# Developer Documentation: Secure Password Generator

### By: Joshua Kovzan – 13243326 – 17/06/2024

## Prerequisite modules:

A black screen with orange and white text

Description automatically generated

Secrets: This is needed when generating passphrases and passwords as its secrets.choice function randomly selects from the character/word pools in a “cryptographically secure” way.

String: the string module is used to append the chosen character pools (i.e. string.ascii\_lowercase, string.digits, string.punctuation etc.) to a list to then eventually be passed into the generate\_password function.

ArgParse: This module was used to allow the user to specify which file the output will be written to when run in a CLI

## Passphrase variables:

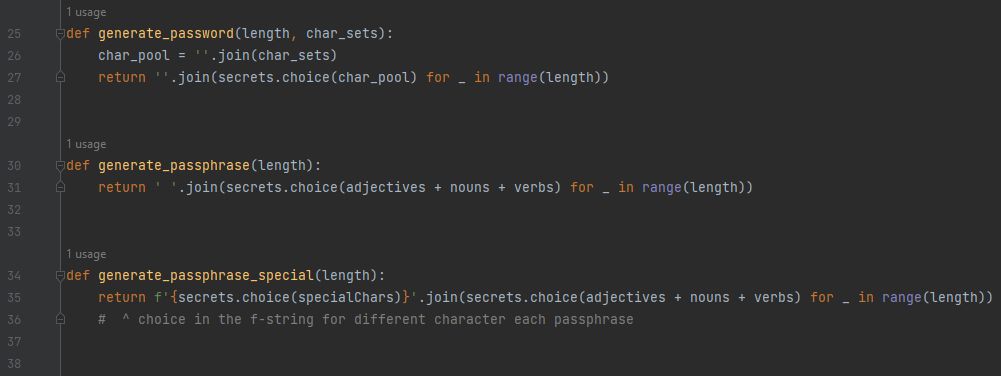
A computer screen shot of a program code

Description automatically generated

There are 3 statements here that open text files (in the same folder as the program) containing different types of words all separated on separate lines which will be read by the program and added into separate variables to be used as words in the program. We chose to use different forms of words to make it easier for the end user to remember, as opposed to using one type of category for words such as colours.

The specialChars variable contains various special characters to replace blank spaces if user chooses to do so.

## Functions:



There are 3 functions written to be used by the main program.

generate\_password choses a character to be added to the password each time the 27th line loops and takes two variables. The first “length” is an integer which will decide the passwords character limit by limiting how many times the loop will run and add to the password. The other is “char\_sets” which will specify what types of characters will be included into the password using chosen string module character pools.

generate\_passphrase uses “length” the same way generate\_password does but here the words are added from the wordlists text files instead of separated by a space

generate\_passphrase\_special functions the same as generate\_passphrase however it uses “specialChars” instead of the blank space between words. Originally both passphrase functions were one, however the special characters function better here as by adding a special.choice(specialChars) will allow each passphrase to contain different special characters between words. Previously, if 10 passphrases were generate and if the first had “/” between words, the following 9 would too.

## Main program:

### Argument Parser

A screen shot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

The program sets up the argument parser for command line use. To be used, the user needs a text file to be written to and selected after the -o or –output options.

The format is:

python ./PasswordGenerator.Complete.py -o [OutputFile]

Which may look something like this in regular use:

F:\CertIV Cybersecurity\Semester 2\Scripting\AT3\Functions\Python Password generator latest stable>python ./PasswordGeneratorComplete.py -o passwords.txt

The results will be added to the text file with each password/passphrase being separated by line.

### User choices:

#### Password vs Passphrase

A screen shot of a computer

Description automatically generated

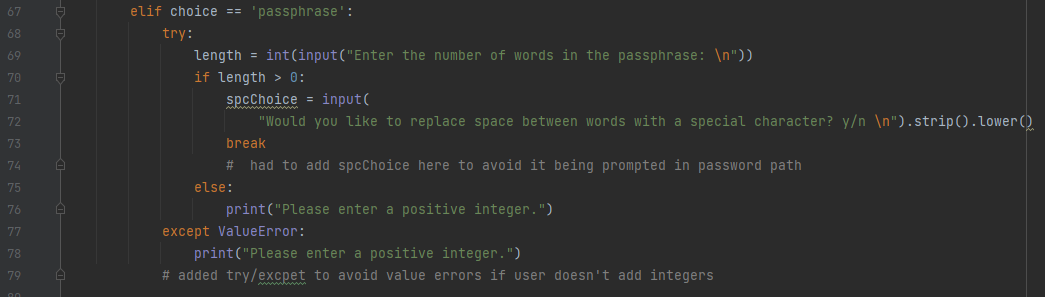
Here the user is prompted enter passphrase or password to choose which form they’d like to generate. It takes the users input, strips it, converts it to lowercase then adds it to the “choice” variable, to then decided which path to take down step 2. If it doesn’t match either options and loops and prompts the user for another attempt.

#### Length

A screen shot of a computer program

Description automatically generated

Here the user input the desired length of the password to eventually be sent to the “generate\_x” functions. Error checking it in place to make sure the entered number is above zero as you cant make a password with negative characters, it can be altered to set any password minimum. There is also error checking for it the user types a number out as a string i.e. “three” or enters a misinput i.e “12\” with the except “ValueError” statement, which will cause them to enter another number.



This passphrase portion is very similar to password however it adds one more input in line 71 if the length was accepted, another choice whether the user would like spaces to be replaced with special characters. Originally the user would be prompted for it even if they were only generating passwords.

#### Password character choices

A screen shot of a computer

Description automatically generated

This section creates a list to contain the character sets used for the password which come from the string module. For each choice (upper, lower, digits, punctuation), the user inputs a y or n, if y the character set will be appended to the list and if n it will be ignored, and the next character set choice will start. If anything, other than y or n gets inputted, the choice loops. If n was chosen for all prompts, the program will display that no character sets were chosen, and to restart now.

#### Total generated:

A computer screen with text

Description automatically generated

Here the user is requested to input a integer to decide how many passwords or phrases should be generated to then be added to the num\_generate variable. It will then check if the num\_generate variable is higher than 0 as there needs to be atleast one password generated, 0 or lower and the user will just be prompted to enter again. If the user fails to enter an integer an except ValueError statement will be hit and then it will loop again with the user being prompted to “enter a positive integer”.

#### Generating and Displaying passwords:

A computer screen shot of a program code

Description automatically generated

First a new list for results will be created and then a for loop started to iterate for the amount of times specified in Step 4 with the num\_generate variable. From there the program will check if the choice variable is “password” or “passphrase”.

If it is a password, it will call the generate\_password function with length and char\_sets as its arguments, and after reaching the length, append the resulting password to the list and looping until num\_generate has been reached.

If it is a passphrase, the program will then check if spcChoice is a y or n, to decide which generate\_passphrase function to call. Regardless of which is chosen length will be sent as its argument to decide how many words will be added onto the passphrase, when that word limit is reached, it will then append the resulting passphrase to the results list, and looped until num\_generate has been reached.

After the passwords/passphrases have been generated, it will then print each result on a new line to be copied/used.