1. Introduction

- The spaceship can reload and shoot bullets
- The invaders will move towards the bottom
- When invaders are hit by bullets. Invaders will be destroyed.
- If 5 invaders came through the bottom line, the player lose.
- If 20 invaders are destroyed, the player will win.
- When the game is over, the player will be asked if he or she wants to play again

2. Design and Implementation

2.1 Objects

1. There are 3 kinds of objects in the game: spaceship, bullet, invader. Each of them can be represented by a customized Turtle object. Since there will be many invaders and bullets. I created classes for them.
2. To make the game more interesting, each invader is created with a random choice of color.

2.2 Functions

1. Move the spaceship

This is done by using “screen listener” and assign new x coordinate value to the spaceship.

```
screen.onkey(go_right, "Right")
```

2. Attach the bullet to the spaceship

This is achieved by assigned the bullet to a dynamic position, which has the same x values with the spaceship and higher.

3. Shoot the bullet and reload

Whenever the space is pressed, the bullet will be “released”. Then a new bullet will be created and attach to the spaceship.

```
    b_list.append(new_bullet)
```

4. Invader and bullet movement

While game is on, all invaders are iterated and move towards the bottom.

All the bullets in the bullet list(b_list) except the last one (the one attached to the spaceship) are iterated. In each iteration, the bullet will move one step closer to the end.

2.3 Game

1. Generate invaders randomly.

```
```

Note: Sometimes (1 out 10) an invader is created.

2. Remove the invaders if they are ‘hit’ by the bullet.

```
        invader.goto(1000, 1000)
```

Note: If the distance between a bullet and invader is too close, the invader will be appointed to a far way position that is not on the screen. The destroyed invader is also removed from the list. Because as time go by, there are more and more invaders in the list, which makes the iteration slow. For the same reason, bullets are removed from the bullet list when they are out of the screen too.

3. Game over

```
destroyed = 0
```

Note: the number of invaders successfully made it to the bottom are tracked. When 5 of them are in the end, game is over. The number of invaders destroyed are tracked too. When 20 of them are destroyed, the play won.

3. Conclusions

- **Discuss what you personally learned from your project.**

1. Nested for loops, in order for the invader and bullet to “meet”. I used a for loop inside a for loop. So each invader is checked with each bullet to see if they are too close. Kind of like the Bubble Sort algorithm that has a O(n2). This can get really slow as there are more and more bullets and invaders. I overcome this by removing useless bullets and invaders from the list.

2. I used stack overflow to figure out how to restart a python program and pop up windows to communicate with the user.

- **Discuss the best features and the shortcomings of the project.**

1. Invaders are randomly generated so it is more like a game.
2. The bullet can go with the spaceship and “be fired”.
3. When the game ends, it asks if you want to play again.

- **Describe any additional features you may want to add in the future.**

1. A scoreboard
2. More difficulty levels
3. Use better images to represent bullet, spaceship and invader.