LE GOUVERNEMENT
DU GRAND-DUCHÉ DE LUXEMBOURG
Ministère de l'Éducation nationale,
de l'Enfance et de la Jeunesse

EXAMEN DE FIN D'ÉTUDES SECONDAIRES CLASSIQUES

2019

CORRIGÉ – BARÈME

BRANCHE	SECTION(S)	ÉPREUVE ÉCRITE
		Durée de l'épreuve : *Date de l'épreuve :*

```
import pygame, sys
from pygame.locals import *
from random import randint, randrange

# I) 2 p.
pygame.init()

WIDTH, HEIGHT = 600, 200
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption('Grenouille - LYCEE - NUMERO')
screen.fill(Color('black'))

FPS = 25
clock = pygame.time.Clock()


# II) 3 p. (1 p. constructeur, 1 p. pour chaque méthode
class Frog:
    def __init__(self, x=300, y=180, color=Color('white')):
        self.x, self.y = x, y
        self.color = color

    def reset_frog(self):
        self.x, self.y = 300, 180
        self.color = Color('White')
        print('frog reset done')

    def draw_frog(self):
        pygame.draw.circle(screen, self.color, (self.x, self.y), 3, 1)

# III) 10 p.
class Car:
    def __init__(self, x=0, y=200, width=20, height=10):    # 2 p.
        self.x, self.y = x, y
        self.width, self.height = width, height
        self.speed = randint(-20, 20)

    def reset_car(self):                                    # 1 p.
        self.x = 0
        self.speed = randint(-20, 20)
        print('car reset done')

    def drive_car(self):                                    # 4 p.
        new_x = self.x + self.speed
        if new_x + self.width < 0:
            self.x = WIDTH - 1
        elif new_x >= WIDTH:
            self.x = -self.width
        else:
            self.x = new_x

    def contains_pixel(self, x, y):                         # 2 p.
        return self.x <= x < self.x + self.width and self.y <= y < self.y + self.height

    def draw_car(self):                                     # 1 p.
        pygame.draw.rect(screen, Color('yellow'), (self.x, self.y, self.width, self.height), 1)

# IV) 20 p.
class Street:
    def __init__(self):                                     #1 p.
        self.list_of_cars = []

    def append_car(self, car):                              # 1 p.
        self.list_of_cars.append(car)

    def speed_up_cars(self, amount):                        # 4 p.
        for car in self.list_of_cars:
```

```
            if car.speed > 0:
                car.speed += amount
            elif car.speed == 0:
                r = 2*randint(0, 1)-1
                car.speed = r * amount
            else:
                car.speed -= amount

    def slow_down_cars(self, amount):                       # 4 p.
        for car in self.list_of_cars:
            if car.speed < -amount:
                car.speed += amount
            elif car.speed <= amount:
                car.speed = 0
            else:
                car.speed -= amount

    def stop_cars(self):                                    # 1 p.
        for car in self.list_of_cars:
            car.speed = 0

    def drive_cars(self):                                   # 1 p.
        for car in self.list_of_cars:
            car.drive_car()

    def draw_cars(self):                                    # 1 p.
        for car in self.list_of_cars:
            car.draw_car()

    def reset_cars(self):                                   # 1 p.
        for car in self.list_of_cars:
            car.reset_car()

    def check_collision(self, frog):                        # 6 p.
        for car in self.list_of_cars:
            test1 = car.contains_pixel(frog.x - 3, frog.y)
            test2 = car.contains_pixel(frog.x + 3, frog.y)
            test3 = car.contains_pixel(frog.x, frog.y - 3)
            test4 = car.contains_pixel(frog.x, frog.y + 3)
            if test1 or test2 or test3 or test4:
                return True
        return False


# V) Programme principal    25 p.
# V) - 1) Préparation
frog = Frog()
street = Street()                                           # 1 p.
for i in range(4):
    street.append_car(Car(0, 40 * (i + 1)))                 # 2 p.

done = False
game_stopped = False
key_pressed = KMOD_NONE
while not done:
    # V) - 2) Réaction aux évènements
    for event in pygame.event.get():
        if event.type == pygame.QUIT:                       # 1 p.
            done = True
        elif event.type == pygame.KEYDOWN:                  # 3 p.
            key_pressed = event.key
            if key_pressed == K_SPACE:
                game_stopped = False
                screen.fill(Color('black'))
                frog.reset_frog()
                street.reset_cars()
        elif event.type == pygame.KEYUP:                    # 4 p.
            key_pressed = KMOD_NONE                                 # 2 p. max. si touche enfoncée ne
                                                                    # fonctionne pas

    if not game_stopped:                                    # usage de la variable game_stopped
        if key_pressed == K_LEFT:                           # (dernière question)
            frog.x -= 2
        elif key_pressed == K_RIGHT:
            frog.x += 2
        elif key_pressed == K_UP:
            frog.y -= 2
        elif key_pressed == K_DOWN:
```

```
            frog.y += 2
        elif key_pressed == K_a:                          # 4 p.
            street.speed_up_cars(1)                         # 2 p. max. si touche enfoncée ne
        elif key_pressed == K_s:                          # fonctionne pas
            street.slow_down_cars(1)

    # V) - 3) Actions répétées à chaque itération
    screen.fill(Color('black'))                           # 1 p.
    street.drive_cars()
    street.draw_cars()                                    # 1 p.
    if street.check_collision(frog):
        street.stop_cars()
        frog.color = Color('red')                         # 2 p.
        game_stopped = True                                # 5 p. (variable game_stopped +  usage)
    if frog.y - 3 < 0:
        street.stop_cars()

        frog.color = Color('green')
        game_stopped = True                               # 1 p.
    frog.draw_frog()
    pygame.display.update()
    clock.tick(FPS)

pygame.quit()
sys.exit()
```