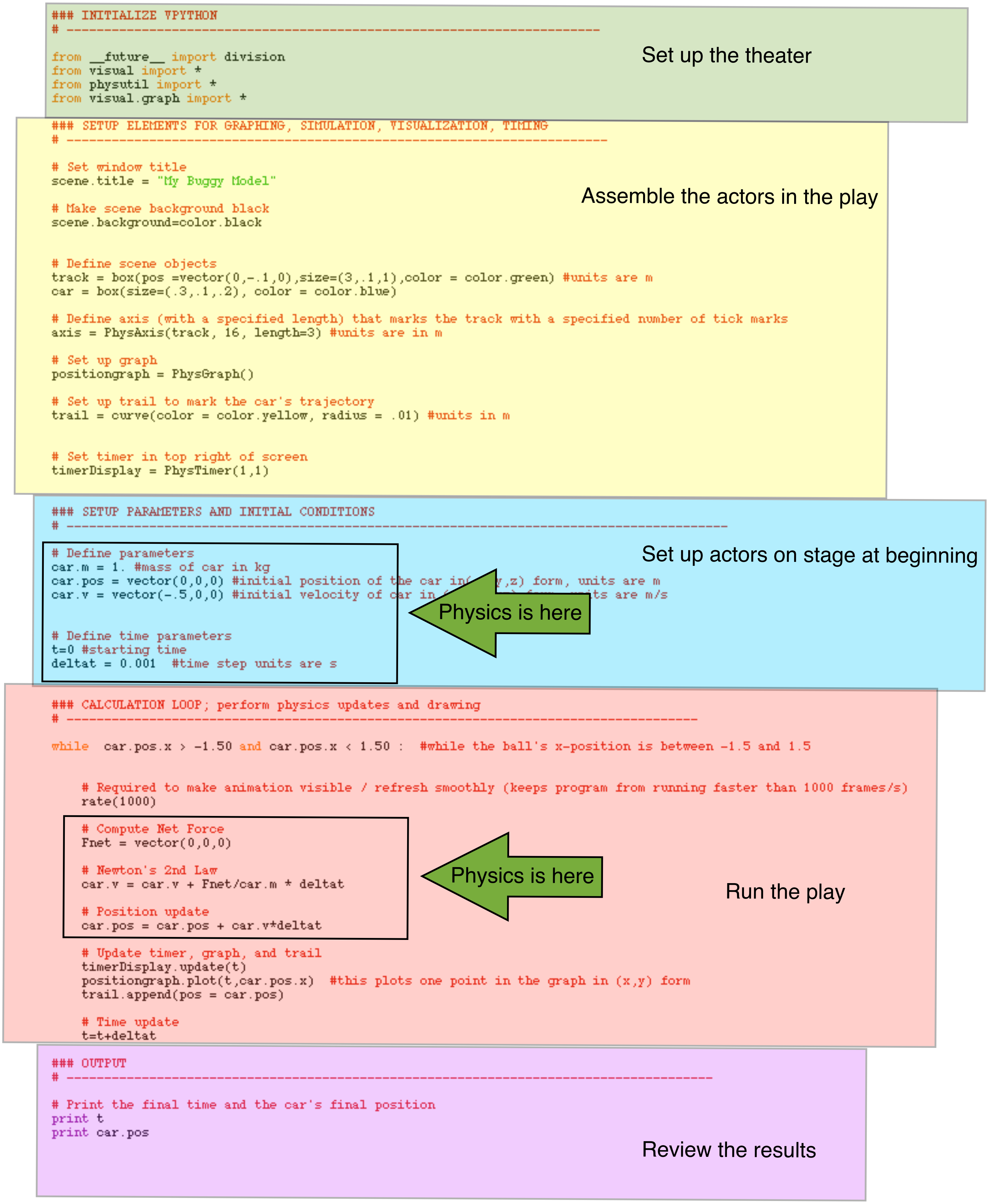

```
### INITIALIZE VPYTHON
# -----------------------------------------------------------------------

from __future__ import division
from visual import *
from physutil import *
from visual.graph import *
```

Set up the theater

```
### SETUP ELEMENTS FOR GRAPHING, SIMULATION, VISUALIZATION, TIMING
# ------------------------------------------------------------------------

# Set window title
scene.title = "My Buggy Model"

# Make scene background black
scene.background=color.black


# Define scene objects
track = box(pos =vector(0,-.1,0),size=(3,.1,1),color = color.green) #units are m
car = box(size=(.3,.1,.2), color = color.blue)

# Define axis (with a specified length) that marks the track with a specified number of tick marks
axis = PhysAxis(track, 16, length=3) #units are in m

# Set up graph
positiongraph = PhysGraph()

# Set up trail to mark the car's trajectory
trail = curve(color = color.yellow, radius = .01) #units in m


# Set timer in top right of screen
timerDisplay = PhysTimer(1,1)
```

Assemble the actors in the play

```
### SETUP PARAMETERS AND INITIAL CONDITIONS
# ----------------------------------------------------------------------------------------

# Define parameters
car.m = 1. #mass of car in kg
car.pos = vector(0,0,0) #initial position of the car in(x,y,z) form, units are m
car.v = vector(-.5,0,0) #initial velocity of car in (x,y,z) form, units are m/s


# Define time parameters
t=0 #starting time
deltat = 0.001  #time step units are s
```

Set up actors on stage at beginning

Physics is here

```
### CALCULATION LOOP; perform physics updates and drawing
# ------------------------------------------------------------------------------------

while  car.pos.x > -1.50 and car.pos.x < 1.50 :  #while the ball's x-position is between -1.5 and 1.5


    # Required to make animation visible / refresh smoothly (keeps program from running faster than 1000 frames/s)
    rate(1000)

    # Compute Net Force
    Fnet = vector(0,0,0)

    # Newton's 2nd Law
    car.v = car.v + Fnet/car.m * deltat

    # Position update
    car.pos = car.pos + car.v*deltat

    # Update timer, graph, and trail
    timerDisplay.update(t)
    positiongraph.plot(t,car.pos.x)  #this plots one point in the graph in (x,y) form
    trail.append(pos = car.pos)

    # Time update
    t=t+deltat
```

Physics is here

Run the play

```
### OUTPUT
# ---------------------------------------------------------------------------------

# Print the final time and the car's final position
print t
print car.pos
```

Review the results