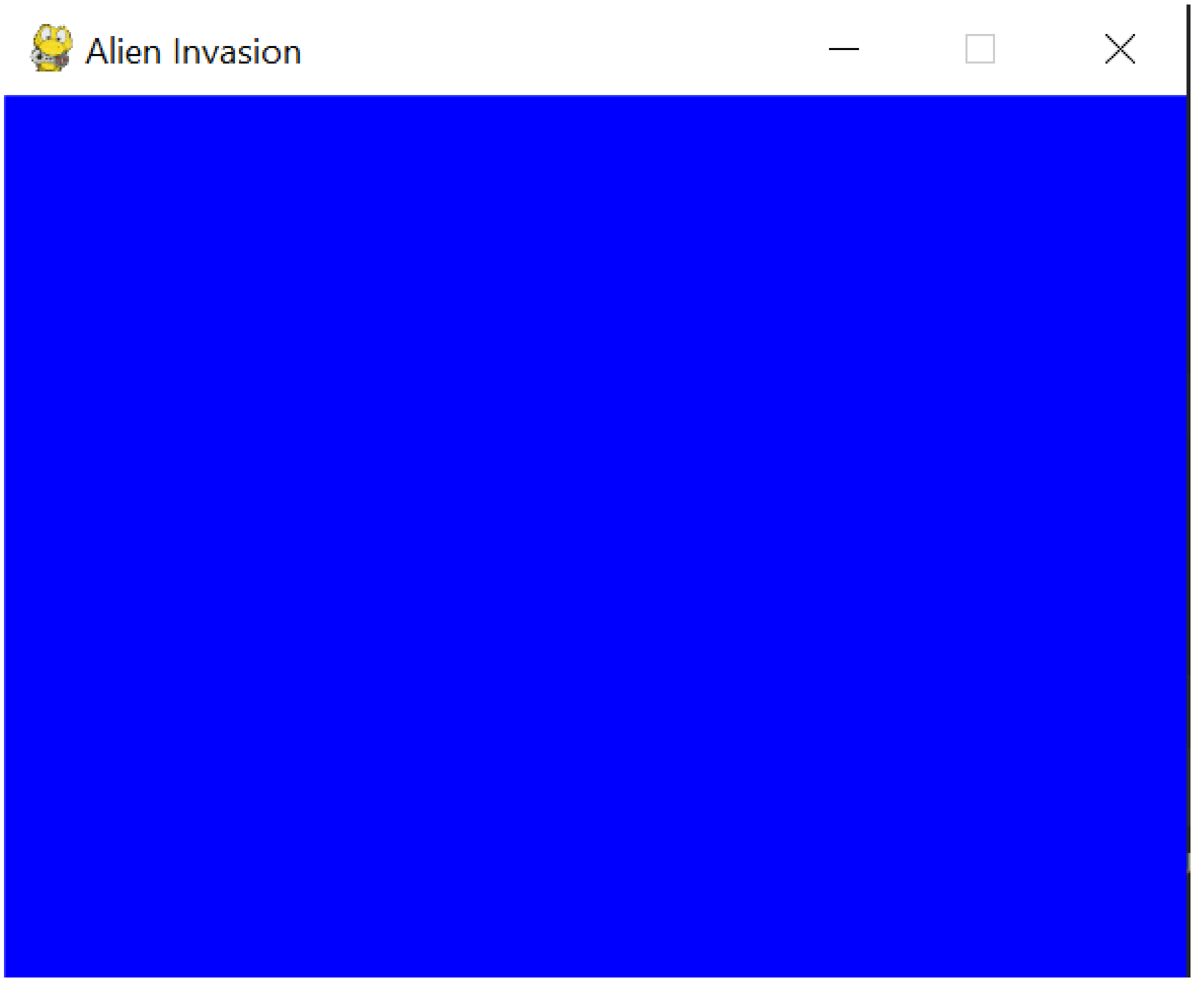
Alien Invasion (Chapters 12)

Create a seperate file in your alien invasion game for each of the following questions. Paste the content of the file into a Jupyter Notebook cell along with a screen shot of the game (2 cells per answer)

12-1. Blue Sky: Make a Pygame window with a blue background

In [2]:
```
# code

import pygame
import time

pygame.init()
screen = pygame.display.set_mode((400, 300))
pygame.display.set_caption('Alien Invasion')
bg_color = (0, 0, 255)
screen.fill(bg_color)
pygame.display.flip()

time.sleep(15)
```

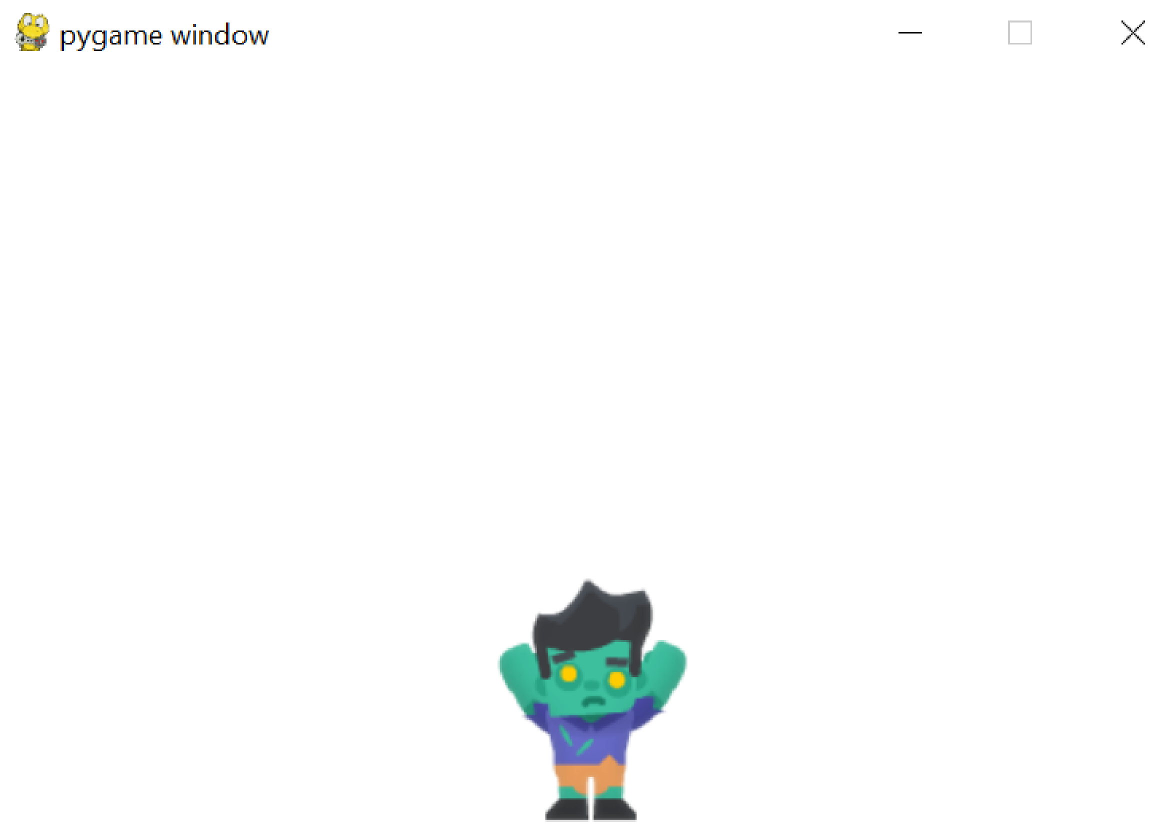
screen shot



12-2. Game Character: Find a bitmap image of a game character you like or convert an image to a bitmap. Make a class that draws the character at the center of the screen and match the background color of the image to the background color of the screen or vice versa

In [2]:
```
# code
import pygame
import time

class GameCharacter:
    def __init__(self, screen):
        self.screen = screen
        self.image = pygame.image.load('images/zombie.png')
        self.image_rect = self.image.get_rect()
        screen_rect = screen.get_rect()
        self.image_rect.center = screen_rect.center
    def show_character(self):
        self.screen.blit(self.image, self.image_rect)
pygame.init()
screen = pygame.display.set_mode((500,500))
screen. fill((255,255,255))

zombie_man = GameCharacter(screen)
zombie_man.show_character()

pygame.display.flip()
time.sleep(5)
```

screen shot

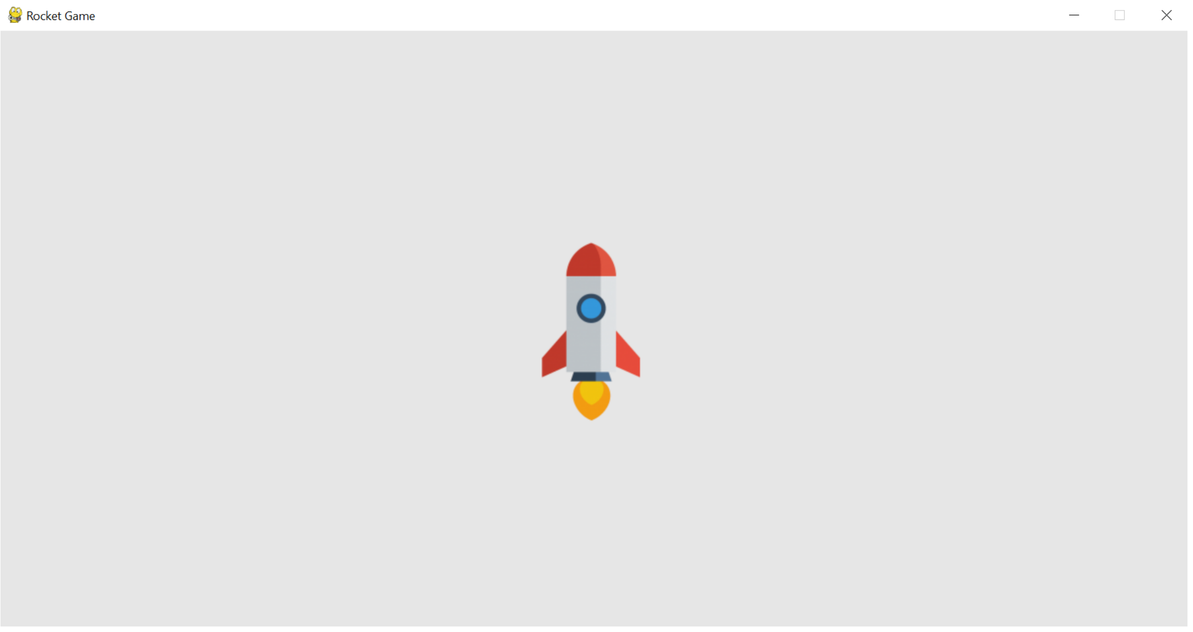


12-4. Rocket: Make a game that begins with a rocket in the center of the screen. Allow the player to move the rocket up, down, left, or right using the four arrow keys. Make sure the rocket never moves beyond any edge of the screen.

In [2]:
```
# code
class Settings:
    """A class to store all settings for Alien Invasion."""

    def __init__(self):
        """Initialize the games settings."""
        self.screen_width = 1200
        self.screen_height = 600
        self.bg_color = (230, 230, 230)

        self.rocket_speed = 1.5
import pygame
```

```
class Rocket:
    def __init__(self, ai_game):
        self.screen = ai_game.screen
        self.settings = ai_game.settings
        self.screen_rect = ai_game.screen.get_rect()
        self.image = pygame.image.load("images/red_rocket.bmp")
        self.rect = self.image.get_rect()
        self.rect.center = self.screen_rect.center

        self.x = float(self.rect.x)
        self.y = float(self.rect.y)
        self.moving_right = False
        self.moving_left = False
        self.moving_up = False
        self.moving_down = False

    def update(self):
        if self.moving_right and self.rect.right < self.screen_rect.right:
            self.x += self.settings.ship_speed
        if self.moving_left and self.rect.left > 0:
            self.x -= self.settings.ship_speed
        self.rect.x = self.x
        if self.moving_up and self.rect.top > 0:
            self.y -= self.settings.ship_speed
        if self.moving_down and self.rect.bottom < self.screen_rect.bottom:
            self.y += self.settings.ship_speed
        self.rect.y = self.y

    def blitme(self):
        self.screen.blit(self.image, self.rect)
import sys
import pygame
from settings import Settings
from rocket import Rocket

class RocketGame:
    def __init__(self):
        pygame.init()
        self.settings = Settings()
        self.screen = pygame.display.set_mode((self.settings.screen_width, self.settin
        pygame.display.set_caption("Rocket Game")
        self.rocket = Rocket(self)

    def run_game(self):
        while True:
            self._check_events()
            self.rocket.update()
            self._update_screen()

    def _check_events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                self._check_keydown_events(event)
            elif event.type == pygame.KEYUP:
                self._check_keyup_events(event)
    def _check_keydown_events(self,event):
            if event.key == pygame.K_RIGHT:
                self.rocket.moving_right = True
            elif event.key == pygame.K_LEFT:
                self.rocket.moving_left = True
            elif event.key == pygame.K_UP:
                self.rocket.moving_up = True
            elif event.key == pygame.K_DOWN:
                self.rocket.moving_down = True
            elif event.key == pygame.K_q:
                sys.exit()

    def _check_keyup_events(self, event):
                if event.key == pygame.K_RIGHT:
                    self.rocket.moving_right = False
                elif event.key == pygame.K_LEFT:
                    self.rocket.moving_left = False
                elif event.key == pygame.K_UP:
                    self.rocket.moving_up = False
                elif event.key == pygame.K_DOWN:
                    self.rocket.moving_down = False

    def _update_screen(self):
        self.screen.fill(self.settings.bg_color)
        self.rocket.blitme()
        pygame.display.flip()

if __name__ == "__main__":
    rg = RocketGame()
    rg.run_game()
```

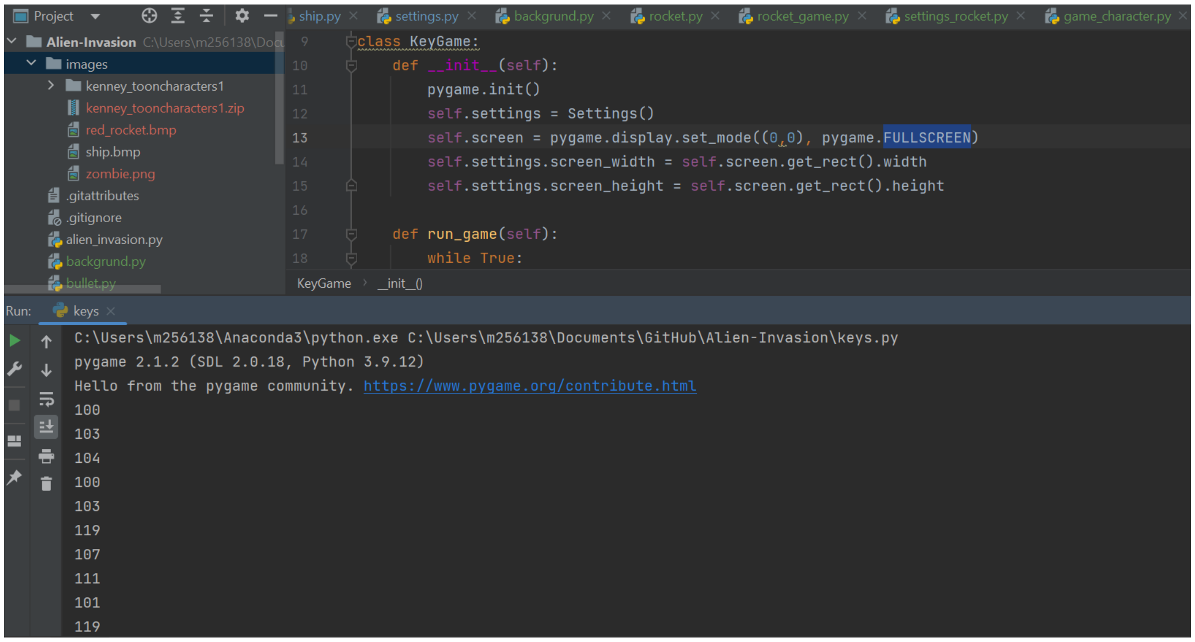
screen shot



12-5. Keys: Make a Pygame file that creates an empty screen. In the event loop, print the `event.key` attribute whenever a `pygame.KEYDOWN` event is detected. Run the program and press various keys to see how Pygame responds. Your screen shot should be the text output from the Pycharm consolse (not the game screen)

In [2]:
```python
# code
import sys
import pygame

class Settings:
    def __init__(self):
        self.screen_width = 1000
        self.screen_height = 600
        self.bg_color = (255,255,255)
class KeyGame:
    def __init__(self):
        pygame.init()
        self.settings = Settings()
        self.screen = pygame.display.set_mode((0,0), pygame.FULLSCREEN)
        self.settings.screen_width = self.screen.get_rect().width
        self.settings.screen_height = self.screen.get_rect().height

    def run_game(self):
        while True:
            self._check_events()
            self._update_screen()
    def _check_events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                self._check_keydown_events(event)
    def _check_keydown_events(self, event):
        print(event.key)
        if event.key == pygame.K_q:
            sys.exit()
    def _update_screen(self):
        self.screen.fill(self.settings.bg_color)
        pygame.display.flip()
if __name__ == '__main__':
    kg = KeyGame()
    kg.run_game()
```

screen shot



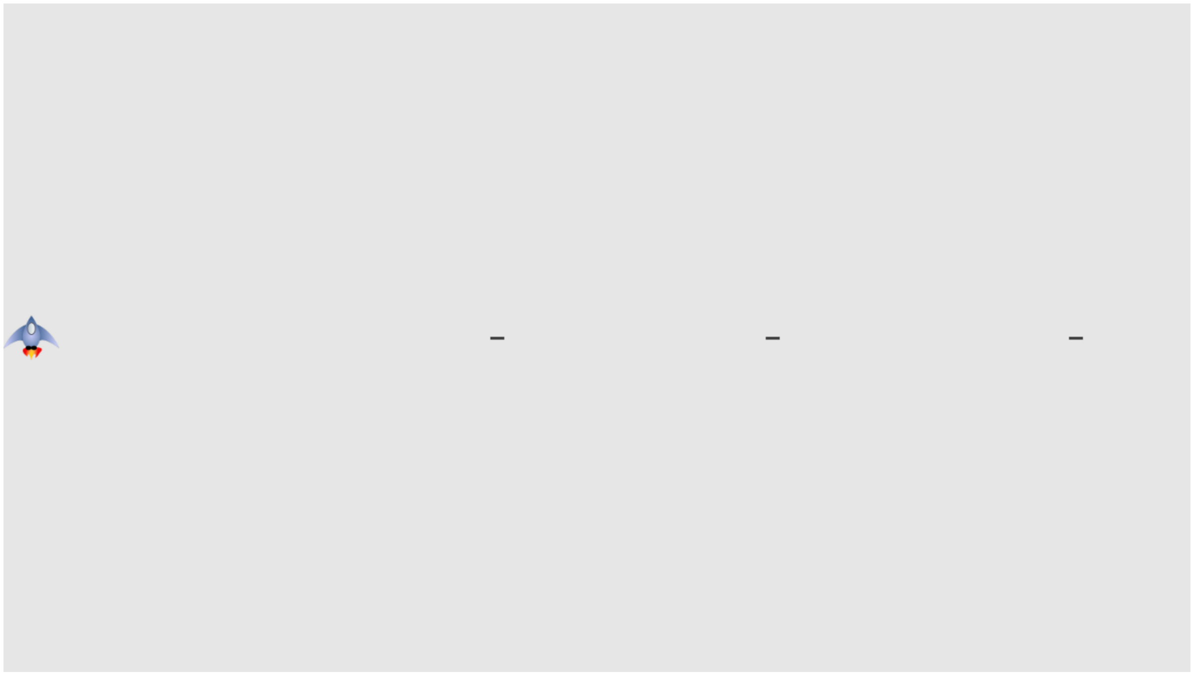
```python
class KeyGame:
    def __init__(self):
        pygame.init()
        self.settings = Settings()
        self.screen = pygame.display.set_mode((0,0), pygame.FULLSCREEN)
        self.settings.screen_width = self.screen.get_rect().width
        self.settings.screen_height = self.screen.get_rect().height

    def run_game(self):
        while True:
```

```
C:\Users\m256138\Anaconda3\python.exe C:\Users\m256138\Documents\GitHub\Alien-Invasion\keys.py
pygame 2.1.2 (SDL 2.0.18, Python 3.9.12)
Hello from the pygame community. https://www.pygame.org/contribute.html
100
103
104
100
103
119
107
111
101
119
```

12-6. Sideways Shooter: Write a game that places a ship on the left side of the screen and allows the player to move the ship up and down. Make the ship fire a bullet that travels right across the screen when the player presses the spacebar. Make sure the bullets are deleted once they disappear off the screen

In [2]:
```
# code
import sys
import pygame
from pygame.sprite import Sprite

class Bullet(Sprite):
    def __init__(self, ss_game):
        super().__init__()
        self.screen = ss_game.screen
        self.settings = ss_game.settings
        self.color = self.settings.bullet_color
        self.rect = pygame.Rect(0,0, self.settings.bullet_width, self.settings.bullet_
        self.rect.midright = ss_game.ship.rect.midright
        self.x = float(self.rect.x)
    def update(self):
        self.x += self.settings.bullet_speed
        self.rect.x = self.x
    def draw_bullet(self):
        pygame.draw.rect(self.screen, self.color, self.rect)
class Settings:
    def __init__(self):
        self.screen_width = 1000
        self.screen_height = 600
        self.bg_color = (230,230,230)
        self.ship_speed = 1.5
        self.bullet_speed = 3.0
        self.bullet_width = 15
        self.bullet_height = 3.0
        self.bullet_color = (60,60,60)
        self.bullets_allowed = 3

class Ship:
    def __init__(self, ss_game):
        self.screen = ss_game.screen
        self.settings = ss_game.settings
        self.screen_rect = ss_game.screen.get_rect()
        self.image = pygame.image.load('images/ship.bmp')
        self.rect = self.image.get_rect()
        self.rect.midleft = self.screen_rect.midleft
        self.y = float(self.rect.y)
        self.moving_up = False
        self.moving_down = False
    def update(self):
        if self.moving_up and self.rect.top > 0:
            self.y -= self.settings.ship_speed
        if self.moving_down and self.rect.bottom < self.screen_rect.bottom:
            self.y += self.settings.ship_speed
        self.rect.y = self.y
    def blitme(self):
        self.screen.blit(self.image, self.rect)
class SidewaysShooter:
    def __init__(self):
        pygame.init()
        self.settings = Settings()
        self.screen = pygame.display.set_mode((0,0), pygame.FULLSCREEN)
        self.settings.screen_width = self.screen.get_rect().width
        self.settings.screen_height = self.screen.get_rect().height
        pygame.display.set_caption("Sideways Shooter")
        self.ship = Ship(self)
        self.bullets = pygame.sprite.Group()
    def run_game(self):
        while True:
            self._check_events()
            self.ship.update()
            self._update_bullets()
            self._update_screen()
    def _update_bullets(self):
        self.bullets.update()
        for bullet in self.bullets.copy():
            if bullet.rect.left >= self.settings.screen_width:
                self.bullets.remove(bullet)
    def _check_events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                self._check_keydown_events(event)
            elif event.type == pygame.KEYUP:
                self._check_keyup_events(event)
    def _check_keydown_events(self,event):
        if event.key == pygame.K_UP:
            self.ship.moving_up = True
        elif event.key == pygame.K_DOWN:
            self.ship.moving_down = True
        elif event.key == pygame.K_SPACE:
            self._fire_bullet()
        elif event.key == pygame.K_q:
            sys.exit()

    def _check_keyup_events(self,event):
        if event.key == pygame.K_UP:
            self.ship.moving_up = False
        elif event.key == pygame.K_DOWN:
            self.ship.moving_down = False

    def _fire_bullet(self):
        if len(self.bullets) < self.settings.bullets_allowed:
            new_bullet = Bullet(self)
            self.bullets.add(new_bullet)

    def _update_screen(self):
        self.screen.fill(self.settings.bg_color)
        self.ship.blitme()
        for bullet in self.bullets.sprites():
            bullet.draw_bullet()
        pygame.display.flip()
if __name__ == '__main__':
    ss = SidewaysShooter()
    ss.run_game()
```

screen shot



```
In [ ]:
```