```
import sys # Importerer "sys": anvendes til at afslutte spillet.
from time import sleep # Importerer "sleep" fra "time": anvendes til
at pause
                       # spillet.

import pygame # Importerer "pygame": anvendes til håndtering af
grafik, input og
              # styring af spillet.

from settings import Settings # Importerer vores Settings-klasse:
                              # håndterer spilindstillingerne.
from game_stats import GameStats # Importerer vores GameStats-
klasse:
                                 # gemmer/håndterer
spilstatistikker.
from scoreboard import Scoreboard #---: viser scoren.
from button import Button #---: generer og viser "Start!"-knappen.
from ship import Ship #---: En object-repræsentation af rumskibet.
from bullet import Bullet #---: En object-repræsentation af skud som
en liste,
                          #     der opdateres.
from alien import Alien #---: Repræsenterer målene og opdaterer dem,
hvis ramt.


class AlienInvasion:
    """ Den generelle klasse, der styrer spillet og ressourcerne"""

    def __init__(self):
        """
        Initialiserer spillet og starter de
        nødvendige recourser (spillets objekter)
        """
        pygame.init()
        self.clock = pygame.time.Clock()
        self.settings = Settings()

        # Oprette spil-skærmen, baseret på dimensionerne
        # angivet i settings-klassen
        self.screen = pygame.display.set_mode(
            (self.settings.screen_width,
self.settings.screen_height))
        # Printer titlen på vores spilvindue.
        pygame.display.set_caption("Alien Invasion")

        # Starter vores spilstatistik og scoreboard.
        self.stats = GameStats(self)
        self.sb = Scoreboard(self)

        # Initialiserer/opretter skib-objektet
        self.ship = Ship(self)

        # Opretter *Grupper* til at håndtere dublikerede objecter
(skud, aliens)
        self.bullets = pygame.sprite.Group()
        self.aliens = pygame.sprite.Group()

        # Opretter *Gruppen* af aliens objecter.
        self._create_fleet()

        # Sætter default status på spille til "inaktiv". Status
bliver "aktiv"
        # når vi strykker på "Start!"-knappen
        self.game_active = False

        # Opretter knapper, der gør det muligt at gøre spillet
"aktivt"
        self.play_button = Button(self, "Start!")

    def run_game(self):
        """
        Påbegynder den løkke, hvor i spillet kører
        og løbende opdaterer sig selv baseret på input
        """
        while True:
            # Tjekker kontinuerligt inputs (fra mus eller tastetur)
            self._check_events()

            # Når spillet er aktivt:
            if self.game_active:
                # Opdateres skibets position
                self.ship.update()
                # Opdateres skuds position og tjekker om skud
                # rammer andre opbjekter (aliens)
                self._update_bullets()
                # Opdateres aliens positions og tjekker om skud
rammer alines
                # objecter.
                self._update_aliens()

            # Opdaterer skærmen iht. positioner og evt.
            # kollision mellem objecter (skud, aliens)
            self._update_screen()
            # Begrænser opdateringshastigheden i løkken til 60FPS.
            self.clock.tick(60)

    def _check_events(self):
        """
        Funktionen, der håndterer/tjekker alle inputs fra mus og
tastetur,
        og løbende hændelser i spillet.
        """
        # Tjekker alle de inputs, der er foretaget siden sidste
opdatering
        # (basere på 60FPS raten)
        for event in pygame.event.get():
            # Hvis spilvinduet lukkes (trykker rød X), slutter
spillet.
```

```
            if event.type == pygame.QUIT:
                sys.exit()
            # Hvis en knap trykkes NED
            elif event.type == pygame.KEYDOWN:
                self._check_keydown_events(event)
            # Hvis en trykket knap SLIPPES.
            elif event.type == pygame.KEYUP:
                self._check_keyup_events(event)
            # Hvis der trykkes på MUSEN (højreklik), tjekkes der om
der
            # rykkes på den definerede play-knap.
            elif event.type == pygame.MOUSEBUTTONDOWN:
                mouse_pos = pygame.mouse.get_pos()
                self._check_play_button(mouse_pos)

    def _check_play_button(self, mouse_pos):
        """
        Starter et nyt spil, når man med MUSEN trykker på start-
knappen.
        """
        button_clicked =
self.play_button.rect.collidepoint(mouse_pos)
        # Hvis knappen trykkes og spillet ikke allerede er aktivt.
        if button_clicked and not self.game_active:
            # Nulstiller/begynder med default "settings"
            self.settings.initialize_dynamic_settings()

            # Nulstiller spilstatistikkerne
            self.stats.reset_stats()
            self.sb.prep_score()
            self.sb.prep_level()
            self.sb.prep_ships()
            self.game_active = True

            # Fjerner alle gamle skud og aliens fra forrige spil
            self.bullets.empty()
            self.aliens.empty()

            # Opretter en ny "flåde" af aliens og starter rumskipets
            # placering i midten (centreret)
            self._create_fleet()
            self.ship.center_ship()

            # Gør MUSEN usynlig mens spillet kører.
            pygame.mouse.set_visible(False)

    def _check_keydown_events(self, event):
        """
        Metoden, der tager tryk på TASTETUR som inputs.
        """
        # Hvis HØJRE-pil trykkes bevæger skibet sig til højre.
        if event.key == pygame.K_RIGHT:
            self.ship.moving_right = True
        #-------#
        elif event.key == pygame.K_LEFT:
            self.ship.moving_left = True
        # Hvis Q-tasten trykkes, slutter spillet.
        elif event.key == pygame.K_q:
            sys.exit()
        # Hvis MELLEMRUM-tasten trykkes, skyder skibet et bullet-
object.
        elif event.key == pygame.K_SPACE:
            self._fire_bullet()

    def _check_keyup_events(self, event):
        """
        Metoden, der tager et tastetur-SLIP som input
        """
        # Stopper bevægelse mod højre, når højre-pil SLIPPES.
        if event.key == pygame.K_RIGHT:
            self.ship.moving_right = False
        #--------------#
        elif event.key == pygame.K_LEFT:
            self.ship.moving_left = False

    def _fire_bullet(self):
        """
        Metoden, der affyrer et skud, når mellemrum trykkes,
        såfremt der er plads til flere skud i listen, baseret
        på settings-parametrene.
        """
        # Tjekker først at antallet af "aktive" skud-objecter,
        # ikke overskrider grænsen.
        if len(self.bullets) < self.settings.bullets_allowed:
            # Opretter et nyt skud-object.
            new_bullet = Bullet(self)
            # Tilføjer skudet til listen/Gruppen af aktive skud.
            self.bullets.add(new_bullet)

    def _update_bullets(self):
        """
        Opdaterer den løbende position af skudende,
        og håndterer en kollision mellem objecter (aliens, bullets)
        """
        # Opdaterer skudenes kontinuerlige position
        self.bullets.update()

        # Fjerne skud, der har forladt skærmens parametre.
        for bullet in self.bullets.copy():
            if bullet.rect.bottom <= 0:
                self.bullets.remove(bullet)

        # Tjekker om et skud rammer et alien object.
        self._check_bullet_alien_collisions()

    def _check_bullet_alien_collisions(self):
        """
        Håndterer kollisionen mellem skud og aliens.
```

```
        """
        # Fjerner både skuddet og alien objectet i tilfælde af
kollision.
        collisions = pygame.sprite.groupcollide(
                self.bullets, self.aliens, True, True)

        # Ved tilfælde af kollision:
        if collisions:
            for aliens in collisions.values():
                # Tilføjer point for hver fjende, der rammes.
                self.stats.score += self.settings.alien_points *
len(aliens)
            # Opdaterer scoretavlen.
            self.sb.prep_score()
            # Tjekker om en ny highscore er opnået.
            self.sb.check_high_score()

        # Hvis alle aliens er fjernet, start næste niveau:
        if not self.aliens:
            # Fjern alle skud på skærmen.
            self.bullets.empty()
            # Opret ny "flåde" af aliens.
            self._create_fleet()
            # Øger spillets hastighed.
            self.settings.increase_speed()

            # Viser det nye opdaterede niveau på scoretavlen.
            self.stats.level += 1
            self.sb.prep_level()

    def _ship_hit(self):
        """
        Metode, der definerer hvad der sker,
        når skibet rammes af en alien.
        """
        # Tjekker om der er flere liv tilbage.
        if self.stats.ships_left > 0:
            # Reducerer antallet af tilbageværende liv med 1.
            self.stats.ships_left -= 1
            # Opdaterer antallet af synlige skibe på scoreboardet.
            self.sb.prep_ships()

            # Fjerner alle skud
            self.bullets.empty()
            # Fjerner alle aliens
            self.aliens.empty()

            # Opretter ny flåde af aliens
            self._create_fleet()
            # Placerer skibet i midten
            self.ship.center_ship()

            # Sætter spillet på pause i 0,5 sekunder.
            sleep(0.5)

        # Hvis der ikke er flere liv tilbage:
        else:
            # Spil afsluttes
            self.game_active = False
            # MUSEN bliver synlig på ny.
            pygame.mouse.set_visible(True)

    def _update_aliens(self):
        """
        Metode, der kontinuerligt opdaterer aliens position og
        tjekker om der er kollision med rumskibet
        """
        # Tjekker om en alien rammer skærmens kant
        self._check_fleet_edges()
        # Opdaterer position og bevægelse mod ny retning (højre/
venstre)
        self.aliens.update()

        # Tjekker om en fjende rammer skibet
        if pygame.sprite.spritecollideany(self.ship, self.aliens):
            self._ship_hit()

        # Tjekker om en alien rammer bunden af skærmen.
        self._check_aliens_bottom()

    def _check_aliens_bottom(self):
        """
        Metoden, der tjekker om en alien rammer bunden af skærmen
        og hvad der sker i så fald.
        """
        for alien in self.aliens.sprites():
            # Hvis en alien rammer bunden.
            if alien.rect.bottom >= self.settings.screen_height:
                # Fjern et liv fra spilleren og scoreboardet.
                self._ship_hit()
                break

    def _create_fleet(self):
        """
        Metoden, der opretter en "flåde" af aliens
        """
        # Opretter alien objectet
        alien = Alien(self)
        # Henter størrelsen på alien objectet
        alien_width, alien_height = alien.rect.size
        current_x, current_y = alien_width, alien_height

        # Fylder spil-skærmen med fjender, indtil der ikke er plads
til flere
        # givet aliens størrelse.
        while current_y < (self.settings.screen_height - 3 *
alien_height):
            while current_x < (self.settings.screen_width - 2 *
alien_width):
```

```
                self._create_alien(current_x, current_y)  # Opretter
en fjende
                                                          # ved hver
position.
                current_x += 2 * alien_width

            current_x = alien_width
            current_y += 2 * alien_height

    def _create_alien(self, x_position, y_position):
        """
        Opretter aliens og placerer dem på gyldige positioner
        """

        # Tilføjer hver alien til Gruppe af aliens.
        new_alien = Alien(self)
        new_alien.x = x_position
        new_alien.rect.x = x_position
        new_alien.rect.y = y_position
        self.aliens.add(new_alien)

    def _check_fleet_edges(self):
        """
        Metoden, der tjekker om nogle aliens rammer skærmens kant
        og alle skifter retning.
        """
        for alien in self.aliens.sprites():
            # Hvis en alien når skærmens kant, ændres retningen
(højre/venstre)
            if alien.check_edges():
                self._change_fleet_direction()
                break

    def _change_fleet_direction(self):
        """
        Metoden, der får flåden til at ændre retning,
        og får flåden til kontinuerligt at bevæge sig ned ad.
        """
        for alien in self.aliens.sprites():
            alien.rect.y += self.settings.fleet_drop_speed
        self.settings.fleet_direction *= -1

    def _update_screen(self):
        """
        Metoden, der opdaterer skærmen med
        alle initialiserede spil-objecter
        """
        # Fylder skærmen med valgte baggrundsfarve
        self.screen.fill(self.settings.bg_color)

        # Tegner alle skud på skærmen
        for bullet in self.bullets.sprites():
            bullet.draw_bullet()
        # Tegner skibet på skærmen
        self.ship.blitme()
        # Tegner alle alines på skærmen
        self.aliens.draw(self.screen)
        # Viser spillerens score og antal liv
        self.sb.show_score()

        # Hvis spillet er inaktiv, vises Start-knappen
        if not self.game_active:
            self.play_button.draw_button()

        # Opdaterer skærmen med det nyeste indhold
        pygame.display.flip()


if __name__ == '__main__':
    # Opretter en instants af spillet og starter det
    ai = AlienInvasion()
    # Kører hoved-løkken i spillet, der administrerer
    # alle objecter og bevægelser.
    ai.run_game()
```