Project Report

1. Introduction

Project Title: Random Password Generator
Language: Python 3
Purpose: To create a command-line tool that generates cryptographically random passwords with enforced complexity requirements.

2. Methodology

2.1 Algorithm Design

The password generation follows these steps:

1. Import character sets (lowercase, uppercase, digits, punctuation)
2. Accept user input for password length
3. Validate input (numeric, minimum 6 characters)
4. Calculate character distribution:
    - part1 = 30% of length (for uppercase and lowercase)
    - part2 = 20% of length (for digits and punctuation)
5. Shuffle all character pools randomly
6. Select characters from each pool according to calculated portions
7. Shuffle the final character combination
8. Output the generated password

2.2 Input Validation

- **Type checking**: Catches non-numeric input with try-except block
- **Length enforcement**: Loops until user provides value ≥ 6
- **User-friendly prompts**: Clear error messages guide users to correct input

3. Code Structure Analysis

Character Distribution Formula:

- part1 = round(characters_number * 0.3) → 30% allocation
- part2 = round(characters_number * 0.2) → 20% allocation

For a 12-character password:

- part1 = 4 (4 lowercase + 4 uppercase = 8 characters)
- part2 = 2 (2 digits + 2 special chars = 4 characters)

- Total = 12 characters

4. Strengths

 Simple and easy to understand
 No external dependencies
 Enforces password complexity automatically
 Good randomization with shuffle operations
 Robust error handling

5. Areas for Improvement

- Could offer customizable character distribution ratios
- Password strength indicator could be added
- Option to exclude ambiguous characters (0/O, 1/l)
- Copy-to-clipboard functionality
- Save passwords to encrypted file option

6. Testing Results

The program successfully:

- Rejects passwords shorter than 6 characters
- Handles non-numeric input gracefully
- Generates passwords with proper character distribution
- Produces unique passwords on each run

7. Conclusion

This password generator effectively creates secure, random passwords suitable for general use. The implementation demonstrates good practices in input validation and random number generation. It serves as an excellent foundation for more advanced password management tools.