https://github.com/dagoncharov/Snake-game/blob/main/snake.py

Для того чтобы запустить проект, который находится на GitHub, вам потребуется клонировать репозиторий и выполнить несколько шагов. Предполагается, что у вас уже установлен Python и pip.

1. **Клонирование репозитория:**
   * Откройте терминал (или командную строку) на вашем компьютере.
   * Выполните следующую команду для клонирования репозитория:

bashCopy code

git clone https://github.com/username/repository.git

Замените **https://github.com/username/repository.git** на фактическую ссылку на репозиторий.

1. **Переход в директорию проекта:**
   * Перейдите в директорию, которую только что клонировали:

bashCopy code

cd repository

Замените **repository** на название директории вашего проекта.

1. **Установка зависимостей:**
   * Убедитесь, что у вас установлен **pipenv**. Если нет, установите его следующей командой:

bashCopy code

pip install pipenv

* + Затем выполните команду для установки зависимостей из файла **Pipfile.lock**:

bashCopy code

pipenv install

1. **Активация виртуальной среды:**
   * Активируйте виртуальную среду с помощью команды:

bashCopy code

pipenv shell

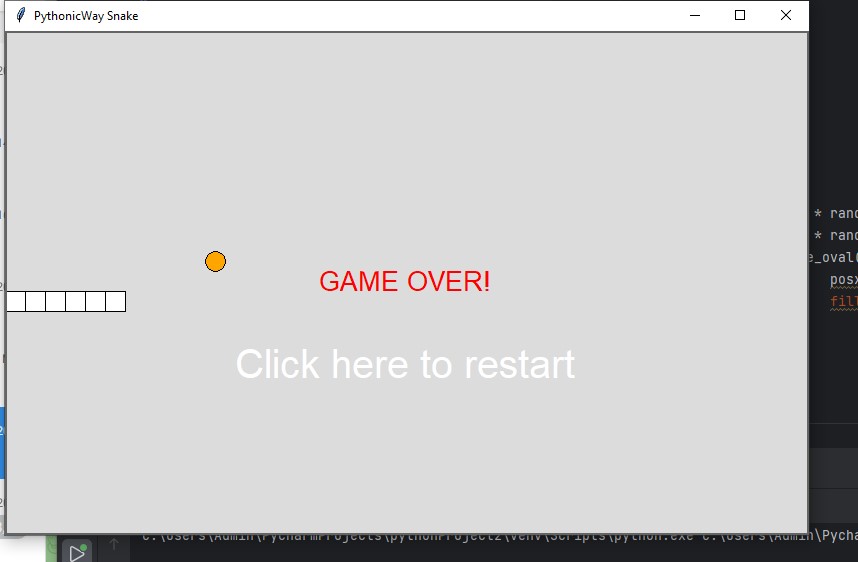
1. **Запуск проекта:**
   * В вашем проекте, вероятно, есть файл, вроде **main.py** или **app.py**, который содержит точку входа в приложение. Выполните его:

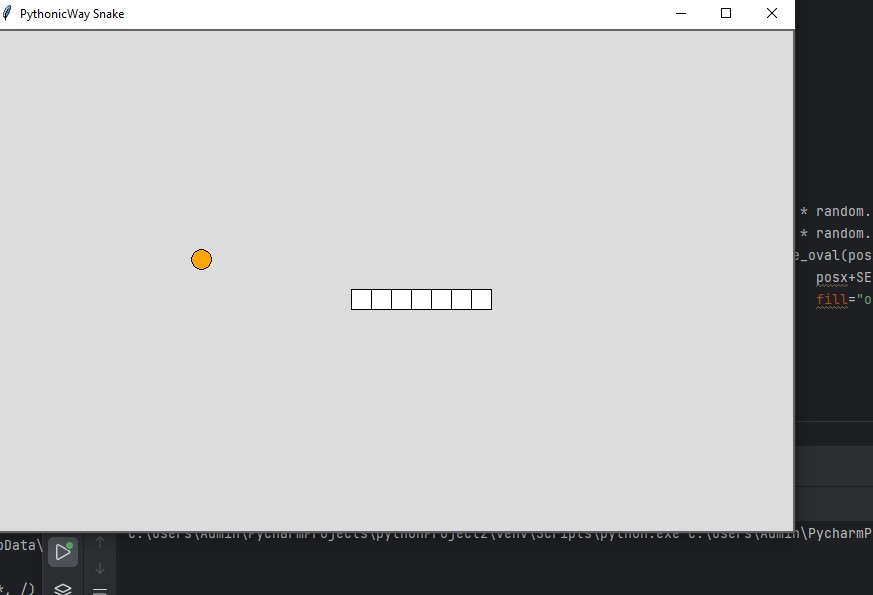
bashCopy code

python main.py

Замените **main.py** на фактическое имя файла вашего приложения.

Теперь ваш проект должен быть запущен. Обратите внимание, что настройки могут немного отличаться в зависимости от структуры проекта и используемых инструментов.





Чтобы запустить игру "Змейка" из проекта, хранящегося на GitHub, вам понадобится выполнить следующие шаги:  
  
1. Склонируйте репозиторий: Скопируйте URL репозитория с игрой "Змейка" на GitHub. Затем откройте командную строку или терминал на вашем компьютере и используйте команду git clone с указанием скопированного URL для клонирования репозитория на ваш локальный компьютер.  
  
Пример:  
````shell  
   git clone https://github.com/username/repository.git  
```  
  
2. Перейдите в директорию проекта: В командной строке или терминале перейдите в директорию, в которую был склонирован репозиторий игры "Змейка".  
  
Пример:  
````shell  
   cd repository  
```  
  
3. Установите зависимости (если необходимо): Если в проекте используются дополнительные зависимости или библиотеки, убедитесь, что они установлены на вашем компьютере. Для установки зависимостей обычно используется файл requirements.txt или setup.py. Выполните команду установки зависимостей в вашем виртуальном окружении или глобально.  
  
Пример:  
````shell  
   pip install -r requirements.txt  
```  
  
4. Запустите игру: Выполните команду для запуска игры. В зависимости от структуры проекта и языка программирования, используемого в игре, команда запуска может отличаться.  
  
Пример (Python):  
````shell  
   python game.py  
```  
  
Пример (JavaScript):  
````  
   node game.js  
```

**Общее описание**

Этот код реализует простую игру "Змейка" с использованием графической библиотеки tkinter в языке программирования Python. Игрок управляет змеей на игровом поле, стараясь съесть шарики (еду) и увеличивая длину змеи. Игра завершается, если змея сталкивается с границами поля или самой собой.

**Глобальные переменные**

* **WIDTH**: Ширина игрового поля.
* **HEIGHT**: Высота игрового поля.
* **SEG\_SIZE**: Размер сегмента змеи.
* **IN\_GAME**: Переменная, отвечающая за состояние игры (True - игра продолжается, False - завершена).

**Классы**

**Segment**

* **Описание:** Представляет собой сегмент змеи.
* **Методы:**
  + **\_\_init\_\_(self, x, y)**: Инициализирует сегмент змеи с координатами **(x, y)** и создает прямоугольник на игровом поле.

**Snake**

* **Описание:** Представляет змею и управляет ею.
* **Методы:**
  + **\_\_init\_\_(self, segments)**: Инициализирует змею с заданными сегментами и устанавливает начальное направление движения вправо.
  + **move(self)**: Отвечает за движение змеи в текущем направлении.
  + **add\_segment(self)**: Добавляет новый сегмент змеи при съедании еды.
  + **change\_direction(self, event)**: Изменяет направление движения змеи в ответ на события клавиш.
  + **reset\_snake(self)**: Удаляет все сегменты змеи.

**GameField**

* **Описание:** Представляет игровое поле и управляет графикой.
* **Методы:**
  + **\_\_init\_\_(self, canvas, width, height)**: Инициализирует игровое поле с указанным холстом, шириной и высотой.
  + **draw\_border(self)**: Рисует границы игрового поля.
  + **draw\_text(self, x, y, text, font='Arial 12', fill='black')**: Рисует текст на игровом поле.
  + **clear(self)**: Очищает игровое поле.
  + **update(self)**: Обновляет отображение игрового поля.

**Функции**

**create\_block()**

* **Описание:** Создает новый шарик (еду) на игровом поле в случайном месте.

**main()**

* **Описание:** Главная функция управления логикой игры.
* **Действия:**
  + Проверяет столкновение с границами поля, съедание шарика и самопересечение змеи.
  + Вызывается снова через **root.after(100, main)**, чтобы обновлять состояние игры каждые 100 миллисекунд.

**clicked(event)**

* **Описание:** Обработчик события клика мыши.
* **Действия:**
  + Сбрасывает состояние игры и запускает новую игру.

**start\_game()**

* **Описание:** Инициализирует начальные условия игры, создает змею и еду, а также устанавливает обработчик клавиш.

**create\_snake()**

* **Описание:** Создает экземпляр змеи с тремя сегментами в начальных позициях.

**Инициализация и запуск**

* Создается основное окно tkinter.
* Инициализируется холст (**Canvas**) для отображения игры.
* Создается экземпляр класса **GameField** для управления графикой игрового поля.
* Устанавливаются обработчики событий для клавиш и мыши.
* Запускается цикл обработки событий tkinter (**root.mainloop()**).

**Примечание:**

* Код частично представлен в сокращенной форме, и для полной понимаемости, возможно, потребуется дополнительный контекст или детализация.

Начало формы

**from tkinter import Tk, Canvas**

**import random**

**WIDTH = 800**

**HEIGHT = 500**

**SEG\_SIZE = 20**

**IN\_GAME = True**

**class Segment(object):**

**# Класс, представляющий сегмент змеи**

**def \_\_init\_\_(self, x, y):**

**self.instance = c.create\_rectangle(x, y, x+SEG\_SIZE, y+SEG\_SIZE, fill="white")**

**class Snake(object):**

**# Класс, представляющий змею**

**def \_\_init\_\_(self, segments):**

**self.segments = segments**

**self.mapping = {"Down": (0, 1), "Right": (1, 0), "Up": (0, -1), "Left": (-1, 0)}**

**self.vector = self.mapping["Right"]**

**def move(self):**

**# Движение змеи**

**for index in range(len(self.segments)-1):**

**segment = self.segments[index].instance**

**x1, y1, x2, y2 = c.coords(self.segments[index+1].instance)**

**c.coords(segment, x1, y1, x2, y2)**

**x1, y1, x2, y2 = c.coords(self.segments[-2].instance)**

**c.coords(self.segments[-1].instance,**

**x1+self.vector[0]\*SEG\_SIZE, y1+self.vector[1]\*SEG\_SIZE,**

**x2+self.vector[0]\*SEG\_SIZE, y2+self.vector[1]\*SEG\_SIZE)**

**# Остальные методы класса Snake аналогичны по структуре - управление змеей.**

**class GameField(object):**

**# Класс, представляющий игровое поле**

**def \_\_init\_\_(self, canvas, width, height):**

**self.canvas = canvas**

**self.width = width**

**self.height = height**

**def draw\_border(self):**

**# Рисование границ поля**

**self.canvas.create\_rectangle(0, 0, self.width, self.height, outline="black", width=2)**

**def draw\_text(self, x, y, text, font='Arial 12', fill='black'):**

**# Рисование текста на поле**

**self.canvas.create\_text(x, y, text=text, font=font, fill=fill)**

**def clear(self):**

**# Очистка игрового поля**

**self.canvas.delete("all")**

**def update(self):**

**# Обновление игрового поля**

**self.clear()**

**self.draw\_border()**

**def create\_block():**

**# Создание шарика (еды)**

**global BLOCK**

**posx = SEG\_SIZE \* random.randint(1, (WIDTH-SEG\_SIZE) // SEG\_SIZE)**

**posy = SEG\_SIZE \* random.randint(1, (HEIGHT-SEG\_SIZE) // SEG\_SIZE)**

**BLOCK = c.create\_oval(posx, posy, posx+SEG\_SIZE, posy+SEG\_SIZE, fill="orange")**

**def main():**

**global IN\_GAME**

**if IN\_GAME:**

**# Логика игры при активном состоянии**

**s.move()**

**head\_coords = c.coords(s.segments[-1].instance)**

**x1, y1, x2, y2 = head\_coords**

**# Проверки на столкновения и взаимодействия**

**if x2 > WIDTH or x1 < 0 or y1 < 0 or y2 > HEIGHT:**

**IN\_GAME = False**

**elif head\_coords == c.coords(BLOCK):**

**s.add\_segment()**

**c.delete(BLOCK)**

**create\_block()**

**else:**

**for index in range(len(s.segments)-1):**

**if head\_coords == c.coords(s.segments[index].instance):**

**IN\_GAME = False**

**root.after(100, main)**

**else:**

**# Обработка завершения игры**

**game\_field.draw\_text(WIDTH/2, HEIGHT/2, "GAME OVER!", font='Arial 20', fill='red')**

**def clicked(event):**

**# Обработчик клика мыши**

**global IN\_GAME**

**s.reset\_snake()**

**IN\_GAME = True**

**c.delete(BLOCK)**

**game\_field.clear()**

**game\_field.update()**

**start\_game()**

**def start\_game():**

**# Начало новой игры**

**global s**

**create\_block()**

**s = create\_snake()**

**c.bind("<KeyPress>", s.change\_direction)**

**main()**

**def create\_snake():**

**# Создание экземпляра змеи**

**segments = [Segment(SEG\_SIZE, SEG\_SIZE),**

**Segment(SEG\_SIZE\*2, SEG\_SIZE),**

**Segment(SEG\_SIZE\*3, SEG\_SIZE)]**

**return Snake(segments)**

**root = Tk()**

**root.title("PythonicWay Snake")**

**c = Canvas(root, width=WIDTH, height=HEIGHT, bg="#DCDCDC")**

**c.grid()**

**game\_field = GameField(c, WIDTH, HEIGHT)**

**game\_field.update()**

**c.focus\_set()**

**c.bind("<Button-1>", clicked)**

**root.mainloop()**