Strong Passwords

Cameron Waddy

Old Dominion University

CYSE-250 Cybersecurity Programming and Networking

Dr. Rashid Ali Khan

December 6, 2021

An area where many people create extremely vulnerable confidential information is from weak passwords. Most people use a person, pet, year, or place that is important to them. Creating password without a wide variety of characteristics also means that you have a weaker password. Another area where a user can be susceptible to an attack is by using repetitive passwords across many domains or devices. To help a user decide if a password is strong enough to be used, we created a program. This program will check the user's input for several things. We will refer to these as flags. These flags are lowercase letters, uppercase letters, special characters, numbers, and the length of the password. Simultaneously, the program will be checking these flags and assigning a score. This assigned score will inform the user of the strength of the password. If the password has a score of 0, the base score, then the password can be easily cracked by a hacker. On the opposite end, if the user's inputted password gains a score of 35, which is the highest score, then the password cannot be easily cracked and is a strong and reliable password to use. The total score of 35 means that all the criteria is met that was previously explained above. For this program, the MacBook Pro was used on the software version macOS Monterery Version 12.0.1. The platform used to compose the program was IntelliJ IDEA 2021.2.3 (Community Edition).

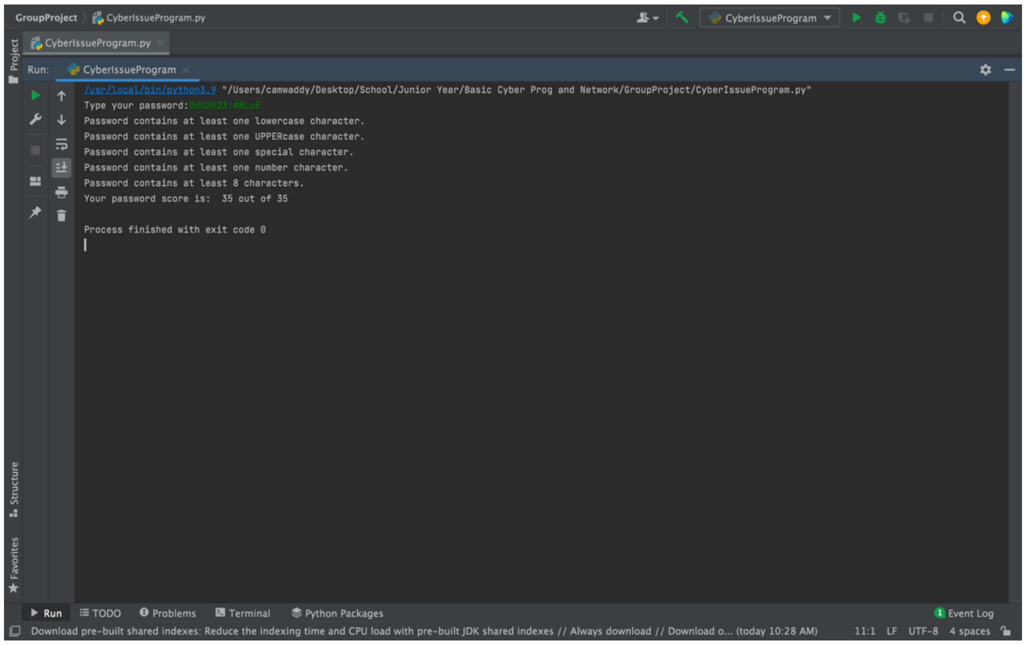
In this program, we assigned the values of 'False' to the several character types. This was so that if a character type was found within the user's password, it would be changed to 'True'. If the value of the character types were changed to 'True', a line would be printed out saying that there was that character in the password. If the character type was not found within the password, the value of the character would continue to be set to false and inform the user by printing that the character was not found and needs to be tried again. This was done by using if/else statements.

At the beginning of the program, we assigned the score value to 0. The value of lowercase, uppercase, special, and number being changed to 'True' adds points to the final score of the password. Separately from the 'True' and False' values, the program checks the length of the password. The length of the password is checked twice in this program. The first time the length is checked is in the fourth block of code. This first check informs the user if the password reaches the minimum of 8 characters, regardless of the characters used. The second check contributes to the score. If the length is reached or exceeds 8 characters, then 10 points are given. The score break down for the other characteristics are 10 points for both uppercase and lowercase being found, if a number is found with uppercase and lowercase then another 10 points will be assigned, if a special character is found then 5 points will be added. Any score under 35 in this program will be considered a weak password.

Having a variety of characters is the key to a strong and successful password. Using this program will help keep users using strong passwords and to be creative in their password creation. I have provided four separate outputs of this code to show the accuracy of the program and the different scores that are produced when foregoing on certain characteristics. In "Output.1" all of the criteria was met using a complex password. The password had lowercase characters, uppercase characters, special characters, numbers, and exceeded eight characters. This gave the password a score of 35 out of 35. In the next execution of code, "Output.2", I used a simple password with all lowercase characters and two numbers. This gave a password score of 10 out of 35. The password can be easily cracked with any hacking tool. There were no uppercase characters, special characters, and it was less than eight characters long. In "Output.3", I included all the types of characters excluding the minimum length needed. This gave a password score of 25 out of 35. In the final execution of code, "Output.4", the inputted password used all uppercase letters with a single number. This had the code notify the user of the missing character types as well as the inadequate length of the password. The final password score of this code is 10 out of 35.

Passwords can be seen as the first line of defense against intruders. Having strong passwords will protect your confidential information. However, using weak passwords will expose you to identity theft among many other attacks or thefts. This program will help younger people understand how to create strong passwords. It is also beneficial for older people who are not well versed in technology and cybersecurity. Although this guides users in the direction for better passwords, it must be reinforced to use different passwords for different sites. The reason to use separate passwords is in the scenario where one password is cracked by an intruder, they will not be able to access other accounts by using the same password. The difficulty of this project in my opinion is medium. The program was produced from using multiple if/else statements, a for loop, and more if statements. I attempted to include a while loop which I struggled with working code when attempting to do so. I wanted the code to re-execute automatically after asking if the user would like to try again, if the user answered yes, then the code would execute again. If the user entered no, then it would exit the program. However, I already had a working program that just needed to be re-run at the end of each single entry. If I was able to include this extra feature then I would say it would have been a harder difficulty.

Output.1

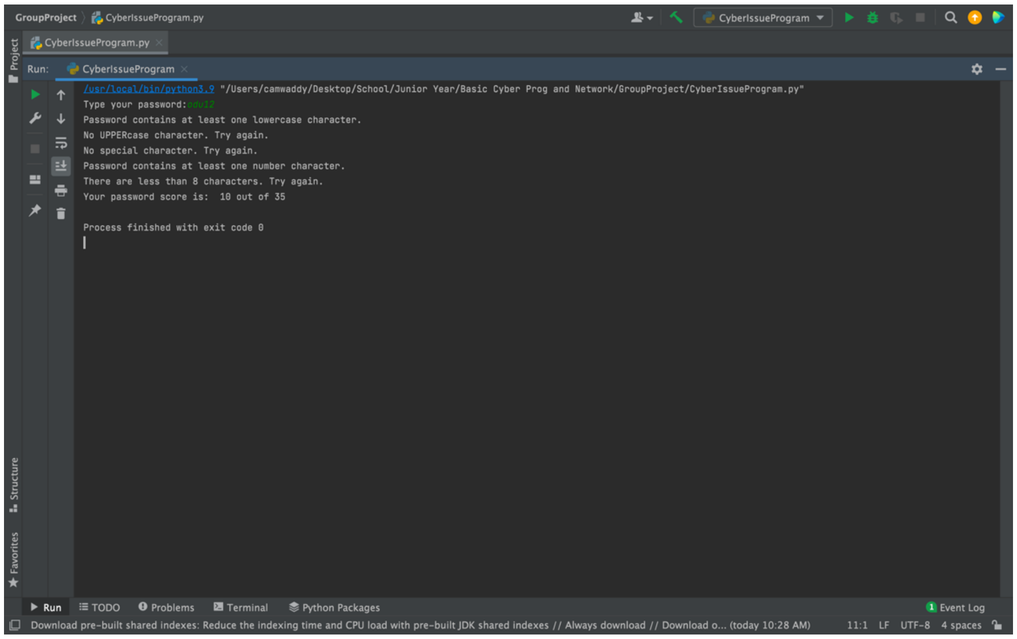


```
/usr/local/bin/python3.9 "/Users/camwaddy/Desktop/School/Junior Year/Basic Cyber Prog and Network/GroupProject/CyberIssueProgram.py"
Type your password:0dU2023!#8LuE
Password contains at least one lowercase character.
Password contains at least one UPPERcase character.
Password contains at least one special character.
Password contains at least one number character.
Password contains at least 8 characters.
Your password score is:  35 out of 35

Process finished with exit code 0
```

GroupProject
CyberIssueProgram.py
CyberIssueProgram
Run:
Project
Structure
Favorites
Run
TODO
Problems
Terminal
Python Packages
Event Log
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK shared indexes // Always download // Download o... (today 10:28 AM)
11:1 LF UTF-8 4 spaces

Output.2

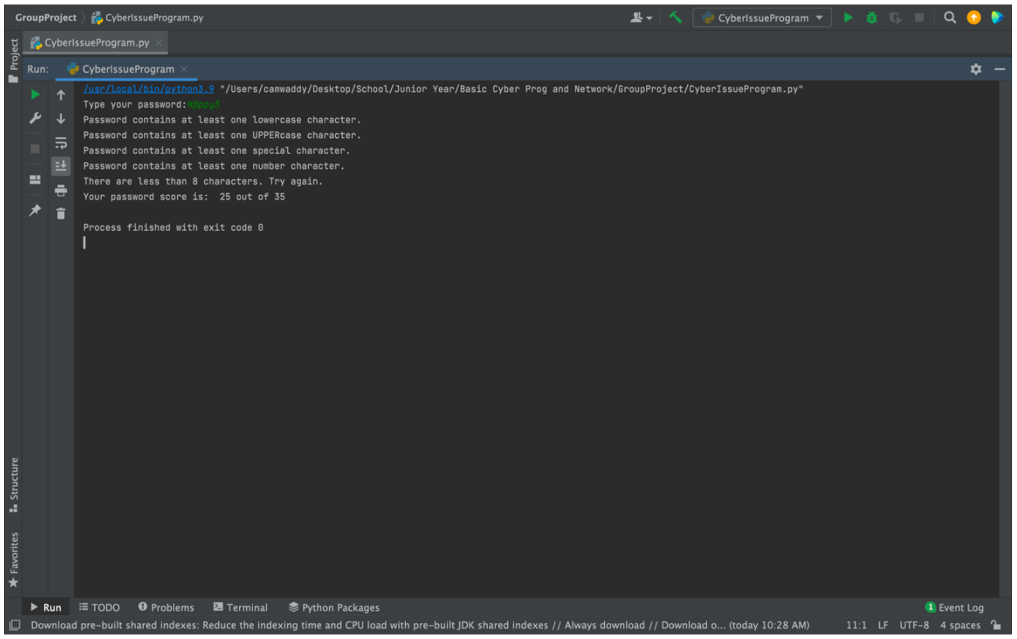


GroupProject
CyberIssueProgram.py
CyberIssueProgram
Run:
Project
Structure
Favorites
Run
TODO
Problems
Terminal
Python Packages
Event Log
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK shared indexes // Always download // Download o... (today 10:28 AM)
11:1 LF UTF-8 4 spaces

```
/usr/local/bin/python3.9 "/Users/camwaddy/Desktop/School/Junior Year/Basic Cyber Prog and Network/GroupProject/CyberIssueProgram.py"
Type your password:odu12
Password contains at least one lowercase character.
No UPPERcase character. Try again.
No special character. Try again.
Password contains at least one number character.
There are less than 8 characters. Try again.
Your password score is:  10 out of 35

Process finished with exit code 0
```

Output.3

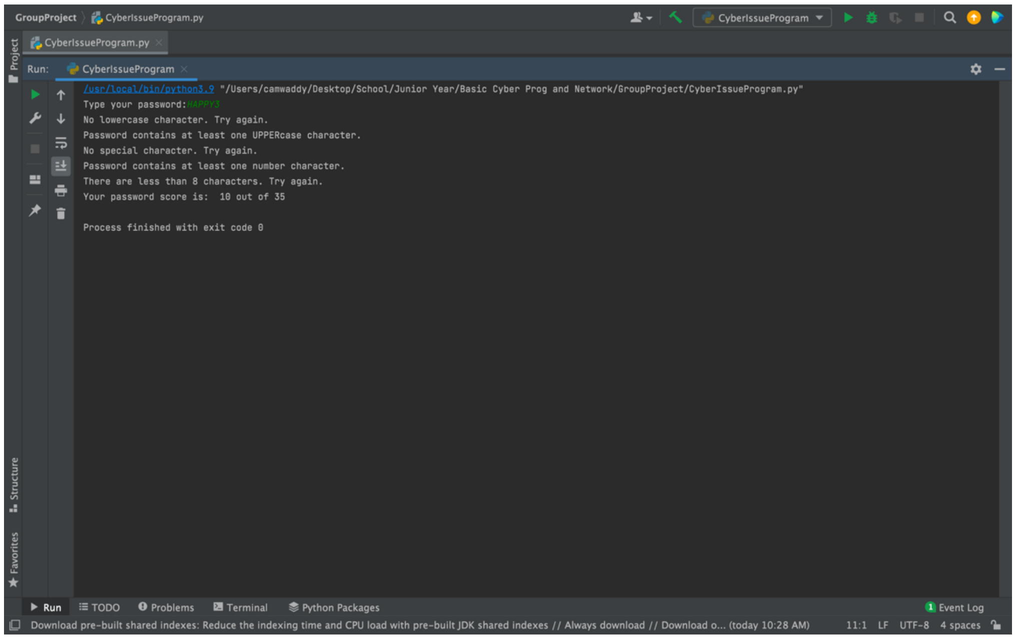


GroupProject
CyberIssueProgram.py
CyberIssueProgram
Run:
Project
Structure
Favorites
Run
TODO
Problems
Terminal
Python Packages
Event Log
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK shared indexes // Always download // Download o... (today 10:28 AM)
11:1 LF UTF-8 4 spaces

```
/usr/local/bin/python3.9 "/Users/camwaddy/Desktop/School/Junior Year/Basic Cyber Prog and Network/GroupProject/CyberIssueProgram.py"
Type your password:H@ppy3
Password contains at least one lowercase character.
Password contains at least one UPPERcase character.
Password contains at least one special character.
Password contains at least one number character.
There are less than 8 characters. Try again.
Your password score is:  25 out of 35

Process finished with exit code 0
```

Output.4



```
/usr/local/bin/python3.9 "/Users/camwaddy/Desktop/School/Junior Year/Basic Cyber Prog and Network/GroupProject/CyberIssueProgram.py"
Type your password:HAPPY3
No lowercase character. Try again.
Password contains at least one UPPERcase character.
No special character. Try again.
Password contains at least one number character.
There are less than 8 characters. Try again.
Your password score is:  10 out of 35

Process finished with exit code 0
```

GroupProject
CyberIssueProgram.py
CyberIssueProgram
Run:
Run
TODO
Problems
Terminal
Python Packages
Event Log
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK shared indexes // Always download // Download o... (today 10:28 AM)
11:1 LF UTF-8 4 spaces

Appendix

```
# This first set of code asks for the users potential password.
# The user will input their password and assign it to password.
# score is used to rate the strength of the password. 0 is the base value
with no strength.
password = input("Type your password:")
score = 0

# This set of code sets the values of the characters to False
lowercase = False
uppercase = False
number = False
special = False

# This set of code checks if the user inputted password has the different
types of characters.
# If the password includes the character, the value will be set to True.
for character in password:
    if character in "abcdefghijklmnopqrstuvwxyz":
        lowercase = True
    elif character in "ABCDEFGHIJKLMNOPQRSTUVWXYZ":
        uppercase = True
    elif character in "! @ # $ % ^ & * ( ) _ - : ; ' , . < > / ? \ | + = ":
        special = True
    elif character in "1234567890":
        number = True


# This section checks if the character types match the user given password
# If the character type is in the user given password, it will print that it
has the character.
# If the user given password does not have the character type, the code will
print that the user must try again.
if lowercase == True:
    print("Password contains at least one lowercase character.""")
else:
    print("No lowercase character. Try again.")
if uppercase == True:
    print("Password contains at least one UPPERcase character.")
else:
    print("No UPPERcase character. Try again.")
if special == True:
    print("Password contains at least one special character.")
else:
    print("No special character. Try again.")
if number == True:
    print("Password contains at least one number character.")
else:
    print("No number character. Try again.")
if len(password) >= 8:
    print("Password contains at least 8 characters.")
else:
    print("There are less than 8 characters. Try again.")

# If a value is true, points will be assigned to the password score (strength
of the password).
# The strongest password will have the value of 35. All scores will be based
out of 35.
if lowercase == True and uppercase == True:
    score = score + 10
if number == True and (lowercase==True or uppercase==True):
    score = score + 10
if special == True:
    score = score + 5
if len(password) >= 8:
    score = score + 10

# This line of code prints the password score out of 35.
print("Your password score is: ", score, "out of 35")
```