```
'''Alien Invasion
There are 30 alien ships arranged in a 5 x 6 uniform grid that spans across the
window width and covers one quarter of the height of the window.
This block of alien ships has some “enemy ships” that are red in color. The goal
of the game is to drain the battery of all the  “enemy ships”
from this block.

The alien ships are represented by a filled ellipse, the color of which is chosen
randomly from a list of colors. At the start of the game, the block of alien
ships moves vertically downwards starting from the top edge of the window. When
the bottom row of the block touches the bottom edge of the window, all alien
ships reverse direction and move upwards. When the top row of the block touches
the top edge of the window, all alien ships reverse direction again. The block of
alien ships continue to move downwards and upwards vertically until the game
ends.

If a player clicks inside a red alien ship, the color of the ship changes from
red to background color of the surface indicating that its battery has been
drained.
The game ends after the battery of all red ships in the block have been drained
in this manner.
'''

import pygame,random


# User-defined functions

def main():
   # initialize all pygame modules (some need initialization)
   pygame.init()
   # create a pygame display window
   pygame.display.set_mode((500, 400))
   # set the title of the display window
   pygame.display.set_caption('Alien Invasion')
   # get the display surface
   w_surface = pygame.display.get_surface()
   # create a game object
   game = Game(w_surface)
   # start the main game loop by calling the play method on the game object
   game.play()
   # quit pygame and clean up the pygame window
   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
      self.colors = ['red','yellow','blue','green','orange']
      self.enemy_color = pygame.Color('red')
      self.rows = 5
      self.columns = 6
      self.alien_grid = []
      self.create_alien_grid()

   def create_alien_grid(self):
      # - self is the Game object
      alien_width = self.surface.get_width()//self.columns
      grid_height = self.surface.get_height()//4
      alien_height = grid_height//self.rows
      for row_index in range(self.rows):
         row = []
         for col_index in range(self.columns):
            color = random.choice(self.colors)
            x = col_index * alien_width
            y = row_index * alien_height
            alien =
Alien(x,y,alien_width,alien_height,pygame.Color(color),self.surface)
            row.append(alien)
         self.alien_grid.append(row)
```

```
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 = True
      if event.type == pygame.MOUSEBUTTONUP and self.continue_game:
         self.handle_mouse_up(event.pos)
def handle_mouse_up(self,position):
   # handles the mouse click
   # Drains the battery of an enemy ship if player clicks inside an enemy ship
   # - self is the Game
   for row in self.alien_grid:
      for alien in row:
         if alien.select(position):
            if alien.get_color() == self.enemy_color:
               alien.drain(self.bg_color)

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 row in self.alien_grid:
      for alien in row:
         alien.draw()
   pygame.display.update() # make the updated surface appear on the display

def update(self):
```

```
      # Update the game objects for the next frame.
      # - self is the Game to update
      for row in self.alien_grid:
         for alien in row:
            alien.move()
      self.check_edge_collision()

   def check_edge_collision(self):
      # checks if any ship in the last row has touched the bottom edge
      # or any ship in the first row has touched the top edge of the window.
      # Reverses the direction of all alien ships if grid touches top edge or
      # bottom edge of the window
      # - self is the Game

      if self.alien_grid[0][0].touch_edge() or
self.alien_grid[self.rows-1][self.columns-1].touch_edge():
         for row in self.alien_grid:
            for alien in row:
               alien.reverse_direction()

   def decide_continue(self):
      # Check and remember if the game should continue
      # - self is the Game to check
      self.continue_game = False
      for row in self.alien_grid:
         for alien in row:
            if alien.get_color() == self.enemy_color:
               self.continue_game = True


class Alien:
      #an object of this class is an Alien ship

      def __init__(self,x,y,width,height,color,surface):
         # initialize the alien ship with pygame.Rect object, color, speed and
surface
         # - x of type int and y of type int are the top left corner coordinates
of an alien ship
         # - width of type int and height of type int are the dimensions of an
alien ship
         # - color of type str is the color of an alien ship
         # - surface of type pygame.Surface is the area on the which an alien
ship is drawn
         self.color = color
```

```
        self.speed = 1
        self.surface = surface
        self.rect = pygame.Rect(x,y,width,height)


    def drain(self,bg_color):
        # sets the color of the ship to the surface background color
        # - self is the Alien ship
        self.color = bg_color

    def draw(self):
        # draws an alien ship
        # - self is the Alien ship

        pygame.draw.ellipse(self.surface,self.color,self.rect)



    def move(self):
        # moves the alien ship in the vertical direction using its speed
        # - self is the Alien ship
        self.rect.move_ip(0,self.speed)

    def reverse_direction(self):
        # reverses the direction of movement of the Alien ship
        self.speed = -self.speed
    def touch_edge(self):
        # Returns True if Alien ship collides with  the bottom edge of window
        # or the top edge of the window
        # False otherwise
        # - self is the Alien ship

        return self.rect.bottom >= self.surface.get_height() or self.rect.top <
0


    def select(self,position):
        # Returns True if position collides with the Alien ship
        # False otherwise
        # - self is the Alien ship
        # - position is the (x,y) location of the click
        return self.rect.collidepoint(position)
```

```
      def get_color(self):
         # Return the color of the Alien ship
         # - self is the Alien ship
         return self.color




main()
```