```
#!/usr/bin/python3

import numpy as np
import matplotlib.pyplot as plt

def f(x, t):
    return 2*t*x

def act(t):
    return 2*np.exp(np.power(t,2.0))

def RungeKutta(func, init, a, b, h=0.025):
    X = []
    T = np.arange(a, b + h, 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 X, T

# PROBLEM 1
v0x = 600*np.cos(3.1415/3)
v0y = 600*np.sin(3.1415/3)
def x(t,c):
    return v0x/c*(1-np.exp(-c*t))
def y(t,c):
    return -(9.81/c)*t+(v0y*c+9.81)/np.power(c,2.0)*(1-np.exp(-c*t))
X1 = []
Y1 = []
t1 = 0
X1.append(v0x*t1)
Y1.append(v0y*t1-0.5*9.81*np.power(t1,2.0))
while Y1[-1] >= 0:
    X1.append(v0x*t1)
    Y1.append(v0y*t1-0.5*9.81*np.power(t1,2.0))
    t1 += 1.0
X2 = []
Y2 = []
t2 = 0
X2.append(x(t2, 0.005))
Y2.append(y(t2,0.005))
while Y2[-1] >= 0:
    X2.append(x(t2, 0.005))
    Y2.append(y(t2,0.005))
    t2 += 1.0
X3 = []
Y3 = []
t3 = 0
X3.append(x(t3, 0.01))
Y3.append(y(t3, 0.01))
while Y3[-1] >= 0:
    X3.append(x(t3, 0.01))
    Y3.append(y(t3, 0.01))
    t3 += 1.0

fig, (ax1, ax2) = plt.subplots(2)
ax1.plot(X1, Y1, 'k-', label="C=0.0")
ax1.plot(X2, Y2, 'k--', label="C=0.005")
ax1.plot(X3, Y3, 'k:', label="C=0.01")
```

```python
ax1.set_title("#1")
ax1.legend()
# plt.plot(X1, Y1)
# plt.plot(X2, Y2)
# plt.plot(X3, Y3)
# plt.show()


# PROBLEM 2
X, T = RungeKutta(f, 2.0, 0, 1, 0.025)
ax2.plot(T, [act(t) for t in T], 'k--', label="Analytical")
ax2.plot(T, X, 'k-', label="Runge-Kutta")
ax2.set_title("#2")
ax2.set_xlabel("X(1)=5.465")
ax2.legend()
fig.set_size_inches(6,9)
# plt.show()
plt.savefig("hw2.ps", papertype="a4")
```