UNIVERSIDAD INTERNACIONAL DEL ECUADOR

Nombre: Hugo David Armas Peralta

Materia: Lógica de programación

Fecha: Quito, 24 de agosto del 2025

PROYECTO FINAL

Introducción

El presente proyecto tiene como objetivo desarrollar un programa en Python capaz de generar contraseñas de manera aleatoria, brindando opciones personalizadas y seguras para el usuario. Todo esto en un contexto donde la seguridad digital es cada vez más importante, por lo cual contar con contraseñas robustas resulta fundamental para proteger información personal y profesional.

Para cumplir este propósito, se implementaron distintas funciones que permiten desde la creación de contraseñas simples hasta configuraciones más específicas, donde el usuario puede elegir el tipo de caracteres que desea incluir (mayúsculas, minúsculas, números y símbolos). El diseño incorpora un menú interactivo y elementos estéticos en consola que facilitan la navegación, mejorando la experiencia del usuario.

Asimismo, el proyecto integra librerías como **random**, para la selección aleatoria de caracteres; **os**, para la limpieza de la pantalla; **time**, para la generación de pausas y animaciones; y **shutil**, para centrar los títulos en la terminal. Con ello, no solo se logra una herramienta funcional, sino también una aplicación práctica que combina seguridad y usabilidad.

Link del Repositorio Github: https://github.com/Dav522/Generador_decontrase-as_seguras

Descripción del proyecto

1. **Importación de librerías**

```
import random
import os          # Para limpiar la pantalla
import time        # Animacion
import shutil      # Para centrar el título según el ancho de la terminal

opcion = ""
```

Se importan las librerías necesarias para el programa:

- random para elegir caracteres de forma aleatoria.
- os para limpiar la pantalla en la consola.
- time para realizar pequeñas pausas o animaciones.
- shutil para obtener el ancho de la terminal y centrar el título.

Además, se inicializa la variable opción que ayuda a controlar el menú principal.

2. **Estructura del menú y diseño estético**

```
while opcion != "0":
    # ====== ESTÉTICA: limpiar pantalla ======
    os.system("cls" if os.name == "nt" else "clear")

    # ====== ESTÉTICA: encabezado centrado con marco ======
    cols = shutil.get_terminal_size((80, 20)).columns  # ancho de terminal (80 por defecto
    titulo = " Generador de Contraseñas "
    borde = "=" * len(titulo)
    print(borde.center(cols))
    print(titulo.center(cols))
    print(borde.center(cols))
    print()  # línea en blanco

    # ===== Menú principal =====
    print("1.- Generar contraseña segura aleatoria")
    print("2.- Generar contraseña segura específica")
    print("0.- Salir")
    print("-" * 40)  # separador estético
    opcion = input("Elige una opción: ").strip()
```

Antes de mostrar el menú, se utiliza os.system("cls" if os.name == "nt" else "clear") para limpiar la consola. Aquí se aplica un operador ternario: si el sistema operativo es Windows (os.name == "nt"), ejecuta cls; de lo contrario (Linux/Mac), ejecuta clear. Después, con shutil.get_terminal_size() se obtiene el ancho de la terminal y se usa .center() para centrar el título y el marco de caracteres "=". Esto permite que nuestro encabezado aparezca alineado en el medio, no importa el tamaño de ventana que salga. Finalmente, se imprime el menú principal con sus opciones numeradas, un separador netamente estetico de guiones ("-" * 40), y se utiliza input() para capturar la opción elegida.

3. Opción 1 Generación de contraseña aleatoria simple sin parámetros

```python
    # ========= Opción 1: contraseña aleatoria simple =========
    if opcion == "1":
        caracteres = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"

        # Validación de longitud (básica, sin try/except)
        while True:
            longitud_txt = input("¿Número de caracteres?: ").strip()
            if not longitud_txt.isdigit():
                print("Longitud inválida. Debe ser un número entero.")
                continue
            longitud = int(longitud_txt)
            if longitud <= 0 or longitud > 64:
                print("Usa una longitud entre 1 y 64.")
                continue
            break

        # ESTÉTICA: pequeña "animación" de generación
        print("\nGenerando", end="", flush=True)
        for _ in range(3):
            time.sleep(0.3)
            print(".", end="", flush=True)
        print("\n")

        contraseña = ""
        for i in range(longitud):
            contraseña += random.choice(caracteres)

        # ESTÉTICA: resultado con caja simple
        caja = "-" * (len(contraseña) + 8)
        print(caja)
        print(f"|  {contraseña}  |")
        print(caja)
        input("\n(Enter para volver al menú) ")
```

En esta sección se implementa la opción 1 del menú, que genera una contraseña usando letras y números.

Condicional if: valida si la opción elegida es "1".

Conjunto de caracteres: se define una cadena fija con todas las letras y números posibles.

Validación de longitud: aqui validamos la longitud de la contraseña deseada

Se usa un bucle while True para pedir la longitud hasta que el usuario ingrese un número válido.

Se emplea .isdigit() para comprobar que el dato sea un número.

Se usan los alteradores de bucle continue (volver a pedir) y break (salir de la validación).

Animación: se usa time.sleep() para mostrar una secuencia de puntos, simulando que la contraseña se está generando.

Construcción de la contraseña:

Con un for en range(longitud) se repite la elección aleatoria de caracteres usando random.choice(). haciendo uso de la funcion choice y la libreria random

Los caracteres se van concatenando en la variable contraseña.

Estética: el resultado se muestra dentro de una “caja” de guiones, calculando su tamaño con len(contraseña).

Entrada de control: input("\n(Enter para volver al menú)") pausa la ejecución para que el usuario pueda ver el resultado antes de regresar.

4. **Opción 2. Generación de contraseña especifica haciendo uso de listas y tuplas y selección por el usuario**

```
    # ========= Opción 2: contraseña específica (con tuplas y lista) =========
    elif opcion == "2":
        tipos_disponibles = (
            ("minúsculas", "abcdefghijklmnopqrstuvwxyz"),
            ("MAYÚSCULAS", "ABCDEFGHIJKLMNOPQRSTUVWXYZ"),
            ("números",    "0123456789"),
            ("símbolos",   "!@#$%^&*()-_=+[]{}<>?")
        )

        seleccionados = []

        print("\nConfigura tu contraseña específica (responde s/n):")
        print("-" * 40)
        for nombre, conjunto in tipos_disponibles:
            r = input(f"¿Incluir {nombre}? (s/n): ").strip().lower()
            if r == "s":
                seleccionados.append(conjunto)

        if len(seleccionados) == 0:
            print("\nDebes seleccionar al menos un tipo de carácter.")
            input("(Enter para volver al menú) ")
            continue

        caracteres = "".join(seleccionados)

        while True:
            longitud_txt = input("¿Número de caracteres?: ").strip()
            if not longitud_txt.isdigit():
                print("Longitud inválida. Debe ser un número entero.")
                continue
            longitud = int(longitud_txt)
            if longitud <= 0 or longitud > 64:
                print("Usa una longitud entre 1 y 64.")
                continue
            break
```

En esta parte se desarrolla la opción 2 del menú, que permite configurar qué tipos de caracteres incluir en la contraseña (minúsculas, mayúsculas, números o símbolos).

Tupla de tuplas:

tipos_disponibles guarda cada categoría como una tupla (nombre, conjunto_de_caracteres).

Esto facilita recorrerlas en un bucle sin necesidad de definir variables separadas.

Lista:

Se crea la lista seleccionados = [] para acumular los conjuntos de caracteres elegidos por el usuario.

Cada vez que el usuario responde “s”, se agrega ese conjunto a la lista con .append().

Bucle for con desempaquetado:

for nombre, conjunto in tipos_disponibles: recorre cada tupla interna.

nombre sirve para mostrar el texto al usuario y conjunto contiene los caracteres a añadir.

Validación:

Si la lista seleccionados queda vacía, se muestra un mensaje y se usa continue para volver al menú.

Unión de datos:

"".join(seleccionados) combina los elementos de la lista en un solo string con todos los caracteres permitidos.

Bucle while de validación de longitud:

Igual que en la opción 1, asegura que el número ingresado sea válido antes de generar la contraseña.

5. **Generación y salida del menú**

```
105         # ESTÉTICA: pequeña “animación” de generación
106         # Uso de funcion de animación
107         print("\nGenerando", end="", flush=True)
108         for _ in range(3):
109             time.sleep(0.3)
110             print(".", end="", flush=True)
111         print("\n")
112
113         contraseña = ""
114         for i in range(longitud):
115             contraseña += random.choice(caracteres)
116
117         caja = "-" * (len(contraseña) + 8)
118         print(caja)
119         print(f"|  {contraseña}  |")
120         print(caja)
121         input("\n(Enter para volver al menú) ")
122
123     elif opcion == "0":
124         print("\nGracias por preferirnos\n")
125         time.sleep(0.6)  # pausa breve para que se vea el mensaje y la animacion
126
127     else:
128         print("\nOpción incorrecta")
129         time.sleep(0.8)  # pausa breve y vuelve al menú y se borra la pantalla
```

En esta parte se muestra cómo se genera la contraseña final y se presenta al usuario con un formato más estético:

Animación:

Se usa un bucle for _ in range(3): junto con time.sleep(0.3) para imprimir puntos poco a poco.

El parámetro end="" evita el salto de línea, y flush=True obliga a mostrar el texto inmediatamente.

Esto simula que la contraseña está siendo procesada.

Construcción de la contraseña:

Se inicializa la variable contraseña = "".

Con un bucle for i in range(longitud): se van agregando caracteres aleatorios mediante random.choice(caracteres).

Estética de salida:

Se calcula una línea de guiones proporcional al tamaño de la contraseña con "-" * (len(contraseña) + 8).

La contraseña se muestra dentro de una “caja” delimitada por guiones.

Se utiliza una f-string f"| {contraseña} |" para incrustar la contraseña en el texto.

Control de flujo:

Se usa input("\n(Enter para volver al menú)") para pausar la ejecución y que el usuario pueda ver el resultado antes de regresar al menú.

Opción de salida e inválida:

elif opcion == "0": muestra un mensaje de despedida con una pausa (time.sleep(0.6)).

else: maneja opciones incorrectas, mostrando un mensaje y esperando un momento antes de limpiar la pantalla.

Implicaciones y limitaciones

Una implicación encontrada al momento de realizar este proyecto fue comprobar que el dato sea número, se empleó el uso de la función ***.isdigit()*** para comprobar que el dato sea número.

Otra limitación encontrada fue la aleatoriedad se empleo el uso de la librería random y de la función choice, con el fin de obtener aleatoriedad única en cada generación de contraseña.

Existe una limitación en el almacenamiento ya que solo se muestran las contraseñas en la consola, no se guardan en ningún archivo ni se cifran para ningún uso posterior, el nivel de seguridad de la contraseña depende de las elecciones que haga el usuario, si el usuario escoge pocas opciones de seguridad la contraseña no será realmente fuerte