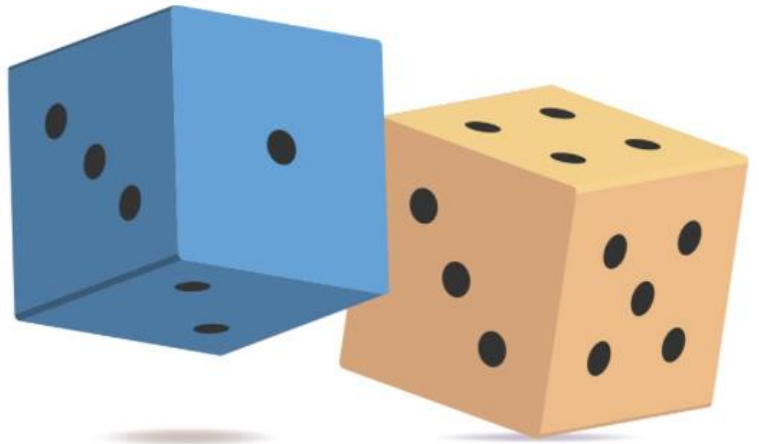
PROYECTO FINAL PROGRAMACION 1

Omar Jose Torrez Moscoso

2025-12-4

LIBRERIAS RAMDOM

PYTHON
RANDOM
MODULE



DATETIME


datetime
dateutil
YYYY MM DD
YEAR MONTH DAY
HH MM SS
HOUR MINUTE SEC
UNIX EPOCH
ISO 8601
COUNTDOWN
08 : 12 : 27
HOUR MINUTE SEC
Real Python

IPYTHON

MÓDULO 1 – CIFRADO Y DESCIFRADO

```python
def encriptar_cesar(texto, desplazamiento = 1):
    resultado = ""
    for letra in texto:
        if 'a' <= letra <= 'z':  # Minúsculas
            inicio = ord('a')
            caracter_cifrado = chr((ord(letra) - inicio + desplazamiento) % 26 + inicio)
        elif 'A' <= letra <= 'Z':  # Mayúsculas
            inicio = ord('A')
            caracter_cifrado = chr((ord(letra) - inicio + desplazamiento) % 26 + inicio)
        else:  # Otros caracteres (números, símbolos, espacios) se mantienen sin cambios
            caracter_cifrado = letra
        resultado += caracter_cifrado
    return resultado
def desencriptar_cesar(texto, desplazamiento = 1):
    resultado = ""
    for letra in texto:
        if 'a' <= letra <= 'z':  # Minúsculas
            inicio = ord('a')
            caracter_cifrado = chr((ord(letra) - inicio - desplazamiento) % 26 + inicio)
        elif 'A' <= letra <= 'Z':  # Mayúsculas
            inicio = ord('A')
            caracter_cifrado = chr((ord(letra) - inicio - desplazamiento) % 26 + inicio)
        else:  # Otros caracteres (números, símbolos, espacios) se mantienen sin cambios
            caracter_cifrado = letra
        resultado += caracter_cifrado
    return resultado
```

- **Objetivo del módulo**
- Implementar un **método de cifrado reversible** que permita:
- Guardar contraseñas cifradas en disco
- Evitar exposición directa de contraseñas
- Cumplir el requisito C del proyecto:
- **¿Por qué este módulo existe?**
- Porque **todo el flujo del sistema depende de cifrar y descifrar contraseñas** antes de guardarlas o mostrarlas:
- La **contraseña maestra** se guarda cifrada.
- Las **contraseñas de servicios** se almacenan cifradas.
- Cada vez que el usuario consulta y elige “mostrar”, se deben **descifrar**.

¿Cómo funciona?

El cifrado usa:

- ord() → convierte una letra en número ASCII
- chr() → convierte el número de vuelta en letra
- Fórmula de desplazamiento circular % 26 para no salir del alfabeto.

Conexiones con otros módulos

Módulo que lo llama	¿Para qué?
Gestión de contraseñas	guarda cifrado y lee descifrado
Contraseña maestra	validar acceso
Revisión de integridad	detectar registros sin cifrar

MÓDULO 2 – CONTRASEÑA MAESTRA

```
password = input("Ingrese la contraseña: ")

password_segura = encriptar_cesar(password)

with open("contrasena.txt", "w") as file:
    file.write(password_segura)

print("Contraseña guardada en 'contrasena.txt'")
```

```
Ingrese la contraseña: contrasena
Contraseña guardada en 'contrasena.txt'
```

```
def verificar_contrasena(texto):
  with open("contrasena.txt", "r") as archivo:
    contrasena_encriptada = archivo.read().strip()
  contrasena_desencriptada = desencriptar_cesar(contrasena_encriptada)
  return texto == contrasena_desencriptada

```

- **Objetivo**
- Proteger el acceso al sistema utilizando:
- Archivo de persistencia (contrasena.txt)
  Validación del usuario
  Cifrado obligatorio
- **Justificación técnica**
- La contraseña maestra debe:
- Ser cifrada → evitar exposición
- Persistir en un archivo → dar continuidad al sistema
- Ser verificada → un usuario no puede entrar sin validación
- Este módulo implementa **seguridad mínima real**, alineado con lo que LastPass o KeePass trabajan.

MÓDULO 3 – VALIDACIÓN DE OPCIONES

```python
def verificar_opciones(opcion, opciones_validas, tipo):
  if len(opciones_validas) != 0:
    while not isinstance(opcion, tipo) or opcion not in opciones_validas:
      try:
        opcion = tipo(opcion)
        if opcion not in opciones_validas:
          raise ValueError
      except ValueError:
        print(f"Valor invalido, Intente nuevamente.")
        opcion = input("Ingrese una opción: ")
  else:
    while not isinstance(opcion, tipo):
      try:
        opcion = tipo(opcion)
        if isinstance(opcion, int) or isinstance(opcion, float):
          if opcion < 0:
            raise ValueError
      except ValueError:
        print(f"Valor invalido, Intente nuevamente.")
        opcion = input("Ingrese una opción: ")
  return opcion
```

Objetivo

Evitar errores del usuario cuando ingresa opciones incorrectas:

- Conversión de tipos
  Reintentos
  Validación dentro de rangos permitidos
  Garantizar integridad de la interacción
- **¿Por qué es importante?**
- Un menú sin validación puede:
- Romper el programa
- Corromper archivos
- Aplicar acciones equivocadas (ej. borrar otro registro)
- Este módulo centraliza toda la **lógica defensiva** del sistema.
- Es un ejemplo claro de **diseño modular descendente** (EC1): toda la validación ocurre en un solo lugar.

MÓDULO 4 – GESTOR DE CONTRASEÑAS

```python
import time
def agregar_contrasena(servicio, usuario, contrasena):
  fecha = time.strftime("%Y-%m-%d")
  while "," in servicio or "," in usuario or "," in contrasena:
    print("No se permiten comas en los campos de servicio, usuario o contraseña.")
    servicio = input("Ingrese el servicio: ")
    usuario = input("Ingrese el usuario: ")
    contrasena = input("Ingrese la contraseña: ")
  with open("contrasenas.txt", "a") as archivo: # Cambiado 'w' a 'a' para añadir al final
    archivo.write(f"{servicio},{usuario},{encriptar_cesar(contrasena)},{fecha}\n")
```

```python
agregar_contrasena("Gmail", "Usuario1", "contraseña")
agregar_contrasena("Discord", "Usuario1", "programacion")
agregar_contrasena("Steam", "Usuario1", "matematica")
agregar_contrasena("Microsoft", "Usuario1", "probabilidades")
agregar_contrasena("Amazon", "Usuario1", "Peliculas")
```

```python
def consultar_contrasenas(desencriptar = False):
  with open("contrasenas.txt", "r") as archivo:
    lineas = archivo.readlines()
    print("Servicio | Usuario | Contraseña | fecha ")
    for linea in lineas[1:]:
      datos = linea.strip().split(",")
      if desencriptar:
        print(f"{datos[0]}, {datos[1]}, {desencriptar_cesar(datos[2])}, {datos[3]}")
      else:
```

MODULO 4

```python
def consultar_contrasenas(desencriptar = False):
  with open("contrasenas.txt", "r") as archivo:
    lineas = archivo.readlines()
    print("Servicio | Usuario | Contraseña | fecha ")
    for linea in lineas[1:]:
      datos = linea.strip().split(",")
      if desencriptar:
        print(f"{datos[0]}, {datos[1]}, {desencriptar_cesar(datos[2])}, {datos[3]}")
      else:
        print(f"{datos[0]}, {datos[1]}, {datos[2]}, {datos[3]}")
    print("1.- Editar contraseña")
    print("2.- Eliminar contraseña")
    print("3.- Salir")
    opcion = input("Ingrese una opción: ")
    opcion = verificar_opciones(opcion, [1, 2, 3], int)
    if opcion == 1:
      clear_output()
      print("Servicio | Usuario | Contraseña | fecha ")
      i = 0
      for linea in lineas[1:]:
        i += 1
        datos = linea.strip().split(",")
        print(f"{i}. {datos[0]}, {datos[1]}, {datos[2]}, {datos[3]}")
      fila = input("Ingrese el número de la contraseña a editar: ")
      fila = verificar_opciones(fila, range(1, i+1), int)
      nueva_contrasena = input("Ingrese la nueva contraseña: ")
      editar_contrasena(fila, nueva_contrasena)
      registrar_acciones("Modificada contraseña", f"para '{datos[0]}'")
    if opcion == 2:
      clear_output()
      print("Servicio | Usuario | Contraseña | fecha ")
```

MODULO 4

```
43        return
```

```
1 def editar_contrasena(fila, contrasena):
2   with open("contrasenas.txt", "r") as archivo:
3     lineas = archivo.readlines()
4   datos = lineas[fila].strip().split(",") # Use strip() to remove existing newline
5   datos[2] = encriptar_cesar(contrasena)
6   lineas[fila] = ",".join(datos) + "\n" # Add newline back
7   with open("contrasenas.txt", "w") as archivo:
8     archivo.writelines(lineas)
```

```
1 def eliminar_contrasena(fila):
2   with open("contrasenas.txt", "r") as archivo:
3     lineas = archivo.readlines()
4   lineas.pop(fila)
5   with open("contrasenas.txt", "w") as archivo:
6     archivo.writelines(lineas)
```

- Manejar el archivo contrasenas.txt que actúa como la **base de datos** del sistema.
- **Qué hace cada una:**

Función	Rol
agregar_contrasena()	Añade una contraseña cifrada con fecha
consultar_contrasenas()	Lista, descifra opcionalmente, abre menú CRUD
editar_contrasena()	Reemplaza la contraseña cifrada
eliminar_contrasena()	Borra la fila completa

Conexión con requisitos

Requisito	Cómo se cumple
B – Gestión	CRUD completo
C – Cifrado	Todas las contraseñas se guardan cifradas
G – Log	Cada operación se registra (en tu menú llamas registrar_acciones)

MÓDULO 5 – ANALIZADOR DE FUERZA DE CONTRASEÑAS

```python
import re

def analizar_fuerza_contrasena(contrasena):
    puntuacion = 0

    longitud = len(contrasena)
    if longitud < 8:
        puntuacion += 0
    elif 8 <= longitud < 12:
        puntuacion += 1
    else:
        puntuacion += 2

    if len(re.findall(r"[A-Z]", contrasena)) < 2:
        puntuacion += 0
    elif 2 <= len(re.findall(r"[A-Z]", contrasena)) < 4:
        puntuacion += 1
    else:
        puntuacion += 2

    if len(re.findall(r"\d", contrasena)) < 2:
        puntuacion += 0
    elif 2 <= len(re.findall(r"\d", contrasena)) < 4:
        puntuacion += 1
    else:
        puntuacion += 2

    if len(re.findall(r"[^a-zA-Z0-9\s]", contrasena)) < 1:
        puntuacion += 0
    elif 1 <= len(re.findall(r"[^a-zA-Z0-9\s]", contrasena)) < 3:
        puntuacion += 1
    else:
```

¿Por qué usamos re?

Porque permite:

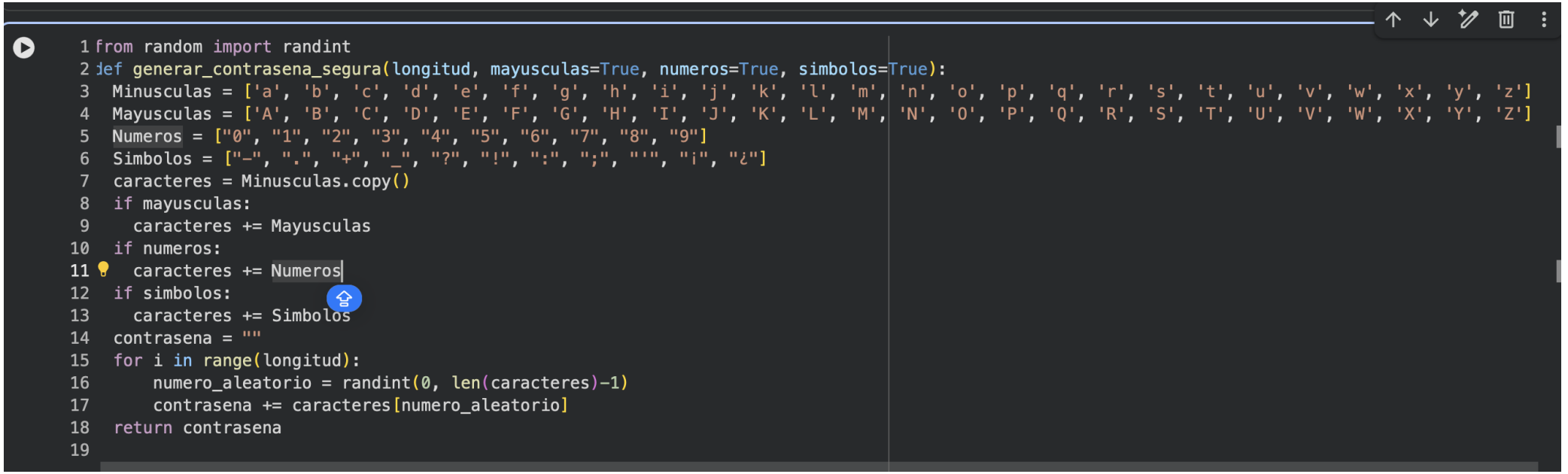
- ✓ Buscar patrones complejos
  ✓ Detectar mayúsculas [A-Z]
  ✓ Detectar números \d
  ✓ Detectar símbolos no alfanuméricos [^a-zA-Z0-9\s]
  ✓ Detectar patrones prohibidos (“password”, “123”, etc.)
- Sin expresiones regulares, tendrías que:
- Revisar carácter por carácter
- Implementar reglas manuales
- Hacer el código poco legible
- re te da **poder compacto y profesional**.

Rol del módulo

- Evaluar la calidad de una contraseña con puntaje:

Longitud
Mayúsculas
Números
Símbolos
Patrones prohibidos

MÓDULO 6 – GENERADOR DE CONTRASEÑAS



```python
from random import randint
def generar_contrasena_segura(longitud, mayusculas=True, numeros=True, simbolos=True):
  Minusculas = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
  Mayusculas = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
  Numeros = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"]
  Simbolos = ["-", ".", "+", "_", "?", "!", ":", ";", "'", "¡", "¿"]
  caracteres = Minusculas.copy()
  if mayusculas:
    caracteres += Mayusculas
  if numeros:
    caracteres += Numeros
  if simbolos:
    caracteres += Simbolos
  contrasena = ""
  for i in range(longitud):
      numero_aleatorio = randint(0, len(caracteres)-1)
      contrasena += caracteres[numero_aleatorio]
  return contrasena

```

EXPLICACION

- **¿Por qué usamos randint?**
- Para seleccionar **índices aleatorios** dentro de los arreglos de caracteres.
- 🎯 **Objetivo del módulo**
- Crear contraseñas seguras basadas en:
- Longitud
- Mayúsculas opcionales
- Números opcionales
- Símbolos opcionales
- Arreglos de caracteres → cumple EC2

MÓDULO 7 – BÚSQUEDA RECURSIVA

```python
def buscar_texto_recursivo(texto_principal, texto_a_buscar):
  if not texto_a_buscar:
    return True
  if len(texto_principal) < len(texto_a_buscar):
    return False
  if texto_principal.startswith(texto_a_buscar):
    return True
  return buscar_texto_recursivo(texto_principal[1:], texto_a_buscar)
```

```python
from datetime import datetime
def revisar_integridad_recursiva(lineas, index, lineas_validas, conteo_reparos):
    if index >= len(lineas):
        return lineas_validas, conteo_reparos

    linea = lineas[index]
    lineastrip = linea.strip()
    problema_hallado = False
    descripcion = ""
    servicio = "desconocido"

    if not lineastrip:
        problema_hallado = True
        descripcion = "Registro vacío encontrado y omitido."
    else:
        parts = lineastrip.split(',')
        if len(parts) != 4:
            problema_hallado = True
            descripcion = "Entrada incompleta o formato no reconocido (número de campos incorrecto)."
        else:
            servicio, usuario, contrasena, fecha = parts

            patrones_prohibidos = [
                "password", "123456", "qwerty", "asdfgh", "123", "abc"
            ]

            tiene_patron_prohibido = False
            # Attempt to decrypt for pattern check, if decryption fails, this will be caught later
            try:
                contrasena_minusculas = desencriptar_cesar(contrasena).lower()
            except:
                contrasena_minusculas = contrasena.lower() # Fallback if not decryptable
```

OBJETIVO

- Implementar una búsqueda **sin usar ciclos**, solo mediante:
- Casos base
  Llamados recursivos
  Corte por longitud
- Este módulo cumple explícitamente el requisito:
- “Debe existir una búsqueda implementada de manera recursiva”

¿Qué técnica usa?

- Sliding window recursiva:

1. Compara el inicio del texto
2. Si no coincide → descarta primer carácter
3. Llama recursivamente al resto del texto
4. Hasta que encuentra o se agota