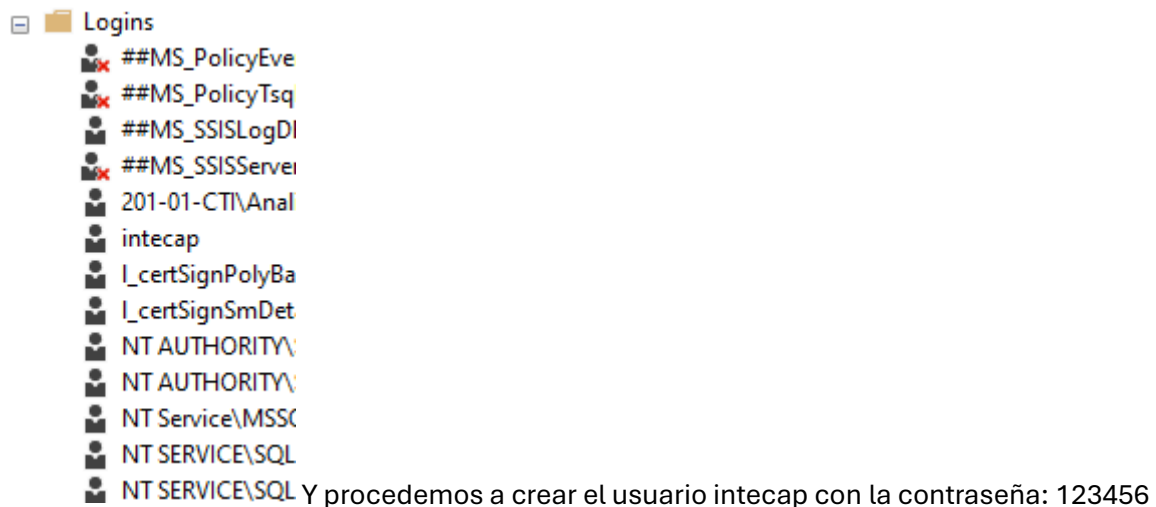


1. Configuración de la base de datos:

- Configurar un sistema de base de datos (por ejemplo, MSSQL, MongoDB) para almacenar la información de los estudiantes.
- Defina la estructura de las tablas de la base de datos para almacenar los datos de los estudiantes, incluidos campos como el ID del estudiante, el nombre, la edad y cualquier otra información relevante.

```
CREATE TABLE [dbo].[ESTUDIANTES](  
    [CODIGO] [int] NULL,  
    [NOMBRE] [varchar](max) NULL,  
    [ENCARGADO] [varchar](max) NULL,  
    [GRADO_ACA] [varchar](max) NULL  
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]  
GO
```

Creamos la tabla de estudiantes con sus respectivos atributos para que funcione la inserción de datos en la DB de control estudiantes



Y procedemos a crear el usuario intecap con la contraseña: 123456

para no complicarse.



También procedemos a crear el documento en mundo de control de estudiantes, y la colección de estudiantes también donde se almacenará los datos del programa

2. Desarrollo de código Python:

- Escribir código Python para establecer una conexión con la base de datos.
- Implementar funciones para manejar el registro, la búsqueda, la actualización y la eliminación de estudiantes dentro de la base de datos utilizando Python.

```
#definir la conexion para mongo
cliente = MongoClient('mongodb://localhost:27017/')
db = cliente['Control_Estudiantes']

estudiantes_col = db['estudiantes']
notas_col = db['notas']

#definir la conexion para sql
conn_str = (
    r'DRIVER={ODBC Driver 17 for SQL Server};'
    r'SERVER=201-01-CTI;'
    r'DATABASE=CONTROL_ESTUDIANTES;'
    r'UID=intecap;'
    r'PWD=123456;'
)

conect = sql.connect(conn_str)
cursor = conect.cursor()
```

Menu:

1. Registro de estudiantes
2. Control de notas
3. Mostrar datos
4. Salir

Este sería el menú principal del programa desde la consola el cual muestra un registro de estudiantes, control de notas, y mostrar datos.

```
Ingrese su opción: 1

Opciones:
1. Crear estudiante
2. Buscar estudiante
3. Actualizar
4. Eliminar
5. Regresar
```

Haremos una demostración

con la opción de crear estudiantes:

```
Ingrese una opción: 1
Ingrese el código de estudiante: 004
Ingrese su nombre: Deisel
Ingrese su encargado: Erick
Ingrese su nivel académico: Universidad
Datos guardados en 'estudiantes.txt'
```

select * from ESTUDIANTES

	CODIGO	NOMBRE	ENCARGADO	GRADO_ACA
1	4	Deisel	Erick	Universidad

```
_id: "004"
codigo : "004"
encargo : "Universidad"
grado : "Erick"
nombre : "Deisel"
```

```
control de estudiantes.py  notas.txt

004|Deisel|Universidad|Erick
```

```
Código | Nombre | Grado | Encargado
-----
004    | Deisel | Universidad | Erick
```

Esto ultimo seria lo

que muestra el programa al momento de darle la opción de mostrar los datos

```

Ingrese una opción: 3
Ingrese el código del estudiante que desea actualizar: 004
Datos actuales:
Código: 004
Nombre: Deisel
Grado: Universidad
Encargo: Erick
Ingrese el nuevo nombre (deje en blanco para no cambiar): Juan
Ingrese el nuevo grado (deje en blanco para no cambiar): Doctor
Ingrese el nuevo encargo (deje en blanco para no cambiar): Pepe
Datos actualizados con éxito.
Datos guardados en 'estudiantes.txt'

```

Aquí procedemos a actualizar los datos que vemos en pantalla

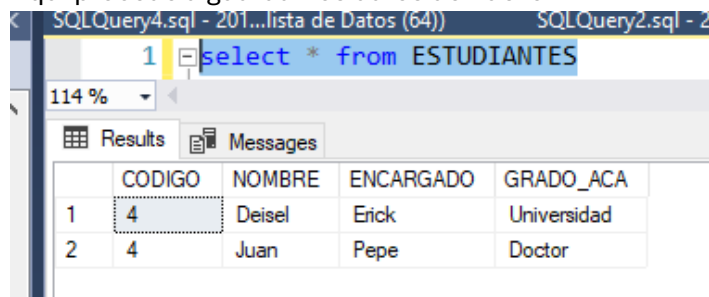
```

Ingrese su opción: 3
Estos son sus datos:

Código | Nombre | Grado | Encargado
-----
004    | Juan   | Doctor | Pepe

```

Aquí procede a guardar los datos de nuevo:



The screenshot shows a SQL query window with the following SQL statement: `select * from ESTUDIANTES`. The results are displayed in a table with the following columns: CODIGO, NOMBRE, ENCARGADO, and GRADO_ACA. The results show two rows of data.

	CODIGO	NOMBRE	ENCARGADO	GRADO_ACA
1	4	Deisel	Erick	Universidad
2	4	Juan	Pepe	Doctor

```

_id: "004"
codigo : "004"
encargo : "Doctor"
grado : "Pepe"
nombre : "Juan"

```

3. Creación del manual del programador:

- Documentar el código Python y su funcionalidad en un manual de programador.
- Incluya explicaciones detalladas, fragmentos de código y ejemplos que demuestren cómo usar el sistema de registro de estudiantes e interactuar con la base de datos.

Por cuestión de simplicidad mostrare solo los def que contiene el código porque son 370 líneas de código entonces para no explicar tantas:

```
def main():  
    estudiantes = cargar_datos_txt() # Carga  
    notas = carga_notas() # Cargar notas al  
  
    while True:  
        print(''  
            Menu:  
            1. Registro de estudiantes  
            2. Control de notas  
            3. Mostrar datos  
            4. Salir  
        '' )
```

Tendríamos nuestro def main el cual mostrara las opciones de menú general y el de estudiantes

```
elif opcion == '4':  
    print('Auto destrucción activada\n T-')  
    intervalo = 1  
    for i in range(10, 0, -1):  
        print(i)  
        time.sleep(intervalo)  
    print('¡¡Boom!!')  
    break
```

Al finalizar se mostrará una pequeña broma donde mostrar un conteo regresivo simulando una autodestrucción

```

1 usage
def estudiante():
    codigo = input("Ingrese el código de estudiante: ")
    nombre = input("Ingrese su nombre: ")
    encargo = input("Ingrese su encargado: ")
    grado = input("Ingrese su nivel académico: ")
    return codigo, nombre, encargo, grado

```

Este es el def de estudiante el cual se encarga de solicitar los datos de los estudiantes

```

def print_estudiantes(estudiantes):
    if not estudiantes:
        print("No hay estudiantes registrados.")
        return

    ancho_codigo = max(len('Código'), max(len(est['codigo']) for est in estudiantes))
    ancho_nombre = max(len('Nombre'), max(len(est['nombre']) for est in estudiantes))
    ancho_grado = max(len('Grado'), max(len(est['grado']) for est in estudiantes))
    ancho_encargo = max(len('Encargado'), max(len(est['encargo']) for est in estudiantes))

    # Encabezado
    print(
        f'{"Código".ljust(ancho_codigo)} | {"Nombre".ljust(ancho_nombre)} | {"Grado".ljust(ancho_grado)} | {"Encargado".ljust(ancho_encargo)}')
    print('-' * (ancho_codigo + ancho_nombre + ancho_grado + ancho_encargo + 9))

    for est in estudiantes:
        print(
            f'{est["codigo"].ljust(ancho_codigo)} | {est["nombre"].ljust(ancho_nombre)} | {est["grado"].ljust(ancho_grado)} | {est["encargo"].ljust(ancho_encargo)}')

    print()

```

Este def imprime los estudiantes que están almacenados, y con un poco de programación le agregamos algo de estética

```

def buscar(estudiantes):
    campo_busqueda = input("Ingrese el campo en el que desea buscar (codigo, nombre, grado): ").strip().lower()
    valor_busqueda = input(f"Ingrese el valor para buscar en el campo '{campo_busqueda}': ").strip()

    if campo_busqueda not in ['codigo', 'nombre', 'grado']:
        print("Campo de búsqueda no válido.")
        return

    encontrados = False
    for est in estudiantes:
        if est.get(campo_busqueda, "").lower() == valor_busqueda.lower():
            print(
                f'{est["codigo"].ljust(10)} | {est["nombre"].ljust(20)} | {est["grado"].ljust(10)} | {est["encargo"].ljust(15)}')
            encontrados = True

    if not encontrados:
        print("No se encontraron resultados.")

```

Este def va a buscar los estudiantes que solicite el usuario y se encarga de buscar ya sea por nombre, código, grado o estudiante

```

def actualizar_estudiante(estudiantes):
    codigo_actualizar = input("Ingrese el código del estudiante que desea actualizar: ").strip()

    estudiante_encontrado = None
    for est in estudiantes:
        if est['codigo'] == codigo_actualizar:
            estudiante_encontrado = est
            break

    if estudiante_encontrado:
        print("Datos actuales:")
        print(f"Código: {estudiante_encontrado['codigo']}")
        print(f"Nombre: {estudiante_encontrado['nombre']}")
        print(f"Grado: {estudiante_encontrado['grado']}")
        print(f"Encargo: {estudiante_encontrado['encargo']}")

        nuevo_nombre = input("Ingrese el nuevo nombre (deje en blanco para no cambiar): ").strip()
        nuevo_grado = input("Ingrese el nuevo grado (deje en blanco para no cambiar): ").strip()
        nuevo_encargo = input("Ingrese el nuevo encargo (deje en blanco para no cambiar): ").strip()

        if nuevo_nombre:
            estudiante_encontrado['nombre'] = nuevo_nombre
        if nuevo_grado:
            estudiante_encontrado['grado'] = nuevo_grado
        if nuevo_encargo:
            estudiante_encontrado['encargo'] = nuevo_encargo

        print("Datos actualizados con éxito.")
        guardar_datos_txt(estudiantes)
    else:
        print("Código del estudiante no encontrado.")

```

El def actualizar estudiante, pues hace lo que dice, actualiza los estudiantes

```

def eliminar(estudiantes):
    print_estudiantes(estudiantes)

    codigo_eliminar = input("Código del estud

    index_to_remove = None
    for i, registro in enumerate(estudiantes):
        if registro['codigo'] == codigo_elimi

```

El def eliminar pues

elimina los estudiantes y por medio de su código

```
def guardar_datos_txt(estudiantes):
    conect = sql.connect(conn_str)
    cursor = conect.cursor()

    try:
```

este def guarda los datos en txt, y
sql

```
1 usage
def cargar_datos_txt():
    estudiantes = []
    try:
        with open('estudiantes.txt', 'r') as f:
            for line in f:
                partes = line.strip().split
                # Asegúrate de que haya exa
```

También tenemos lo
de cargar datos lo cual cumple la función de almacenar los datos guardados y los retiene

```
1 usage
def estudiantes_mongo(codigo, nombre, encargo, grado):
    estudiante = {
        '_id': codigo,
        'nombre': nombre,
```

Esto almacena los datos en mongoDB

```
    guardar_datos_txt(estudiantes, cursor, conect)
    estudiantes_mongo(codigo, nombre, encargo, grado)
elif estu_op == '2':
    buscar(estudiantes)
elif estu_op == '3':
    actualizar_estudiante(estudiantes)
elif estu_op == '4':
```

Aquí
mando a las funciones guardar datos, estudiantes mongo, buscar, y actualizar las cuales
estarían alojadas dentro del menú principal


```

1 usage
def menu_notas(estudiantes, notas):
    while True:
        print(f'''
            Menu de opciones:
            1. Ingresar notas
            2. Mostrar notas
            3. Modificar notas
            4. Eliminar notas
            5. Salida
            ''')

```

Y por último tenemos el menú de notas el cual hace exactamente lo mismo que el de estudiante y tiene las mismas funciones, a continuación, una demostración:

```

Opción a escoger: 1
Ingrese el código del estudiante para ingresar las notas: 004
Ingrese la nota para la unidad 1: 80
Ingrese la nota para la unidad 2: 87
Ingrese la nota para la unidad 3: 97
Ingrese la nota para la unidad 4: 64
Datos guardados en 'notas.txt'

```

```

Notas del estudiante con código 004:
Unidad 1: 80
Unidad 2: 87
Unidad 3: 97
Unidad 4: 64
Promedio de 004: 82.00
-----

```

También la configuración del promedio se hizo de la siguiente forma:

```
def print_notas(notas):
    if not notas:
        print("No hay notas registradas")
        return

    for codigo, nt in notas.items():
        print(f'Notas del estudiante con código {codigo}:')
        print(f'Unidad 1: {nt["unidad_1"]}')
        print(f'Unidad 2: {nt["unidad_2"]}')
        print(f'Unidad 3: {nt["unidad_3"]}')
        print(f'Unidad 4: {nt["unidad_4"]}')
        promedio = sum(n for n in nt.values() if n is not None) / len([n for n in nt.values() if n is not None])
        print(f'Promedio de {codigo}: {promedio:.2f}')
    print('-' * 40)
```

Imprime las notas primero, las de las 4 unidades y luego de eso en la variable promedio almacena las notas las cuales hace el calculo y por ultimo las imprime

```
notas_op == '1':
    codigo = input("Ingrese el código del estudiante para ingresar las notas: ").strip()
    if any(est['codigo'] == codigo for est in estudiantes):
        unidad_1, unidad_2, unidad_3, unidad_4 = nota()
        notas[codigo] = {
            'unidad_1': unidad_1,
            'unidad_2': unidad_2,
            'unidad_3': unidad_3,
            'unidad_4': unidad_4
        }
        guardar_notas(notas)
    else:
        print("Código de estudiante no encontrado.")
if notas_op == '2':
    print_notas(notas)
```

Aquí almacenamos las notas por código de estudiante, este será solicitado antes de ingresar las notas que saco

```
        guardar_notas(notas)
    else:
        print("Código de estudiante no encontrado.")
elif notas_op == '2':
    print_notas(notas)
```

También mandamos a llamar el guardar notas el cual guardara las notas. También podemos ver el print notas el cual enseña las notas que tiene cada estudiante