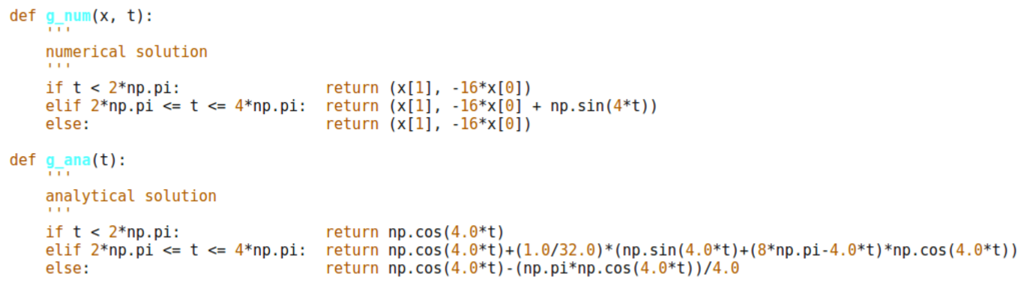
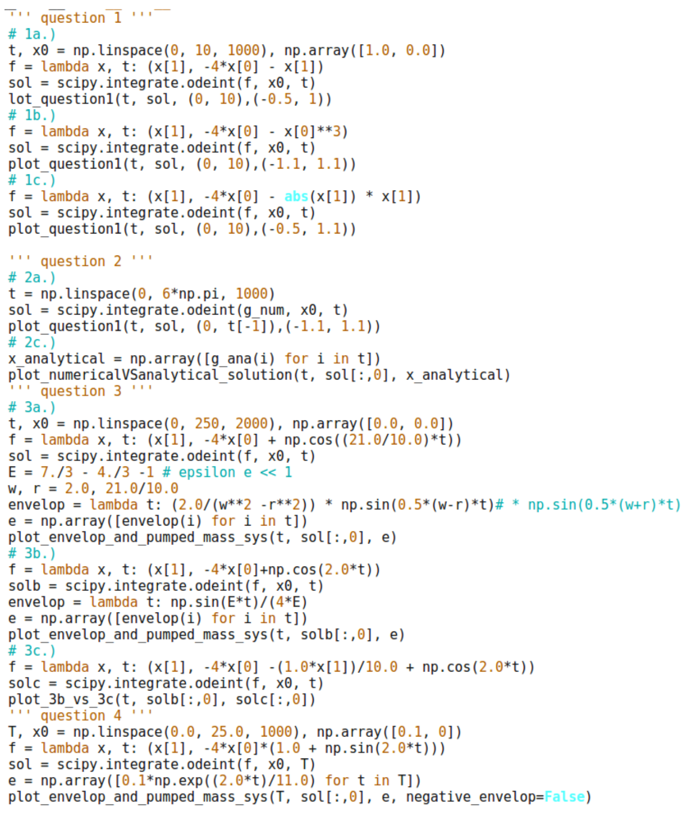
The following is the source code for Assignment05:



```
def g_num(x, t):
    '''
    numerical solution
    '''
    if t < 2*np.pi:                 return (x[1], -16*x[0])
    elif 2*np.pi <= t <= 4*np.pi:   return (x[1], -16*x[0] + np.sin(4*t))
    else:                           return (x[1], -16*x[0])

def g_ana(t):
    '''
    analytical solution
    '''
    if t < 2*np.pi:                 return np.cos(4.0*t)
    elif 2*np.pi <= t <= 4*np.pi:   return np.cos(4.0*t)+(1.0/32.0)*(np.sin(4.0*t)+(8*np.pi-4.0*t)*np.cos(4.0*t))
    else:                           return np.cos(4.0*t)-(np.pi*np.cos(4.0*t))/4.0
```



```
''' question 1 '''
# 1a.)
t, x0 = np.linspace(0, 10, 1000), np.array([1.0, 0.0])
f = lambda x, t: (x[1], -4*x[0] - x[1])
sol = scipy.integrate.odeint(f, x0, t)
lot_question1(t, sol, (0, 10),(-0.5, 1))
# 1b.)
f = lambda x, t: (x[1], -4*x[0] - x[0]**3)
sol = scipy.integrate.odeint(f, x0, t)
plot_question1(t, sol, (0, 10),(-1.1, 1.1))
# 1c.)
f = lambda x, t: (x[1], -4*x[0] - abs(x[1]) * x[1])
sol = scipy.integrate.odeint(f, x0, t)
plot_question1(t, sol, (0, 10),(-0.5, 1.1))

''' question 2 '''
# 2a.)
t = np.linspace(0, 6*np.pi, 1000)
sol = scipy.integrate.odeint(g_num, x0, t)
plot_question1(t, sol, (0, t[-1]),(-1.1, 1.1))
# 2c.)
x_analytical = np.array([g_ana(i) for i in t])
plot_numericalVSanalytical_solution(t, sol[:,0], x_analytical)
''' question 3 '''
# 3a.)
t, x0 = np.linspace(0, 250, 2000), np.array([0.0, 0.0])
f = lambda x, t: (x[1], -4*x[0] + np.cos((21.0/10.0)*t))
sol = scipy.integrate.odeint(f, x0, t)
E = 7./3 - 4./3 -1 # epsilon e << 1
w, r = 2.0, 21.0/10.0
envelop = lambda t: (2.0/(w**2 -r**2)) * np.sin(0.5*(w-r)*t)# * np.sin(0.5*(w+r)*t)
e = np.array([envelop(i) for i in t])
plot_envelop_and_pumped_mass_sys(t, sol[:,0], e)
# 3b.)
f = lambda x, t: (x[1], -4*x[0]+np.cos(2.0*t))
solb = scipy.integrate.odeint(f, x0, t)
envelop = lambda t: np.sin(E*t)/(4*E)
e = np.array([envelop(i) for i in t])
plot_envelop_and_pumped_mass_sys(t, solb[:,0], e)
# 3c.)
f = lambda x, t: (x[1], -4*x[0] -(1.0*x[1])/10.0 + np.cos(2.0*t))
solc = scipy.integrate.odeint(f, x0, t)
plot_3b_vs_3c(t, solb[:,0], solc[:,0])
''' question 4 '''
T, x0 = np.linspace(0.0, 25.0, 1000), np.array([0.1, 0])
f = lambda x, t: (x[1], -4*x[0]*(1.0 + np.sin(2.0*t)))
sol = scipy.integrate.odeint(f, x0, T)
e = np.array([0.1*np.exp((2.0*t)/11.0) for t in T])
plot_envelop_and_pumped_mass_sys(T, sol[:,0], e, negative_envelop=False)
```