```
lasso.fit(X_train, y_train)
y_pred_lasso = lasso.predict(X_test)
r2_lasso = r2_score(y_test, y_pred_lasso)
results.append(['Lasso (L1)', r2_lasso])

# 4. Polynomial Regression (Degree 2 & 3)
poly2 = make_pipeline(PolynomialFeatures(degree=2), LinearRegression())
poly2.fit(X_train, y_train)
y_pred_p2 = poly2.predict(X_test)
r2_p2 = r2_score(y_test, y_pred_p2)
results.append(['Polynomial (deg=2)', r2_p2])

poly3 = make_pipeline(PolynomialFeatures(degree=3), LinearRegression())
poly3.fit(X_train, y_train)
y_pred_p3 = poly3.predict(X_test)
r2_p3 = r2_score(y_test, y_pred_p3)
results.append(['Polynomial (deg=3)', r2_p3])

# Print Results
print("="*65)
print("REGRESSION COMPARISON ON IRIS (petal_length → petal_width)")
print("="*65)
for name, score in results:
    print(f"{name:25} → R² = {score:.5f}")
print("="*65)

# Plot all models
plt.figure(figsize=(12,8))
plt.scatter(X, y, color='blue', alpha=0.6, label='Actual Data')

# Sort for smooth lines
X_plot = np.linspace(X.min(), X.max(), 100).reshape(-1,1)

plt.plot(X_plot, lr.predict(X_plot), 'g-', linewidth=3, label=f'Linear
    ↪ (R²={r2_lr:.4f})')
plt.plot(X_plot, ridge.predict(X_plot), 'r--', linewidth=2, label=f'Ridge
    ↪ (R²={r2_ridge:.4f})')
plt.plot(X_plot, lasso.predict(X_plot), 'm:', linewidth=2, label=f'Lasso
    ↪ (R²={r2_lasso:.4f})')
plt.plot(X_plot, poly2.predict(X_plot), 'c-', linewidth=2, label=f'Poly
    ↪ deg2 (R²={r2_p2:.4f})')
plt.plot(X_plot, poly3.predict(X_plot), 'orange', linewidth=2, label=f'Poly
    ↪ deg3 (R²={r2_p3:.4f})')

plt.xlabel('Petal Length (cm)')
plt.ylabel('Petal Width (cm)')
plt.title('Regularized + Non-Linear Regression Comparison')
plt.legend()
```