```
'''
When this program starts, a new window opens.
The window is 500 x 400 pixels, has a black background
and a title "Colourful Dots".
The game features two dots which bounce around the screen
and a thin vertical white line which divides the screen in half.

There is a score in the top left corner of the window. It says "Score :"
in size 64 white text, followed by the number of seconds which
have passed since the start of the program.

Both dots start at a fixed location, move with constant speed in a straight
line (speed described below for each ball), and bounce off the window edges.

One dot is green when it is completely on the
left half of the screen and blue when it is completely
on the right hand side of the screen.

The other dot is always white when completely on
the left half of the screen and red when completely
on the right half of the screen.

If either dot is in the center of the zone --- i.e. any part of the dot is
touching the dividing line--- the dot turns yellow.

The green/blue dot has a radius of 30 and starts at position [50,75].
It moves twice as fast vertically as it does horizontally.

The white/red dot has a radius of 40 and starts at position [200,100].
It moves twice as fast horizontally as it does vertically.

If both dots are touching the center line at the same time
(i.e. if both dots are yellow), the game ends.
Both dots no longer move.

If the player clicks the close window button, the window closes.
'''
```

```
import pygame

# User-defined functions

def main():
   pygame.init()
   pygame.display.set_mode(|_____???_____|)
   pygame.display.set_caption('Colorful Dots')
   w_surface = pygame.display.get_surface()
   game = Game(w_surface)
   game.play()
   pygame.quit()

# User-defined classes
class Game:
   # An object in this class represents a complete game.

   def __init__(self, surface):
      # Initialize a Game.
      # - self is the Game to initialize
      # - surface is the display window surface object

      # === objects that are part of every game that we will discuss
      self.surface = surface
      self.bg_color = pygame.Color('black')

      self.FPS = 60
      self.game_Clock = pygame.time.Clock()
      self.close_clicked = False
      self.continue_game = True

      # === game specific objects
      # Attributes needed to manage the dot objects
      gyb_dot = Dot(|_____???_____|)
      wyr_dot = Dot(|_____???_____|)
      self.dots = |_____???_____|
      # Attributes needed to draw the line
      self.line_color = pygame.Color("white")
      middle = self.surface.get_width()//2
      self.line_top = [middle, 0 ]
      self.line_bottom = |_____???_____|
      # Attributes needed for score:
      self.score = 0
```

```
    def play(self):
        # Play the game until the player presses the close box.
        # - self is the Game that should be continued or not.

        while not self.close_clicked:  # until player clicks close box
            # play frame
            self.handle_events()
            self.draw()
            if self.continue_game:
                self.update()
                self.decide_continue()
            self.game_Clock.tick(self.FPS) # run at most with FPS Frames Per
Second

    def handle_events(self):
        # Handle each user event by changing the game state appropriately.
        # - self is the Game whose events will be handled

        events = pygame.event.get()
        for event in events:
            if event.type == pygame.QUIT:
                self.close_clicked = |_____???_____|

    def draw(self):
        # Draw all game objects.
        # - self is the Game to draw

        self.surface.fill(self.bg_color) # clear the display surface first
        for |_____???_____|:
            dot.draw()
        pygame.draw.line(self.surface, self.line_color,
                         self.line_top, self.line_bottom)
        |_____???_____|
        pygame.display.update() # make the updated surface appear on the display

    def draw_score(self):
        # Draw the time since the game began as a score
        # in white on the window's background.
        # - self is the Game to draw for.
        score_string = 'Score:' + str(self.score)
        score_font = pygame.font.SysFont('', 64)
        score_image = score_font.render(score_string, True,
                                        pygame.Color('white'), self.bg_color)
        score_top_left_corner = (0, 0)
        self.surface.blit(|_____???_____|)
```

```
    def update(self):
        # Update the game objects.
        # - self is the Game to update

        |_____???_____|
            |_____???_____|
        self.score = pygame.time.get_ticks()//1000

    def decide_continue(self):
        # Check and remember if the game should continue
        # - self is the Game to check

        all_touching = True
        for dot in self.dots:
            if |_____???_____|:
                all_touching = False
        if all_touching:
            self.continue_game = |_____???_____|


class Dot:
    # An object in this class represents a colored circle.

    def __init__(|_____???_____|, colors, radius, center, velocity, surface):
        # Initialize a Dot.
        # - self: the dot object we're creating
        # - colors is a list of string names for the dot colors in the following
        #    order: [left color, center color, right color].
        # - radius is an int indicating the radius of our dot
        # - center is a list containing the x and y int coordinates of the
        #    center of the dot
        # - velocity is a list with the x velocity and the y velocity for our dot
        # - surface is the window's pygame.Surface object

        self.left_color = pygame.Color(colors[0])
        self.center_color = pygame.Color(colors[1])
        self.right_color = pygame.Color(colors[2])
        self.radius = radius
        self.center = center
        self.velocity = velocity
        self.surface = surface

    def draw(self):
        # Draw the Dot.
```

```
        # - self is the Dot to draw

        color = self.pick_color()
        pygame.draw.circle(self.surface, color, self.center, self.radius)

    def pick_color(self):
        # decides which color the dot should have depending on its position.
        # - self is the dot object whose color we need to determine
        # returns a pygame.Color object representing the current color of the dot
        middle = self.surface.get_width()//2
        if self.center[0] + self.radius  < middle:
            |_____???_____|
        elif self.center[0] - self.radius > middle:
            |_____???_____|
        else:
            |_____???_____|

    def move(self):
        # Change the location of the Dot by adding the corresponding
        # speed values to the x and y coordinate of its center
        # - self is the Dot
        size = self.surface.get_size()

        for index in range(len(size)):
            self.center[index] = self.center[index] + self.velocity[index]
            #Check bounce
            left_top_bounce = |_____???_____|
            right_bottom_bounce = |_____???_____|
            if left_top_bounce |__???__| right_bottom_bounce:
                self.velocity[index] = -self.velocity[index]

    def touching_center(self):
        # Checks if the dot is touching the vertical center line of the screen
        # self - the dot object
        # return: True if the dot is touching the center, otherwise False
        middle = self.surface.get_width()//2
        left_limit = middle - self.radius
        right_limit = middle + self.radius
        return self.center[0] <= right_limit |__???__| self.center[0] >=
left_limit


main()
```