МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И

МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное

бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

(МТУСИ)

Отчет по

Проектной Практике

Выполнили:

студенты гр. БВТ2201

Ибрагимов Т.Э.

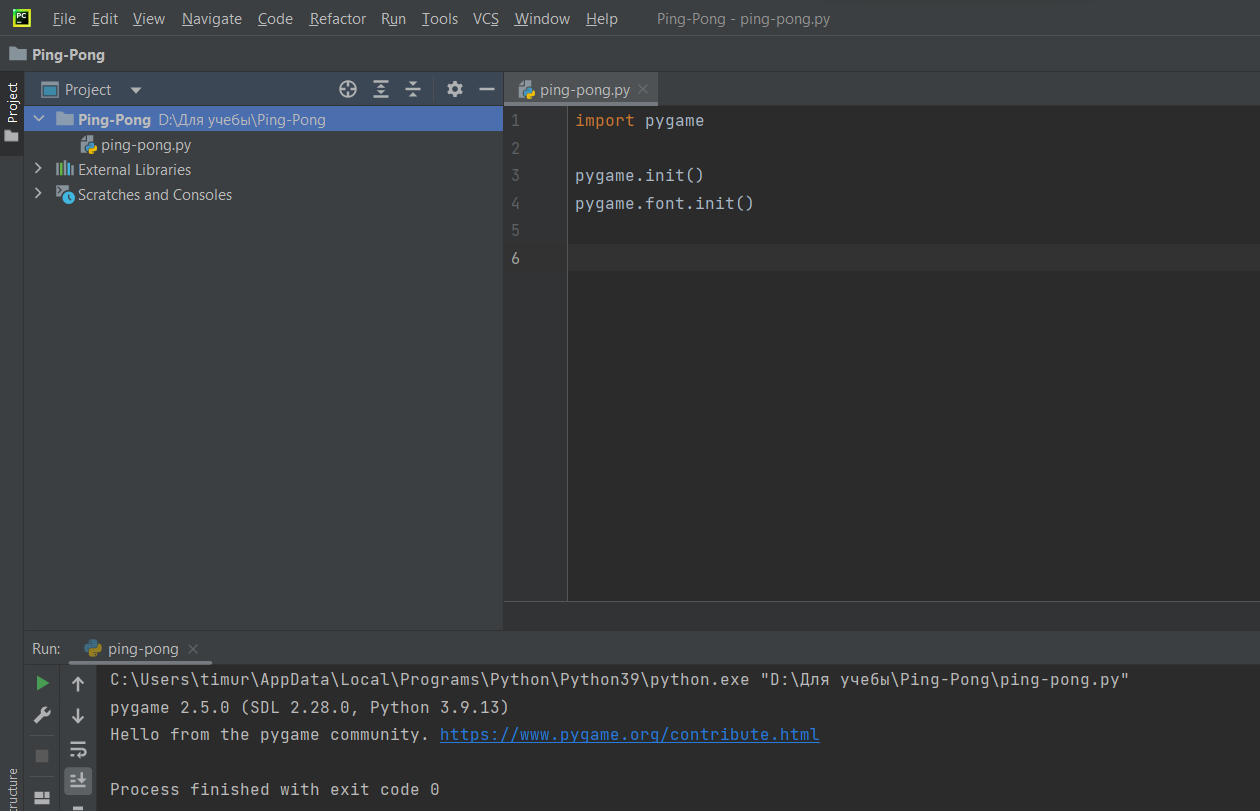
Салаватов Р.С.

12.07.2023 г.

Москва, 2023 г.

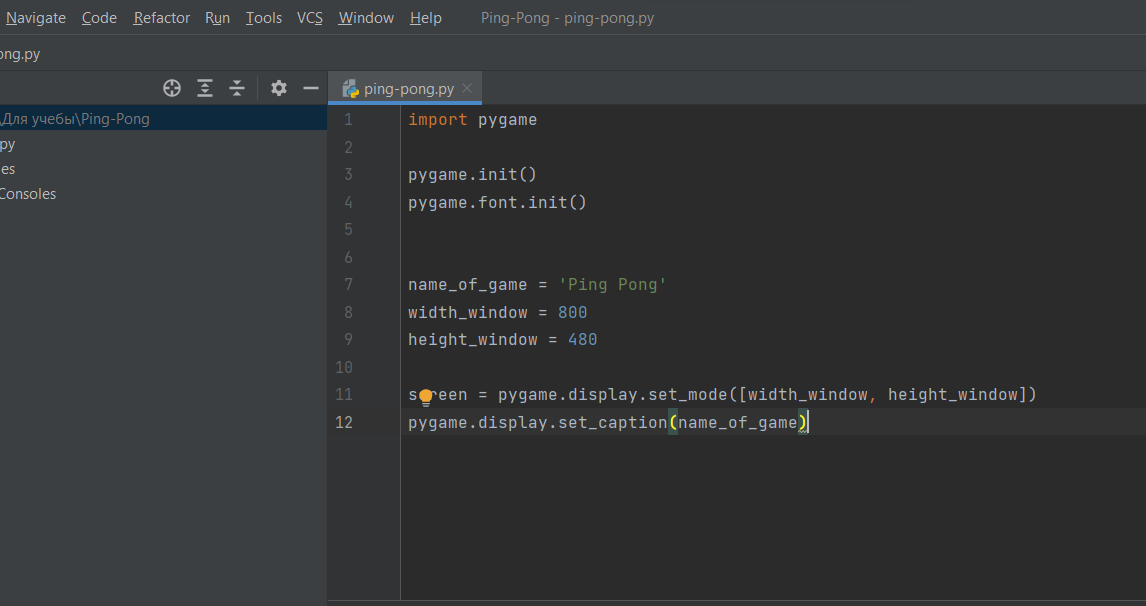
**Проект Ping-Pong на Python**

1. Для начала мы создаем наш py файл в Pycharm и называем его ping\_pong. Так как в игре используется всего два подвижных объекта, то есть платформа и мячик, смысла создавать класса под них особо не имеет смысла.
2. Импортируем библиотеку pygame, с которой мы будем работать, чтобы написать нашу игру. По сути это набор модулей языка программирования Python, предназначенного для написания компьютерных игр и мультимедиа-приложений. \ы

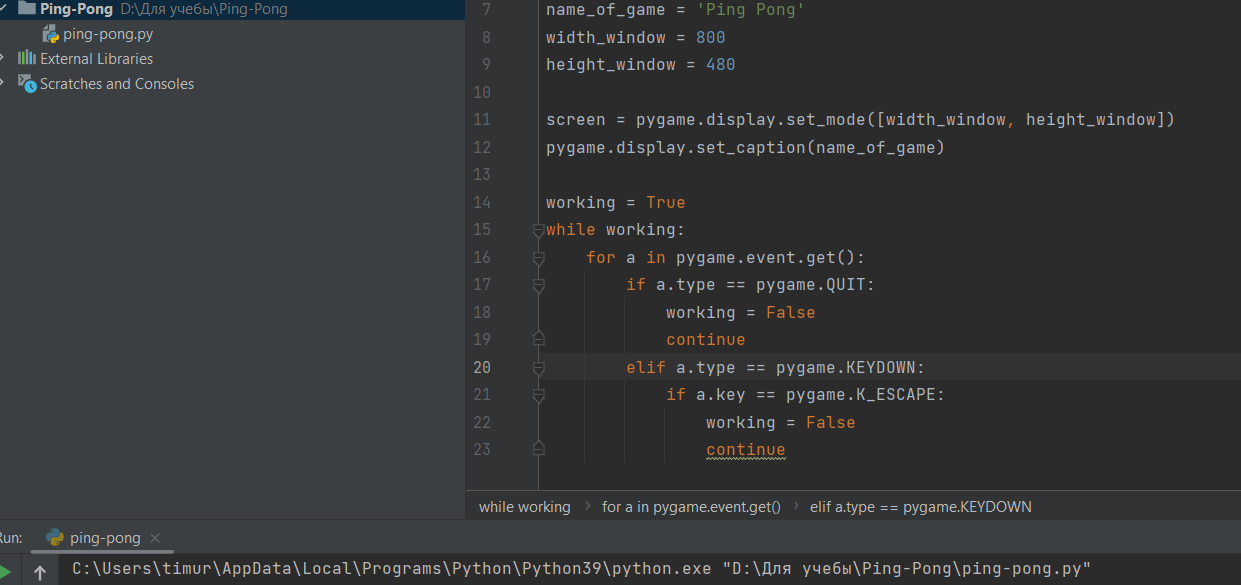


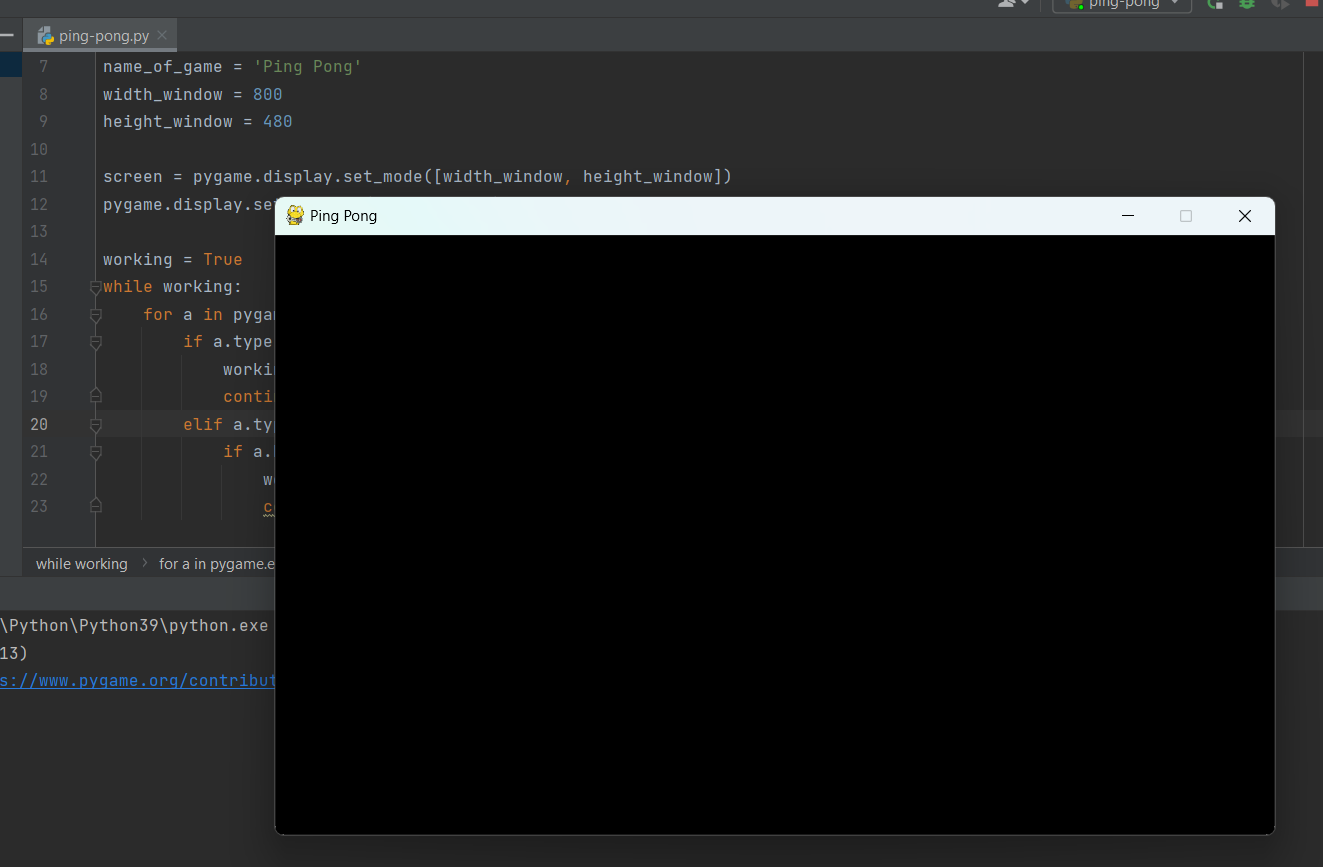
Далее мы инициализируем данную библиотеку и инициализируем шрифт, чтобы отображать счет(очки).

1. Следующим шагом следует создать окно, в котором будет происходить отрисовка объектов. Для этого нам понадобятся 2 переменные: ширина(width), высота(height). Далее эти переменные нужно куда-нибудь поставить, чтобы наш экран выводился. Для этого в pygame есть специальная функция – display. Она отвечает за обновление экрана, за отрисовку изображения и т.д. И вот как раз таки в display есть функция set\_mode(). Она отвечает размер наших заданных размеров окна. Дополнительное можем установить название нашей игры с помощью метода set\_caption. В name\_of\_game записываем название игры.



1. После этого нужно сделать так, чтобы экран не закрывался сам по себе. Создаем переменную working и присваиваем ей значение True. Ставим ее в цикл while. Он будет работать бесконечно, пока переменная working не станет False. В цикле while создаем еще один цикл for, который будет фиксировать какие-либо события. И если событием окажется выход при нажатии на крестик либо при нажатии кнопки Escape, то игра будет закрываться.

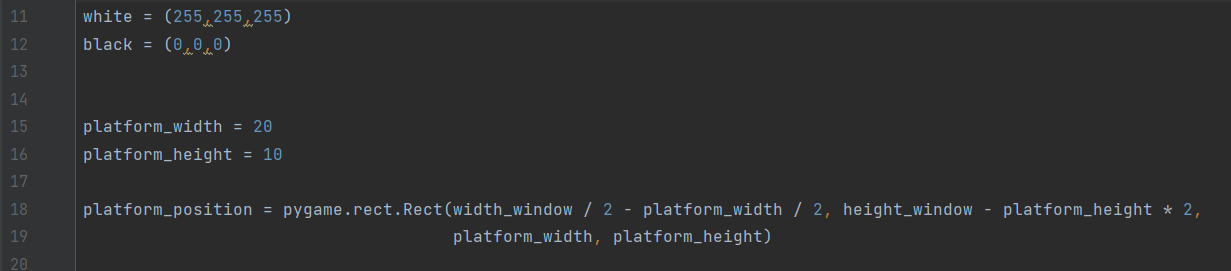




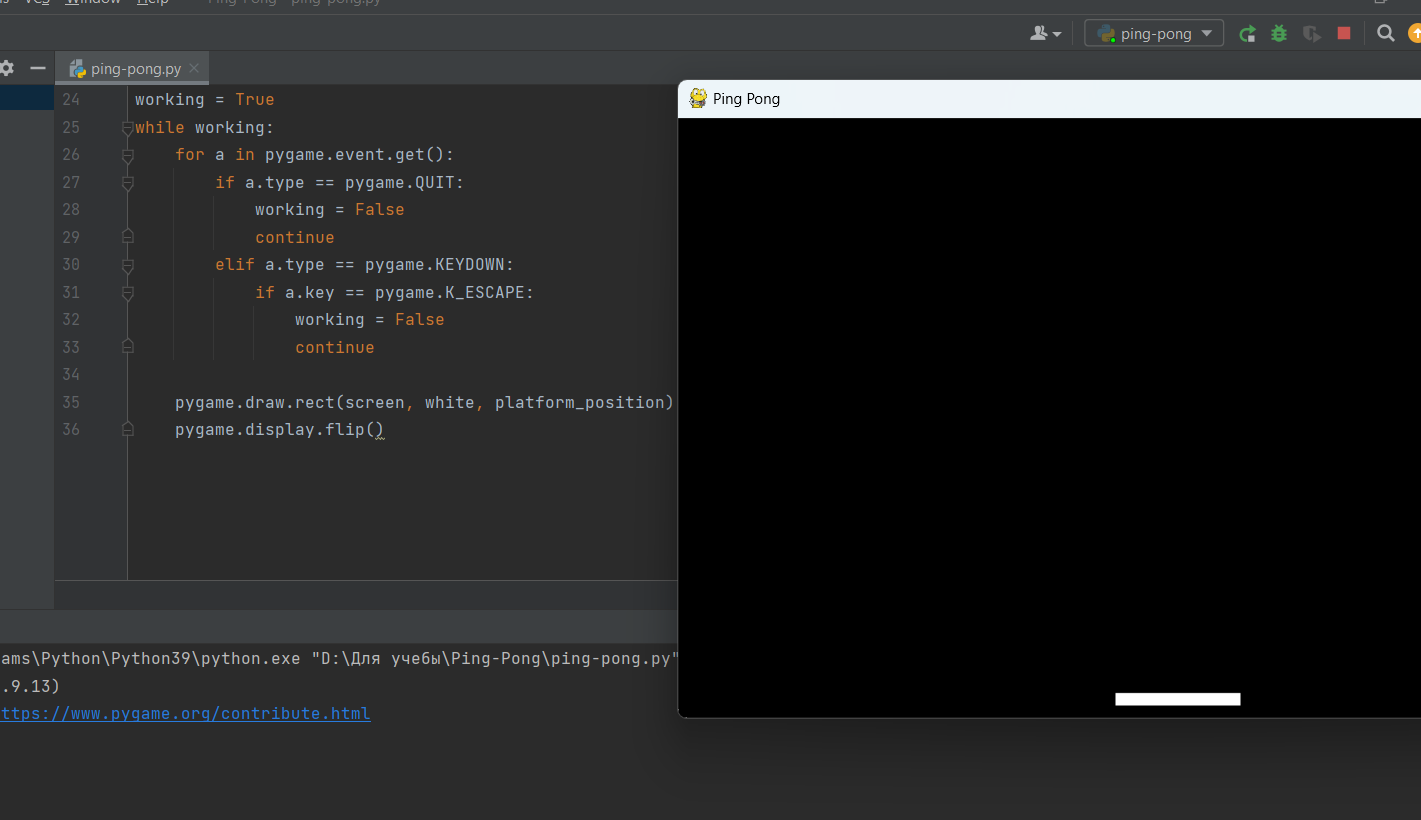
Так же мы можем наблюдать за названием игры

1. Теперь будем создавать сами платформы. Создаем переменные platform\_width и platform\_height, то есть ширина и высота платформ. После этого создаем контейнер прямоугольной области – rect. Он будет проверять наши платформы на столкновениями, в нашем случае для мячика. Так же он содержит 4 величины: позиции и размеры. Позиции нужны для того, чтобы понять где находится платформа.

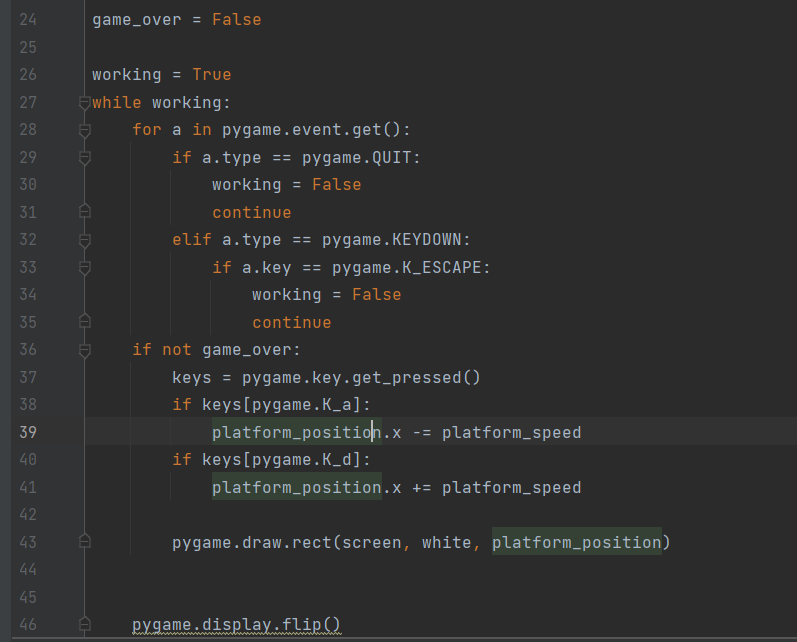
Далее можно создать цвета. В основном будет белый и черный.



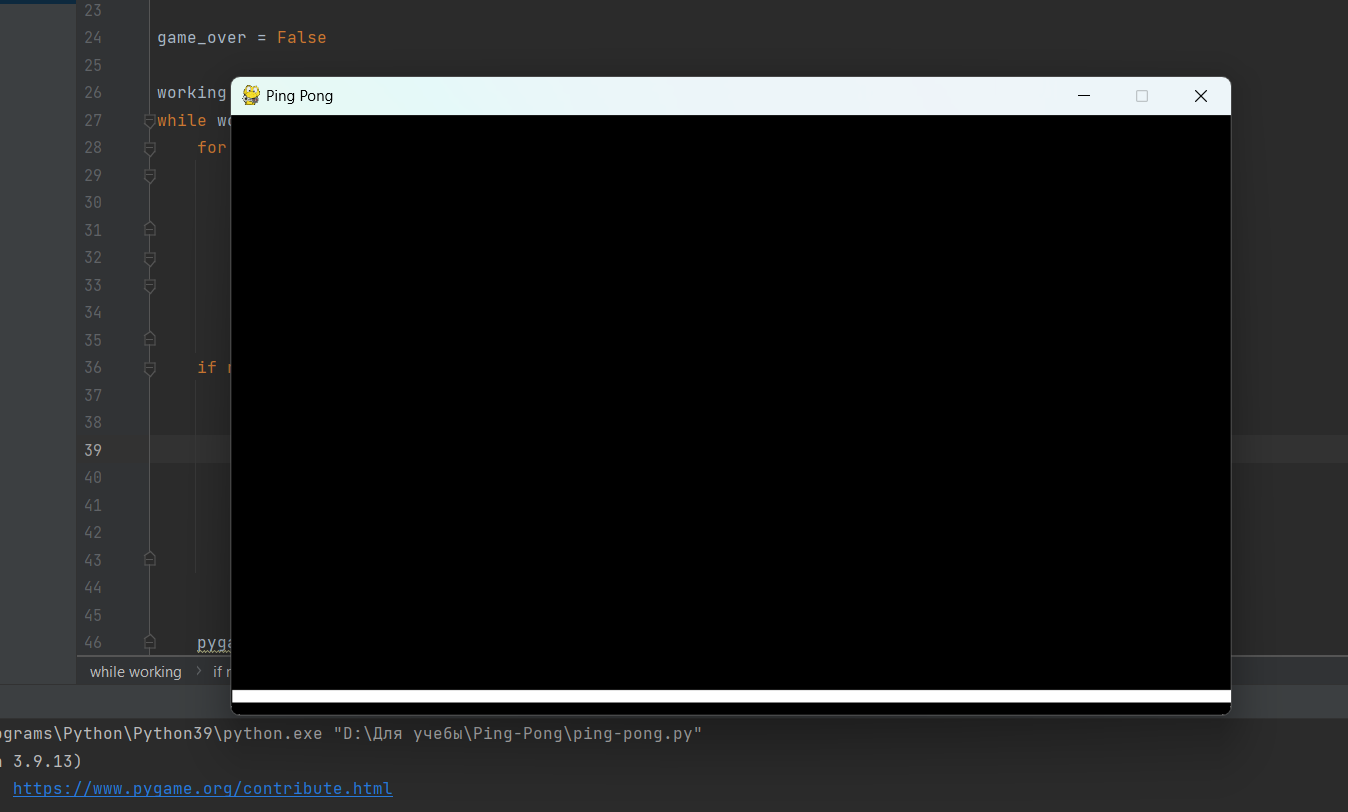
После того как мы сделали инициализировали все необходимые переменные для платформы, ее нужно отрисовать. Для этого будем использовать метод, который находится в pygame – draw. Туда передаем три параметра: экран, на котором будет происходить отрисовка, цвет и позиция платформы. Если запустим, то у нас ничего не изменится, наш экран будет черным. Чтобы это исправить нам понадобится метод flip, который будет обновлять наш созданный экран



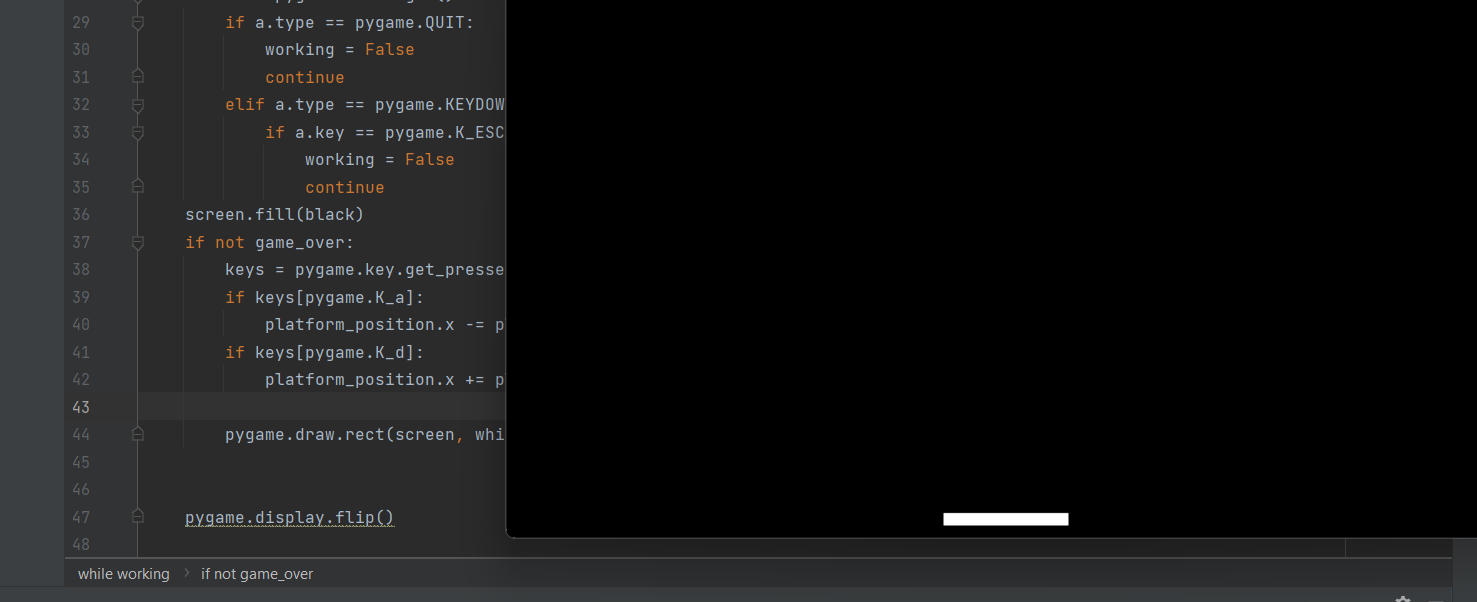
1. Теперь следует прописать логику движения платформы. Что нам для этого нужно. Во-первых клавиши, на которых мы будем нажимать, чтобы двигать платформу. Путь это будут A и D. Так же понадобится скорость платформы. Инициализируем переменную platform\_speed. Далее создаем булевую переменную game\_over, чтобы понимать когда игра будет закончена.



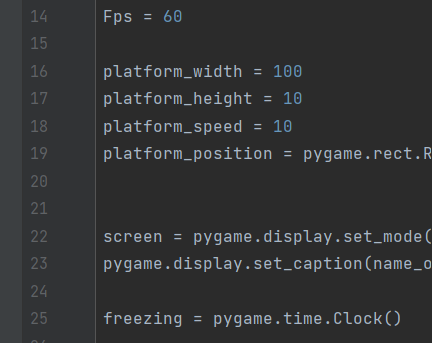
Если игра не закончена, то мы будет отслеживать нажатие каждой клавиши на клавиатуре. И если такими клавишами окажется A и D, то платформа будет двигаться влево или вправо от координаты x.



После нажатия клавиш происходит “расстягивание” платформы. Это происходит потому, что мы отрисовываем только платформу и ничего больше. Чтобы это исправить, можно заполнить наш экран черным цвет с помощью метода fill().

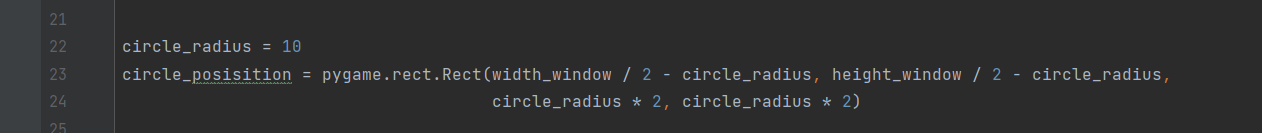


Все работает хорошо, но есть небольшая проблема, наша платформа очень быстро предвигается, даже если скорость достаточно маленькая. Чтобы это исправить, есть специальный метол clock(), который будет искусственно замедлять время работы цикла

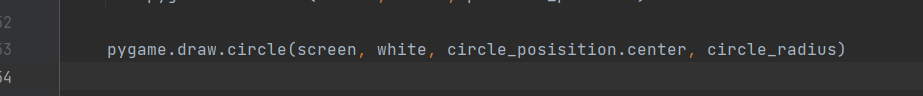


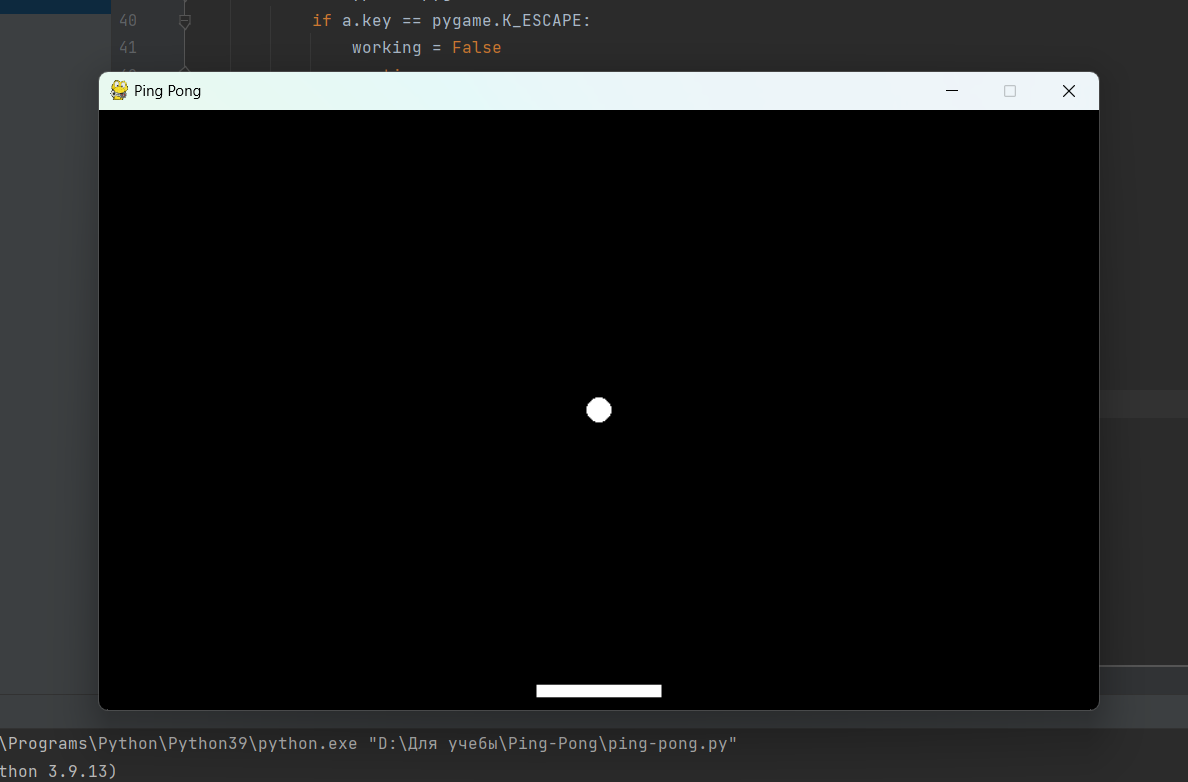
Создаем переменную Fps и freezing, которая будет замедлять наш цикл while. В цикле while прописываем freezing.tick(Fps). Теперь все будет работать корректно на любых компьютерах с разной вычислительной мощностью

1. Теперь переходим к мячику. Опять задаем параметры. Для этого понадобится радиус мячика, его позиция. Изначально его позицию зададим в центре



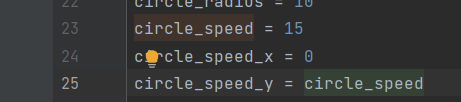
И теперь отрисуем его на экране:



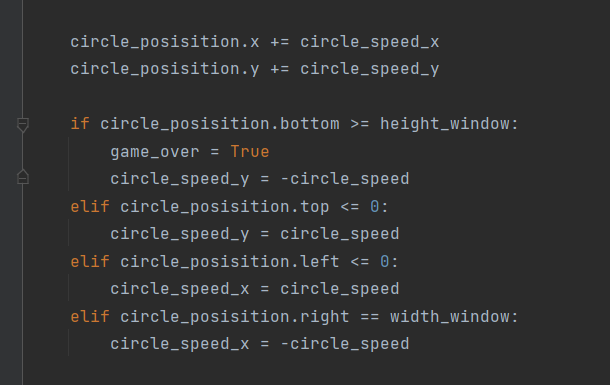


1. После отрисовки нужно задать скорость мячика и передвижение.

Создаем три переменные: circle\_speed, circle\_speed\_x, circle\_speed\_y.

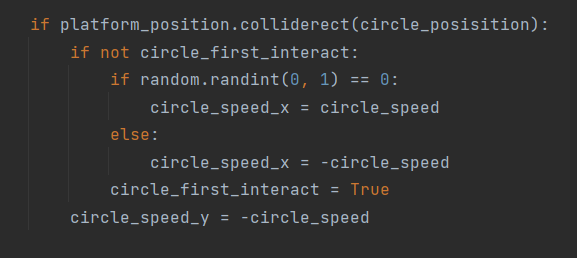


Следующим этапом будет проверка мячика на столкновение с границами экрана и инициализация его скорости



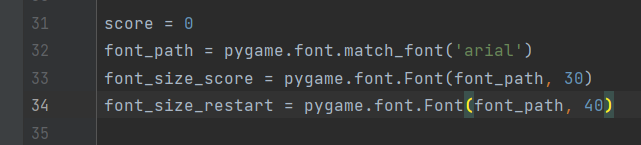
Первые две строчки инициализируют скорость передвижения мячика по оси Х и Y. Дальше проходит проверка. То есть если мячик столкнулся с нижней границей экрана, то игра закончена, но при этом он будет двигаться дальше и отражаться от границ экрана.

1. Теперь будем прописывать логика столкновения мячика с платформой. Сначала импортируем библиотеку random. И создаем переменную circle\_first\_interact. Инициализируем ее как False.



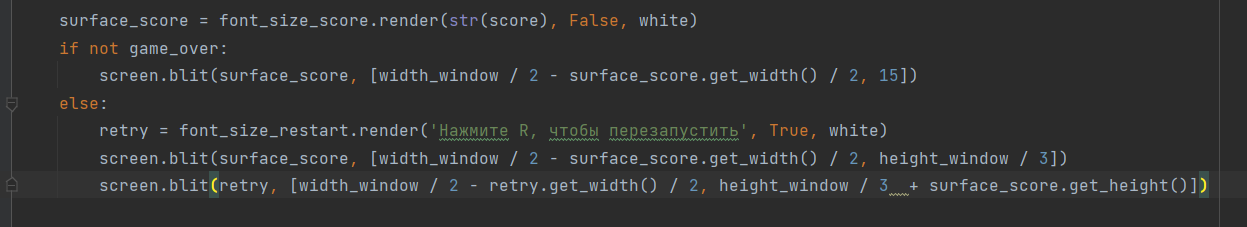
В первой строчке кода проверяем, если колайдер платформы столкнулся с колайдером мячика при этом он сделал это впервые, то выбирается рандомное число от 0 до 1. И если число 0, то мяч полетит вправо, иначе полетит влево.

1. Основную задачу выполнили. Теперь нам осталось только создать очки, то есть счет. Для начала создадим переменную score, в которой будет хранится количество набранных очков. Затем создадим шрифт для счета. Для этого мы сначала получает путь к шрифту, а потом просто этот шрифт вставляем в переменную и задаем размер. Также можно создать еще одну переменную, которая будет хранить надпись “Перезапустить игру”



Следующим шагом мы должны отрисовать интерфейс уже на экране.

Сначала мы инициализируем переменную surface\_score. В ней будет хранится отрисовка очков. Метод render принимает в данном случае 3 значения: строковую переменную либо мы сами можем что-либо написать с помощью кавычек, далее включать сглаживание или нет и цвет. Если игра не проиграна, то с помощью метода blit(), который может отрисовать поверхости, будем выводить очки по центру выше середины. Иначе мы создаем новую перемнную retry, в которой хранится то же самое, что и в переменной surface\_score. Только вместо переменной мы написали текст сами. Также очки никуда не денутся, а просто перейдут ближе к середине и после этого отрисуется надпись проигрыша.



1. В заключение сделаем так, чтобы на рестарте игры значения сбрасывались для платформы, для мячика, для очков.

