Developer Documentation for Secure Password/Passphrase Generator

Introduction 1
Overview 1
Functionality 2
Generating a password 2
Generating a passphrase 3
The Algorithm / Developer documentation 4
Functions 4
Variables 4
Input handling 5
Output handling 5
Future maintenance and development 5

Introduction

Welcome to the developer documentation for the secure Password/Passphrase Generator!

This documentation serves developers who are interested in understanding the code and workflow. The documentation will go through an overview of what the script does and how it works, and explanations of its workflow and automation needs.

Overview

The secure Password/Passphrase generator is a python script which generates secured randomly generated passwords and passphrases, how they are generated is based on the user preference and customization (Symbols, Numbers, etc.). Generated Password/Passphrase will optionally output the results.

Functionality

The secure password/passphrase generator works by first asking the user to generate either a password or a passphrase. This selection will loop until a given choice is inputted by writing the corresponding input. If an invalid response is given, the program will loop again to ask the user for a valid response. If either ‘password’ or ‘passphrase’ is given, the loop will break and the corresponding functions will be executed, this will begin the main automation process of the program. If ‘password’ was the input, it will execute the ‘generate_password()’ function else if ‘passphrase’ was the input it will execute the ‘generate_passphrase()’

Generating a password

If the user chooses to generate a password, they will be prompted to input and answer questions prompted by the script on how they would like their password to be generated. It starts off by initializing a variable called **characters**. It is currently blank, but it will be used to store the results for the next set of inputs. Next the script will ask for the **length** of the password, the length is an integer variable which will store a number, next the script will ask to use for **uppercase** letters or not, this variable is saved as either a ‘yes’ or a ‘no’ if ‘yes’ the ‘characters’ variable will add and assign uppercase letters to it, next the script will ask for **lowercase** letters, if ‘yes’ the ‘characters’ variable will add and assign lowercase letters to it, next the script will ask for **numbers**, if ‘yes’ the ‘characters’ variable will add and assign numbers from 0-9 to it, next the script will ask for **symbols**, if ‘yes’ the ‘characters’ variable will add and assign punctuation characters to it, next the script will ask for the **amount** which is how many passwords you want to generate, next the script will ask for **output**, if ‘yes’ the results that will be printed in the console will be saved into a text file. Finally, once all inputs have been added and assigned to the ‘characters’ variable, a for loop iterates how many times the **amount** variable equals to, inside that for loop a **password** variable is assigned with the secrets module join method to create randomly generated passwords based on the value in the amount variable and inside another for loop, iterates each character for how many numbers there are in the **length** variable, the length variable will decide how many characters to generate for the password. Next the script prints the passwords onto the console screen notifying the user they've been printed; the printed passwords will be the randomly generated passwords automatically created by the inputs. Next the script performs another check if **output** is 'yes' and if it is, it creates a text file called "passwords.txt" and prints the resulting generated passwords into the created text file

Generating a passphrase

If the user chooses to generate a passphrase, they will be prompted to input and answer questions by the script, the first question is asking the **number of words in the passphrase they would like,** and if variable is 0 print and continue the loop till input is greater than 0 and break out of loop. The script will then ask the user to choose the **amount of numbers between each word,** the user types in the response. Next, the code will ask if the user would like to **capitalize each word,** also while loops that loop until the input is yes or no and do something, if invalid it'll loop over to ask again - this variable is saved as either a **'yes' or a 'no'**. Then the code will ask if the user would like to have **random capitalization,** this variable is saved as either a **'yes' or a 'no',** if yes set random capitalization. After that, the console displays and asks the user the **amount of passphrases they would like to generate** and if variable is 0 print and continue the loop till input is greater than 0 and break out of loop. Lastly, the script asks the individual if the **results will output to a text file,** this variable is saved as either a **'yes' or a 'no'**. At the end, the **console will print out the desired passphrase configurations,** depending on whether the results were output to a text file, you can find the **passphrases.txt** with all the passphrases in that file – and it will loop how many times a generated passphrase is displayed based on the amount variable. Also, the code will print on screen to notify the user their passphrases have been output to a text file.

The Algorithm / Developer documentation

Functions

generate_password() - Function responsible for generating a password

- Uses module **secret** for securely selecting characters
- Uses module **string** for character sets (ascii_uppercase, ascii_lowercase, digits, punctuation)
- Outputs results to console and optionally a text file

generate_passphrase() - Function responsible for generating a passphrase

- Uses module **random** to randomly choose words from the wordlist and amount of numbers
- Outputs results to console and optionally a text file

Variables

length – Used for inputting an integer length

uppercase – Used for checking if all capitalization is true or false

lowercase – Used for checking if lowercase is true or false

number – Used for inputting an integer value

symbol – Used for checking if symbol is true or false

amount – Used for inputting an integer value

capwords – Used to check if capitalization is true or false

randcap – Used to check if random capitalization is true or false

output – Used to check if output is true or false

password – Used to assign the characters variable

characters – Used to add and assign character sets

wordlist – Used to assign a predefined list of words

passphrase – Used to assign the characters

words – Used to check the number of words

word – Used to choose a random word from the word list

PASS – Used to check if the selection is a password or passphrase

Input handling

If a response is yes or no, the code gives the answer. Whereas if there is invalid syntax or something different was written (for example a number when you should put words or yes/no), the code then gives an error, and you must change your answer and the response will display/print again.

- Response is handled by a yes or a no, to ensure only validated responses are given.
- The script catches and informs the user of any invalid inputs or zero length selections
- Input is validated by using **try** and **except** blocks to handle non valid inputs

Output handling

The passwords/passphrases are optionally downloaded to the directory of the secure password generator python code. You can choose either yes to download to directory or choose not to and keep changes unsaved, but just print out the passwords/passphrases.

- Results are printed to the console screen
- Results are optionally outputted to a text file

Future maintenance / further development

After the initial release of the script, we plan to keep maintenance of our code for future fixes and changes if needed. Developers who may be interested in making modifications and changes to our code, may do so by forking our current repository into their own and shall do as they wish with the script.

If there are developers who wish to get in contact with us on the script and wish to share their changes and have them implemented into our main repository, please contact one of the emails provided below:

13176443@gordontafe.edu.au

13235673@gordontafe.edu.au