```python
# HW 1 | Question 3 | Part C | Conor X Devlin

import math

root_3 = math.sqrt(3.0)

def loss_function(w):
    # L(w)
    L = (1.0/3.0)*(w**3) - w
    return L

def gradient_loss_function(w):
    # ∇L(wt)
    g_L = w**2 - 1.0
    return g_L

def product_operator(w_t, low_bound=-root_3, high_bound=root_3):
    # Π(x) = { bounds check }
    prod_op = min(high_bound, max(low_bound, w_t))
    return prod_op

def proj_gd_step(w, eta):
    # Π[ wt − η∇L(wt) ]
    w_t_1 = w - (eta * gradient_loss_function(w))
    prod_bounds_check = product_operator(w_t_1)
    return prod_bounds_check

def run_proj_gd(w_init, eta=(1.0/(2.0*root_3)),eps=1e-10, eps_grad=1e-10,
max_iter=100000):
    ws = [w_init]
    Ls = [loss_function(w_init)]
    iters = 0
    while iters < max_iter:
        # W_t_1 next w
        w_next = proj_gd_step(ws[-1], eta)
        ws.append(w_next)
        Ls.append(loss_function(w_next))
        iters += 1

        # Difference between most recent iterates
        dw = abs(ws[-1] - ws[-2])

        # Magnitude of the gradient
        g = abs(gradient_loss_function(ws[-1]))

        # Bounds check, is ws[-1] within the interval
        x = (-root_3 < ws[-1] < root_3)

        # Check for iterate, gradient magnitude deltas per run to say within defined eps
        bounds
        if dw <= eps and (not x or g <= eps_grad):
            break

    wT = ws[-1]
    out = {
        "w0": w_init,
        "eta": eta,
        "iterations": iters,
        "w_final": wT,
        "L_final": loss_function(wT),
        "grad_final": gradient_loss_function(wT),
        "hit_boundary": (abs(wT) >= root_3 - 1e-15),
        "converged": (iters < max_iter)
    }

    print(f"\n=== PGD run (w0={w_init:+.6f}, eta={eta:.6f}) ===")
    print(f"Converged: {out['converged']}  in {out['iterations']} iters")
    print(f"w_T = {out['w_final']:+.12f}")
    print(f"L(w_T) = {out['L_final']:+.12f}")
```

```
    print(f"|gradL(w_T)| = {abs(out['grad_final']):.6e}")
    print(f"Hit boundary? {out['hit_boundary']}")
    return out, ws, Ls

def main():
    eta = float(1.0 / (2.0*root_3))
    print(f"\n### PGD RUN ETA: {eta}###")
    for w_init in [-0.9, -1.1, -1.0]:
        run_proj_gd(w_init, eta=eta, eps=1e-10, eps_grad=1e-10, max_iter=100000)

main()
```