**Python Journal Template**

**Directions:** Follow the directions for each part of the journal template. Include in your response all the elements listed under the Requirements section. Prompts in the Inspiration section are not required; however, they may help you to fully think through your response.

Remember to review the Touchstone page for entry requirements, examples, and grading specifics.

**Name: Jason Victor**

**Date: 08/02/23**

**Final Replit Program Share Link:**

Complete the following template. Fill out all entries using complete sentences.

## PART 1: Defining Your Problem

|  |
| --- |
| **Task**  State the problem you are planning to solve.  **Requirements**   * Describe the problem you are trying to solve for. * Describe any input data you expect to use. * Describe what the program will do to solve the problem. * Describe any outputs or results the program will provide.   **Inspiration**  When writing your entry below ask yourself the following questions:   * Why do you want to solve this particular problem? * What source(s) of data do you believe you will need? Will the user need to supply that data, or will you get it from an external file or another source? * Will you need to interact with the user throughout the program? Will users continually need to enter data in and see something to continue? * What are your expected results or what will be the end product? What will you need to tell a user of your program when it is complete? |
| To protect online accounts from potential hacking attempts, I will create a program that generates stronger passwords. A key requirement of the program is the ability to customize password generation based on user input. In particular, I seek to determine how many uppercase and lowercase letters, special characters, and numbers the user prefers in their passwords.    By using a randomized approach, the program creates unique and robust passwords while adhering to user-specified criteria. Passwords must meet security standards, including a minimum length and a variety of characters.    The output of the program is a randomly generated strong password, which users can use to fortify the security of their accounts.    This endeavor is motivated by the desire to provide users with a simple yet powerful tool to enhance their online security. A strong password is essential for protecting data and personal information in today's digital age, where cyber threats are prevalent.    The primary data source for the program is user inputs collected interactively during execution. This allows the program to satisfy each user's requirements.    In conclusion, the password strengthener program equips users with the means to fortify their accounts. This makes them less susceptible to cyber-attacks and ensures greater peace of mind in their online interactions. |

## PART 2: Working Through Specific Examples

|  |
| --- |
| **Task**  Write down clear and specific steps to solve a simple version of your problem you identified in Part 1.  **Requirements**  Complete the three steps below **for at least two distinct examples/scenarios**.   * State any necessary input data for your simplified problem. * Write clear and specific steps in English (not Python) detailing what the program will do to solve the problem. * Describe the specific result of your example/scenario.   **Inspiration**  When writing your entry below ask yourself the following questions:   * Are there any steps that you don’t fully understand? These are places to spend more time working out the details. Consider adding additional smaller steps in these spots. * Remember that a computer program is very literal. Are there any steps that are unclear? Try giving the steps of your example/scenario to a friend or family member to read through and ask you questions about parts they don’t understand. Rewrite these parts as clearly as you can. * Are there interesting edge cases for your program? Try to start one of your examples/scenarios with input that matches this edge case. How does it change how your program might work? |
| **Scenario 1: Password Strengthener with Fixed Length - 8 characters**   1. Initialize an empty string to store the generated password. 2. Generate a random uppercase letter and add it to the password string. 3. Generate a random lowercase letter and add it to the password string. 4. Generate a random special character (!, @, #, $, etc.) and add it to the password string. 5. Generate a random number (0-9) and add it to the password string. 6. Fill the remaining 4 characters with randomly chosen characters (uppercase, lowercase, special, or numbers). 7. Mix the password string to make sure there is a randomized order. 8. The generated password is now ready! 9. An example of a possible generated password: "Aa@5bcD3"   **Scenario 2: Password Strengthener with Customizable Length - 12 characters**   1. Prompt the user to enter their desired password length of 12 characters. 2. Initialize an empty string to store the generated password. 3. Generate a random uppercase letter and add it to the password string. 4. Generate a random lowercase letter and add it to the password string. 5. Generate a random special character (!, @, #, $, etc.) and add it to the password string. 6. Generate a random number (0-9) and add it to the password string. 7. Calculate the number of characters remaining to reach the desired password length. 8. Use a loop to fill the remaining characters with randomly chosen characters (uppercase, lowercase, special, or numbers). 9. Mix the password string to ensure a randomized order. 10. The generated password is now ready! 11. An example of a possible generated password with a desired length of 12 characters: "1B@cD4#r7T9z" |

## PART 3: Generalizing Into Pseudocode

|  |
| --- |
| **Task**  Write out the general sequence your program will use, including all specific examples/scenarios you provided in Part 2.  **Requirements**   * Write pseudocode for the program in English but refer to Python program elements where they are appropriate. The pseudocode should represent the full functionality of the program, not just a simplified version. Pseudocode is broken down enough that the details of the program are no longer in any paragraph form. One statement per line is ideal.   **Help with writing pseudocode**   * Here are a few links that can help you write pseudocode with examples. Remember to check out part 3 of the Example Journal Template Submission if you have not already. Note: everyone will write pseudocode differently. There is no right or wrong way to write it other than to make sure you write it clearly and in as much detail as you can so that it should be easy to convert it to code later.   + <https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/>   + <https://www.wikihow.com/Write-Pseudocode>   **Inspiration**  When writing your entry below ask yourself the following questions:   * Do you see common program elements and patterns in your specific examples/scenarios in Part 2, like variables, conditionals, functions, loops, and classes? These should be part of your pseudocode for the general sequence as well. * Are there places where the steps for your examples/scenarios in Part 2 diverged? These may be places where errors may occur later in the project. Make note of them. * When you are finished with your pseudocode, does it make sense, even to a person that does not know Python? Aim for the clearest description of the steps, as this will make it easier to convert into program code later. |
| # Function to generate a strong password based on user preferences  Function generate\_strong\_password():  Set length = 8 # Default password length  Set uppercase\_letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"  Set lowercase\_letters = "abcdefghijklmnopqrstuvwxyz"  Set special\_characters = "!@#$%^&\*()\_-+=<>?/:;{}[]|"  Set numbers = "0123456789"  Set password = ""    # Prompt the user for custom password length  Input desired\_length  If desired\_length > 8:  Set length = desired\_length    # Generate required characters  Set password = password + Randomly select one character from uppercase\_letters  Set password = password + Randomly select one character from lowercase\_letters  Set password = password + Randomly select one character from special\_characters  Set password = password + Randomly select one character from numbers    # Fill the remaining characters with random selections  Set remaining\_length = length - 4  Repeat remaining\_length times:  Set random\_character = Randomly select one character from uppercase\_letters, lowercase\_letters, special\_characters, or numbers  Set password = password + random\_character    # Shuffle the password characters to randomize the order  Shuffle characters in password    # Return the generated password  Return password    # Option 1 - Password Strengthener with Fixed Length (8 characters)  Set generated\_password = generate\_strong\_password()  Print "Generated Password: " + generated\_password    # Option 2 - Password Strengthener with Customizable Length (ex. 12 characters)  Print "Enter desired password length: "  Input desired\_length  Set generated\_password = generate\_strong\_password(desired\_length)  Print "Generated Password: " + generated\_password |

## PART 4: Testing Your Program

|  |
| --- |
| **Task**  While writing and testing your program code, describe your tests, record any errors, and state your approach to fixing the errors.  **Requirements**   * For at least one of your test cases, describe how your choices for the test helped you understand whether the program was running correctly or not.   For each error that occurs while writing and testing your code:   * Record the details of the error from Replit. A screenshot or copy-and-paste of the text into the journal entry is acceptable. * Describe what you attempted in order to fix the error. Clearly identify what approach was the one that worked.   **Inspiration**  When writing your entry below ask yourself the following questions:   * Have you tested edge cases and special cases for the inputs of your program code? Often these unexpected values can cause errors in the operation of your program. * Have you tested opportunities for user error? If a user is asked to provide an input, what happens when they give the wrong type of input, like a letter instead of a number, or vice versa? * Did the outcome look the way you expected? Was it formatted correctly? * Does your output align with the solution to the problem you coded for? |
| The first version of my password strengthener program that asked the user for a desired password length did not handle the input of letters correctly. Please see the screenshot below for this problem:    I added the following try and except block to handle the ValueErrorException.  # Option 2 - Password Strengthener with Customizable Length  while True:  try:  desired\_length\_2 = int(input("Enter desired password length (customized length): "))  if desired\_length\_2 < 8:  print("Password length must be at least 8 characters.")  else:  break  except ValueError:  print("Invalid input. Please enter an integer.")    generated\_password\_2 = generate\_strong\_password(desired\_length\_2)  print("Generated Password (customized length):", generated\_password\_2)  Also, it did not handle the input of zero or a negative number correctly. It would give 8 characters for both scenarios. Please see the screenshots below for this problem:    However, adding the try and except block resolved the issue. See below: |

## PART 5: Commenting Your Program

|  |
| --- |
| **Task**  Submit your full program code, including thorough comments describing what each portion of the program should do when working correctly.  **Requirements**   * The purpose of the program and each of its parts should be clear to a reader that does not know the Python programming language.   **Inspiration**  When writing your entry, you are encouraged to consider the following:   * Is each section or sub-section of your code commented to describe what the code is doing? * Give your code with comments to a friend or family member to review. Add additional comments to spots that confuse them to make it clearer. |
| # Import the random module to generate random values  import random    def generate\_strong\_password(desired\_length=8):  # Define sets of characters to be used for generating the password  uppercase\_letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"  lowercase\_letters = "abcdefghijklmnopqrstuvwxyz"  special\_characters = "!@#$%^&\*()\_-+=<>?/:;{}[]|"  numbers = "0123456789"  password = ""    # Determine the password length based on user input, defaulting to 8 characters if not specified  if desired\_length > 8:  length = desired\_length  else:  length = 8    # Ensure that the generated password contains at least one character from each category  password += random.choice(uppercase\_letters) # Add a random uppercase letter  password += random.choice(lowercase\_letters) # Add a random lowercase letter  password += random.choice(special\_characters) # Add a random special character  password += random.choice(numbers) # Add a random number    # Calculate the number of characters remaining to reach the desired password length  remaining\_length = length - 4    # Create a pool of characters containing all possible characters  character\_pool = uppercase\_letters + lowercase\_letters + special\_characters + numbers    # Generate the remaining characters for the password  for count in range(remaining\_length):  random\_character = random.choice(character\_pool)  password += random\_character    # Shuffle the characters in the password to randomize the order  password\_list = list(password)  random.shuffle(password\_list)  shuffled\_password = "".join(password\_list)    # Return the generated password  return shuffled\_password    # Option 1 - Password Strengthener with Fixed Length (8 characters)  generated\_password\_1 = generate\_strong\_password()  print("Generated Password (8 characters):", generated\_password\_1)    # Option 2 - Password Strengthener with Customizable Length  while True:  try:  desired\_length\_2 = int(input("Enter desired password length (customized length): "))  if desired\_length\_2 < 8:  print("Password length must be at least 8 characters.")  else:  break  except ValueError:  # Catch the ValueError exception if user enters a non-integer value for desired\_length\_2  print("Invalid input. Please enter an integer.")    # Generate the password with the user-specified length  generated\_password\_2 = generate\_strong\_password(desired\_length\_2)  print("Generated Password (customized length):", generated\_password\_2) |

## PART 6: Your Completed Program

|  |
| --- |
| **Task**  Provide the Replit link to your full program code.  **Requirements**   * The program must work correctly with all the comments included in the program.   **Inspiration**   * Check before submitting your touchstone that your final version of the program is running successfully. |
| https://replit.com/@JasonVictor3/Password-Strengthener# |