Part 1:

1)
```
class Spaceship {
   constructor(){
    this.velocity = new createVector(0, 0);
    this.location = new createVector(width/2, height/2);
    this.acceleration = new createVector(0, 0);
    this.maxVelocity = 5;
    this.bulletSys = new BulletSystem();
    this.size = 50;
 }
```

2)
```
   //bullet collisions
   //YOUR CODE HERE (3-4 lines approx)
   var bulletSys = spaceship.bulletSys;
   var bullets = bulletSys.bullets;
   for(var i=0; i<bullets.length;i++){
   for(var j=0;j<asteroids.locations.length;j++){
    var asteriodsLoc = asteroids.locations[j];
     var asteriodsDiam = asteroids.diams[j];
     if(isInside(asteriodsLoc,asteriodsDiam,bullets[i],bulletSys.diam)){
      asteroids.destroy(j);
      score += 1;
       }
      }
```

3)
```
var spaceship;
var asteroids;
var atmosphereLoc;
var atmosphereSize;
var earthLoc;
var earthSize;
var starLocs = [];

var score = 0;

/////////////////////////////////////////////////
function setup() {
createCanvas(1200,800);
 spaceship = new Spaceship();
 asteroids = new AsteroidSystem();

//location and size of earth and its atmosphere

 atmosphereLoc = new createVector(width/2, height*2.9);
  atmosphereSize = new createVector(width*3, width*3);
  earthLoc = new createVector(width/2, height*3.1);
  earthSize = new createVector(width*3, width*3);

}
```

4)
```
applyForce(f){
 this.acceleration.add(f);
} // sets how fast the spaceship will move

interaction(){
   if (keyIsDown(LEFT_ARROW)){
   this.applyForce(createVector(-0.1, 0));
     }
    if (keyIsDown(RIGHT_ARROW)){
     // YOUR CODE HERE (1 line)
         this.applyForce(createVector(0.1, 0));
     }
    if (keyIsDown(UP_ARROW)){
    // YOUR CODE HERE (1 line)
        this.applyForce(createVector(0, -0.1));
     }
     if (keyIsDown(DOWN_ARROW)){
     // YOUR CODE HERE (1 line)
     this.applyForce(createVector(0, 0.1));
     }
  }
```

<u>Part 2:</u>

```
def getVariance(vals):

# Number of observations

   n = len(vals)

# Mean of the data

   mean = sum(vals) / n

# Square deviations

   deviations = [(x - mean) ** 2 for x in vals]

# Variance

   variance = sum(deviations) / n

   return variance


def getMedian(vals):

   n = len(vals)

   index = n // 2
```

```
    # Sample with an odd number of observations
    if n % 2:
         return sorted(vals)[index]
    # Sample with an even number of observations
    return sum(sorted(vals)[index - 1:index + 1]) / 2


def getMode(vals):
    frequency = {}


    for value in vals:
        frequency[value] = frequency.get(value, 0) + 1


    most_frequent = max(frequency.values())


    modes = [key for key, value in frequency.items()
                  if value == most_frequent]


    return modes


import unittest
import snakestats


class TestForSnakeStats(unittest.TestCase):


    def test_getVariance(self):
        theVariance = snakestats.getVariance([10,20,30])
        self.assertEqual(theVariance, 66.66666666666667)




unittest.main(argv=['ignored','-v'], exit=False)
```

```
import unittest

import snakestats


class TestForSnakeStats(unittest.TestCase):


    def test_getMedian(self):

        theMedian = snakestats.getMedian([10,20,30])

        self.assertEqual(theMedian, 20)



unittest.main(argv=['ignored','-v'], exit=False)


import unittest

import snakestats


class TestForSnakeStats(unittest.TestCase):


    def test_getMode(self):

        theMode = snakestats.getMode([1000,2000,3000,4000,4000,4000,4000])

        self.assertEqual(theMode, [4000])



unittest.main(argv=['ignored','-v'], exit=False)
```