```
function [x, y_RK4] = ODE_Runge_Kutta_4(dydx, a, b, n, h, y_ini)
% ODE_Runge_Kutta_4 solves 1st order initial value ODE with Runge-Kutta
% Fourth Order's Method
% dydx - First Order Differential Equation
% a    - starting point of a range
% b    - ending point of a range
% h    - step size
% n    - number of intervals

% Initialize y vectors;
%y_RK4 = zeros(n, 1);

% Store all the x values in a vector form.
if a > b
    h = -h;
end
%x = a : h : b;

% Initial value of x
x(1) = a;

% Initial value of y
y_RK4(1) = y_ini;

% Apply Runge-Kutta 4th Order Method
for i = 1 : n

    x(i+1) = x(i) + h;

    K_1 = dydx(x(i), y_RK4(i));

    new_x = x(i) + h / 2;

    y_K1 = y_RK4(i) + K_1 / 2 * h;

    K_2 = dydx(new_x, y_K1);

    y_K2 = y_RK4(i) + K_2 / 2 * h;

    K_3 = dydx(new_x, y_K2);

    y_K3 = y_RK4(i) + K_3 * h;

    K_4 = dydx(x(i + 1), y_K3);

    y_RK4(i + 1) = y_RK4(i) + h / 6 * (K_1 + 2 * K_2 + 2 * K_3 + K_4);

end

end
```