы зачастую не осознаем того, что начинаем во многом зависеть от работоспособности этих дорогостоящих кусков металла. Уже сегодня компьютеры отвечают за наведение и запуск ядерных ракет, за банковские переводы многомиллионных сумм денег и многие другие системы, ошибки, в работе которых дорого обходятся людям. К сожалению, даже этим не ограничивается зависимость человека от компьютера.

С появлением Web-технологии компьютер начинают использовать совершенно новые слои населения Земли. Можно выделить две наиболее характерные группы, находящиеся на разных социальных полюсах, которые были стремительно вовлечены в новую технологию, возможно, даже помимо их собственного желания. С одной стороны, это были представители элитарных групп общества - руководители крупных организаций, президенты банков, топ - менеджеры, влиятельные государственные чиновники и т. д. С другой стороны, это были представители широчайших слоев населения - домохозяйки, пенсионеры, дети.

Вот и в карточные игры внедряются технологии, они приобретают популярность в online. Любители азартных игр постоянно ищут способ испытать свои способности и нервы. Многие профессиональные игроки высчитывают вероятность в голове. Однако существуют калькуляторы которые помогают им в этом.

**1 Обзор используемых технологий**

* 1. **Flask**

Flask является микрофреймворком для создания вебсайтов на языке Python. В основу статьи положен перевод из официальной документации Flask. Поэтому в ней имеется обращение от первого лица, то есть от создателя фреймворка Армина Ронахера.

«Микро» в фреймворке относится не только к простоте и небольшому размеру базы, но это также может означать тот факт, что он не предлагает вам много проектных решений. Несмотря на то, что Flask использует шаблонизаторы, мы не будем принимать подобные решения для вашего хранилища данных или других частей. Тем не менее, для нас термин «микро» не означает, что встя реализация должна вписываться в один файл.  
  
Одним из проектных решений во Flask является то, что простые задачи должны быть простыми; они не должны занимать много кода, и это не должно ограничивать вас. Поэтому мы сделали несколько вариантов дизайна, некоторые люди могут посчитать это удивительным и даже странным. Например, Flask использует локальные треды внутри объектов, так что вы не должны передавать объекты в пределах одного запроса от функции к функции, оставаясь в безопасном треде. Хоть это и очень простой подход, который позволяет сэкономить время, такое решение может вызвать некоторые проблемы для слишком больших приложений, поскольку изменения в этих локальных тредах-объектах могут произойти где угодно в этом треде. Для того, чтобы решить эти проблемы, мы не стали скрывать от вас локальные треды-объекты, вместо этого мы охватываем их и предоставляем вам много инструментов, чтобы сделать работу с ними настолько приятной, насколько это возможно.

Во Flask многие вещи предварительно сконфигурированы, на основе общей базовой конфигурации. Например, шаблоны и статические файлы сохранены в подкаталогах в пределах исходного дерева. Вы также можете изменить это, но обычно этого не требуется.

Основная причина почему Flask называется «микрофреймворком» — это идея сохранить ядро простым, но расширяемым. В нем нет абстрактного уровня базы данных, нет валидации форм или всего того, что уже есть в других библиотеках. Однако, Flask поддерживает расширения, которые могут добавить необходимую функциональность и имплементирует их так, как будто они уже были встроены изначально. В настоящее время уже есть расширения: формы валидации, поддержка закачки файлов, различные технологии аутентификации и многие другие.

* 1. **Python**

**Python** в русском языке распространено название пито́н или, более правильно, **па́йтон**) — Высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время Стандартная библиотека включает большой набор полезных функций.

Python поддерживает структурное, обобщенное объектно-ориентированное, функциональное и аспектно-ориентированное программирование. Основные архитектурные черты — [динамическая типизация](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), [автоматическое управление памятью](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), полная [интроспекция](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), механизм [обработки исключений](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), поддержка [многопоточных вычислений](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C), высокоуровневые [структуры данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Поддерживается разбиение программ на [модули](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), которые, в свою очередь, могут объединяться в пакеты.

Эталонной реализацией Python является интерпретатор [CPython](https://ru.wikipedia.org/wiki/CPython), поддерживающий большинство активно используемых платформ. Он распространяется под [свободной лицензией](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая. Есть [реализация интерпретатора для JVM](https://ru.wikipedia.org/wiki/Jython) с возможностью [компиляции](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80), [CLR](https://ru.wikipedia.org/wiki/IronPython), [LLVM](https://ru.wikipedia.org/wiki/LLVM), другие независимые реализации. Проект [PyPy](https://ru.wikipedia.org/wiki/PyPy" \o "PyPy) использует [JIT-компиляцию](https://ru.wikipedia.org/wiki/JIT-%D0%BA%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F), которая значительно увеличивает скорость выполнения Python-программ.

Python — активно развивающийся [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), новые версии с добавлением/изменением языковых свойств выходят примерно раз в два с половиной года. Язык не подвергался официальной стандартизации, роль стандарта де-факто выполняет [CPython](https://ru.wikipedia.org/wiki/CPython" \o "CPython), разрабатываемый под контролем автора языка. В настоящий момент Python занимает второе место в [рейтинге TIOBE](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81_TIOBE) с показателем 12,12 %. Аналитики отмечают, что это самый высокий балл Python за все время его присутствия в рейтинге.

* 1. **Jinja2**

**Jinja** (произносится как [дзиндзя](https://ru.wikipedia.org/wiki/%D0%94%D0%B7%D0%B8%D0%BD%D0%B4%D0%B7%D1%8F" \o "Дзиндзя)) — это [шаблонизатор](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%82%D0%BE%D1%80" \o "Шаблонизатор) для [языка программирования Python](https://ru.wikipedia.org/wiki/Python_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)). Он подобен шаблонизатору [Django](https://ru.wikipedia.org/wiki/Django" \o "Django), но предоставляет Python-подобные выражения, обеспечивая исполнение шаблонов в [песочнице](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%81%D0%BE%D1%87%D0%BD%D0%B8%D1%86%D0%B0_(%D0%B1%D0%B5%D0%B7%D0%BE%D0%BF%D0%B0%D1%81%D0%BD%D0%BE%D1%81%D1%82%D1%8C)). Это текстовый шаблонизатор, поэтому он может быть использован для создания любого вида разметки, а также исходного кода. Лицензирован под [BSD лицензией](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_BSD).

Шаблонизатор Jinja позволяет настраивать теги, фильтры, тесты и глобальные переменные. Также, в отличие от шаблонизатора Django, Jinja позволяет конструктору шаблонов вызывать функции с аргументами на объектах.

Простой пример шаблона:

**from** **jinja** **import** from\_string

tmpl = from\_string(u'''**\**

<html>

<head><title>{{ variable|escape }}</title></head>

<body>

{**% f**or item in item\_list %}

{{ item }}{**% i**f not loop.last %},{**% e**ndif %}

{**% e**ndfor %}

</body>

</html>''')

print tmpl.render(

variable='Value with <unsafe> data',

item\_list=[1, 2, 3, 4, 5, 6]

)

Результат в HTML:

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Value with **&lt;**unsafe**&gt;** data</**title**>

</**head**>

<**body**>

1,

2,

3,

4,

5,

6

</**body**>

</**html**>

Помимо веб-фреймворков ([Flask](https://ru.wikipedia.org/wiki/Flask_(%D0%B2%D0%B5%D0%B1-%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA)" \o "Flask (веб-фреймворк))), Jinja2 используется и, например, в [системе управления конфигурациями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%BA%D0%BE%D0%BD%D1%84%D0%B8%D0%B3%D1%83%D1%80%D0%B0%D1%86%D0%B8%D1%8F%D0%BC%D0%B8) [SaltStack](https://ru.wikipedia.org/wiki/SaltStack" \o "SaltStack) и [Ansible](https://ru.wikipedia.org/wiki/Ansible" \o "Ansible).

* 1. **HTML**

**HTML** (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *HyperText Markup Language* — «язык [гипертекстовой](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%B5%D1%80%D1%82%D0%B5%D0%BA%D1%81%D1%82) разметки») — стандартизированный язык разметки веб-страниц во [Всемирной паутине](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0). Код HTML интерпретируется [браузерами](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80); полученная в результате интерпретации страница отображается на экране монитора компьютера или мобильного устройства.

Язык HTML до 5-й версии определялся как приложение [SGML](https://ru.wikipedia.org/wiki/SGML) (стандартного обобщённого языка разметки по стандарту [ISO](https://ru.wikipedia.org/wiki/ISO) 8879). Спецификации HTML5 формулируются в терминах [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model) (объектной модели документа).

Строгим вариантом HTML является [XHTML](https://ru.wikipedia.org/wiki/XHTML), он наследует синтаксис [XML](https://ru.wikipedia.org/wiki/XML) и является приложением языка XML в области разметки гипертекста.

HTML-страницы, как правило, открываются браузерами обмениваясь с сервером информацией по протоколу [HTTP](https://ru.wikipedia.org/wiki/HTTP) или [HTTPS](https://ru.wikipedia.org/wiki/HTTPS), в виде простого текста или с использованием [шифрования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

**2 Описание структуры веб-приложения**

* 1. **Создание страницы html**

Веб-страница состоит из трёх частей: Информация о версии HTML, Шапка веб-страницы, в которой содержится техническая информация ( <head> ), Тело веб-страницы ( <body> )

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** http-equiv="Content-Type" content="text/html; charset=utf-8" />

...

</**head**>

<**body**>

...

</**body**>

</**html**>

**2.2 Создание и подключение css**

### Правила построения CSS

В первых трёх случаях подключения стилей CSS к документу (см. выше) каждое правило CSS из файла имеет две основные части — *селектор* и *блок объявлений*. *Селектор*, расположенный в левой части правила до знака «{» определяет, на какие части документа (возможно, специально обозначенные) распространяется правило. *Блок объявлений* располагается в правой части правила. Он помещается в фигурные скобки, и, в свою очередь, состоит из одного или более *объявлений*, разделённых знаком «;». Каждое *объявление* представляет собой сочетание *свойства CSS* и *значения*, разделённых знаком «:». Селекторы могут группироваться в одной строке через запятую. В таком случае свойство применяется к каждому из них.

**селектор**, **селектор** {

свойство: значение;

свойство: значение;

свойство: значение;

}

писание стилей находится в отдельном файле, оно может быть подключено к документу посредством элемента <link>, включённого в элемент <head>:

<!DOCTYPE html>

<**html**>

<**head**>

.....

<link rel="stylesheet" href="{{ url\_for('static', filename ='css/style.css') }}">

</**head**>

<**body**>

.....

</**body**>

</**html**>

**2.3 Создание Flask приложения**

На этом этапе создаем небольшое простейшее веб-приложение для примера внутри файла Python и запускаем его для начала работы сервера, который отобразит определенную информацию в браузере.

Содаем файл app.py. Этот файл app.py будет служить минимальным примером того, как обрабатывать запросы HTTP. Внутри него импортируем [объект Flask](https://flask.palletsprojects.com/en/1.1.x/api/#flask.Flask) и создаем функцию, которая возвращает ответ HTTP. Напишите следующий код внутри файла app.py​​​:

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def hello():

return 'Hello, World!'

В предыдущем блоке необходимо предварительно импортировать объект Flask из пакета flask. Затем создать экземпляр приложения Flask с именем app. Передать специальную переменную \_\_name\_\_, которая содержит имя текущего модуля Python. Она указывает экземпляру его расположение. Это необходимо, так как Flask устанавливает ряд путей за кадром.

Создав экземпляр app, начинаем использовать его для обработки поступающих веб-запросов и отправки ответов пользователю. @app.route​​​ — это [декоратор](https://en.wikipedia.org/wiki/Python_syntax_and_semantics#Decorators), который превращает стандартную функцию Python в функцию просмотра Flask, конвертирующую возвращаемое значение функции в ответ HTTP, который отображается клиентом HTTP, например веб-браузером. Мы передаем значение '/' в @app.route()​​​ для обозначения того, что эта функция будет отвечать на веб-запросы для URL /, который является основным URL-адресом.

1. **Демонстрация работы веб-приложения**

* 1. **Покер**

**Правила игры в покер** – это первый шаг любого, кто только встал на путь изучения покера.

Каждому игроку в Техасском Холдеме раздается по две карты.

После раздачи **начинается торговля**, которая проходит в несколько раундов.

После первого круга **торговли**, если в раздаче остается больше одного человека, то, по правилам игры в покер, на стол кладутся три общие открытые карты, которые называют **флоп**. Общие карты нужно использовать для составления комбинаций.

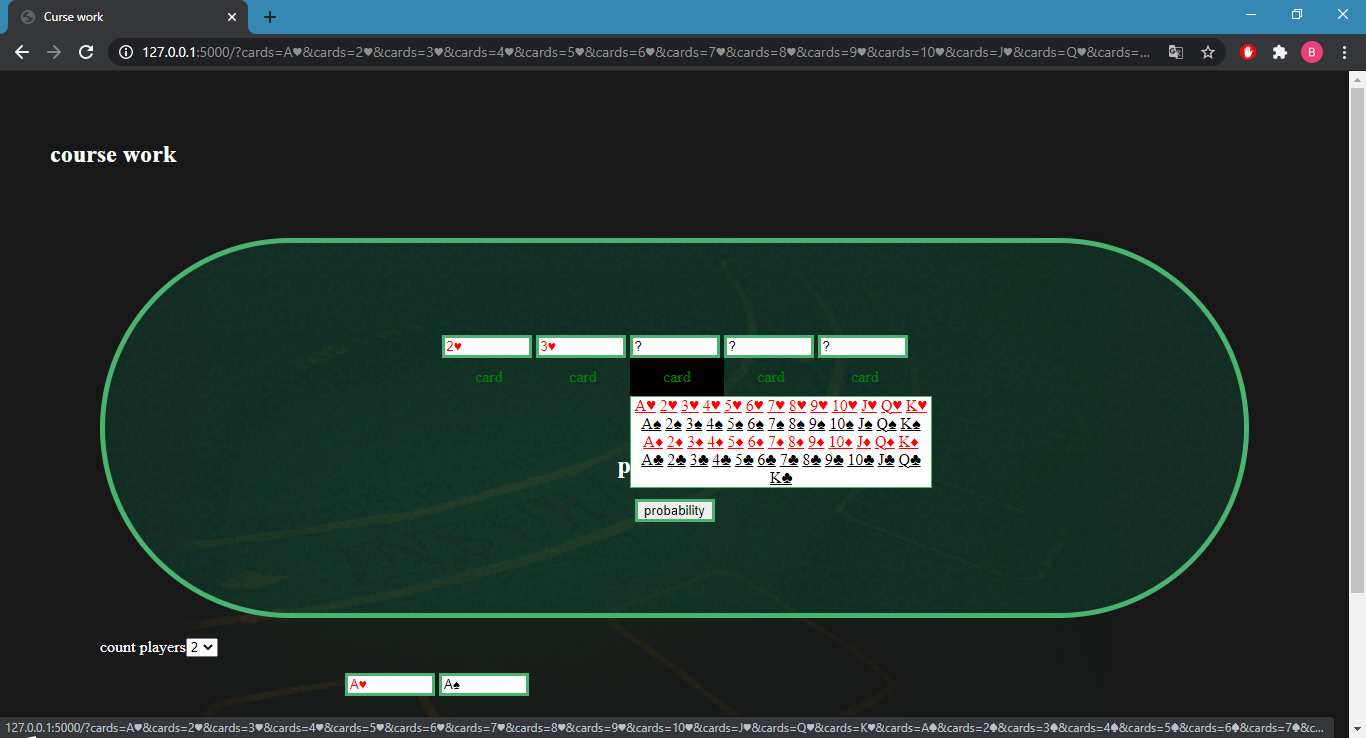
Если и после этого раунда в раздаче остается больше одного человека, то кладут еще одну общую карту, которую называют **терном**. Аналогично флопу, после сдачи терна проводится еще один круг торговли.

После терна, если это необходимо, кладут последнюю общую карту – **ривер**.

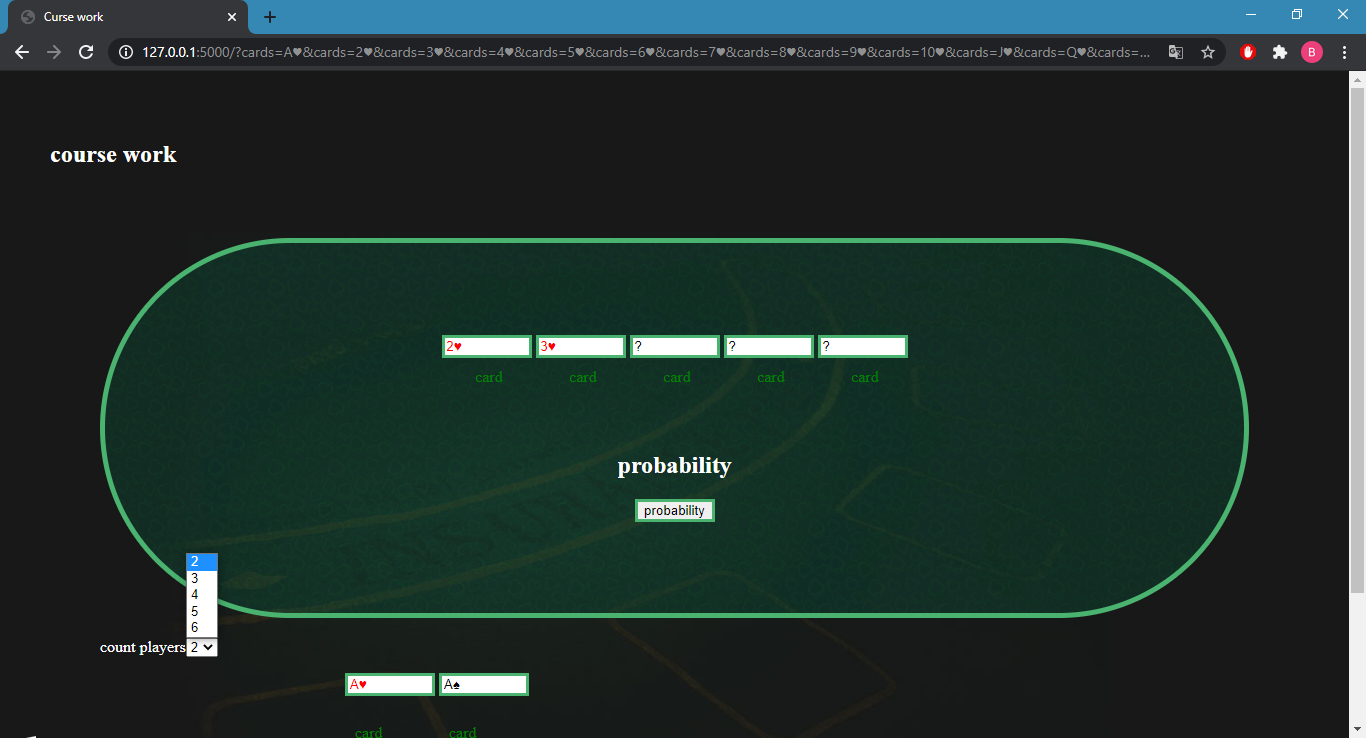
Чтобы понимать, как играть в покер, нужно запомнить старшинство данных покерных комбинаций:

1. **«Роял Стрит Флеш»** – 5 самых старших одномастных карт.
2. **«Стрит Флеш»** – 5 карт одной масти по порядку.
3. **«Каре»** – 4 карты одного ранга.
4. **«Фулл Хаус»** – комбинация, включающая в себя «Пару» и «Тройку» одновременно.
5. **«Флеш»** – 5 одномастных карт.
6. **«Стрит»** – 5 собранных по порядку карт любой масти.
7. **«Сет»** или **«Тройка»** – 3 карты одного ранга.
8. «**Две пары»** – 4 карты, среди которых собраны по 2 одинаковых по рангу.
9. **«Пара»** – это 2 одинаковые карты.
10. Самая младшая комбинация **«Старшая карта»** – это 1 карта. Чем она выше по рангу, тем вероятнее ее победа.
    1. **Заполнение данных**

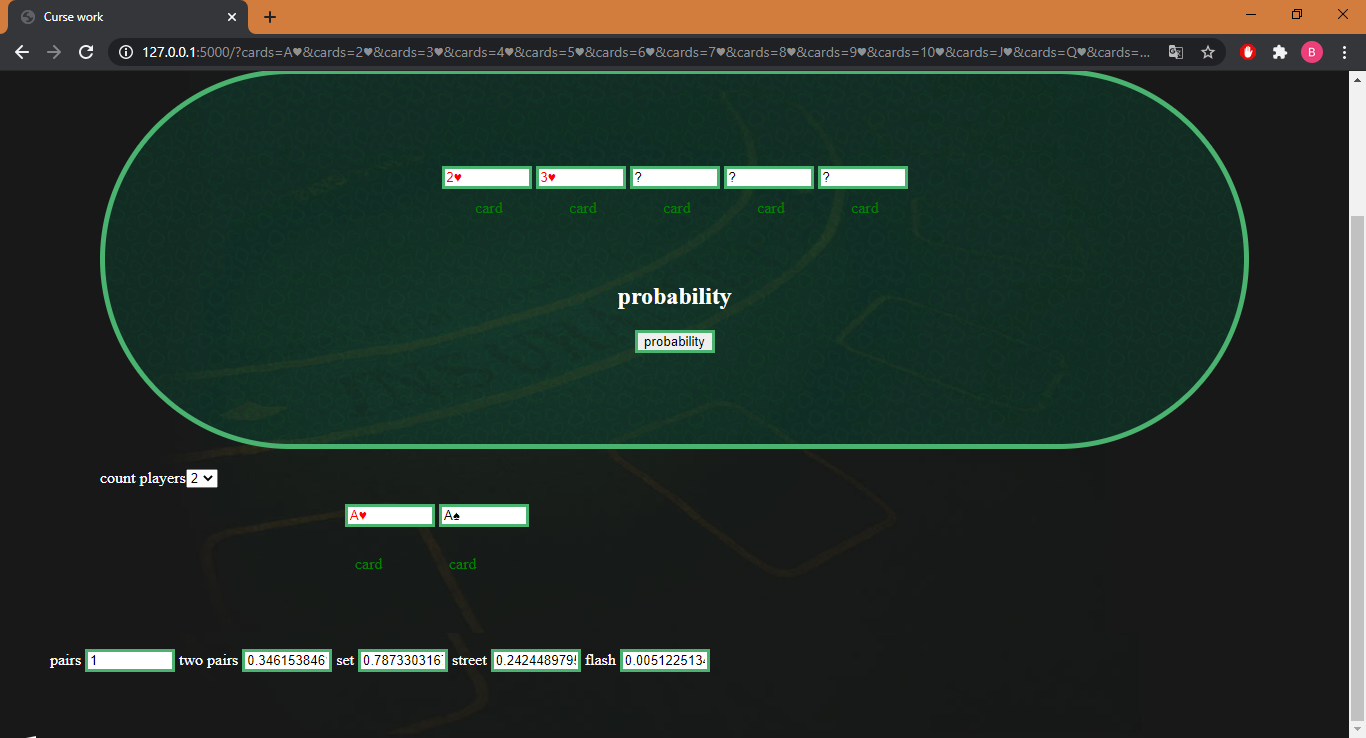
Перед тем как получить вероятность нужно выбрать 2 свои карты и карты на столе. Сделать это можно с помощью списка который будет выпадать при наведении курсора на вкладку card.



Затем можно выбрать количество игроков (по умолчаню их два).



Потом нажимаем кнопку probability. Ниже получаем вероятность выпадения комбинации, если она уже есть то значение будет 1.



**Заключение**

Во время выполнения курсового проекта были изучены такие технологии как Flask, Jinja2, HTML, css,. В курсовом проекте Flask использовался для создания сервера и визуализации шаблонов, Python для подсчета вероятностей, jinja2 для создания шаблонов, css для создания стилей.

Этот веб-сайт может быть выложена на хостинге . В будущем он имеет большие возможности для улучшения и продвижения в определенные массы. После доработки всех и добавления большего функционала, будет создан полезный сайт, который еще больше погрузит людей в игру.

# Список использованных источников

1. Программирование на языке Python, официальная документация по flask [Электронный ресурс]. – URL: http:// <https://flask.palletsprojects.com>
2. Stack Overflow — как [систему вопросов и ответов](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%B2%D0%BE%D0%BF%D1%80%D0%BE%D1%81%D0%BE%D0%B2_%D0%B8_%D0%BE%D1%82%D0%B2%D0%B5%D1%82%D0%BE%D0%B2_(%D0%B2%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81)) о [программировании](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [Электронный ресурс]. – URL: <https://ru.stackoverflow.com>
3. Html и css [Электронный ресурс]. URL: <http://htmlbook.ru>
4. Официальная документация Jinja2 [Электронный ресурс]. URL: https://jinja.palletsprojects.com
5. Курносенко, Н.М. Теория вероятностей и математическая статистика: практическое пособие для студентов экономических специальностей заочного факультета / Н.М. Курносенко, В.Е. Евдокимович ; Мин-во образования РБ, Гомельский гос. ун-т им. Ф. Скорины. - Гомель : ГГУ им. Ф.Скорины, 2010. - 154 с.

**Веб-Приложение**

**Код**

1. Теги для html страницы

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Curse work</title>

<link rel="stylesheet" href="{{ url\_for('static', filename ='css/style.css') }}">

</head>

<body>

<section class="poker-table--container" style>

<h2 style="color:white">course work</h2>

<form method="post" action="{{url\_for('calculate')}}">

<div class="poker-table--container">

<div class="poker-table">

<input type="text" name="tablecard1" value="{{diction['tablecard1']}}" class="inputsize"

style="color : {{diction['color1']}}">

<input type="text" name="tablecard2" value="{{diction['tablecard2']}}" class="inputsize"

style="color : {{diction['color2']}}">

<input type="text" name="tablecard3" value="{{diction['tablecard3']}}" class="inputsize"

style="color : {{diction['color3']}}">

<input type="text" name="tablecard4" value="{{diction['tablecard4']}}" class="inputsize"

style="color : {{diction['color4']}}">

<input type="text" name="tablecard5" value="{{diction['tablecard5']}}" class="inputsize"

style="color : {{diction['color5']}}">

<ul class="menu">

<li><a href=# style="color:green;">card</a>

<ul class="submenu">

<li><a href=# style="color: red "></a>

<ul>

{% for card in cards %}

{% if card[-1] in ['♠', '♣']%}

<li style="display: inline-block; left: 0;"><a

href="{{url\_for('tablecard1',value=card)}}"

style="color: black">{{card}}</a></li>

{% elif card[-1] in ['♥', '♦']%}

<li style="display: inline-block;"><a href="{{url\_for('tablecard1',value=card)}}"

style="color: red">{{card}}</a></li>

{% endif %}

{% endfor %}

</ul>

</li>

</ul>

</li>

<li><a href=# style="color:green;">card</a>

<ul class="submenu">

<li><a href=# style="color: red "></a>

<ul>

{% for card in cards %}

{% if card[-1] in ['♠', '♣']%}

<li style="display: inline-block; left: 0;"><a

href="{{url\_for('tablecard2',value=card)}}"

style="color: black">{{card}}</a></li>

{% elif card[-1] in ['♥', '♦']%}

<li style="display: inline-block;"><a href="{{url\_for('tablecard2',value=card)}}"

style="color: red">{{card}}</a></li>

{% endif %}

{% endfor %}

</ul>

</li>

</ul>

</li>

<li><a href=# style="color:green;">card</a>

<ul class="submenu">

<li><a href=# style="color: red "></a>

<ul>

{% for card in cards %}

{% if card[-1] in ['♠', '♣']%}

<li style="display: inline-block; left: 0;"><a

href="{{url\_for('tablecard3',value=card)}}"

style="color: black">{{card}}</a></li>

{% elif card[-1] in ['♥', '♦']%}

<li style="display: inline-block;"><a href="{{url\_for('tablecard3',value=card)}}"

style="color: red">{{card}}</a></li>

{% endif %}

{% endfor %}

</ul>

</li>

</ul>

</li>

<li><a href=# style="color:green;">card</a>

<ul class="submenu">

<li><a href=# style="color: red "></a>

<ul>

{% for card in cards %}

{% if card[-1] in ['♠', '♣']%}

<li style="display: inline-block; left: 0;"><a

href="{{url\_for('tablecard4',value=card)}}"

style="color: black">{{card}}</a></li>

{% elif card[-1] in ['♥', '♦']%}

<li style="display: inline-block;"><a href="{{url\_for('tablecard4',value=card)}}"

style="color: red">{{card}}</a></li>

{% endif %}

{% endfor %}

</ul>

</li>

</ul>

</li>

<li><a href=# style="color:green;">card</a>

<ul class="submenu">

<li><a href=# style="color: red "></a>

<ul>

{% for card in cards %}

{% if card[-1] in ['♠', '♣']%}

<li style="display: inline-block; left: 0;"><a

href="{{url\_for('tablecard5',value=card)}}"

style="color: black">{{card}}</a></li>

{% elif card[-1] in ['♥', '♦']%}

<li style="display: inline-block;"><a href="{{url\_for('tablecard5',value=card)}}"

style="color: red">{{card}}</a></li>

{% endif %}

{% endfor %}

</ul>

</li>

</ul>

</li>

</ul>

<br><br>

<h1 style="color: white">probability</h1>

<input type="submit" name="probability" value="probability" class="inputsize">

</div>

<p style="color: white"> count players<select name="players">

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

<option value="6">6</option>

</select></p>

<input type="text" name="card1" value="{{diction['card1']}}" class="inputsize "

style="color : {{diction['color11']}}; left: 245px;">

<input type="text" name="card2" value="{{diction['card2']}}" class="inputsize"

style="color : {{diction['color12']}}; left: 245px;">

<br>

<br>

<ul class="menu">

<li><a href=# style="color:green;">card</a>

<ul class="submenu">

<li><a href=# style="color: red "></a>

<ul>

{% for card in cards %}

{% if card[-1] in ['♠', '♣']%}

<li style="display: inline-block; left: 0;"><a

href="{{url\_for('card1',value=card)}}"

style="color: black">{{card}}</a></li>

{% elif card[-1] in ['♥', '♦']%}

<li style="display: inline-block;"><a href="{{url\_for('card1',value=card)}}"

style="color: red">{{card}}</a></li>

{% endif %}

{% endfor %}

</ul>

</li>

</ul>

</li>

<li><a href=# style="color:green;">card</a>

<ul class="submenu">

<li><a href=# style="color: red "></a>

<ul>

{% for card in cards %}

{% if card[-1] in ['♠', '♣']%}

<li style="display: inline-block; left: 0;"><a

href="{{url\_for('card2',value=card)}}"

style="color: black">{{card}}</a></li>

{% elif card[-1] in ['♥', '♦']%}

<li style="display: inline-block;"><a href="{{url\_for('card2',value=card)}}"

style="color: red">{{card}}</a></li>

{% endif %}

{% endfor %}

</ul>

</li>

</ul>

</li>

</ul>

</div>

<p style="color: white">pairs

<input type="text" name="tablecard1" value="{{diction['pairs']}}" class="inputsize">

two pairs

<input type="text" name="tablecard2" value="{{diction['two\_pairs']}}" class="inputsize">

set

<input type="text" name="tablecard3" value="{{diction['sets']}}" class="inputsize">

street

<input type="text" name="tablecard4" value="{{diction['street']}}" class="inputsize">

flash

<input type="text" name="tablecard5" value="{{diction['flash']}}" class="inputsize">

</p>

</form>

</section>

</body>

</html>

1. Css стили

.poker-table {

border: 5px solid #4ab36f;

padding: 8%;

border-radius: 10000px;

text-align: center;

margin: 0 0 20px;

background-image: url(../photos/table.png);

background-size: 60px auto;

}

.poker-table--container {

background: url(../photos/background.jpg) center center no-repeat;

background-size: cover;

padding: 50px;

}

.redtext{

color: red;

}

.container {

width: 90%;

max-width: 900px;

margin: 10px auto;

}

.inputsize{

border: 3px solid #4ab36f;

/\*float: center;\*/

align: center;

position: relative;

width: 80px;

}

body {

//background-image: url(../photos/table.png);

//font: 14px 'Verdana';

margin: 0;

padding: 0;

}

ul {

display: block;

margin: 0;

padding: 0;

list-style: none;

align: center;

}

ul:after {

display: block;

content: ' ';

clear: both;

float: none;

}

ul.menu > li {

float: left;

position: relative;

left: 245px;

width: 94px;

}

ul.menu > li > a {

display: block;

padding: 10px;

color: white;

//background-image: url(../photos/table.png);

text-decoration: none;

}

ul.menu > li > a:hover {

background-color: black;

}

ul.submenu {

display: none;

position: absolute;

width: 300px;

left: 0;

border: 1px solid #4ab36f;

background: white;

//margin: 0;

//display: inline;

}

ul.submenu > li {

display: block;

}

ul.submenu > li > a {

display: block;

/\*padding: 10px;

/\*color: red;\*/

text-decoration: none;

}

ul.submenu > li > a:hover {

text-decoration: underline;

}

ul.menu > li:hover > ul.submenu {

display: block;

}

1. **app.py (основной файл где происходит инициализация flask)**

from flask import Flask, render\_template, url\_for, redirect, request

from calculate import Probability

app = Flask(\_\_name\_\_)

cards = ['A♥', '2♥', '3♥', '4♥', '5♥', '6♥', '7♥', '8♥', '9♥', '10♥', 'J♥', 'Q♥', 'K♥',

'A♠', '2♠', '3♠', '4♠', '5♠', '6♠', '7♠', '8♠', '9♠', '10♠', 'J♠', 'Q♠', 'K♠',

'A♦', '2♦', '3♦', '4♦', '5♦', '6♦', '7♦', '8♦', '9♦', '10♦', 'J♦', 'Q♦', 'K♦',

'A♣', '2♣', '3♣', '4♣', '5♣', '6♣', '7♣', '8♣', '9♣', '10♣', 'J♣', 'Q♣', 'K♣']

suits = ['♥', '♠', '♦', '♣']

diction = {}

diction['card1'] = 'A♥'

diction['card2'] = 'A♠'

diction['tablecard1'] = '?'

diction['tablecard2'] = '?'

diction['tablecard3'] = '?'

diction['tablecard4'] = '?'

diction['tablecard5'] = '?'

diction['color11'] = 'red'

diction['color12'] = 'black'

diction['probability'] = 0

prob = Probability()

@app.route('/', methods=['GET', 'POST'])

def main():

return render\_template('main.html', cards=cards, suits=suits, diction=diction)

# card in hend

@app.route('/card1/<value>', methods=['GET', 'POST'])

def card1(value):

diction['card1'] = value

if value[-1] in ['♥', '♦']:

diction['color11'] = 'red'

else:

diction['color11'] = 'black'

return redirect(url\_for('main', cards=cards, suits=suits, diction=diction))

@app.route('/card2/<value>', methods=['GET', 'POST'])

def card2(value):

diction['card2'] = value

if value[-1] in ['♥', '♦']:

diction['color12'] = 'red'

else:

diction['color12'] = 'black'

return redirect(url\_for('main', cards=cards, suits=suits, diction=diction))

# talbecards

@app.route('/tablecard1/<value>', methods=['GET', 'POST'])

def tablecard1(value):

diction['tablecard1'] = value

if value[-1] in ['♥', '♦']:

diction['color1'] = 'red'

else:

diction['color1'] = 'black'

return redirect(url\_for('main', cards=cards, suits=suits, diction=diction))

@app.route('/tablecard2/<value>', methods=['GET', 'POST'])

def tablecard2(value):

diction['tablecard2'] = value

if value[-1] in ['♥', '♦']:

diction['color2'] = 'red'

else:

diction['color2'] = 'black'

return redirect(url\_for('main', cards=cards, suits=suits, diction=diction))

@app.route('/tablecard3/<value>', methods=['GET', 'POST'])

def tablecard3(value):

diction['tablecard3'] = value

if value[-1] in ['♥', '♦']:

diction['color3'] = 'red'

else:

diction['color3'] = 'black'

return redirect(url\_for('main', cards=cards, suits=suits, diction=diction))

@app.route('/tablecard4/<value>', methods=['GET', 'POST'])

def tablecard4(value):

diction['tablecard4'] = value

if value[-1] in ['♥', '♦']:

diction['color4'] = 'red'

else:

diction['color4'] = 'black'

return redirect(url\_for('main', cards=cards, suits=suits, diction=diction))

@app.route('/tablecard5/<value>', methods=['GET', 'POST'])

def tablecard5(value):

diction['tablecard5'] = value

if value[-1] in ['♥', '♦']:

diction['color5'] = 'red'

else:

diction['color5'] = 'black'

return redirect(url\_for('main', cards=cards, suits=suits, diction=diction))

# culculate probability

@app.route('/calculate', methods=['GET', 'POST'])

def calculate():

dealler = cards[:]

diction['players'] = int(request.form.get('players'))

diction['card1'] = request.form.get('card1')

diction['card2'] = request.form.get('card2')

for i in range(1, 6):

diction['tablecard' + str(i)] = request.form.get('tablecard' + str(i))

print(diction)

diction['pairs'] = prob.pairs(diction)['prob\_all']

diction['two\_pairs'] = prob.two\_pairs(diction)['prob\_all']

diction['sets'] = prob.sets(diction)['prob\_all']

diction['street'] = prob.street(diction)['prob\_all']

diction['flash'] = prob.flash(diction)['prob\_all']

return redirect(url\_for('main', cards=cards, suits=suits, diction=diction))

1. calculate.py. Здесь размесчен класс Provbability который содержит функции подсчета вероятностей.

class Probability():

def pairs(self, diction):

combin = {}

combin['prob\_pairs'] = 0

combin['pocked'] = 0

tablecards = [diction['tablecard' + str(i)] for i in range(1, 6)]

tablebefore = [card[:-1] for card in tablecards if card != '?']

print('pairs' + str(tablebefore))

free\_pos = 5 - sum([1 for i in tablecards if i != '?'])

if diction['card1'][:-1] == diction['card2'][:-1]:

combin['card'] = diction['card1'][:-1]

combin['prob\_pairs'] = 1

combin['pocked'] = 1

combin['prob\_all'] = 1

return combin

elif len(set(tablebefore)) != len(tablebefore):

for tablecard in tablebefore:

if tablebefore.count(tablecard) > 1:

combin['card'] = tablecard

combin['prob\_pairs'] = 1

combin['prob\_all'] = 1

return combin

else:

for card in [diction['card1'], diction['card2']]:

if tablebefore.count(card[:-1]) == 1:

combin['card'] = card

combin['pocked'] = 0

combin['prob\_pairs'] = 1

combin['prob\_all'] = 1

return combin

combin['prob\_pairs'] += 3 / (52 - (5 - free\_pos) + (diction['players'] - 1) \* 2) \* free\_pos

combin['prob\_all'] = combin['prob\_pairs'] + 3 / (

52 - (5 - free\_pos) + (diction['players'] - 1) \* 2) \* free\_pos \* (5 - free\_pos)

return combin

def two\_pairs(self, diction):

combin = {}

combin['prob\_two\_pairs'] = 0

tablecards = [diction['tablecard' + str(i)] for i in range(1, 6)]

tablebefore = [card[:-1] for card in tablecards if card != '?']

free\_pos = 5 - sum([1 for i in tablecards if i != '?'])

pairs = self.pairs(diction)

if pairs['pocked'] == 1:

if len(set(tablebefore)) != len(tablebefore):

combin['prob\_two\_pairs'] = 1

combin['prob\_all'] = 1

return combin

else:

combin['prob\_two\_pairs'] = 3 / (52 - (5 - free\_pos) + (diction['players'] - 1) \* 2) \* free\_pos

combin['prob\_all'] = combin['prob\_two\_pairs'] \* (5 - free\_pos)

return combin

else:

if tablebefore.count(diction['card1'][:-1]) == 1 and tablebefore.count(diction['card2'][:-1]) == 1:

combin['prob\_two\_pairs'] = 1

combin['prob\_all'] = 1

return combin

elif (tablebefore.count(diction['card1'][:-1]) == 1 or tablebefore.count(diction['card1'][:-1])):

if len(set(tablebefore)) != len(tablebefore):

combin['prob\_two\_pairs'] = 1

combin['prob\_all'] = 1

return combin

else:

combin['prob\_two\_pairs'] = 3 / (52 - (5 - free\_pos) + (diction['players'] - 1) \* 2) \* free\_pos

combin['prob\_all'] = combin['prob\_two\_pairs'] \* (4 - free\_pos)

return combin

elif len(set(tablebefore)) != len(tablebefore):

for tablecard in tablebefore:

if tablebefore.count(tablecard) == 2 and len(

[card for card in tablebefore if card != tablecard]) != len(

set([card for card in tablebefore if card != tablecard])):

combin['prob\_two\_pairs'] = 1

combin['prob\_all'] = 1

return combin

combin['prob\_two\_pairs'] = 3 / (52 - (5 - free\_pos) + (diction['players'] - 1) \* 2) \* free\_pos

combin['prob\_all'] = combin['prob\_two\_pairs'] \* (4 - free\_pos)

return combin

else:

if free\_pos <= 1:

combin['prob\_all'] = 0

return combin

else:

combin['prob\_two\_pairs'] += 3 / (

52 - (5 - free\_pos) + (diction['players'] - 1) \* 2) \* free\_pos \* 3 / (

52 - (5 - free\_pos)) \* free\_pos

combin['prob\_all'] = combin['prob\_two\_pairs'] \* (5 - free\_pos)

return combin

def sets(self, diction):

combin = {}

combin['prob\_set'] = 0

combin['pocked'] = 0

tablecards = [diction['tablecard' + str(i)] for i in range(1, 6)]

tablebefore = [card[:-1] for card in tablecards if card != '?']

free\_pos = 5 - sum([1 for i in tablecards if i != '?'])

print('set' + str(tablebefore))

pairs = self.pairs(diction)

if pairs['prob\_pairs'] == 1 and tablebefore.count(pairs['card'][:-1]) == 1:

combin['prob\_set'] = 1

combin['prob\_all'] = 1

return combin

elif len(set(tablebefore)) == 5 - free\_pos - 2 and self.two\_pairs(diction)['prob\_two\_pairs'] != 1:

combin['prob\_set'] = 1

combin['prob\_all'] = 1

return combin

elif len(set(tablebefore)) == 5 - free\_pos - 2 and self.two\_pairs(diction)['prob\_two\_pairs'] == 1:

combin['prob\_set'] += 2 / (52 - 1 - (5 - free\_pos) + (diction['players'] - 1) \* 2) \* free\_pos

combin['prob\_all'] = combin['prob\_set'] \* 2

return combin

else:

for card in [diction['card1'], diction['card2']]:

if tablebefore.count(card[:-1]) == 2:

combin['prob\_set'] = 1

combin['prob\_all'] = 1

return combin

else:

for tablecard in tablecards:

if tablebefore.count(card[:-1]) == 2:

combin['prob\_set'] = 1

combin['prob\_all'] = 1

return combin

elif card[:-2] == tablecard[:-1]:

combin['prob\_set'] += 2 / (

52 - 1 - (5 - free\_pos) + (diction['players'] - 1) \* 2) \* free\_pos

else:

combin['prob\_set'] += 3 / (

52 - (5 - free\_pos) + (diction['players'] - 1) \* 2) \* free\_pos \* 2 / (

52 - 1 - (5 - free\_pos) + (

diction['players'] - 1) \* 2) \* free\_pos

if free\_pos == 2:

combin['prob\_all'] = combin['prob\_set'] + 3 / (

52 - (5 - free\_pos) + (diction['players'] - 1) \* 2) \* free\_pos \* 2 / (

52 - 1 - (5 - free\_pos) + (diction['players'] - 1) \* 2) \* free\_pos \* (

5 - free\_pos)

return combin

else:

combin['prob\_all'] = combin['prob\_set']

return combin

def street(self, diction):

# street= ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K']

combin = {}

combin['prob\_street'] = 0

combin['prob\_all'] = 0

combin['pocked'] = 0

tablecards = [diction['tablecard' + str(i)] for i in range(1, 6)]

tablecards.append(diction['card1'])

tablecards.append(diction['card2'])

tablebefore = [card[:-1] for card in tablecards if card != '?']

print('street' + str(tablebefore))

for i in range(len(tablebefore)):

if tablebefore[i] == 'A':

if '2' not in tablebefore and 2 not in tablebefore:

tablebefore[i] = 14

else:

tablebefore[i] = 1

if tablebefore[i] == 'Q':

tablebefore[i] = 12

if tablebefore[i] == 'Q':

tablebefore[i] = 12

if tablebefore[i] == 'J':

tablebefore[i] = 11

else:

tablebefore[i] = int(tablebefore[i])

print('street' + str(tablebefore))

free\_pos = 5 - sum([1 for i in tablecards if i != '?'])

len\_street = 1

maxlen = 1

setcards = list(set(tablebefore))

for i in range(len(setcards) - 1):

if int(setcards[i]) == int(setcards[i + 1]) - 1:

len\_street += 1

if len\_street > maxlen:

maxlen = len\_street

else:

len\_street = 1

if maxlen == 5:

combin['prob\_all'] = 1

elif 5 - maxlen < free\_pos:

combin['prob\_all'] = 0

else:

combin['prob\_all'] = sum(

[3 / (52 - i + 1 - (5 - free\_pos) + (diction['players'] - 1) \* 2) for i in range(1, 6 - maxlen)]) \* 2

return combin

def flash(self, diction):

# street= ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K']

combin = {}

combin['prob\_flash'] = 0

combin['prob\_all'] = 0

combin['pocked'] = 0

tablecards = [diction['tablecard' + str(i)] for i in range(1, 6)]

tablecards.append(diction['card1'])

tablecards.append(diction['card2'])

tablebefore = [card[-1] for card in tablecards if card != '?']

print(tablebefore)

free\_pos = 7 - sum([1 for i in tablecards if i != '?'])

len\_flash = 1

maxlen = 1

for suit in tablebefore:

len\_street = tablebefore.count(suit)

if len\_street > maxlen:

maxlen = len\_street

if maxlen == 5:

combin['prob\_all'] = 1

elif 5 - maxlen > free\_pos:

print('here')

combin['prob\_all'] = 0

else:

combin['prob\_all'] = 1

for i in range(free\_pos):

combin['prob\_all'] \*= (13 - i - maxlen) / (52 - i + 1 - (5 - free\_pos) + (diction['players'] - 1) \* 2)

return combin