|  |
| --- |
| Домашнее задание |

**ЗАДАНИЕ**

**Результат**

файл с расширением .py .docx .pdf .doc или ссылка на google-документы, google-colab, github

**ЗАДАНИЕ 1**

Выбрать класс(Любой из списка или предложить свой, можно без согласования, но по объему работ такой же как и в списке) и описать его на Python

1. Описать класс «домашняя библиотека». Предусмотреть возможность работы с произвольным числом книг, поиска книги по какому-либо признаку (например, по автору или по году издания), добавления книг в библиотеку.
2. Описать класс, реализующий десятичный счетчик, который может увеличивать или уменьшать свое значение на единицу в заданном диапазоне. Предусмотреть инициализацию счетчика значениями по умолчанию и произвольными значениями. Счетчик имеет два метода: увеличения и уменьшения, — и свойство, позволяющее получить его текущее состояние. Написать программу, демонстрирующую все возможности класса.
3. в pygame реализовать класс стена, добавить стены в игру, ограничить передвижение игрока по стенам.

**Дополнительные задачи**

1. Задача на взаимодействие между классами. Разработать систему «Автобаза». **Диспетчер** распределяет заявки на **Рейсы** между **Водителями** и назначает для этого **Автомобиль**. **Водитель** может сделать заявку на ремонт. **Диспетчер** может отстранить **Водителя** от работы. **Водитель** делает отметку о выполнении **Рейса** и состоянии **Автомобиля.**
2. Класс Абонент: Идентификационный номер, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет, Кредит, Время междугородных и городских переговоров; Конструктор; Методы: установка значений атрибутов, получение значений атрибутов, вывод информации. Создать массив объектов данного класса. Вывести сведения относительно абонентов, у которых время городских переговоров превышает заданное. Сведения относительно абонентов, которые пользовались междугородной связью. Список абонентов в алфавитном порядке.

**РЕШЕНИЕ**

**ЗАДАНИЕ 1**

Выбрать класс(Любой из списка или предложить свой, можно без согласования, но по объему работ такой же как и в списке) и описать его на Python

3. в pygame реализовать класс стена, добавить стены в игру, ограничить передвижение игрока по стенам.

main.py (\*полный код ниже):

*# упор в стену*if window.get\_rect().top == player.rect.top:  
 player.check\_walls(top=1, bottom=0, left=0, right=0)  
if window.get\_rect().left == player.rect.left:  
 player.check\_walls(top=0, bottom=0, left=1, right=0)  
if window.get\_rect().right == player.rect.right:  
 player.check\_walls(top=0, bottom=0, left=0, right=1)  
if window.get\_rect().bottom == player.rect.bottom:  
 player.check\_walls(top=0, bottom=1, left=0, right=0)

Player.py (\*полный код ниже):

*# граничные положения игрока*def check\_walls(self, top, bottom, left, right):  
 *# Y вниз* self.rect.y -= bottom  
 *# Y вверх* self.rect.y += top  
 *# X влево* self.rect.x += left  
 *# X вправо* self.rect.x -= right

\*main.py

import pygame  
from Player import Player *# наш созданный класс Player.py*from Enemy import \*  
pygame.init() *# точка входа и настройка игры  
  
# просто переменные*FPS = 60  
WIDTH = 800  
HEIGHT = 600  
COLOR = (255, 0, 0) *# цвет в формате rgb этот красный  
  
# создание игрока через класс Player*player = Player(100, 100) *# (100, 100) - размер  
  
#*enemy = Enemy(100, 100)  
  
*# создаём группу спрайтов*all\_sprites = pygame.sprite.Group()  
sprites\_enemy = pygame.sprite.Group()  
  
*#*window = pygame.display.set\_mode((WIDTH, HEIGHT)) *# задаем размеры окну*clock = pygame.time.Clock() *# задаем время игры сейчас оно нужно для фпс*run = True *# управление игрой  
  
#*all\_sprites.add(player, enemy)  
sprites\_enemy.add(enemy)  
  
*# сама игра*while run:  
 for i in pygame.event.get(): *# отслеживаем события, которые просходят в окне нажатие, закрытие и т.дю* if i.type == pygame.QUIT: *# если окно закрыли* run = False *# выходим из цикла  
  
 # kill при соприкосновении  
 # if enemy.rect.left == player.rect.right \  
 # and enemy.rect.top == player.rect.top:  
 # player.kill()* hit = pygame.sprite.spritecollide(player, sprites\_enemy, False)  
 if hit:  
 player.kill()  
  
 *# упор в стену* if window.get\_rect().top == player.rect.top:  
 player.check\_walls(top=1, bottom=0, left=0, right=0)  
 if window.get\_rect().left == player.rect.left:  
 player.check\_walls(top=0, bottom=0, left=1, right=0)  
 if window.get\_rect().right == player.rect.right:  
 player.check\_walls(top=0, bottom=0, left=0, right=1)  
 if window.get\_rect().bottom == player.rect.bottom:  
 player.check\_walls(top=0, bottom=1, left=0, right=0)  
  
 window.fill(COLOR) *# красим окно здесь можно потом картинки загружать* all\_sprites.draw(window) *#* player.update() *#* enemy.update() *#* pygame.display.update() *# обновляем экран каждую итерацию* clock.tick(FPS) *# плавно обновляем*pygame.quit() *# корректное завершение работы*

\*Player.py

import pygame  
BLACK = (0, 0, 0) *# цвет в формате rgb этот красный*class Player(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, width=50, height=50):  
 pygame.sprite.Sprite.\_\_init\_\_(self) *# создание спрайта - игровой объект* self.image = pygame.Surface((width, height)) *# размеры игрока* self.image.fill(BLACK) *# цвет игрока* self.rect = self.image.get\_rect() *# получаем положение игрока  
 # self.image.get\_width = 50 # ширина игрока* self.rect.center = (100, 100) *# передаем координаты расположения центра игрока  
  
 # отрисовка игрока* def update(self):  
 self.run() *#  
  
 # передвижение игрока* def run(self):  
 keystate = pygame.key.get\_pressed()  
 *# узнаём текущее положение и добовляем перемещение по Y вниз* if keystate[pygame.K\_DOWN]:  
 self.rect.y += 1  
 *# Y вверх* if keystate[pygame.K\_UP]:  
 self.rect.y -= 1  
 *# X влево* if keystate[pygame.K\_LEFT]:  
 self.rect.x -= 1  
 *# X вправо* if keystate[pygame.K\_RIGHT]:  
 self.rect.x += 1  
  
 *# граничные положения игрока* def check\_walls(self, top, bottom, left, right):  
 *# Y вниз* self.rect.y -= bottom  
 *# Y вверх* self.rect.y += top  
 *# X влево* self.rect.x += left  
 *# X вправо* self.rect.x -= right