Создание графического окна приложения.

Мы будем работать с библиотекой tkinter, в котором окно задается следующим образом:

[?](http://pythonicway.com/python-games/python-arcade/18-python-snake)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | from tkinter import \*    # Создаем окно  root = Tk()  # Устанавливаем название окна  root.title("PythonicWay Snake")    # Запускаем окно  root.mainloop() |

 Обратите внимание, что весь остальной код должен находиться до строки root.mainloop()

Объявление вспомогательных глобальных переменных.

Тут все просто, для облегчения последующей работы создадим следующие [переменные](http://pythonicway.com/python-basic-syntax):

[?](http://pythonicway.com/python-games/python-arcade/18-python-snake)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | # ширина экрана  WIDTH = 800  # высота экрана  HEIGHT = 600  # Размер сегмента змейки  SEG\_SIZE = 20  # Переменная отвечающая за состояние игры  IN\_GAME = True |

Установка на окне области для рисования.

Область для рисования в tkinter реализована при помощи класса Canvas, им и воспользуемся.

[?](http://pythonicway.com/python-games/python-arcade/18-python-snake)

|  |  |
| --- | --- |
| 1  2  3  4  5 | # создаем экземпляр класса Canvas (его мы еще будем использовать) и заливаем все зеленым цветом  c = Canvas(root, width=WIDTH, height=HEIGHT, bg="#003300")  c.grid()  # Наводим фокус на Canvas, чтобы мы могли ловить нажатия клавиш  c.focus\_set() |

Если вы все делали правильно, то запустив полученный код получите следующую картину



Создание классов сегмента и змеи:

Класс сегмента змейки.

Сегмент змейки будет простым прямоугольником, созданным при помощи метода create\_rectangle класса Canvas модуля tkinter.

[?](http://pythonicway.com/python-games/python-arcade/18-python-snake)

|  |  |
| --- | --- |
| 1  2  3  4  5 | class Segment(object):      def \_\_init\_\_(self, x, y):          self.instance = c.create\_rectangle(x, y,                           x+SEG\_SIZE, y+SEG\_SIZE,                           fill="white") |

Класс змейки.

**Змейка** у нас будет набором сегментов. У нее будут методы движения, изменения направления и добавления сегмента.

[?](http://pythonicway.com/python-games/python-arcade/18-python-snake)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51 | class Snake(object):      def \_\_init\_\_(self, segments):          self.segments = segments            # список доступных направлений движения змейки          self.mapping = {"Down": (0, 1), "Up": (0, -1),                                  "Left": (-1, 0), "Right": (1, 0) }          # изначально змейка двигается вправо          self.vector = self.mapping["Right"]        def move(self):           """ Двигает змейку в заданном направлении """             # перебираем все сегменты кроме первого           for index in range(len(self.segments)-1):                segment = self.segments[index].instance                x1, y1, x2, y2 = c.coords(self.segments[index+1].instance)                # задаем каждому сегменту позицию сегмента стоящего после него                c.coords(segment, x1, y1, x2, y2)             # получаем координаты сегмента перед "головой"           x1, y1, x2, y2 = c.coords(self.segments[-2].instance)             # помещаем "голову" в направлении указанном в векторе движения           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)        def change\_direction(self, event):          """ Изменяет направление движения змейки """            # event передаст нам символ нажатой клавиши          # и если эта клавиша в доступных направлениях          # изменяем направление          if event.keysym in self.mapping:              self.vector = self.mapping[event.keysym]        def add\_segment(self):          """ Добавляет сегмент змейке """            # определяем последний сегмент          last\_seg = c.coords(self.segments[0].instance)            # определяем координаты куда поставить следующий сегмент          x = last\_seg[2] - SEG\_SIZE          y = last\_seg[3] - SEG\_SIZE            # добавляем змейке еще один сегмент в заданных координатах          self.segments.insert(0, Segment(x, y)) |

Если вы осилили эти два класса и более того, поняли что, как и почему, то поздравляю - самая трудная часть позади. Уже сейчас можно создавать змейку. Вставте следующие строчки, но обязательно после строк c.grid()

[?](http://pythonicway.com/python-games/python-arcade/18-python-snake)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # создаем набор сегментов  segments = [Segment(SEG\_SIZE, SEG\_SIZE),              Segment(SEG\_SIZE\*2, SEG\_SIZE),              Segment(SEG\_SIZE\*3, SEG\_SIZE)]    # собственно змейка  s = Snake(segments) |

Вот так выглядит наша игра на данный момент.



Создание вспомогательных функций.

Для начала напишем [функцию](http://pythonicway.com/python-functions) создания яблок (или что там наша змея будет есть). Не забудьте импортировать модуль random, чтобы все работало

[?](http://pythonicway.com/python-games/python-arcade/18-python-snake)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | def create\_block():      """ Создает блок в случайной позиции на карте """      global BLOCK      posx = SEG\_SIZE \* (random.randint(1, (WIDTH-SEG\_SIZE) / SEG\_SIZE))      posy = SEG\_SIZE \* (random.randint(1, (HEIGHT-SEG\_SIZE) / SEG\_SIZE))        # блок это кружочек красного цвета      BLOCK = c.create\_oval(posx, posy,                            posx + SEG\_SIZE,                            posy + SEG\_SIZE,                            fill="red") |

Теперь основная функция main, которая будет управлять игровым процессом.

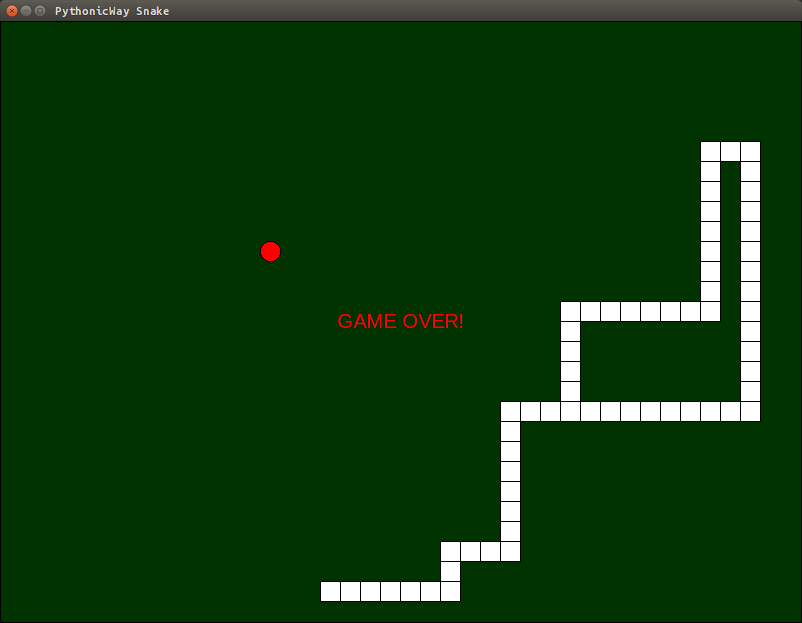
[?](http://pythonicway.com/python-games/python-arcade/18-python-snake)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | def main():      global IN\_GAME        if IN\_GAME:          # Двигаем змейку          s.move()            # Определяем координаты головы          head\_coords = c.coords(s.segments[-1].instance)          x1, y1, x2, y2 = head\_coords            # Столкновение с границами экрана          if x1 < 0 or x2 > WIDTH or y1 < 0 or y2 > HEIGHT:              IN\_GAME = False            # Поедание яблок          elif head\_coords == c.coords(BLOCK):              s.add\_segment()              c.delete(BLOCK)              c.create\_block()            # Самоедство          else:              # Проходим по всем сегментам змеи              for index in range(len(s.segments)-1):                  if c.coords(s.segments[index].instance) == head\_coords:                      IN\_GAME = False        # Если не в игре выводим сообщение о проигрыше      else:            c.create\_text(WIDTH/2, HEIGHT/2,                                text="GAME OVER!",                                font="Arial 20",                                fill="#ff0000") |

 Последний шаг - правильно обработать нажатия клавиш. Привяжем метод класса Snake change\_direction() к Canvas:

[?](http://pythonicway.com/python-games/python-arcade/18-python-snake)

|  |  |
| --- | --- |
| 1 | c.bind("<KeyPress>", s.change\_direction) |



Вот такая у нас получилась **змейка на Python**. Попробуйте немного усовершенствовать игру, например, добавить счет, ускорение игры, отредактировать функцию create\_block чтобы новые яблоки не появлялись на самой змейке, или реализовать систему уровней.