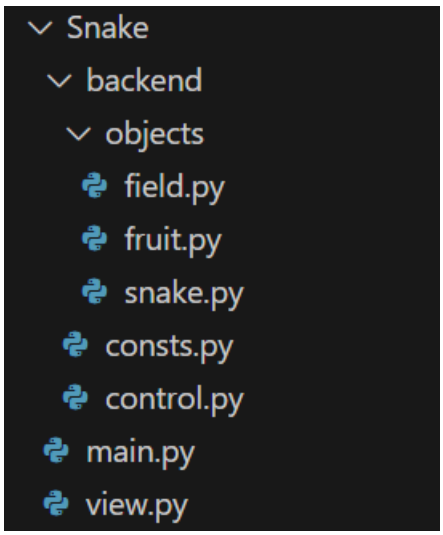
Опираясь на код, написанный на занятии (см. Иллюстрации ниже), реализуйте следующее задание:

1. Необходимо создать счётчик количества очков (количество съеденных фруктов) пользователя.

2. После столкновения змейки со стенкой выводить на экран количество очков пользователя при помощи всплывающего окна.

Код программы с занятия:

Программа имеет следующую файловую структуру:



# Main.py:

```python
import pygame
import sys
import backend.consts as const
from backend.objects.field import Field
from backend.objects.snake import Snake
from backend.objects.fruit import Fruit
from backend.control import Control
from view import View

class Main:
    def __init__(self):

        pygame.init()
        self.WINDOW_SIZE = const.WINDOW_SIZE
        self.screen = pygame.display.set_mode(self.WINDOW_SIZE)
        self.view = View(self.screen)
        self.snake = Snake()
        self.gameField = Field(400,760,40,40)
        self.fruit = Fruit ()
        self.control = Control(self.view,self.snake,self.fruit,self.gameField)

    def gameLoop(self):
        game = True
        while game:
            for event in pygame.event.get(): # получаем действия от игрока
                if event.type == pygame.QUIT: # если был нажат крестик
                    pygame.quit() # выйти из библиотеки
                    sys.exit() # завершить работу программы

            game=self.control.backProcess()
            self.control.frontProcess()
            pygame.display.flip()
            pygame.time.Clock().tick(7)

if __name__ == '__main__':
    Main().gameLoop()
```

# View.py:

```python
from pygame import draw

class View:
    def __init__(self,screen):
        self.screen = screen

    def printRectList(self,rectList,color,width_borders = 0):
        for rect in rectList:
            draw.rect(self.screen, color, rect,width_borders)
```

# Consts.py:

```python
YELLOW = (255,255,0)
WHITE = (255,255,255)
BLACK = (0,0,0)
DARK = (128,128,128)

RECT_SIZE = 40

WINDOW_SIZE = (400, 760)
```

# Controls.py:

```python
from pygame import key
from pygame.locals import *
from .consts import *

# проверка вхождения одного списка в другой
def check_unification(list1,list2):
    for element in list1:
        if element in list2:
            return True
    else:
        return False

class Control:
    def __init__(self,view,snake,fruit,gameField):
        self.view=view
        self.snake=snake
        self.fruit = fruit
        self.gameField = gameField
        self.returnValue = True

    def backProcess(self):

        keys = key.get_pressed()
        action = None

        if keys[K_LEFT]: action = 'LEFT'
        if keys[K_RIGHT]: action = 'RIGHT'
        if keys[K_UP]: action = 'UP'
        if keys[K_DOWN]: action = 'DOWN'

        self.snake.setDirection(action)
        self.snake.moving()

        if check_unification([self.fruit.getRect()],self.snake.getListRects())==True:
            self.fruit.changePosition()
            self.snake.grow()

        self.returnValue = self.checkBorders()

        return self.returnValue

    def frontProcess(self):
        self.view.screen.fill(DARK)
        self.view.printRectList(self.gameField.getListRects(), WHITE,1)
        self.view.printRectList([self.fruit.getRect()],self.fruit.color)
        self.view.printRectList(self.snake.getListRects(),self.snake.color)

    def checkBorders(self):
        returnValue=True
        for rect in self.snake.getListRects():
            if ((rect.x+RECT_SIZE) > WINDOW_SIZE[0]) or rect.x < 0:
                returnValue = False
            elif ((rect.y+RECT_SIZE) > WINDOW_SIZE[1]) or rect.y < 0:
                returnValue = False
        return returnValue
```

# field.py

```python
from pygame import Rect

class Field:
    def __init__(self,window_x,window_y,width,height):

        self._rects = []

        for x in range(0,window_x+1,width):
            for y in range(0,window_y+1,height):
                self._rects.append(Rect(x, y, width, height))

    def getListRects(self):
        return self._rects
```

# Fruit.py

```python
from ..consts import RECT_SIZE,YELLOW
from pygame import Rect
from random import randint

class Fruit:
    def __init__(self):
        self._rect=Rect(randint(0,9)*RECT_SIZE,randint(0,19)*RECT_SIZE,RECT_SIZE,RECT_SIZE)
        self.color = YELLOW

    def changePosition(self):
        self._rect.x = randint(0,9) * RECT_SIZE
        self._rect.y = randint(0,19) * RECT_SIZE

    def getRect(self):
        return self._rect
```

# Snake.py

```python

from ..consts import *
from pygame import Rect

class Snake:
    def __init__(self):
        self._rects = [Rect(4*RECT_SIZE, 8*RECT_SIZE, RECT_SIZE, RECT_SIZE)]
        self.color = BLACK
        self.direction = 'RIGHT'  # Начальное направление

    def setDirection(self, action):
        if action in ['LEFT', 'RIGHT', 'UP', 'DOWN']:
            self.direction = action

    def moving(self):
        head = self._rects[0].copy()  # Копируем голову змеи


        if self.direction == 'LEFT': head.x -= RECT_SIZE
        if self.direction == 'RIGHT': head.x += RECT_SIZE
        if self.direction == 'UP': head.y -= RECT_SIZE
        if self.direction == 'DOWN': head.y += RECT_SIZE

        self._rects.insert(0, head)  # Добавляем новую голову
        self._rects.pop(-1)  # Удаляем последний сегмент

    def grow(self):
        # Увеличиваем змейку, добавляя сегмент в конец
        rect = self._rects[-1].copy()
        self._rects.append(rect)

    def getListRects(self):
        return self._rects
```