model = Model(...)  
dt, nt = <timestepping parameters>  

# Define source and receiver geometry  
src = RickerSource(...)  
rec = Receiver(...)  

# Create forward and gradient operators  
op_fwd = forward(model, src, rec, order)  
op_grad = gradient(model, rec, order)  

# Run FWI with gradient descent  
for i in range(0, fwi_iterations):  
    # Compute functional value and gradient  
    # for the current model estimate  
    phi, direction = fwi_gradient(model.m)  

    # Artificial Step length for gradient descent  
    alpha = .005 / np.max(direction)  

    # Update the model estimate and enforce  
    # minimum/maximum values  
m_updated = model.m.data - alpha*direction  
model.m.data[:] = box_constraint(m_updated)