PasswordGenerator.py

Panoramica

La classe `PasswordGenerator` fornisce funzionalità per generare password sicure basate su set di caratteri definiti dall'utente. Include anche metodi per calcolare la complessità, l'entropia, la forza della password generata e il tempo necessario per violarla con un attacco brute force.

Attributi

- `lowercase`: Lista di lettere minuscole
- `uppercase`: Lista di lettere maiuscole
- `digits`: Lista di cifre
- `specialChars`: Lista di caratteri speciali
- `charsAvailable`: Lista di caratteri disponibili per la generazione della password basata sulle scelte dell'utente
- `password`: La password generata come stringa

Metodi

`__init__(self)`

Inizializza gli attributi della classe `PasswordGenerator`.

`buildCharsAvailable(self, choices)`

Costruisce una lista di caratteri disponibili per la generazione della password basata sulle scelte dell'utente.

- **Parametri**:
  - `choices` (lista di int): Lista contenente le scelte dei tipi di carattere (1 per lettere minuscole, 2 per lettere maiuscole, 3 per cifre, 4 per caratteri speciali).

- **Restituisce**:
  - `charsAvailable` (lista di str): Lista di caratteri disponibili per la generazione della password.

`buildPassword(self, length, charsAvailable)`

Genera una password della lunghezza specificata utilizzando i caratteri dal set disponibile.

- **Parametri**:
  - `length` (int): La lunghezza desiderata della password.
  - `charsAvailable` (lista di str): Lista di caratteri disponibili per la generazione della password.
- **Restituisce**:
  - `password` (str): La password generata.

`userInput(self, charTypes)`

Converte l'input dell'utente per le scelte dei set di caratteri in una lista di interi.

- **Parametri**:
  - `charTypes` (lista di str): Lista delle scelte dei set di caratteri inserite dall'utente.
- **Restituisce**:
  - `newList` (lista di int): Lista delle scelte valide e non duplicate dei set di caratteri.

`pwStrength(self, password)`

Calcola l'entropia (forza) della password generata.

- **Parametri**:
  - `password` (str): La password generata.
- **Restituisce**:
  - `entropy` (float): L'entropia della password in bit

`calcViolation(self, entropy)`

Calcola il numero di tentativi e il tempo necessario per violare la password generata utilizzando un attacco brute force.

- **Parametri**:
  - `entropy` (float): L'entropia della password in bit
- **Restituisce**:
  - `violationInfo` (str): Informazioni sul numero di tentativi e il tempo necessario per un attacco brute force.

Esempio di utilizzo

```
import random
import math

# Istanziare la classe PasswordGenerator
password_gen = PasswordGenerator()

# Input dell'utente per le scelte dei set di caratteri:
choices = password_gen.userInput(['1', '2', '3', '4'])

# Costruisce la lista di caratteri disponibili per generare la password
chars_available = password_gen.buildCharsAvailable(choices)

# Genera una password di lunghezza 12
password = password_gen.buildPassword(12, chars_available)
print(f"Password Generata: {password}")

# Calcola l'entropia della password generata
entropy = password_gen.pwStrength(password)
print(f"Entropia della Password: {entropy} bit")

# Calcola il tempo necessario per violare la password
violation_info = password_gen.calcViolation(entropy)
print(violation_info)
```