

Reporte del Proyecto del propedéutico de Programación

GENERADOR DE CONTRASEÑAS

Se desarrolló un programa en Python que genera contraseñas seguras para los usuarios, con la capacidad de compartir las contraseñas generadas a través de la comunicación mediante sockets.

Se redactaron dos códigos, uno estaría montado en el servidor donde contiene la función de generar una contraseña segura a partir de los requisitos que envía el cliente. Se explicará detalladamente los pasos para tener este código:

Crear un servidor que acepte conexiones de múltiples clientes:

1. Primero, se importa el módulo socket, que proporciona una interfaz para la creación de sockets.
2. A continuación, se define la dirección IP del servidor y el número de puerto al que los clientes se conectarán.
3. Después, se crea un objeto socket utilizando socket.socket(). El primer argumento (socket.AF_INET) especifica que se utilizará el protocolo IPv4, y el segundo argumento (socket.SOCK_STREAM) indica que se utilizará un socket de flujo.
4. Luego, se llama a la función bind() del objeto socket para asociar el socket con la dirección IP y el número de puerto especificados anteriormente.
5. Se llama a la función listen() del objeto socket para poner el socket en modo de escucha, lo que significa que está listo para aceptar conexiones entrantes.
6. Se crea una lista llamada conexiones que se utilizará para almacenar todas las conexiones entrantes.
7. Se inicia un bucle while que se ejecutará indefinidamente hasta que el programa se detenga manualmente.
8. Dentro del bucle, se llama a la función accept() del objeto socket para aceptar una nueva conexión entrante. La función accept() devuelve una tupla que contiene un objeto de conexión (conn) y la dirección del cliente (addr).
9. Se agrega el objeto de conexión a la lista de conexiones.
10. Finalmente, se imprime un mensaje indicando que se ha establecido una conexión con el cliente y se muestra su dirección IP (addr).

```
import socket

HOST = '127.0.0.1' # dirección IP del servidor
PORT = 65432 # número de puerto

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conexiones = []
    while True:
        conn, addr = s.accept()
        conexiones.append(conn)
        print('Conexión establecida por', addr)
```

Establecer un protocolo de comunicación entre el servidor y los clientes para solicitar y enviar contraseñas:

1. El bucle while True se utiliza para esperar continuamente nuevas conexiones entrantes.
2. Cuando se recibe una nueva conexión, se llama a la función accept() del objeto socket para aceptarla. La función accept() devuelve una tupla que contiene un objeto de conexión (conn) y la dirección del cliente (addr).
3. Se utiliza la sentencia with conn: para crear un contexto en el que el objeto de conexión conn está disponible. Esto asegura que el objeto de conexión se cierre correctamente al final del bloque.
4. Se imprime un mensaje indicando que se ha establecido una conexión con la dirección del cliente.
5. Se inicia un bucle while True para recibir datos del cliente y enviar contraseñas aleatorias.
6. Dentro del bucle, se llama a la función recv() del objeto de conexión para recibir datos del cliente. La función recv() devuelve los datos recibidos como bytes.
7. Se utiliza la sentencia if not data: para comprobar si no se han recibido datos. Si no se han recibido datos, significa que el cliente ha cerrado la conexión, por lo que se sale del bucle interior.
8. Se decodifican los datos recibidos utilizando la función decode() para convertirlos en una cadena.
9. Se dividen los requisitos de contraseña en una lista utilizando la función split(). En este caso, se asume que los requisitos de contraseña se han enviado como una cadena separada por comas.
10. Se llama a la función generar_contrasena() para generar una contraseña aleatoria que cumpla con los requisitos especificados.

11. Se envía la contraseña generada al cliente utilizando la función `sendall()` del objeto de conexión. La función `sendall()` envía los datos especificados como bytes.
12. El bucle interior continúa hasta que el cliente cierra la conexión o no se reciben datos. Luego, el programa vuelve al bucle exterior y espera una nueva conexión entrante.

```
while True:
    conn, addr = s.accept()
    with conn:
        print('Conexión establecida por', addr)
        while True:
            data = conn.recv(1024)
            if not data:
                break
            requisitos = data.decode().split(',')
            contrasena = generar_contrasena(requisitos)
            conn.sendall(contrasena.encode())
```

Utilizar listas para almacenar los diferentes conjuntos de caracteres y generar contraseñas seguras al azar basadas en los requisitos especificados por los clientes:

Se define una función llamada `generar_contrasena()` que recibe una lista de requisitos para generar una contraseña aleatoria.

1. La función recibe una lista llamada `requisitos` que contiene cinco elementos: longitud, mayúsculas, minúsculas, números y caracteres especiales. Cada uno de estos elementos es una cadena que representa un valor booleano (True o False).
2. Se define una cadena vacía llamada `alfabeto` que se utilizará para construir el alfabeto a partir del cual se generará la contraseña.
3. Se utiliza una serie de sentencias `if` para comprobar los requisitos de la contraseña y añadir los caracteres correspondientes al alfabeto. Si un requisito es True, se añade el conjunto correspondiente de caracteres (`string.ascii_uppercase` para mayúsculas, `string.ascii_lowercase` para minúsculas, `string.digits` para números y `string.punctuation` para caracteres especiales) al alfabeto.
4. Se utiliza la función `random.choice()` junto con una comprensión de lista para generar una contraseña aleatoria. La comprensión de lista utiliza la función `random.choice()` para elegir un carácter aleatorio del alfabeto para cada

posición en la contraseña. La longitud de la contraseña se especifica mediante el argumento `int(longitud)`.

5. Se devuelve la contraseña generada como una cadena.

```
import random
import string

def generar_contrasena(requisitos):
    longitud, mayusculas, minusculas, numeros, caracteres_especiales = requisitos
    alfabeto = ''
    if mayusculas == 'True':
        alfabeto += string.ascii_uppercase
    if minusculas == 'True':
        alfabeto += string.ascii_lowercase
    if numeros == 'True':
        alfabeto += string.digits
    if caracteres_especiales == 'True':
        alfabeto += string.punctuation
    contrasena = ''.join(random.choice(alfabeto) for i in range(int(longitud)))
    return contrasena

requisitos = (8, 'True', 'True', 'True', 'False')
contrasena = generar_contrasena(requisitos)
print(contrasena)

1Jr2l6Im
```

Ahora para la parte del cliente, se elabora un código para conectar al servidor y enviar la lista de requisitos para la generación de la contraseña segura. Al recibir respuesta del servidor, con la contraseña generada, la muestra en terminal y se guarda en un archivo .txt la contraseña. A continuación una explicación detallada.

1. La primera línea importa el módulo `socket`, que proporciona una API para crear y utilizar sockets de red.
2. Luego, se define la dirección IP del servidor en la variable `HOST`. En este caso, es `127.0.0.1`, que es la dirección IP de loopback, lo que significa que el servidor y el cliente se ejecutan en la misma máquina.
3. La variable `PORT` define el número de puertos al que se conectará el cliente.
4. El bloque `with` se utiliza para crear un objeto de socket utilizando la función `socket.socket()`. El primer argumento, `socket.AF_INET`, especifica que se utilizará la familia de direcciones IPv4. El segundo argumento, `socket.SOCK_STREAM`, especifica que se utilizará el protocolo TCP.
5. La función `s.connect((HOST, PORT))` se utiliza para conectarse al servidor en la dirección IP y el puerto especificados. Si la conexión se establece correctamente, el cliente puede enviar y recibir datos a través del socket `s`.

```
import socket

HOST = '127.0.0.1' # dirección IP del servidor
PORT = 65432 # número de puerto

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
```

Después se lee la entrada del usuario utilizando la función `input()`. El mensaje que se muestra en la consola pide al usuario que ingrese los requisitos de la contraseña, que se especifican como una cadena separada por comas que indica la longitud, si debe tener mayúsculas, minúsculas, números y caracteres especiales.

La segunda línea utiliza el objeto de socket `s` para enviar los requisitos de la contraseña al servidor. Primero, se llama al método `encode()` en la cadena `requisitos` para convertirla en una secuencia de bytes, que es el formato en el que se envían los datos a través del socket. Luego, se llama al método `sendall()` en el objeto de socket `s`, que envía los datos a través del socket hasta que se han enviado todos los datos o se produce un error.

```
requisitos = input('Ingrese los requisitos de la contraseña (longitud,mayusculas,minusculas,numeros,caracteres especiales): ')
s.sendall(requisitos.encode())
```

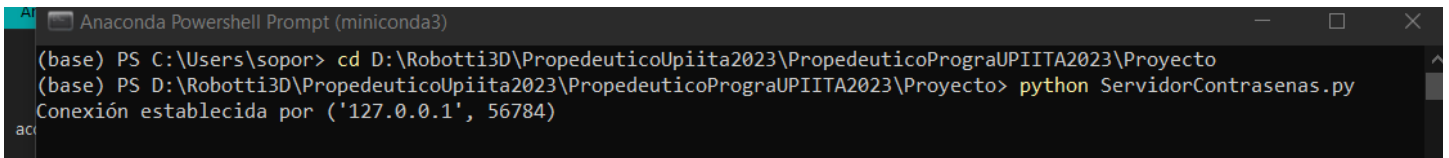
El siguiente fragmento de código recibe datos a través de un socket y los escribe en un archivo de texto.

1. La primera línea utiliza la función `open()` para crear un archivo llamado `contrasenas.txt` en modo de escritura (`'w'`). El modo de escritura sobrescribirá el archivo si ya existe o lo creará si no existe.
2. El archivo se abre en un bloque `with`, lo que significa que se cerrará automáticamente al final del bloque, incluso si ocurre una excepción.
3. Dentro del bloque `with`, se utiliza un bucle `while` para recibir datos a través del socket `s`. La función `recv()` se utiliza para recibir datos del socket en bloques de hasta 1024 bytes. Los datos se almacenan en la variable `data`.
4. La instrucción `if not data:` verifica si no hay más datos que recibir. Si `data` es una cadena vacía, se rompe el bucle `while`.
5. La última línea escribe los datos recibidos en el archivo de texto utilizando el método `write()` del objeto de archivo `f`. Primero, se llama al método `decode()` en la cadena `data` para convertir los datos recibidos en una cadena. Luego, se agrega un salto de línea (`'\n'`) al final de la cadena para separar cada conjunto de datos en una nueva línea en el archivo.

```
with open('contrasenas.txt', 'w') as f:
    while True:
        data = s.recv(1024)
        if not data:
            break
        f.write(data.decode() + '\n')
```

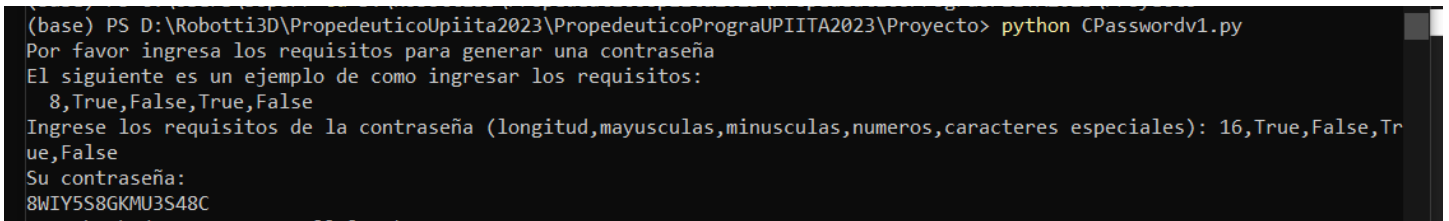
Resultados:

Se presentan capturas de pantalla con el código funcionando.



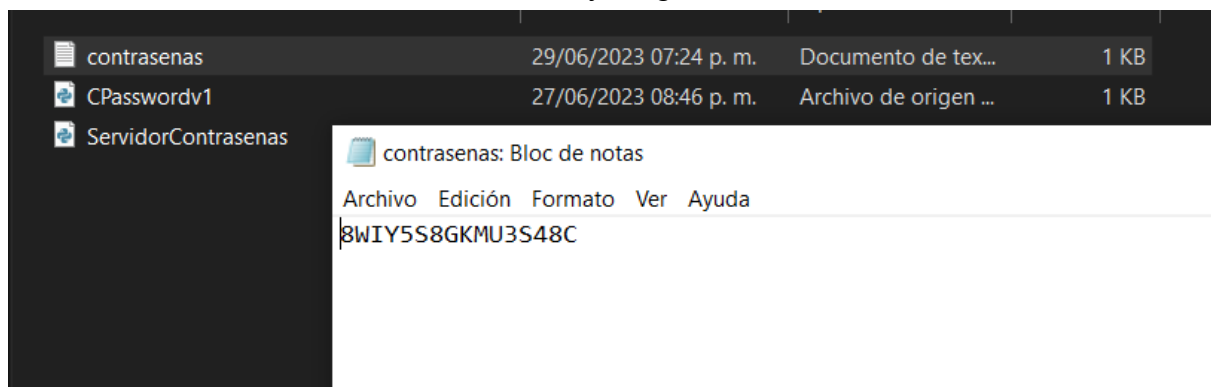
```
Anaconda PowerShell Prompt (miniconda3)
(base) PS C:\Users\sopor> cd D:\Robotti3D\PropedeuticoUpiita2023\PropedeuticoPrograUPIITA2023\Proyecto
(base) PS D:\Robotti3D\PropedeuticoUpiita2023\PropedeuticoPrograUPIITA2023\Proyecto> python ServidorContrasenas.py
Conexión establecida por ('127.0.0.1', 56784)
```

Se inicia el ServidorContrasenas.py



```
(base) PS D:\Robotti3D\PropedeuticoUpiita2023\PropedeuticoPrograUPIITA2023\Proyecto> python CPasswordv1.py
Por favor ingresa los requisitos para generar una contraseña
El siguiente es un ejemplo de como ingresar los requisitos:
8,True,False,True,False
Ingresa los requisitos de la contraseña (longitud,mayusculas,minusculas,numeros,caracteres especiales): 16,True,False,True,False
Su contraseña:
8WIY5S8GKMU3S48C
```

Se ejecuta el código del cliente CPasswordv1.py, Se muestra un ejemplo de cómo el usuario tiene que ingresar los requisitos de la contraseña segura. Se solicitó una contraseña de longitud de 16 caracteres, con mayúsculas y números. Se obtiene esta contraseña: 8WIY5S8GKMU3S48C y se guarda en un block de notas.



Para ver los códigos completos y finales revisar el repositorio en GitHub:

<https://github.com/GusRojas/PropedeuticoPrograUPIITA2023/tree/main/Proyecto>

Conclusión

Se lograron los objetivos del proyecto, haciendo uso de lo visto en clases, uso de listas, cadenas, sockets, condicionales y definir funciones. En el cliente al ya tener como un error sobre un objeto no lo detecta como socket, y eso pasa al cerrar el puerto sin embargo no afecta al resultado de la obtención de la contraseña y el salvarla en un block de notas de manera automática se puede conectar otro cliente al servidor y solicitar su contraseña sin problemas.

Gracias por el curso propedéutico, fueron enriquecedoras las pláticas y los comentarios sobre entrar a una maestría y de investigación, los pros y contras. Yo en lo personal agradezco la atención y lo que me hizo pensar bien las cosas, de mi parte doy gracias a todos los docentes pero mi decisión es no seguir en el proceso de ingreso al posgrado en UPIITA. Realizaré el ingreso en futuros meses a otra institución ya que mi objetivo no cambia, el realizar una maestría, solo cambio la institución. No es un adiós es un hasta luego y como dice la frase “Arrieros somos y en el camino andamos”. Muchas gracias y hasta pronto.

Fuentes:

Python. (2021). socket — Low-level networking interface. Recuperado el 29 de junio de 2023, de <https://docs.python.org/3/library/socket.html>

Python Software Foundation. (2023). Random — Generate pseudo-random numbers — Python 3.10.0 documentation. Recuperado el 29 de junio de 2023, de <https://docs.python.org/3/library/random.html>

Python Software Foundation. (2023). Módulo string - Constantes de cadenas de caracteres. Recuperado el 29 de junio de 2023, de <https://docs.python.org/es/3/library/string.html>