Track I picked and why it matters:

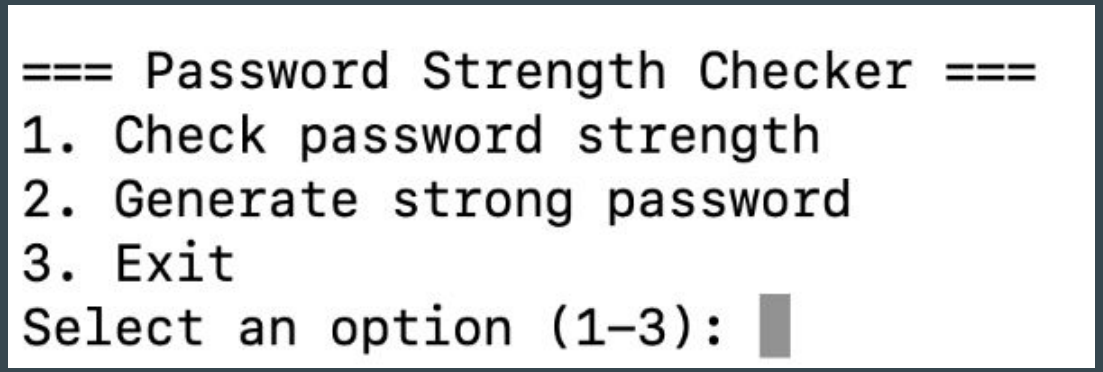
While I chose **cybersecurity** because I was generally curious, I've become increasingly interested in understanding how systems are protected behind the scenes. Along with the fact that there are more and more career opportunities becoming available as technology advances.

Career Examples and the Companies Hiring Them:

- **Cybersecurity Analyst**
  Commonly found at: *Microsoft, Sony, Bank of America*
- **Penetration Tester**
  Employed by: *Tesla, IBM, CrowdStrike*
- **SOC Analyst (Security Operations Center Analyst)**
  Seen at: *Cisco, Google, U.S. Department of Defense*
- **Threat Intelligence Analyst**
  In demand at: *Mandiant, Meta, Palantir*

Python Password Strength Checker

(Python Terminal-Based App)



Why this project?

- Strong passwords are a crucial part of cybersecurity.
- This tool helps users evaluate their password security and generate strong passwords.
- Briefly informs users about what makes a password strong.

Features Overview:

- There's an interactive menu: check a password, generate one, or exit.
- If a password is weak, the app gives improvement suggestions.
- It uses a custom scoring system based on length and character variety.

```
=== Password Strength Checker ===
1. Check password strength
2. Generate strong password
3. Exit
Select an option (1-3): 2
```

```
Select an option (1-3): 1
Enter your password: fycug

Strength Score: 1 / 6
Feedback: ❌ Weak Password

Suggestions to improve:
- Password is too short. Use at least 8 characters.
- Add uppercase letters.
- Include numbers.
- Include special characters (e.g. !@#$%).
```

```
Strength Score: 3 / 6
Feedback: ⚠️Moderate Password

Suggestions to improve:
- Add lowercase letters.
- Include special characters (e.g. !@#$%).
```

```
Enter your password: AJwnvvownv234567!@#$

Strength Score: 6 / 6
```

```
Enter your password: Aj12345!@#

Strength Score: 5 / 6
Feedback: ✅ Strong Password
```

Programming Requirements:

- ***Code Documentation***
  Inline comments throughout explain each function's purpose and logic

- **Descriptive Variable Names:**
  Used clear, meaningful names like "user_password, strength_score, feedback_messages".

- **Three Data Types:**
  Strings hold the password text, **Integers** track the password's strength numerically, and **Booleans** confirm if the password includes required character types.

- **Decision-Making Structures:**
  Used "if", "elif", and "else" to score strength and guide user interaction.

```python
# Password Strength Checker CLI Tool
# Author: AJ
# Purpose: Evaluates password strength and gives improvement suggestions using custom logic

import string
import random

def evaluate_password_strength(user_password):
    """
    Evaluates the strength of a given password based on length, character types, and uniqueness.
    Returns a tuple: (strength_score, feedback_messages)
    """
    strength_score = 0
    feedback_messages = []

    # Booleans to track character variety
    has_lowercase = any(char.islower() for char in user_password)
    has_uppercase = any(char.isupper() for char in user_password)
    has_digit = any(char.isdigit() for char in user_password)
    has_symbol = any(char in string.punctuation for char in user_password)

    # Evaluate length
    if len(user_password) >= 12:
        strength_score += 2
    elif len(user_password) >= 8:
        strength_score += 1
    else:
        feedback_messages.append("Password is too short. Use at least 8 characters.")

    if has_lowercase:
        strength_score += 1
    else:
        feedback_messages.append("Add lowercase letters.")
```

Programming Requirements cont.:

- **Loops for Repeated Tasks:**
  "while True" loop enables continuous user access without restarting the program

- **Reusable Functions:**
  Defined"evaluate_password_strength()" and "generate_strong_password()" for clean, modular logic

- **List (Sequence) Use:**
  Stored improvement tips in "feedback_messages" and iterated through them with a "for" loop

- ***Code Documentation***
  Inline comments throughout explain each function's purpose and logic

```
    Generates a secure password using lowercase, uppercase, digits, and symbols.
    """
    character_pool = string.ascii_letters + string.digits + string.punctuation
    generated_password = ''.join(random.choice(character_pool) for _ in range(length))
    return generated_password

# Main application loop
while True:
    print("\n=== Password Strength Checker ===")
    print("1. Check password strength")
    print("2. Generate strong password")
    print("3. Exit")
```

```
def evaluate_password_strength(user_password):
    """
    Evaluates the strength of a given password based on length, character types, and uniqueness.
    Returns a tuple: (strength_score, feedback_messages).
    """
    strength_score = 0
```

```
def generate_strong_password(length=16):
    """
    Generates a secure password using lowercase, uppercase, digits, and symbols.
    """
```

```
# Evaluate length
if len(user_password) >= 12:
    strength_score += 2
elif len(user_password) >= 8:
    strength_score += 1
else:
    feedback_messages.append("Password is too short. Use at least 8 characters.")

if has_lowercase:
    strength_score += 1
else:
    feedback_messages.append("Add lowercase letters.")

if has_uppercase:
    strength_score += 1
else:
    feedback_messages.append("Add uppercase letters.")

if has_digit:
    strength_score += 1
else:
    feedback_messages.append("Include numbers.")

if has_symbol:
    strength_score += 1
else:
    feedback_messages.append("Include special characters (e.g. !@#$%).")

return strength_score, feedback_messages
```

Insert demonstration here

python3 password_strength_checker.py

Whats next for me/Formal actionable steps:

- Complete 6-month program with focus on practical application and real-world simulations
- Build a professional cybersecurity portfolio (GitHub repos, case studies)
- Pursue certifications like **CompTIA Security+** (often listed as a requirement for entry-level roles — it shows I understand network security, compliance, and threats.)
- Begin applying for internships or junior analyst roles by **early 2026**