

Algoritmo de generación de contraseñas seguras en Python

Karlos Díaz

Mayo de 2025

1 Descripción del algoritmo:

Este algoritmo desarrollado y publicado por el usuario [chandramauliguptach](#) en el portal online [Geeks For Geeks](#) demuestra una manera simple y básica para generar una contraseña segura haciendo uso del lenguaje de programación **Python**. Para lograr este objetivo, el código genera aleatoriamente **letras, números y caracteres especiales**, para posteriormente almacenarlos en una variable que será retornada al usuario.

2 Análisis de código:

Para lograr comprender el funcionamiento de este algoritmo, es necesario analizar el código línea por línea, notaremos que es posible desglosar el algoritmo en una serie de pasos:

1. Importar librerías:

```
1 import string
2 import random
```

Estas líneas de código se encargaran de importar las librerías necesarias. *string* para acceder a los distintos caracteres ASCII y *random* para seleccionar aleatoriamente entre estos.

2. Almacenar los distintos caracteres en listas:

```
3 s1 = list(string.ascii_lowercase)
4 s2 = list(string.ascii_uppercase)
5 s3 = list(string.digits)
6 s4 = list(string.punctuation)
```

Se crean 4 variables dedicadas a almacenar los distintos tipos de variables: **ascii_lowercase** para letras minúsculas, **ascii_uppercase** para letras mayúsculas, **ascii_digits** para números y **ascii_punctuation** para caracteres especiales.

3. Solicitar longitud de la contraseña:

```
7 while True:
8     try:
9         user_input = input("How many characters do you want
in your password? ")
10        characters_number = int(user_input)
11        if characters_number < 8:
12            print("Your number should be at least 8.")
13        else:
14            break
15    except:
16        print("Please, Enter numbers only.")
```

Estas líneas de código le permiten al usuario elegir la cantidad de caracteres deseados en la contraseña, además se asegura que la cantidad **mínima** ingresada sea de 8 caracteres y que el input sea **numérico**.

4. Mezclar las listas de caracteres

```
17 random.shuffle(s1)
18 random.shuffle(s2)
19 random.shuffle(s3)
20 random.shuffle(s4)
```

Se utiliza la función **random.shuffle()** para aleatorizar las listas en búsqueda de mayor seguridad.

5. Calcular distribución de caracteres

```
21 part1 = round(characters_number * (30/100))
22 part2 = round(characters_number * (20/100))
```

se crean dos variables: **part1** para almacenar el 30% y **part2** para el 20% de la longitud de caracteres

6. Generar la contraseña

```
23 result = []
24 for x in range(part1):
25     result.append(s1[x])
26     result.append(s2[x])
27
28 for x in range(part2):
29     result.append(s3[x])
30     result.append(s4[x])
```

Se crea la lista **result** para almacenar la contraseña generada, luego con 2 bucles **for** se van agregando los caracteres haciendo uso de la función **.append()**, luego, las variables **part1** y **part2** sirven para distribuir en un 30% letras y un 20% números y caracteres especiales.

7. Mezclar y mostrar resultado final

```
31 random.shuffle(result)
32 password = "".join(result)
33 print("Strong Password:", password)
```

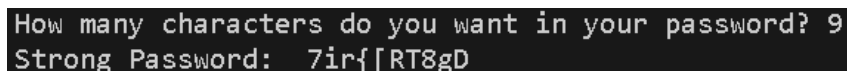
Se vuelve a utilizar la función `random.shuffle()` pero esta vez en el resultado final para romper el patrón definido en los bucles `for`. Finalmente la contraseña generada se transforma a **string** mediante la función `.join()` y se imprime en la consola de el usuario

3 Ventajas del Algoritmo:

- Longitud mínima de contraseña segura
- Mezcla de 4 tipos diferentes de caracteres
- aleatorización de caracteres

4 Desventajas del Algoritmo:

- Sin límite máximo de caracteres
- No se verifica la repetición de caracteres
- En el paso 5, la función `.round()` en algunos casos puede generar problemas y hacer que el algoritmo retorne una contraseña con una longitud mayor o menor a la solicitada debido al modo en que se redondean los números decimales.



```
How many characters do you want in your password? 9
Strong Password: 7ir{[RT8gD
```

Figure 1: Código retorna una contraseña de 10 caracteres cuando el usuario pide 9

- Vulnerabilidades en la función `.random()`, desde la documentación de python se nos avisa que esta función no debe ser utilizada para fines de seguridad.

5 Conclusión:

Si bien este algoritmo puede ser usado para uso educativo, no es recomendado para uso personal ni comercial, debido a la presencia de **vulnerabilidades** tanto en las propias librerías de **Python** como en el código en si, dejando al usuario expuesto a ataques informáticos.

6 Bibliografía:

- GeeksforGeeks. (2023, 10 enero). *Create a Random Password Generator using Python*. GeeksforGeeks. <https://www.geeksforgeeks.org/create-a-random-password-generator-using-python/>
- *Python 3.13 documentation*. (s. f.). Python Documentation. <https://docs.python.org/3/>