PERIODO : 202450 mayo– Septiembre 2024

ASIGNATURA :

TEMA :

ESTUDIANTE :

NIVEL-PARALELO - NRC:

DOCENTE :

FECHA DE ENTREGA :

SANTO DOMINGO – ECUADOR

1. Introducción

Este documento proporciona una guía detallada para comprender y utilizar el generador de contraseñas desarrollado en Python. El generador crea contraseñas alfanuméricas de 10 caracteres, asi garantizando que no se repitan caracteres en contraseñas generadas consecutivamente con cada ejecución.

El código está estructurado en una clase denominada '*PasswordGenerator*', la misma que se encarga de manejar la ejecución de contraseñas y la gestión de caracteres que se usaron previamente. En el desarrollo de este documento se detallaran por secciones específicas en código explicando su funcionalidad; también se añadirá los resultados esperados en cada ejecución.

2. Objetivos

Objetivo General:

Desarrollar un generador de contraseñas en Python que cree contraseñas alfanuméricas de 10 caracteres de manera aleatoria y sin repetición de caracteres en contraseñas consecutivas, garantizando la seguridad y unicidad de las contraseñas generadas

Objetivos Específicos:

- Implementar una clase en Python que gestione la generación de contraseñas para garantizar que cada carácter utilizado no se repita dentro de la misma contraseña y que no se utilicen caracteres en generaciones de contraseñas posteriores.
- Validar que el conjunto de caracteres permitidos es suficiente para crear al menos tres contraseñas distintas de 10 caracteres, asegurando robustez y manejabilidad en la creación de contraseñas.

3. Desarrollo / Marco Teórico/ Práctica

3.1: Importaciones:

- **random:** Se utiliza para seleccionar caracteres de manera aleatoria.
- **string:** Proporciona listas de caracteres alfabéticos y numéricos, facilitando la creación de conjuntos de caracteres permitidos.

```
1  import random  # Importa la librería random para generar números aleatorios
2  import string  # Importa la librería string para obtener caracteres alfanuméricos
3
```

3.2 Clase 'PasswordGenerator'

La clase PasswordGenerator encapsula la lógica para la generación de contraseñas y el manejo de caracteres usados.

- **self.used_characters**: Un conjunto que almacena los caracteres ya utilizados en contraseñas previas.
- **self.password_length**: Define la longitud de la contraseña (10 caracteres).
- **self.allowed_characters**: Lista de caracteres permitidos (letras mayúsculas, minúsculas y dígitos).
- **Validación de caracteres únicos**: Asegura que hay suficientes caracteres únicos para generar al menos tres contraseñas sin repetición.

```
4  class PasswordGenerator:
5      def __init__(self):
6          # Inicializa el conjunto de caracteres utilizados previamente
7          self.used_characters = set()
8          # Define la longitud de la contraseña
9          self.password_length = 10
10         # Define los caracteres permitidos (letras mayúsculas, minúsculas y dígitos)
11         self.allowed_characters = list(string.ascii_letters + string.digits)
12
13         # Verifica que hay suficientes caracteres únicos para generar al menos tres contraseñas
14         if len(self.allowed_characters) < 3 * self.password_length:
15             raise ValueError("No hay suficientes caracteres únicos para generar tres contraseñas distintas de 10 carac
16
```

3.3: Método 'generate_password':

- **available_characters:** Lista de caracteres disponibles, excluyendo los ya utilizados.
- **Validación de caracteres disponibles:** Asegura que hay suficientes caracteres disponibles para generar una nueva contraseña.
- **Generación de contraseña:** Utiliza random.sample para seleccionar 10 caracteres únicos aleatoriamente.

- **Actualización de caracteres usados:** Agrega los caracteres de la contraseña generada al conjunto de caracteres utilizados.

```
17      def generate_password(self):
18          # Filtra los caracteres disponibles eliminando los ya usados
19          available_characters = [char for char in self.allowed_characters if char not in self.used_characters]
20
21          # Verifica que hay suficientes caracteres disponibles para generar una nueva contraseña
22          if len(available_characters) < self.password_length:
23              raise ValueError("No hay suficientes caracteres disponibles para generar una nueva contraseña.")
24
25          # Genera una contraseña de 10 caracteres seleccionando aleatoriamente de los caracteres disponibles
26          password = ''.join(random.sample(available_characters, self.password_length))
27          # Actualiza el conjunto de caracteres utilizados
28          self.used_characters.update(password)
29          return password
```

3.4: Método 'generate_multiple_passwords':

- **count:** Número de contraseñas a generar.
- **passwords:** Lista que almacena las contraseñas generadas.

```
31      def generate_multiple_passwords(self, count):
32          # Genera múltiples contraseñas
33          passwords = []
34          for _ in range(count):
35              passwords.append(self.generate_password())
36          return passwords
```

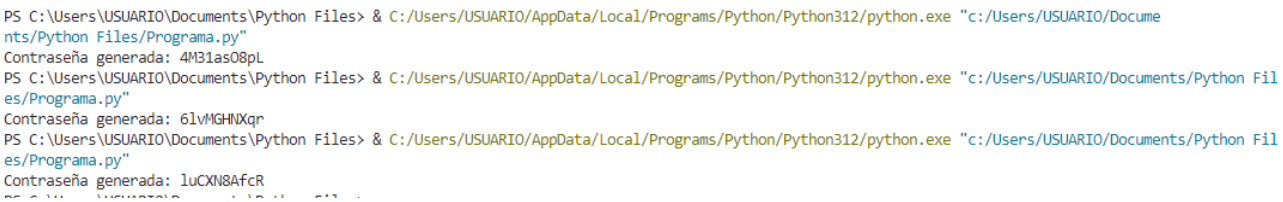
3.5: Uso del Generador de Contraseñas

- **Instanciación:** Crea una instancia del generador de contraseñas.
- **Generación de contraseña:** Llama al método generate_password para generar una contraseña.
- **Manejo de excepciones:** Captura y muestra cualquier 'ValueError' que pueda ocurrir (por ejemplo, si no hay suficientes caracteres disponibles).

```
38  # Crear una instancia del generador de contraseñas
39  password_generator = PasswordGenerator()
40
41  # Generar tres contraseñas y manejarlas de forma segura
42  try:
43      password = password_generator.generate_password()
44      print(f"Contraseña generada: {password}")
45  except ValueError as e:
46      print(e)
```

Resultado Esperado

Al ejecutar 3 veces el archivo, deberías ver la salida de una contraseña generada aleatoriamente:



Consideraciones Finales

- **Robustez**: El generador de contraseñas es robusto y maneja adecuadamente los casos en los que no hay suficientes caracteres disponibles.
- **Sostenibilidad**: La estructura modular y el manejo de excepciones aseguran que el código sea fácil de mantener y extender.
- **Ética**: El uso de contraseñas seguras es crucial para la protección de datos personales y corporativos. Este generador asegura contraseñas únicas y seguras, contribuyendo a la ciberseguridad.

4. Conclusiones

- El generador de contraseñas, creado en Python, puede crear contraseñas seguras y únicas de 10 caracteres alfanuméricos. El generador de contraseñas cumple con las especificaciones de no repetir caracteres ni dentro de una contraseña ni en contraseñas generadas consecutivamente.
- El código es robusto, sostenible y fácil de mantener gracias a su estructura modular y manejo adecuado de excepciones, lo que permite extensiones y mejoras futuras sin comprometer la funcionalidad actual.

5. Recomendaciones

- Permitir configuraciones adicionales, como la inclusión de caracteres especiales, la personalización de la longitud de la contraseña, y opciones para cumplir con diferentes políticas de seguridad de contraseñas.
- Evaluar la integración del generador de contraseñas en aplicaciones que requieran la creación de contraseñas seguras, como sistemas de gestión de usuarios o

aplicaciones web, proporcionando una capa adicional de seguridad en la protección de datos.

6. Bibliografía/ Referencias

- *"NIST Special Publication 800-63B: Digital Identity Guidelines"*. National Institute of Standards and Technology (NIST)https://pages.nist.gov/800-63-3/sp800-63b.html
- *"random — Generate pseudo-random numbers"*. Python Software Foundation. https://docs.python.org/3/library/random.html

7. Legalización de documento

Nombres y Apellidos:

CI:

Firma: Imagen de su firma