**Um script para geração de senhas fortes e amigáveis**

**Resumo:** Este projeto tem como objetivo criar um gerador de senhas seguras e ao mesmo tempo fáceis de lembrar. Através de um script Python que utiliza duas frases como entrada: uma Frase de Configuração, para definir os parâmetros do algoritmo, e uma Frase Base, que é transformada em uma senha complexa. Através da classe PasswordGenerator, a Frase Base sofre uma série de transformações, como substituições, inversões e inserções, parametrizadas pelas regras criadas a partir da Frase de Configuração. O projeto também inclui a implementação de um método para calcular a força da senha gerada, estimando o tempo necessário para quebrá-la por força bruta.

**Objetivo:** Desenvolver um gerador de senhas robusto em Python que utilize duas frases como entrada: uma para configurar o gerador e outra para ser a base da geração da senha. O projeto visa aplicar conceitos de segurança da informação na criação de senhas fortes, explorando técnicas como substituições, inserções e transformações complexas.

**Funcionalidades principais:**

A partir de uma Frase (uma string) e de um conjunto de configurações (um dict) o sistema deverá ser capaz de gerar uma senha segura. Sugestão de entrada para o programa:

* frase\_base: A frase que servirá de base para gerar a senha.
* frase\_chave: Uma frase que será usada para gerar as configurações ou regras de substituição dos caracteres da senha.
* salto: Outra frase, uma string, usada para aumentar ainda mais a entropia da senha.
* comprimento: Qual deve ser o comprimento da senha gerada.

A senha gerada deve possuir, de forma equilibrada: (1) letras maiúsculas e minusculas; (2) dígitos 0..9 e (3) caracteres especiais: !#$%&()\*+-;<=>?@^\_{|}~ (sugestão).

O sistema deverá ser capaz de informar ao usuário a força da senha gerada.

* Força da Senha, um número entre 0..100. Quanto maior o número maior a força da senha.
* Uma descrição, Senha [Insegura, Muito Fraca, Fraca, Moderada, Forte, Muito Forte, Ultra Forte]
* Tempo para quebra da senha por força bruta supondo uma quantidade de testes de senha hipotética.

**Exemplo de uso:**

python main.py `

--base\_phrase "Esta é uma frase base para gerar a senha" `

--key\_phrase "Esta é a frase chave usada na criptografia" `

--salt "Meu Salt Super Seguro" `

--pwd\_length 16 `

--extended\_chars False

Usando .........................: ConfigGeneratorDefault + PwdGeneratorDefault

Senha gerada ...................: {c)d2|XO9EWs~FGo

Comprimento da senha ...........: 16

Força da senha (0..100).........: 92 Ultra Strong

Tempo estimado para quebra .....: aprox. 19,72 quatrilhões (10^15) de anos

Número de combinações possíveis.: 6,22 nonilhões (10^30)

Número de símbolos por caractere: 84

Testes por segundo .............: 10 milhões

**Gerador da regra de substituição dos caracteres:**

Define as substituições de caracteres com base na frase chave. A classe tem como entrada (key\_phrase) uma string, utilizada para gerar o dict de substituição de caracteres similar a { ‘a’: ‘e’, ‘b’: ‘f’, ‘c’: ‘t’, … }.

**Gerador de Senha:**

Define a lógica para geração de senhas seguras. Tem como entrada base\_phase (str), a frase base usada para gerar a senha, subst\_dct (dict), o dicionário para substituição de caracteres, e salt (str) usada para aumentar entropia da senha. O salt se mescla a base\_phase para bagunçar ainda mais a senha gerada. Logo, uma mesma base\_phase pode gerar senhas bastante distintas mesmo com pequenas mudanças no salt.

Gera uma senha segura com base nos argumentos a seguir:

base\_phrase: str # Frase Base usada para gerar a senha

key\_phrase: str # Frase Chave usada para gerar o dict de substituições de chars

salt: str = "" # Uma string, ou lista de chars, usada para aumentar a entropia da senha

pwd\_length: int = 16 # Tamanho da senha gerada. Default 16 chars

extended\_chars: bool = False # Quando True, usa todos os char UTF-8 visíveis na senha, p.ex. 😊

cfg\_generator\_class: Type[SubstRulesInterface] = SubstRulesDefault # Define a classe SubstRules

pwd\_generator\_class: Type[PwdGeneratorInterface] = PwdGeneratorDefault # PwdGenerator

**IMPORTANTE.** Por segurança, para evitar que o usuário perca sua senha devido a erros de digitações, tais como pontuações (p.ex. vírgulas), acentos incorretos etc. As entradas base\_phase, key\_phase e salt são normalizadas. Neste caso,

* São removidos os sinais de pontuação;
* As quebras de linhas são substituídas por espaços;
* Os espaços duplicados são removidos;
* As strings são convertidas em ANSI, os acentos são removido;
* As strings são convertidas em minúsculo.

Para obter dados estatísticos da senha, use o método calc\_stats, que retornará a força senha de 0..100, a força da senha de forma textual (p.ex. Strong), o tempo para quebra da senha por força bruta etc.

**Cálculo de força da senha**

Tem como entrada a senha gerada e retorna a força da senha, um inteiro de 0..100. A classe incorpora as principais premissas para calcular a força da senha:

* Penalidade por repetição de caracteres, por padrões de teclado, por palavras comuns;
* Pontuação (score) por comprimento da senha, distribuição dos caracteres (maiúsculas, minusculas, dígitos e caracteres especiais).