PYGAME.

ЦЕЛЬ: **изучить принцип наследования классов**

ПРИМЕЧАНИЕ:

ПЛАНИРОВАНИЕ

1. Наследование.

За прошлые занятия мы создали спрайта игрока и спрайты платформ. Можно подметить, что у игрока есть некоторые свойства, которых нет у платформы, а с дальнейшей разработкой игры, у спрайта игрока будет появляться все больше свойств, которых нет у платформы. А для объектов платформ и игрока выделяется одинаковое количество памяти для хранения каждого свойства. Это не оптимально. Поэтому лучше будет создать для спрайтов игрока и платформы отдельные классы.

Представим, что мы хотим создать отдельный класс для платформ с конструктором и различными свойствами и добавим новое свойство – **time**. Пускай спустя некоторое время, после того как спрайт игрок запрыгнул на платформу, платформа будет начинать падать. Свойство t**ime** будет как раз-таки равно количеству секунд, через которое платформа начнет падать под весом игрока.

```
class Platform():
        self.image = pygame.image.load(img)
        self.speed = speed
        self.rect =  self.image.get_rect().height
        time = 5
```

Можно заметить, что почти все свойства повторяются как у класса Sprite. Чтобы не повторять свойства и методы того класса, по подобию которого вы создаете новый класс в ООП реализован механизм наследования. Вновь создаваемый класс можно сделать наследником уже существующего класса.

Записывается это так:

```
class Platform(Sprite):
        time = 5
```

Теперь класс Platform унаследует все из класса Sprite, все поля, конструктор, методы.

Но если мы хотим собственные свойства спрайта тоже использовать в конструкторе. То есть свойство **time** задавать при создании объекта класса Platform.

Делается это довольно просто:

```
class Platform(Sprite):
    def __init__(self, x, y, img, time):
        Sprite.__init__(self, x, y, 0, img)
        self.time = time
```

Вызываем конструктор родительского класса внутри конструктора нашего класса, чтобы задать необходимые свойства.

! Проблемная задача. Внести изменения в программу, чтобы был родительский класс Sprite и остальные классы – наследники.

2. Рефлексия

- Сегодня познакомились с принципом наследования и модифицировали нашу программу с учетом этого принципа.