Lab 3

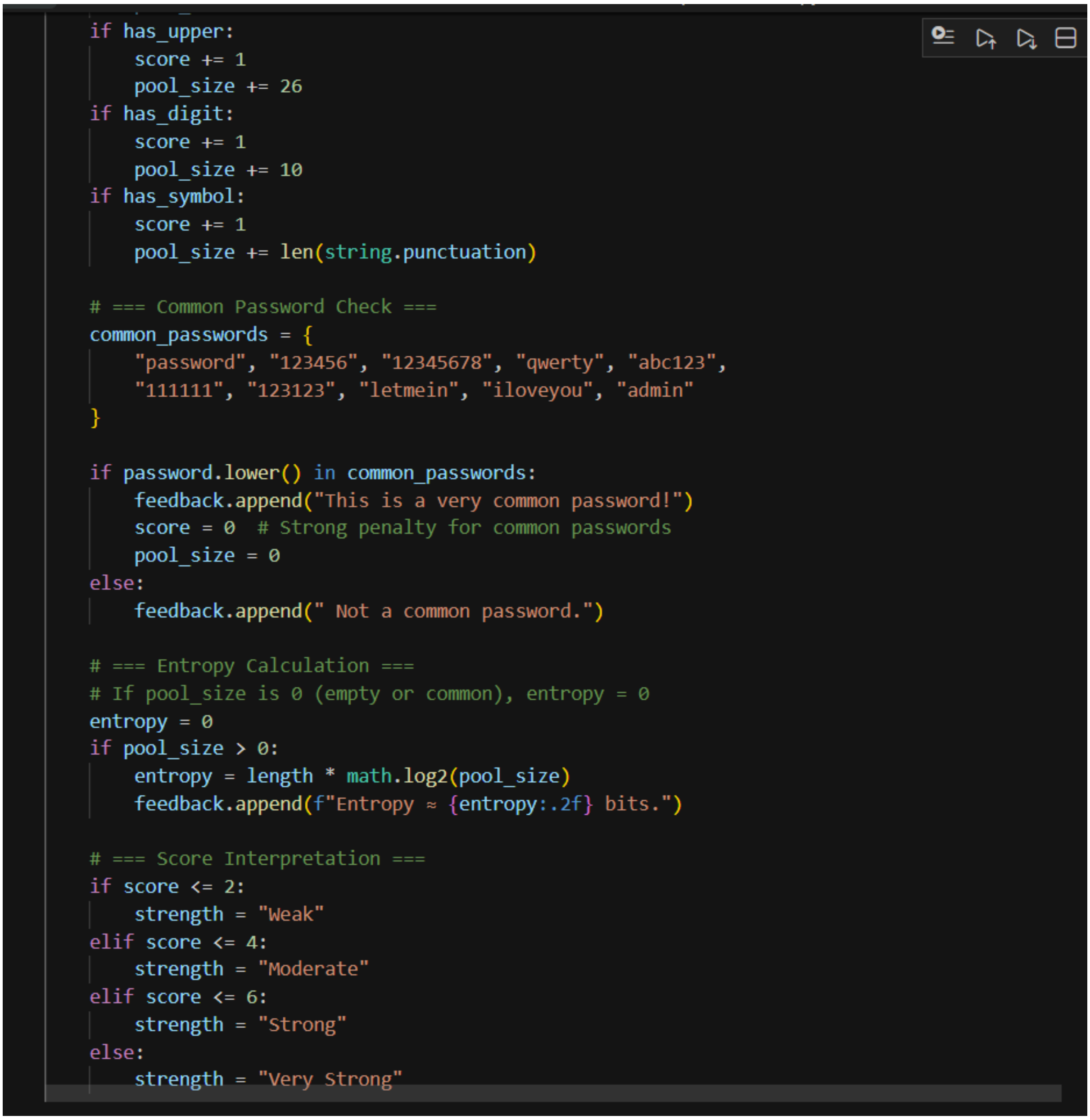
This lab focuses on **how modern authentication systems protect user accounts**. Instead of relying on simple passwords, the lab demonstrates how to build a **secure, multi-layered authentication system** using:

- Password strength analysis
- Secure password hashing (bcrypt)
- Salt and pepper techniques
- Two-Factor Authentication (2FA) using TOTP
- Brute-force attack simulation
- A completely secure user registration and login system

The aim is to show **why weak authentication is dangerous** and how **real-world systems defend against attacks**.

Lab 3

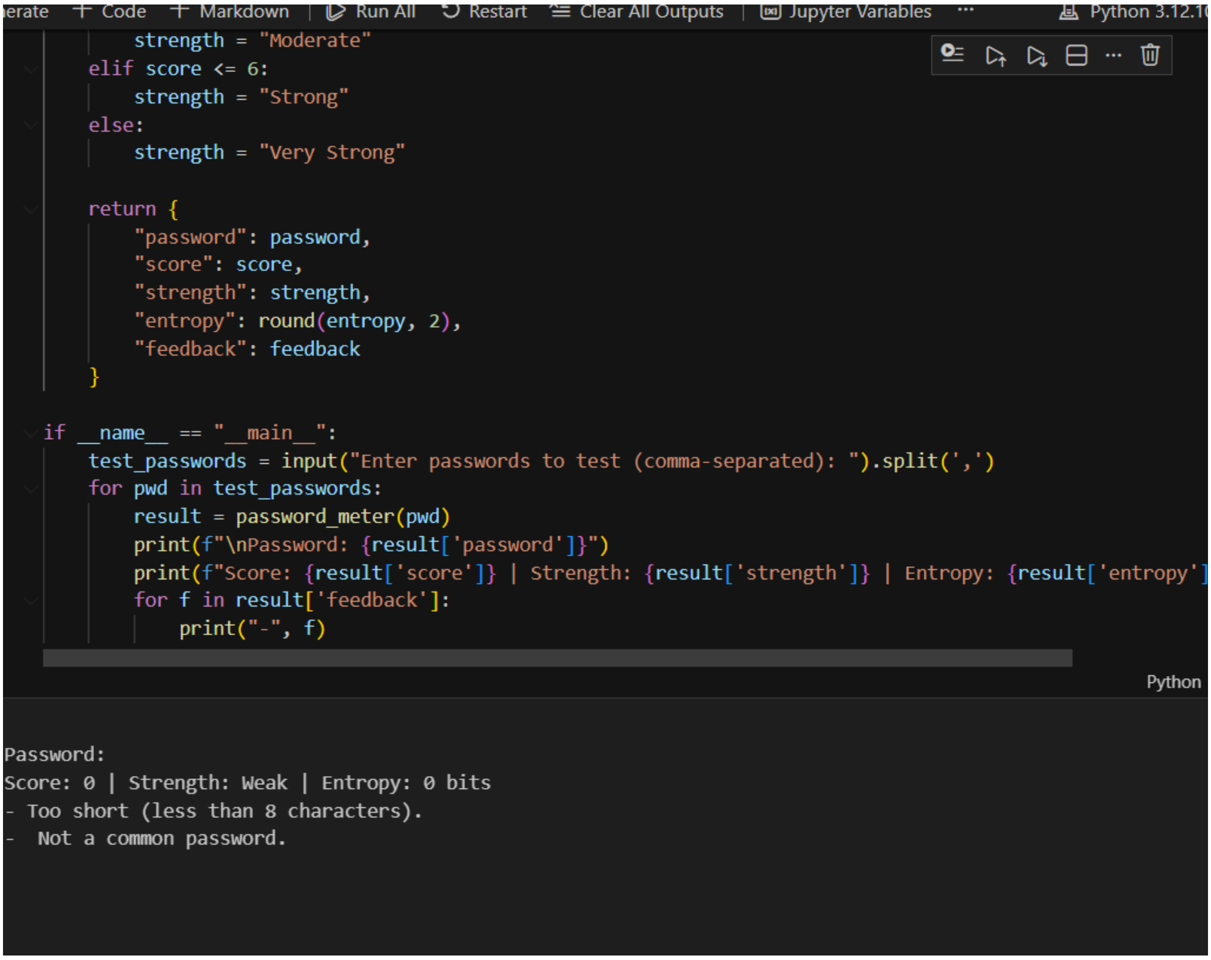
Initial code for the password meter - checking the length and character variety



```
    if has_upper:
        score += 1
        pool_size += 26
    if has_digit:
        score += 1
        pool_size += 10
    if has_symbol:
        score += 1
        pool_size += len(string.punctuation)

    # === Common Password Check ===
    common_passwords = {
        "password", "123456", "12345678", "qwerty", "abc123",
        "111111", "123123", "letmein", "iloveyou", "admin"
    }

    if password.lower() in common_passwords:
        feedback.append("This is a very common password!")
        score = 0  # Strong penalty for common passwords
        pool_size = 0
    else:
        feedback.append(" Not a common password.")

    # === Entropy Calculation ===
    # If pool_size is 0 (empty or common), entropy = 0
    entropy = 0
    if pool_size > 0:
        entropy = length * math.log2(pool_size)
        feedback.append(f"Entropy ≈ {entropy:.2f} bits.")

    # === Score Interpretation ===
    if score <= 2:
        strength = "Weak"
    elif score <= 4:
        strength = "Moderate"
    elif score <= 6:
        strength = "Strong"
    else:
        strength = "Very Strong"
```

Common Password Checks, penalty for common password, and calculation of entropy and if-else statements on password score

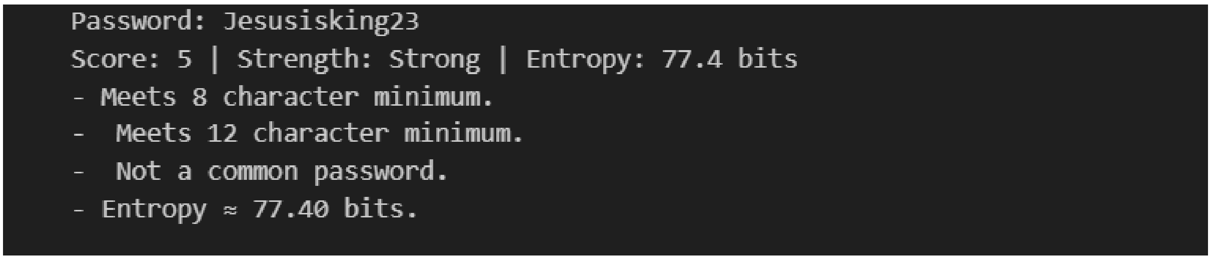
Lab 3



```python
        strength = "Moderate"
    elif score <= 6:
        strength = "Strong"
    else:
        strength = "Very Strong"

    return {
        "password": password,
        "score": score,
        "strength": strength,
        "entropy": round(entropy, 2),
        "feedback": feedback
    }

if __name__ == "__main__":
    test_passwords = input("Enter passwords to test (comma-separated): ").split(',')
    for pwd in test_passwords:
        result = password_meter(pwd)
        print(f"\nPassword: {result['password']}")
        print(f"Score: {result['score']} | Strength: {result['strength']} | Entropy: {result['entropy']
        for f in result['feedback']:
            print("-", f)
```

```
Password:
Score: 0 | Strength: Weak | Entropy: 0 bits
- Too short (less than 8 characters).
-  Not a common password.
```

If else stattements for output of the password score



```
Password: Jesusisking23
Score: 5 | Strength: Strong | Entropy: 77.4 bits
- Meets 8 character minimum.
-  Meets 12 character minimum.
-  Not a common password.
- Entropy ≈ 77.40 bits.
```

Trialled this as a password. It meets the 8-character minimum and 12-character minimum, so it is deemed a strong password. Is this really a strong password? I guess it depends on the context.