Theory:

Calculation: In the zip file

a) Write the motion and measurement equations for this system and identify the matrices necessary for the Kalman filter, i.e., A, B, Q, H, R.

$$x_k = x_{k-1} + T * vx_{k-1} + \frac{T^2}{2} * ax_{k-1}$$
$$y_k = y_{k-1} + T * vy_{k-1} + \frac{T^2}{2} * ay_{k-1}$$
$$vx_k = vx_{k-1} + T * ax_{k-1}$$
$$vy_k = vy_{k-1} + T * ay_{k-1}$$

$$z_k = Hx_k + R$$

```
State Transition Matrix:
 [[1.  0.  0.5 0. ]
 [0.  1.  0.  0.5]
 [0.  0.  1.  0. ]
 [0.  0.  0.  1. ]]
Control Matrix:
 [[0.125 0.   ]
 [0.    0.125]
 [0.5   0.   ]
 [0.    0.5  ]]
Control Vector:
 [[30]
 [20]]
Process Noise Covariance Matrix:
 [[0.078125 0.       0.       0.      ]
 [0.       0.0625   0.       0.      ]
 [0.       0.       1.25     0.      ]
 [0.       0.       0.       1.      ]]
Measurement Noise Covariance Matrix:
 [[2 0]
 [0 1]]
Measurement Matrix:
 [[1 0 0 0]
 [0 1 0 0]]
```

b) Apply the Kalman filter to estimate the position and velocity of the object for the two first iterations given the measurements returned by the algorithm are as follows:

```
Predict iteration 1:
[[ 3.75]
 [ 2.5 ]
 [15.  ]
 [10.  ]]
Correct iteration 1:
[[ 4.73898858]
 [17.95620438]
 [15.65252855]
 [20.2189781 ]]
P_1:
[[1.58238173 0.         1.04404568 0.        ]
 [0.         0.88321168 0.         0.58394161]
 [1.04404568 0.         8.63988581 0.        ]
 [0.         0.58394161 0.         8.08029197]]
Predict iteration 2:
[[16.31525285]
 [30.56569343]
 [30.65252855]
 [30.2189781 ]]
Correct iteration 2:
[[ 6.17079266]
 [16.08061767]
 [19.46648645]
 [11.34984458]]
P_2:
[[1.41729389 0.         1.56281446 0.        ]
 [0.         0.78020656 0.         1.01634413]
 [1.56281446 0.         5.69842634 0.        ]
 [0.         1.01634413 0.         4.38062769]]
```

Programming:

Video demo: [Link]

Source code: within the zip file

a) I have the following measurement covariance matrix:

```python
import numpy as np
x = np.array([20, 25, 20, 10, 0, 0, 0, 0, 0, 0, 0, 0])
y = np.array([10, 30, 10, 10, 0, 0, 0, 0, 0, 0, 0, 0])
print('Variance of error x:', np.std(x)**2)
print('Variance of error y:', np.std(y)**2)
```

```
Variance of error x: 88.02083333333334
Variance of error y: 75.00000000000001
```

I decided to decrease the y error as using above measurement covariance matrix did not perform well

```python
self.Q = np.matrix([[20,0],
                    [0,20]])
```

```python
    self.R = np.matrix([[50,0],
                        [0,50]])
```

b)

Model:

$$x_k = x_{k-1} + T * vx_k$$
$$y_k = y_{k-1} + T * vy_k$$

Matrices:

```python
    def __init__(self):
        self.dt = 1
        self.A = np.matrix([[1,0],
                            [0,1]])
        self.B = np.matrix([[1,0],
                            [0,1]])
        self.u = np.matrix([[1],
                            [1]])
        self.Q = np.matrix([[20,0],
                            [0,20]])

        self.H = np.matrix([[1,0],
                            [0,1]])
        self.R = np.matrix([[50,0],
                            [0,50]])

        self.x = np.matrix([[320],
                            [180]])
        self.P = np.matrix([[1,0],
                            [0,1]])
        self.I = np.eye(2)
```