PYGAME.

ЦЕЛЬ: **Вспомнить или узнать, что такое конструктор класса**

ПРИМЕЧАНИЕ:

ПЛАНИРОВАНИЕ

1. Конструктор

Давайте возьмем снова наш пример с автомобилем. Мы создали класс – автомобиль:

Класс Автомобиль()

И потом объект (экземпляр) этого класса:

МойАвтомобиль = Автомобиль()

А далее, если нам понадобилось изменить свойства нашего объекта и, чтобы это сделать мы писали команды:

МойАвтомобиль.скорость = 100

МойАвтомобиль.цвет = белый

И т.д.

Если свойств много, то довольно неудобно все их задавать отдельными командами для каждого нового объекта.

Было бы здорово сразу при создании объекта в скобках указывать значения свойств. Например:

МойАвтомобиль = Автомобиль(белый, 100, ручная)

И после этой команды создается объект МойАвтомобиль, со свойствами:

Цвет – белый

Скорость – 100 км/час

Коробка переключения скоростей – ручная

И в Python реализован этот принцип. Но для того, чтобы так можно было создавать экземпляры класса, необходимо при описании класса использовать конструктор. Для этого используется специальная функция или другими словами – метод, которая сама по себе и есть «конструктор».

Ее имя: __init__()

Именно с двойными подчеркиваниями. **Вот пример:**

```
Класс Автомобиль():
        ФУНКЦИЯ __init__(Я, ЦВЕТ, СКОРОСТЬ, ДВИГАТЕЛЬ):
                Автомобиль.цвет = ЦВЕТ
                Автомобиль.скорость = СКОРОСТЬ
```

Автомобиль.двигатель = ДВИГАТЕЛЬ

В таком виде, мы, как бы записали конструктор, который будет создавать объект со свойствами, которые мы можем записать в скобках. Теперь мы можем создавать объекты одной строкой:

МойАвтомобиль = Автомобиль(белый, 100, ручная)

! Проблемная задача: В Python изменить программу, так, чтобы в ней использовался конструктор класса Sprite.

Результат, который должен получиться:

```
import pygame

class Sprite():

    def __init__(self, x, y, speed, img):

        self.image = pygame.image.load(img)

        self.speed = speed

        self.x = x

        self.y = y


w = pygame.display.set_mode ((1279, 700))

Player = Sprite(100, 100, 1, 'Images/Player.png')

game = True

while game:

    for ev in pygame.event.get ():

        if ev.type == pygame.QUIT:

            game = False

    keys = pygame.key.get_pressed()

    if keys[pygame.K_RIGHT]:

        Player.x += Player.speed

    elif keys[pygame.K_LEFT]:

        Player.x -= Player.speed

    w.fill((0, 0, 0))

    w.blit(Player.image, (Player.x, Player.y))

    pygame.display.update()

pygame.quit ()
```

!

2. pygame.transform.flip()

В pygame есть функция, которая отражает изображение либо по вертикали, либо по горизонтали или сразу по двум осям:

pygame.transform.flip(Surface, BoolX, BoolY)

Surface – поверхность с изображением

BoolX, BoolY – булевые переменные, определяющие отражать или нет по соответствующей оси.

! Проблемная задача: сделать так, чтобы персонаж при движении смотрел в ту сторону, в которую двигается.

Решение:

В конструктор спрайта добавляем:

```
self.imgR  = pygame.image.load(img)
self.imgL = pygame.transform.flip(self.image, True, False)
```

В игровом цикле:

```
if keys[pygame.K_RIGHT]:
        self.rect.x += self.speedx
        self.image = self.animGoR[self.cadr // 2 % 5]
        self.image = self.imgR
    elif keys[pygame.K_LEFT]:
        self.rect.x -= self.speedx
        self.image = self.animGoL[self.cadr // 2 % 5]
        self.image = self.imgL
```

3. Рефлексия

- Узнали, что такое конструктор класса

- Научились применять на практике этот принцип

- Познакомились с функцией: **pygame.transform.flip()**