DOCUMENTACIÓN TÉCNICA

Generador de Claves de Alta Seguridad (GCS-Core)

Visión General del Sistema

El **GCS-Core** es un script de backend diseñado para la generación determinista-aleatoria de credenciales. A diferencia de los generadores puramente aleatorios (Random String Generators), este sistema utiliza un enfoque híbrido:

1. **Input del Usuario (Semilla):** Mantiene una base mnemotécnica para facilitar la recordación.
2. **Inyección de Ruido:** Introduce entropía calculada matemáticamente para alcanzar una longitud objetivo (*Target Length*) de 20 caracteres.

Arquitectura y Flujo de Datos

El script sigue un flujo lineal de procesamiento de cadenas (String Manipulation Pipeline).

Diagrama de Flujo Lógico:

[INICIO]

⬇

[INPUT] -> Normalización (.upper) y Truncado (Max 9 chars)

⬇

[CÁLCULO] -> Definición de cortes (slices) y espacio sobrante

⬇

[SEGMENTACIÓN] -> División de la semilla en 3 vectores

⬇

[INYECCIÓN] -> Generación de ruido aleatorio por bloque

⬇

[ENSAMBLAJE] -> Unión con separadores ("-") y Recorte final

⬇

[SALIDA] -> Print a STDOUT

Diccionario de Datos

Variables críticas utilizadas en el ámbito global del script.

Variable	Tipo	Origen	Descripción Técnica
caracteres_mix	str	Librería	Universo de caracteres (a-Z, 0-9, !@#...). Fuente de entropía.
palabra_cl	str	Usuario	La semilla. Se normaliza a mayúsculas y se trunca a len <= 9.
secciones_palabra	list	Calculado	Array que contiene los 3 fragmentos de la palabra original.
espacio_sobrante	int	Calculado	Cantidad de caracteres aleatorios necesarios para llegar al *Target* (20).
ruido	str	Random	Cadena temporal de caracteres aleatorios generada en cada iteración.
clave_segura	str	Salida	El string final resultante tras el join y el slice.

Lógica del Algoritmo (Core Logic)

El algoritmo se basa en la Distribución Equitativa de Espacio.

A. Truncamiento de Entrada (Safety Cap)

palabra_cl = palabra_cl[:9]

Se limita la entrada a 9 caracteres. Esto asegura que la variable espacio_sobrante nunca sea negativa (evitando *index out of bounds* o errores lógicos), dado que el objetivo es 20 y se reservan 2 caracteres para guiones.

B. Segmentación Dinámica (Slicing)

Se utilizan divisiones enteras (//) para dividir la palabra en 3 partes, independientemente de su longitud.

C. Cálculo de Relleno

El sistema calcula cuántos caracteres "basura" debe generar basándose en el espacio vacío restante:

espacio_sobrante = 20 - len(palabra_cl) - 2

cantidad_random = espacio_sobrante // 3

(El - 2 representa los guiones separadores que se añadirán al final.)

5.Requisitos de Mantenimiento

Modificar la Longitud de la Clave

Si se desea cambiar la longitud final (actualmente 20), se deben ajustar dos constantes en el código:

1. Línea de cálculo de espacio: Cambiar el valor 20 por el deseado (ej. 30). espacio_sobrante = 30 - len(palabra_cl) - 2
2. Línea de recorte final: Ajustar el slicing final. clave_segura = clave_segura[:30]

Personalizar Caracteres Permitidos

Para restringir símbolos especiales (para sistemas legacy que no soportan @ o $), edite la variable caracteres_mix:

- **Código actual:** caracteres_mix = string.ascii_letters + string.digits + string.punctuation
- **Versión sin símbolos:** caracteres_mix = string.ascii_letters + string.digits

6. ANEXO TÉCNICO: DESGLOSE DE CÓDIGO

Módulos y Constantes

```
import random
import string
```

- Líneas 1-2: Importación de librerías estándar.
    - random: Motor de generación de números pseudoaleatorios (*PRNG*).
    - string: Contenedor de constantes ASCII.

```
caracteres_mix = string.ascii_letters + string.digits + string.punctuation
```

Línea 3: Definición del **Espacio Muestral (Pool de Entropía)**.

- Concatena letras (a-Z), dígitos (0-9) y signos de puntuación (!@#...). Esto maximiza la robustez de la contraseña frente a ataques de fuerza bruta.

Ingesta y Normalización

```
palabra_cl = input('Ingrese una palabra para su clave: ').upper()
```

Línea 4: Captura de datos (Input Stream).

- Muestra el prompt al usuario.
- .upper(): **Normalización.** Convierte toda la entrada a mayúsculas para mantener consistencia visual en la salida.

```
palabra_cl = palabra_cl[:9]
```

- **Línea 5: Truncamiento de Seguridad (Safety Cap).**
    - Aplica un *slice* estricto para conservar solo los primeros 9 caracteres.
    - *Propósito:* Prevenir errores de desbordamiento negativo en el cálculo de espacio sobrante más adelante.

```
longitud = len(palabra_cl)
corte1 = longitud // 3
corte2 = 2 * (longitud // 3)
```

Líneas 6-8: Cálculo de índices de corte.

- // 3: Utiliza la **división entera** para determinar el tamaño base de cada tercio de la palabra.
- Define los puntos exactos donde la "tijera virtual" cortará la palabra.

```
parte1 = palabra_cl[:corte1]
parte2 = palabra_cl[corte1:corte2]
parte3 = palabra_cl[corte2:]
```

Líneas 9-11: Ejecución del **Slicing**.

- Genera tres sub-cadenas (substrings) basadas en los índices calculados anteriormente.

```
secciones_palabra = [parte1, parte2, parte3]
```

Línea 12: Estructuración de datos.

- Encapsula las tres partes en una **Lista** para permitir su procesamiento iterativo.

Cálculo de Entropía y Bucle Principal

```
partes_finales = []
```

Línea 13: Inicialización del acumulador (lista vacía) que almacenará los bloques procesados.

```
espacio_sobrante = 20 - len(palabra_cl) - 2
```

Línea 14: Cálculo del Delta (Padding Calculation).

- Determina cuántos caracteres aleatorios se necesitan.
- Fórmula: Target(20) - Semilla(Usuario) - Separadores(2).

```
for parte in secciones_palabra:
```

- **Línea 15:** Inicio del Bucle Iterativo. Recorre cada uno de los 3 fragmentos de la palabra.

```
cantidad_random = espacio_sobrante // 3
```

Línea 16: Distribución de carga.

- Divide el espacio sobrante total entre los 3 bloques para que el "ruido" se reparta equitativamente.

```
ruido = "".join(random.choices(caracteres_mix, k=cantidad_random))
```

Línea 17: Generación de Entropía.

- random.choices(..., k=...): Selecciona caracteres al azar del pool caracteres_mix. Permite repetición (reemplazo).
- "".join(...): Convierte la lista de caracteres resultante en un solo String.

```
bloque = ruido + parte
```

Línea 18: Ofuscación (Masking).

- Concatena el ruido generado *antes* del fragmento de la palabra del usuario para ocultarlo visualmente.

```
partes_finales.append(bloque)
```

- **Línea 19:** Almacenamiento. Agrega el bloque terminado a la lista partes_finales.

Ensamblaje y Salida (Output)

```
clave_segura = "-".join(partes_finales)
```

Línea 20: Unificación.

- Transforma la lista en un solo String, insertando un guion - entre cada elemento.

```
clave_segura = clave_segura [:20]
```

Línea 21: Recorte Final (Final Polish).

- Garantiza que la longitud no exceda los 20 caracteres bajo ninguna circunstancia (por ejemplo, si los redondeos de la división generaron un carácter extra).

```
print(f"\nTu clave segura es: {clave_segura}")
print(f"Longitud actual: {len(clave_segura)} caracteres")
```

Líneas 22-23: Salida a Consola.

- Utiliza f-strings para interpolar la variable y mostrar el resultado final y su longitud al usuario.

7. CONCLUSIONES Y RECOMENDACIONES

Conclusiones del Proyecto

Tras el desarrollo, implementación y análisis del software GCS-Core v2.0 (Generador de Claves Seguras), se establecen las siguientes conclusiones:

1. Equilibrio entre Seguridad y Usabilidad (User-Centric Security): El algoritmo logra resolver la dicotomía clásica de la seguridad informática: crear una contraseña robusta sin sacrificar la capacidad de recordarla. Al utilizar una "semilla" mnemotécnica (palabra del usuario) rodeada de entropía artificial, se facilita la adopción de hábitos de seguridad más higiénicos por parte del usuario final.
2. Eficacia de la Manipulación de Cadenas en Python: El uso de técnicas de *slicing* (recorte) y concatenación dinámica demostró ser eficiente para el manejo de estructuras de texto. La segmentación matemática en tres bloques permite una distribución uniforme del "ruido" aleatorio, evitando patrones predecibles visualmente en la contraseña final.
3. Aumento Significativo de la Entropía: En comparación con la versión 1.0, la inclusión del módulo string.punctuation y string.digits en la versión 2.0 incrementa exponencialmente el espacio muestral de la contraseña. Una clave de 20 caracteres con este set de caracteres es computacionalmente costosa de vulnerar mediante ataques de fuerza bruta estándar.

4. Limitaciones de la Pseudoaleatoriedad: Se identifica que el módulo random de Python utiliza el algoritmo *Mersenne Twister*, el cual es determinista y no criptográficamente seguro para entornos de alto riesgo (banca gubernamental, secretos de estado). Sin embargo, para el propósito general del usuario (redes sociales, correos, cuentas personales), el nivel de seguridad es sobresaliente y adecuado.

Recomendaciones Futuras

- **Migración a secrets:** Para una futura versión 3.0, se recomienda reemplazar la librería random por la librería secrets de Python, diseñada específicamente para criptografía segura.
- **Interfaz Gráfica (GUI):** Implementar una interfaz visual usando librerías como Tkinter o PyQt para mejorar la experiencia del usuario y abandonar la consola de comandos.
- **Gestión de Portapapeles:** Añadir una función que copie automáticamente la contraseña generada al portapapeles (clipboard) del sistema operativo.

7. BIBLIOGRAFÍA Y REFERENCIAS

Para el desarrollo de este software y la redacción de su documentación técnica, se han consultado las siguientes fuentes y estándares:

Documentación Oficial

- **Python Software Foundation.** (2024). *The Python Standard Library: random — Generate pseudo-random numbers*. Recuperado de docs.python.org
- **Python Software Foundation.** (2024). *The Python Standard Library: string — Common string operations*. Recuperado de docs.python.org

Estándares de Seguridad

- **NIST (National Institute of Standards and Technology).** (2017). *NIST Special Publication 800-63B: Digital Identity Guidelines*. (Sección sobre memorización y longitud de secretos).
- **OWASP (Open Web Application Security Project).** (2023). *Password Storage Cheat Sheet & Authentication Guidelines*.

Recursos de Programación

- **Van Rossum, G., & Drake, F. L.** (2009). *Python 3 Reference Manual*. CreateSpace. (Conceptos sobre inmutabilidad de strings y listas).
- **Sweigart, A.** (2019). *Automate the Boring Stuff with Python: Practical Programming for Total Beginners* (2nd Ed.). No Starch Press. (Lógica de manipulación de texto).