```python
import numpy as np
import random
import pylab


def f(x, t):
    return -x**3 + np.sin(t)


def RungeKutta(func, init, a, b, h=0.1):
    # h = abs(b - a) / N
    X = []
    T = np.arange(a, b, h)
    x = init
    for t in T:
        X.append(x)
        k1 = h * func(x, t)
        k2 = h * func(x + k1 / 2, t + h / 2)
        k3 = h * func(x + k2 / 2, t + h / 2)
        k4 = h * func(x + k3, t + h)
        x += (1 / 6) * (k1 + 2 * k2 + 2 * k3 + k4)
    return T, X


def RungeKutta2(f1, f2, x_init, y_init, a, b, h=0.1):
    X = []
    Y = []
    T = np.arange(a, b, h)
    x = x_init
    y = y_init
    for t in T:
        # TODO REmove THis!
        if (x > np.pi):
            x -= 2 * np.pi
        if (x < -np.pi):
            x += 2 * np.pi
        X.append(x)
        Y.append(y)
        k1 = h * f1(x, y, t)
        l1 = h * f2(x, y, t)
        k2 = h * f1(x + k1 / 2, y + l1 / 2, t + h / 2)
        l2 = h * f2(x + k1 / 2, y + l1 / 2, t + h / 2)
        k3 = h * f1(x + k2 / 2, y + l2 / 2, t + h / 2)
        l3 = h * f2(x + k2 / 2, y + l2 / 2, t + h / 2)
        k4 = h * f1(x + k3, y + l3, t + h)
        l4 = h * f2(x + k3, y + l3, t + h)
        x += (1 / 6) * (k1 + 2 * k2 + 2 * k3 + k4)
        y += (1 / 6) * (l1 + 2 * l2 + 2 * l3 + l4)
    return T, X, Y


def main():
    h = 0.1
    g = 9.8
    l = 9.8
    q = 0.5
    fD = 1.2
    t_max = 100
    OmegaD = 2 / 3
    ep = 1e-6
    f1 = lambda theta, omega, t: omega
    f2 = lambda theta, omega, t: -(g / l) * np.sin(theta) - q * omega + fD * np.sin(OmegaD * t)
```

```python

    # Part a
    print("A")
    T1, Theta1, Omega1 = RungeKutta2(f1, f2, 0.2, 0, 0, t_max, h)
    T2, Theta2, Omega2 = RungeKutta2(f1, f2, 0.2 + ep, 0, 0, t_max, h)
    theta = [np.log(np.fabs(Theta1[i] - Theta2[i])) for i in range(len(Theta1))]
    pylab.plot(np.arange(0, t_max, h), theta)
    pylab.show()

    # Part b
    print("B")
    theta = []
    for i in range(0, 100):
        print(">>{}".format(i))
        theta_0 = random.uniform(0.2, 0.21)
        T1, Theta1, Omega1 = RungeKutta2(f1, f2, theta_0, 0, 0, t_max, h)
        T2, Theta2, Omega2 = RungeKutta2(f1, f2, theta_0 + ep, 0, 0, t_max, h)
        if theta:
            theta = [
                theta[i] + np.log(np.fabs(Theta1[i] - Theta2[i]))
                for i in range(len(Theta1))
            ]
        else:
            theta = [
                np.log(np.fabs(Theta1[i] - Theta2[i]))
                for i in range(len(Theta1))
            ]

    theta = [x / 100 for x in theta]
    E_x = 0.0
    E_y = 0.0
    E_xx = 0.0
    E_xy = 0.0
    for pt in zip(np.arange(0, t_max, h), theta):
        E_x += pt[0]
        E_y += pt[1]
        E_xx += (pt[0]**2)
        E_xy += (pt[0] * pt[1])
    N = len(np.arange(0, t_max, h))
    E_x /= N
    E_y /= N
    E_xx /= N
    E_xy /= N
    m = (E_xy - E_x * E_y) / (E_xx - E_x**2)
    c = (E_xx * E_y - E_x * E_xy) / (E_xx - E_x**2)
    print("Y={}*x+{}".format(m,c))
    pylab.plot(np.arange(0, t_max, h), theta)
    pylab.plot(
        np.arange(0, t_max, h), [m * x + c for x in np.arange(0, t_max, h)])
    pylab.show()


if __name__ == "__main__":
    main()
```