Pontificie Universidad
Católica del Ecuador

Materia:
Programación II

Semestre:
Tercero “A”

Nombre:
Victor Toasa

Tema:
Proyecto final

Ingeniero:
Edison Fernando Meneses Torres

Proyecto Integrador: Sistema de Gestión Escolar de Emergencia

En este proyecto se mostrará el proyecto final de la materia de programación II, en donde se explicará todo el funcionamiento del código realizado.

Primer código:

Gestión de reportes médicos (pilas)

Lo primero que se ha realizado es el apartado para generar un reporte médico de los estudiantes mediante el uso de pilas y listas enlazadas.

```
class nodo_reportes_medicos:
    def __init__(self,nombre, apellido,tipo, observacion):
        self.tipo=tipo
        self.observacion=observacion
        self.nombre=nombre
        self.apellido=apellido
        self.next=None
class reportes_paciente:
    def __init__(self):
        self.top=None
    def agregar_reportes(self,nombre,apellido, tipo, observacion):
        nuevo_nodo=nodo_reportes_medicos(nombre,apellido, tipo, observacion)
        nuevo_nodo.next=self.top
        self.top=nuevo_nodo
```

En la primera parte de este código, encontraremos las clases principales, con las cuales trabajaremos, en este caso, el tipo de enlace que será usando en el simple. La clase “nodo_reportes_medicos”, es la forma en la que se crea el nodo para guardar en nuestra lista de enlace simple.

Mientras que la segunda clase “reportes_paciente” se guardara todas las acciones que realizara este código.

Como primera parte, tenemos una función el cual será nuestro constructor de la clase, seguida de esta, se muestra otra función en donde se guardará los nuevos nodos mediante el enlace simple, el cual, a su vez al ser una pila, respetamos la estructura LIFO (último en entrar, primero en salir).

```python
def elimnar_reporye(self):
    if self.top==None:
        print("No hay Datos dentro de la lista de registros")
    else:
        opc4=input("Desea borrar el primero registro de la lista? ingrese esas opciones: Si o No: ").lower().strip()
        while opc4!="no" and opc4!="si":
            print("Ingrese una opcion valida")
            opc4=input("Desea borrar el primero registro de la lista? ingrese esas opciones: Si o No: ").lower().strip()

        if opc4=="si":
            plato_elimar=self.top
            print(f"""
                El registro:
                Nombre: {plato_elimar.nombre}
                Fecha: {plato_elimar.apellido}
                Tipo de enfermedad: {plato_elimar.tipo}
                Sintomas presentados: {plato_elimar.observacion}

                Ha sido eliminado de la lista
                """)
            self.top=plato_elimar.next
        if opc4=="no":
            print("Volviendo al inicio")
            opc5=input("Precione enter para coninuar")
            return
```

Continuando con el código, encontramos un apartado en el cual, vamos a eliminar nodos (registros médicos).

Este código, se va encargar de verificar si la lista en donde estamos guardando los nodos, no este vacía, y pueda cumplir con sus funciones correctamente, del mismo modo este cuenta con una condición para que, al momento de borrar algún nodo, se confirme esta decisión.

Para su eliminación, se cambiar el puntero “self.top” para que ya no apunte a él, sino al siguiente elemento. Seguido de esto la liberación de memoria es un proceso automático de Python que ocurre después, cuando ya no hay referencias al nodo desconectado.

```python
def mostrar_registro(self):
    actual=self.top
    print("Reporte Medicos de los Estudiante")
    while actual is not None:
        print(f"""
                \n
                ----------------------------------------------------
                Nombre: {actual.nombre}
                Fecha: {actual.apellido}
                Tipo de enfermedad: {actual.tipo}
                Sintomas presentados: {actual.observacion}
                ----------------------------------------------------
                """)
        actual=actual.next
```

Para mostrar el contenido guardado en las listas simples, se usa un ciclo el cual recorre nuestra lista, mostrando el contenido de la forma LIFO, dando detalles de como esta compuesto el nodo o nuestros registros,

```
    def ingreso_tareas(self):
        try:
            opc=int(input("\nIngrese la opcion a realizar: "))
        except ValueError:
            print("Valores incorrecto solo numeros")
            return self.ingreso_tareas()
        while opc>4 or opc<1 :
            print("Ingrese solo valores del 1 al 4")
            try:
                opc=int(input("Ingrese la opcion a realizar : "))
            except ValueError:
                print("Valores incorrecto solo numeros")
        return opc
    def continar(self):
        opc2=input("Precione enter para continuar: ")
```

Para terminar la parte de las clases, se ha integrado una función para seleccionar un numero entre 1 y 4, esto con el fin de usarlo en el menú que se realizó, este consiste en envolvernos en un bucle en el caso de ingresar un número que no corresponde a la limitación dada.

```
clase_usar=reportes_paciente()

while True:
    print("""
          GESTOR DE REPORTES MEDICOS
          1. INGRESO NUEVO PACIENTE
          2. ELIMNAR REPROTE
          3. VER TODOS LOS REPORTES
          4. SALIR
          """)
    opc3=clase_usar.ingreso_tareas()

    if opc3==1:
        nombre_paciente=input("Ingrese el nombre del estudiante: ")
        apelludo_pacinete=input("Ingresar el apellido del estudiante: ")
        tipo_paciente=input("Ingrese el tipo de enfermedado problema de salud que presente el estudiante: ")
        descripcion_paciente=input("Ingrese los sintomas que ha tenido durante las ultimas 24 horas: ")
        clase_usar.agregar_reportes(nombre_paciente, apelludo_pacinete,tipo_paciente,descripcion_paciente)
        clase_usar.continar()
    if opc3==2:
        clase_usar.elimnar_reporye()
    if opc3==3:
        print("Mostrando todo los reportes meducos de los estudiantes")
        clase_usar.mostrar_registro()
        clase_usar.continar()
    if opc3==4:
        print("Saliendo del programa")
        break
```

Al terminar con las clases y sus funcionamientos principales, encontramos el menú de opciones, el cual nos ayudara a controlar todo el código, para esta parte lo primero que se debe realizar es llamar la clase en donde guardamos todas las funciones.

La primera opción que nos permite realizar este código, es el registro del estudiante, y el guardado de sus datos.

La segunda opción nos permite realizar la eliminación de los registros o nodo.

La tercera opción, ayuda a mostrar todos los registros ingresados.

Y como cuarta opción, tenemos la salida del programa, con el fin de terminar el programa cuando ya no necesitamos usarlo.

Segundo código:
Simulación de evacuación escolar (colas)
Para la segunda parte del proyecto, se ha creado una simulación de evacuación de estudiantes cuando exista una emergencia, usando las colas para su forma de guardado.

```
import time

class EstudianteEvacuacion:
    def __init__(self, nombre, aula, id_estudiante):
        self.nombre = nombre
        self.aula = aula
        self.id_estudiante=id_estudiante
        self.siguiente=None

class ColaDeAtencionSimple:
    def __init__(self):
        self.cabeza = None
        self.cola = None

    def esta_vacia(self):
        return self.cabeza is None
```

Para esta segunda parte del proyecto, de la misma forma, se crea dos clases, el primero ayuda a la creación del nodo, y el segundo nos ayuda con las funciones principales del programa, aparte de este se ha usado una librería el cual es “time”, con el fin de simular la evacuación. Al observar esta parte tenemos la primera función el cual nos ayuda a saber si la lista está o no vacía.

```
def insertar(self, nombre, aula, id_estudiante):
    nuevo_nodo = EstudianteEvacuacion(nombre, aula, id_estudiante)
    if self.cabeza is None:
        self.cabeza = nuevo_nodo
        self.cola = nuevo_nodo
    else:
        self.cola.siguiente = nuevo_nodo
        self.cola = nuevo_nodo
```

Para realizar el guardado de los datos de nuestro nodo, usaremos la misma lógica de antes, pero en este caso un poco diferente, dado a que en este caso se trabaja con colas, y para esta ocasión se usara el principio FIFO (primero en entrar primero en salir).

El funcionamiento del código, se basa en ir aguardando el nodo al final de la lista en el caso de tener la lista con elemento, en el caso de tener una lista vacía, el primero nodo, simplemente se guarda y se vuelve el principal.

```
def avanzar(self):
    atendido=self.cabeza
    if self.cabeza is None:
        print("--- La cola está vacía. No hay nadie para atender. ---")
        return None
    while atendido is not None:
        print("Se simulara una evaciacion")
        print(f"""
            El estudiante: {atendido.nombre}
            Del Aula: {atendido.aula}
            Con su ID: {atendido.id_estudiante}
            {time.sleep(1)}
            Ha sido correctamente evacuados, continuando con los siguientes
            """)

        atendido=atendido.siguiente
```

Esta segunda función, nos permite recorrer la lista y este es el que nos ayudará a simular la evacuación, mostrando el primer estudiante evacuado, así hasta el final, con el único detalle que se tendrá un segundo para continuar con el siguiente estúdiate.

De esta forma al pasar por el primer estudiante evacuado cambiara el nodo principal y continuara con el siguiente, de este modo hasta llegar al final.

```
def mostrar_en_espera(self):
    actual = self.cabeza
    if self.cabeza is None:
        print("La cola de atención está vacía.")
        return



    while actual is not None:
        print("Los estudiantes que estan en espera son los siquientes ")
        print(f"""

          El estudiante: {actual.nombre}
          Del Aula: {actual.aula}
          Con su ID: {actual.id_estudiante}
          ---------------------------------------
          """)
        actual=actual.siguiente
```

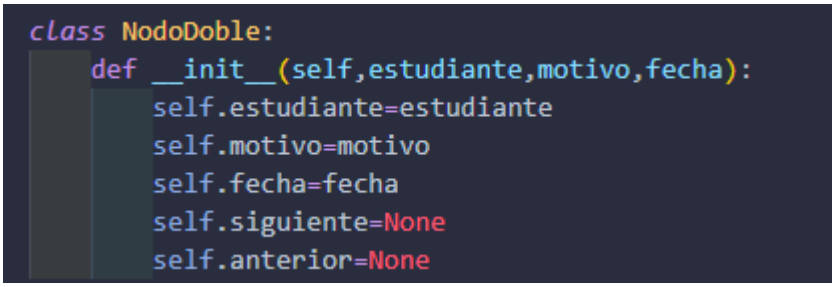
Por último, tenemos la función que no permite mostrar todos los datos de todos los estudiantes, que se han registrado hasta ese momento.

Esta parte se ha usado la misma lógica del anterior código.

```
cola_usar=ColaDeAtencionSimple()

while True:
        print("""
========= MENÚ EVACUACION =========
1. Agregar Estudiantes
2. Simular evacuacion
3. Mostrar los estudiantes, en esta simulacion
4. Sali""")
        opcion = input("Seleccione una opción: ").strip().lower()
        if opcion == "1":
            nombre_estudiante= input("Ingrese el nombre del estudiante: ")
            aula_estudiante=input("Ingrese el aula del estudiante: ")
            id_estudiante=input("Ingrese el id del estudiante: ")
            cola_usar.insertar(nombre_estudiante,aula_estudiante,id_estudiante)
        if opcion=="2":
            cola_usar.avanzar()
        if opcion=="3":
            cola_usar.mostrar_en_espera()
        if opcion=="4":
            print("Saleindo.........")
            time.sleep(2)
            break
```

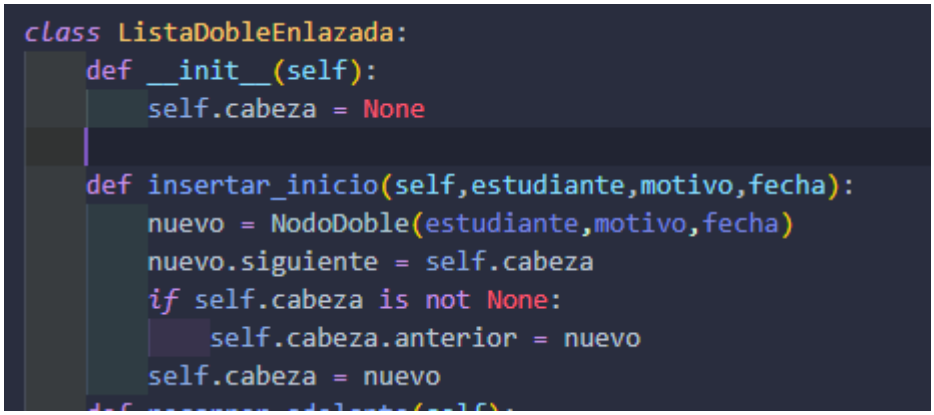
Finalmente tenemos su menú, el cual conta de 4 opciones:
El primero se usa para ingresar y guardar los datos que se ingresaran por teclado.
El segundo se usará para, simular la evacuación.
El tercero tiene el fin de mostrar todos sus elementos.
Y el cuarto tendrá solo la función de terminar el programa con un retraso de dos segundos.

Tercer código:
Registro de Visitas Médicas (LISTA DOBLE)
Para este tercer parte se usará listas dobles para nuestro código, con el fin de realizar un tipo de navegación entre registros médicos, con el fin de recorrer hacia adelante y hacia atrás.



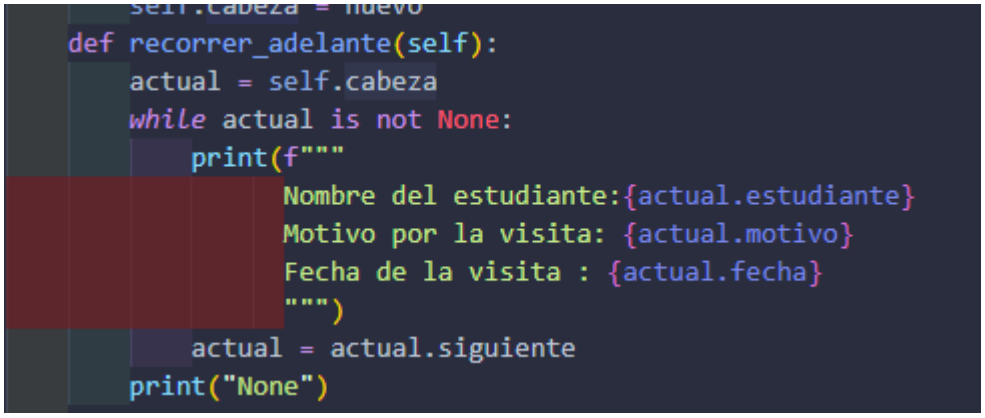
```
class NodoDoble:
    def __init__(self,estudiante,motivo,fecha):
        self.estudiante=estudiante
        self.motivo=motivo
        self.fecha=fecha
        self.siguiente=None
        self.anterior=None
```

En esta clase define la estructura básica de un nodo que forma parte de una lista doblemente enlazada. Cada nodo guarda la información relacionada con un estudiante: su nombre, el motivo de su visita médica y la fecha en que ocurrió. Además, cuenta con dos referencias o enlaces: uno hacia el nodo siguiente (siguiente) y otro hacia el nodo anterior (anterior), lo que permite recorrer la lista tanto hacia adelante como hacia atrás. Esta clase actúa como el "bloque de construcción" principal de la lista.



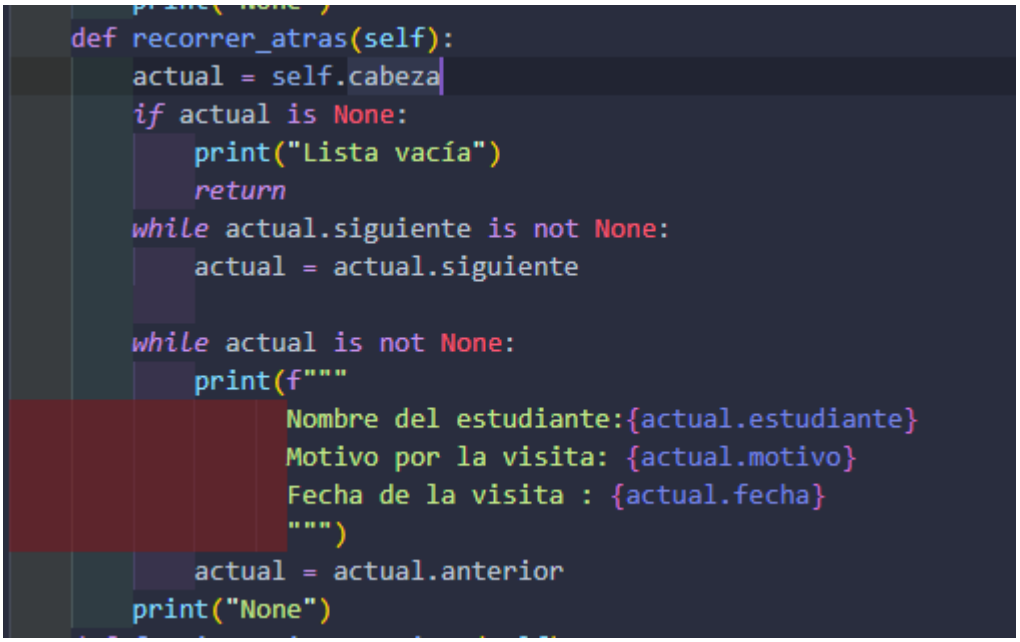
```
class ListaDobleEnlazada:
    def __init__(self):
        self.cabeza = None

    def insertar_inicio(self,estudiante,motivo,fecha):
        nuevo = NodoDoble(estudiante,motivo,fecha)
        nuevo.siguiente = self.cabeza
        if self.cabeza is not None:
            self.cabeza.anterior = nuevo
        self.cabeza = nuevo
```

Para la siguiente función, permite insertar un nuevo nodo al inicio de la lista. Primero se crea un nodo nuevo con los datos proporcionados del estudiante. Luego, se enlaza ese nuevo nodo al actual primer nodo de la lista, si existe, y se actualiza su referencia anterior para que apunte hacia el nuevo nodo. Finalmente, se actualiza la cabeza de la lista para que apunte al nodo recién insertado. Así, el nuevo nodo pasa a ser el primero de la lista y la estructura se mantiene conectada correctamente en ambas direcciones.



```
    def recorrer_adelante(self):
        actual = self.cabeza
        while actual is not None:
            print(f"""
                  Nombre del estudiante:{actual.estudiante}
                  Motivo por la visita: {actual.motivo}
                  Fecha de la visita : {actual.fecha}
                  """)
            actual = actual.siguiente
        print("None")
```

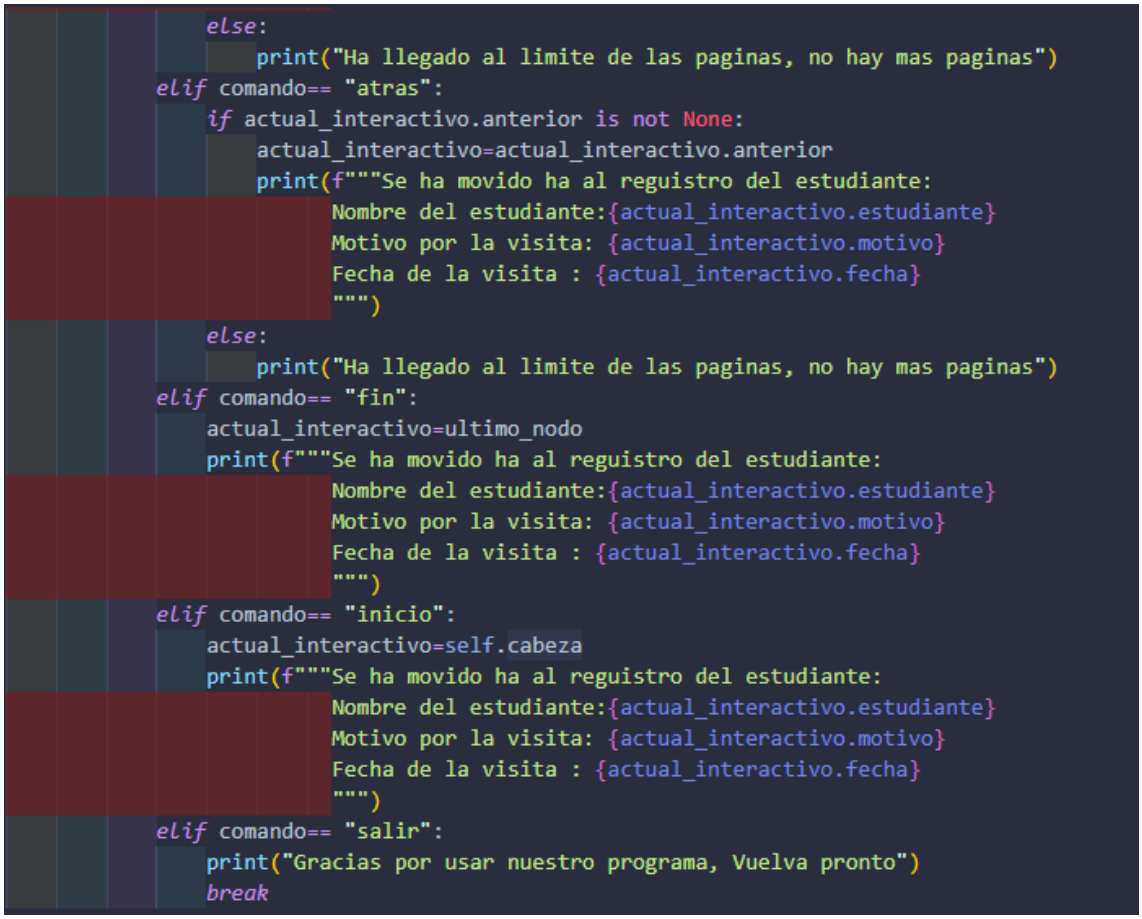
Esta función permite recorrer e imprimir todos los registros almacenados en la lista doblemente enlazada desde el principio hasta el final. Comienza desde la cabeza de la lista y avanza nodo por nodo utilizando el enlace siguiente, mostrando en cada paso la información correspondiente del estudiante. El recorrido termina cuando se llega a un nodo cuyo valor siguiente es None, lo que indica el final de la lista.



```
    def recorrer_atras(self):
        actual = self.cabeza
        if actual is None:
            print("Lista vacía")
            return
        while actual.siguiente is not None:
            actual = actual.siguiente

        while actual is not None:
            print(f"""
                  Nombre del estudiante:{actual.estudiante}
                  Motivo por la visita: {actual.motivo}
                  Fecha de la visita : {actual.fecha}
                  """)
            actual = actual.anterior
        print("None")
```

Continuando con esta función, permite recorrer la lista en sentido inverso, es decir, desde el último nodo hasta el primero. Para ello, primero se avanza desde la cabeza hasta el final de la lista (siguiendo los enlaces siguiente), y una vez ahí, se comienza a retroceder utilizando los enlaces anterior. Durante este recorrido hacia atrás se imprime la información contenida en cada nodo. Esta función es útil cuando se quiere mostrar los registros en orden cronológico desde el más antiguo hasta el más reciente.

```python
def funciones_interactivas(self):
    if self.cabeza is None:
        print("""La lista esata vacia, primero ingrese nuevas registro\n""")
        return
    actual_interactivo=self.cabeza
    ultimo_nodo=self.cabeza

    if ultimo_nodo is not None:
        while ultimo_nodo.siguiente is not None:
            ultimo_nodo=ultimo_nodo.siguiente

    print(f"""El primer registro en el que se encuentra es:
          Nombre del estudiante:{actual_interactivo.estudiante}
          Motivo por la visita: {actual_interactivo.motivo}
          Fecha de la visita : {actual_interactivo.fecha}

          """)

    while True:

        comando=input("Ingrese la accion que desea realizar: ").lower().strip()

        if comando=="adelante":
            if actual_interactivo.siguiente is not None:
                actual_interactivo=actual_interactivo.siguiente
                print(f"""Se ha movido ha al reguistro del estudiante:
                      Nombre del estudiante:{actual_interactivo.estudiante}
                      Motivo por la visita: {actual_interactivo.motivo}
                      Fecha de la visita : {actual_interactivo.fecha}
                      """)
            else:
```



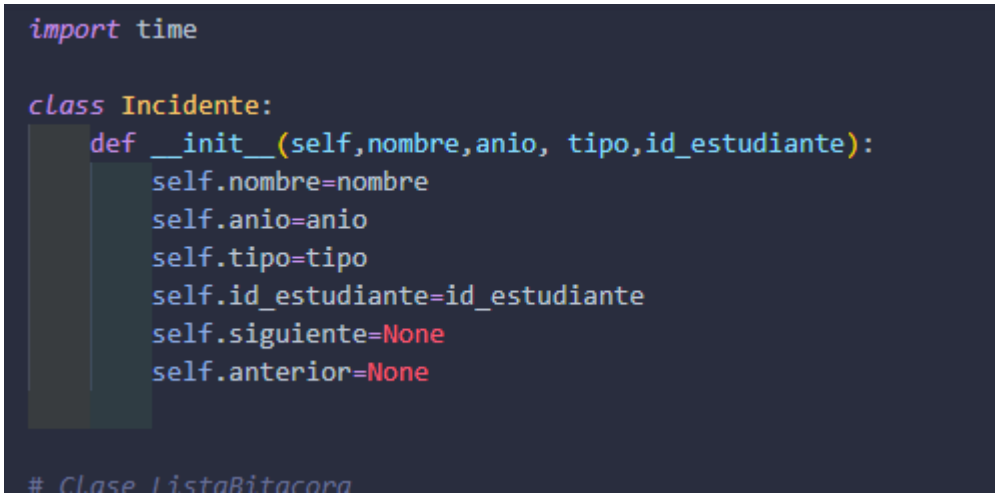
```
            else:
                print("Ha llegado al limite de las paginas, no hay mas paginas")
        elif comando== "atras":
            if actual_interactivo.anterior is not None:
                actual_interactivo=actual_interactivo.anterior
                print(f"""Se ha movido ha al reguistro del estudiante:
                      Nombre del estudiante:{actual_interactivo.estudiante}
                      Motivo por la visita: {actual_interactivo.motivo}
                      Fecha de la visita : {actual_interactivo.fecha}
                      """)
            else:
                print("Ha llegado al limite de las paginas, no hay mas paginas")
        elif comando== "fin":
            actual_interactivo=ultimo_nodo
            print(f"""Se ha movido ha al reguistro del estudiante:
                      Nombre del estudiante:{actual_interactivo.estudiante}
                      Motivo por la visita: {actual_interactivo.motivo}
                      Fecha de la visita : {actual_interactivo.fecha}
                      """)
        elif comando== "inicio":
            actual_interactivo=self.cabeza
            print(f"""Se ha movido ha al reguistro del estudiante:
                      Nombre del estudiante:{actual_interactivo.estudiante}
                      Motivo por la visita: {actual_interactivo.motivo}
                      Fecha de la visita : {actual_interactivo.fecha}
                      """)
        elif comando== "salir":
            print("Gracias por usar nuestro programa, Vuelva pronto")
            break
```

Esta función ofrece una navegación interactiva por los registros de la lista doblemente enlazada. Primero verifica si hay registros; si no los hay, se detiene. En caso contrario, muestra el primer registro y permite al usuario desplazarse con comandos. El comando "adelante" avanza al siguiente nodo, "atras" retrocede al nodo anterior, "inicio" regresa al primer nodo, y "fin" va al último. El comando "salir" termina la navegación. Esta función convierte la lista en una especie de visor interactivo, ideal para explorar los registros paso a paso sin verlos todos de golpe.

```
lista=ListaDobleEnlazada()

while True:

    print("""Bienvenidos al programa interactivo\n
          1. Ingreso de registro medico
          2. Comenzar a navegar entre registros
          3. Ver registro completo de estudiantes
          4. Salir
            """)

    opc=input("Ingrese una opcion a realizar: ")

    if opc=="1":
        nombre_estudiante=input("Ingrese el nombre del estudiante: ")
        motivos_visita=input("Ingrese el motivo de la visita: ")
        fecha_visita=input("Ingrese la fecha de la visita DD/MM/AA: ")
        lista.insertar_inicio(nombre_estudiante,motivos_visita,fecha_visita)
        print("Se ha guardado con exito")
    if opc=="2":
        print("A continuacion, navegara por la paginas ingresadas")
        print("Comando que se peuden usar: adelante; atras; inicio; fin;\n")
        lista.funciones_interactivas()
    if opc=="3":
        print("Su historias de navegacion completa es: ")
        print("\nListado completo ")
        lista.recorrer_adelante()
    if opc=="4":
        print("Saliendo gracias por su uso")
        break
```

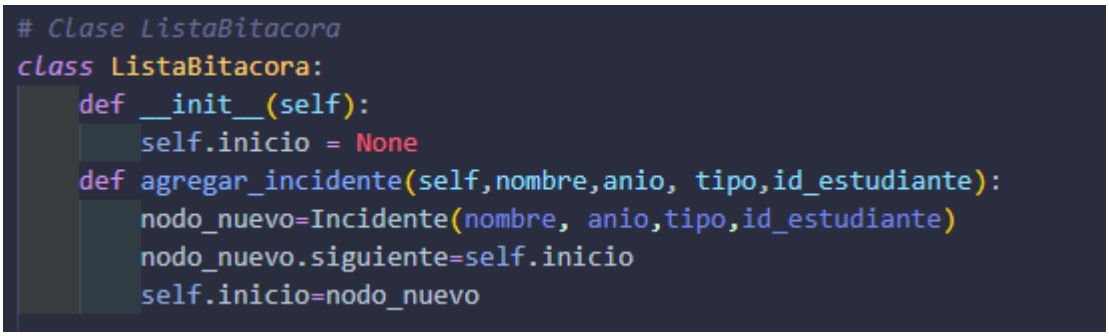
Este es el menú principal del programa, que permite al usuario interactuar con el sistema. Se presenta un menú con cuatro opciones dentro de un bucle para que el usuario pueda realizar varias acciones antes de salir.
La opción 1 permite registrar un nuevo estudiante, solicitando su nombre, el motivo de su visita y la fecha, y luego insertando el nodo en la lista.
La opción 2 abre la navegación interactiva entre los registros ya ingresados.
La opción 3 imprime todos los registros en orden desde el más reciente al más antiguo.
Finalmente, la opción 4 termina el programa con un mensaje de despedida.

Cuarto código:
Bitácora de Incidentes Escolares (LISTA SIMPLE)
Por último, presentamos el cuarto código, el cual consta esta enlazado mediante las listas simples, está siendo más fáciles de entender e integrar.



```
import time

class Incidente:
    def __init__(self,nombre,anio, tipo,id_estudiante):
        self.nombre=nombre
        self.anio=anio
        self.tipo=tipo
        self.id_estudiante=id_estudiante
        self.siguiente=None
        self.anterior=None


# Clase ListaBitacora
```

Esta clase define la estructura de un nodo que representa un incidente ocurrido a un estudiante. Cada nodo guarda el nombre del estudiante, el año del incidente, el tipo de incidente y su ID. También cuenta con dos punteros: uno hacia el nodo siguiente (siguiente) y otro hacia el anterior (anterior), aunque en el funcionamiento del código actual solo se usa el enlace siguiente. Esta estructura permite gestionar una lista enlazada donde cada nodo contiene la información de un incidente registrado.



```
# Clase ListaBitacora
class ListaBitacora:
    def __init__(self):
        self.inicio = None
    def agregar_incidente(self,nombre,anio, tipo,id_estudiante):
        nodo_nuevo=Incidente(nombre, anio,tipo,id_estudiante)
        nodo_nuevo.siguiente=self.inicio
        self.inicio=nodo_nuevo
```

La función permite agregar un nuevo incidente al inicio de la lista. Crea un nodo con los datos del estudiante y enlaza ese nodo de forma que apunte al actual primer nodo de la lista. Después, actualiza el puntero inicio para que el nuevo nodo se convierta en el primero. Esta forma de inserción mantiene los registros más recientes al comienzo de la lista.

```python
def eliminar_incidente(self, nombre_buscar):
    actual=self.inicio
    while actual and actual.nombre !=nombre_buscar:
        actual=actual.siguiente

    if actual is None:
        print("La lista esta vacia, primeor ingrese valores :D")
        return
    if actual.anterior is None:
        self.inicio=actual.siguiente
        if self.inicio:
            self.inicio.anterior=None
    else:
        actual.anterior.siguiente=actual.siguiente
        if actual.siguiente:
            actual.siguiente.anterior=actual.anterior
    print(f"El reguistro del estudiante; {nombre_buscar} ha sido eliminado ")
```

Mientras que esta función busca eliminar un incidente específico según el nombre del estudiante. Primero recorre la lista hasta encontrar el nodo cuyo nombre coincida con el ingresado. Si no se encuentra, se muestra un mensaje de advertencia. Si el nodo encontrado es el primero de la lista, simplemente se actualiza el puntero inicio. En caso de que esté en otra posición, se actualizan los enlaces del nodo anterior y del siguiente para que se salten al nodo eliminado. Al final, se informa que el incidente ha sido eliminado. Aunque la clase Incidente tiene puntero anterior, no se enlaza en agregar_incidente, por lo que esta función puede no funcionar correctamente con listas verdaderamente dobles.

```python
def buscar_incidente(self, nombre_encontrar):
    actual=self.inicio
    encontrado=False
    while actual:
        if actual.nombre==nombre_encontrar:
            print(f"""
                  Registro del estudiante buscado:
                  Nombre del estudiante:{actual.nombre}
                  Año del del reguistro :{actual.anio}
                  Tipo de incidente: {actual.anio}
                  Id del estudiante: {actual.id_estudiante}

                  """)
            encontrado=True
            break
        actual=actual.siguiente
    if not encontrado:
        print(f"El estudiante: {nombre_encontrar} no ha sido encontrado")
```

Esta función permite buscar un incidente en la lista usando el nombre del estudiante. Recorre la lista desde el inicio hasta encontrar el nodo que coincida con el nombre. Si lo encuentra, muestra todos los datos del incidente. Si no se encuentra ningún registro con ese nombre, se muestra un mensaje indicando que el estudiante no fue hallado. Hay un pequeño error en la impresión: se repite actual.anio donde debería ir actual.tipo.

```python
    def mostrar_incidentes(self):
        actual=self.inicio
        if self.inicio==None:
            print("La lista esta vacia, ingrese primero estudiantes")
            return

        while actual is not None:
            print("Los estudiantes que estan dentro de la lista para la evacuacion son los siguinetes: ")
            print(f"""
                  Nombre del estudiante: {actual.nombre}
                  ID del estudiante: {actual.id_estudiante}
                  Año del incidente: {actual.anio}
                  Tipo de incidente: {actual.tipo}
                  ----------------------------------------------
                  """)
            actual=actual.siguiente
```

Esta función muestra todos los incidentes registrados en la lista enlazada. Comienza desde el primer nodo (inicio) y recorre toda la lista utilizando el enlace siguiente. Por cada nodo encontrado, imprime la información del estudiante y del incidente. Si la lista está vacía, lo informa antes de intentar recorrerla.

```python
cola_usar=ListaBitacora()
while True:
        print("""
========= MENÚ EVACUACION =========
1. Agregar incidente
2. Eliminar incidente
3. Buscar indicente de los estudiantes
4. Mostrar todos los incidentes
5. Salir """)

        opcion = input("Seleccione una opción: ").strip().lower()
        if opcion == "1":
            nombre_estudiante= input("Ingrese el nombre del estudiante: ")
            anio_incidente =input("Ingrese el año que sucedio el incidente: ")
            id_estudiante=input("Ingrese el id del estudiante: ")
            tipo_incidente=input("INgrese el tipo de incidente que paso: ")
            cola_usar.agregar_incidente(nombre_estudiante,anio_incidente,tipo_incidente,id_estudiante)
        if opcion=="2":
            nombre_eliminar=input("Ingrese el nombre del estudiante que quiere eliminar: ").lower().strip()
            cola_usar.eliminar_incidente(nombre_eliminar)
        if opcion=="3":
            nombre_buscar=input("Ingrese el nombre del estudiante que quiero buscar")
            cola_usar.buscar_incidente(nombre_buscar)
        if opcion=="4":
            cola_usar.mostrar_incidentes()
        if opcion=="5":
            print("Saleindo.........")
            time.sleep(2)
            break
```

Este es el menú principal del programa que permite al usuario gestionar una bitácora de evacuación con los incidentes registrados de los estudiantes. A través de un ciclo “while True”, se muestran cinco opciones:
Agregar un nuevo incidente registrando el nombre, año, tipo e ID del estudiante.
Eliminar un incidente buscando por el nombre del estudiante.
Buscar un incidente específico usando el nombre del estudiante.
Mostrar todos los incidentes registrados hasta el momento.
Salir del programa, con una pequeña pausa de 2 segundos antes de finalizar.
El menú es interactivo y permite que el usuario realice múltiples operaciones antes de salir, ofreciendo una simulación básica de un sistema de control de incidentes para evacuaciones escolares.

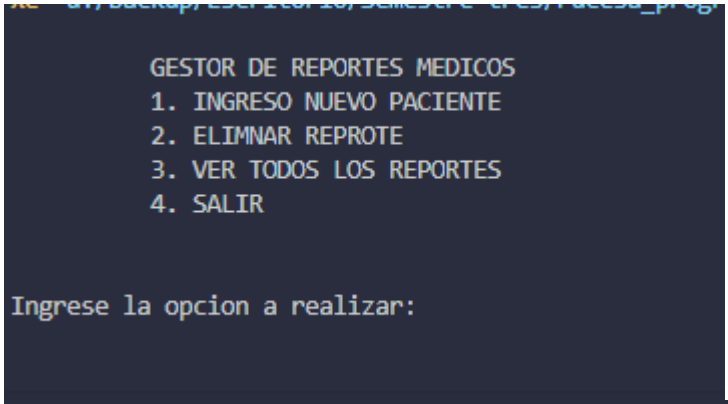
Para finalizar tenemos un menú global en el cual controla todas las acciones que se han presentado hasta el momento.

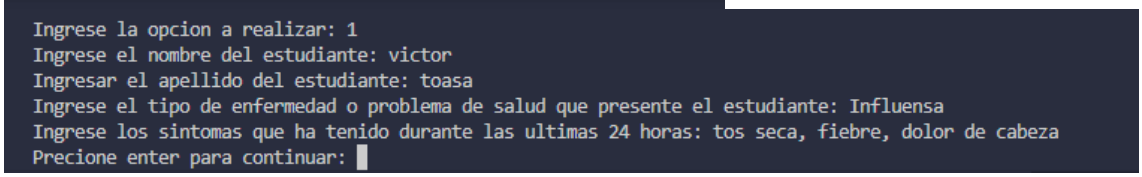
```
while True:
    print("""////Sistema de Gestión Escolar de Emergencia/////

        Bienvenidos a nuestro sistema de gestion
        en donde podremos realizar diferentes tipos de listados
        para los diferente tipos de problemas que puede suceder
        dentro de un colegio o escuela

        Las opciones que se pueden ralizar son los siguientes:
        1. Reportes medicos Estudiantes
        2. Evacuacion escolar ordenada
        3. Ver los registros medicos de los estudiantes
        4. Ver y almacenar bitacoras de incidentes escolares
        5. Salir
        """
        )

    opc=int(input("Ingrese una opcion que se desee realizar: "))

    while opc>=5 and opc<=1:
        print("Ingrese una opcion valida")
        opc=int(input("Ingrese una opcion que se desee realizar: "))


    clase_usar=reportes_paciente()
    cola_usar=ColaDeAtencionSimple()
    lista=ListaDobleEnlazada()
    cola_usar2=ListaBitacora()
```

En este apartado podremos encontrar todas las opciones que se pueden realizar, este es el menú es el que contiene a los demás sub menús, y debajo de este tenemos la condición que se usó, para poder validar la entrada por teclado

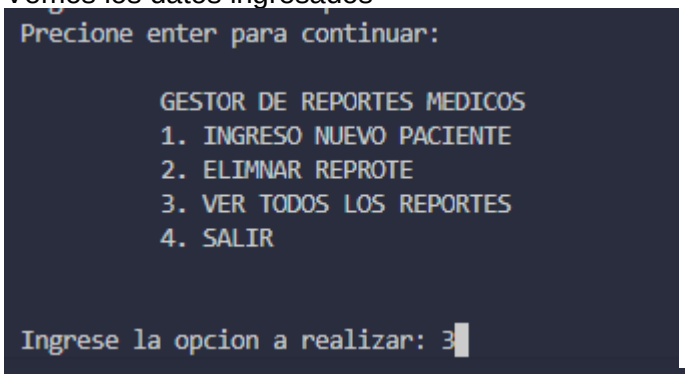
Debajo de este menú encontramos la clases que se han usado a lo largo de este proyecto, el cual se lo utilizara el so sub menús

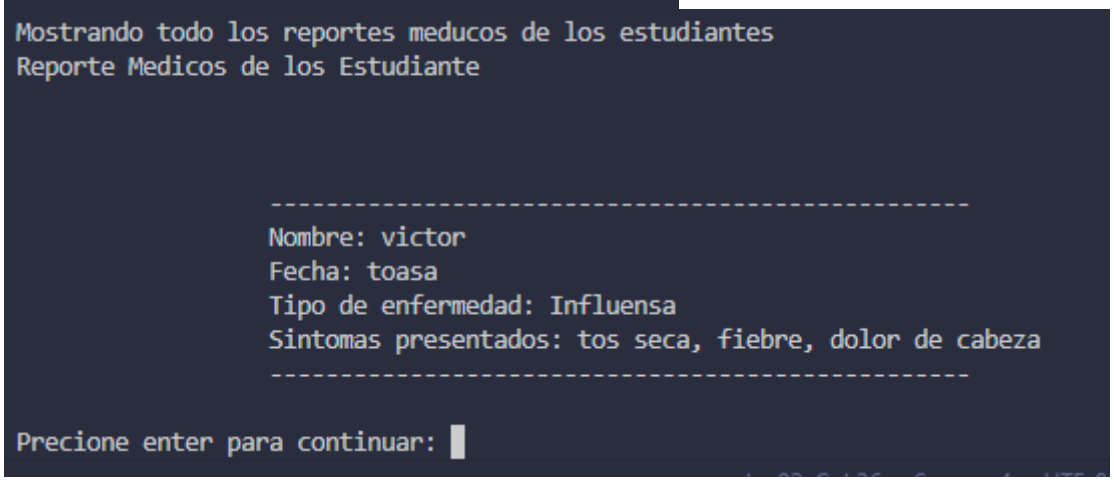
Ejecución del primer programa:
Primero ingresamos los datos





Vemos los datos ingresados





Eliminamos los datos

```
        GESTOR DE REPORTES MEDICOS
        1. INGRESO NUEVO PACIENTE
        2. ELIMNAR REPROTE
        3. VER TODOS LOS REPORTES
        4. SALIR


Ingrese la opcion a realizar: 2
Desea borrar el primero registro de la lista? ingrese esas opciones: Si o No:
```

```
Ingrese la opcion a realizar: 2
Desea borrar el primero registro de la lista? ingrese esas opciones: Si o No: SI

                El registro:
                Nombre: victor
                Fecha: toasa
                Tipo de enfermedad: Influensa
                Sintomas presentados: tos, fiebre, dolor de cabeza

                Ha sido eliminado de la lista
```

Ejecución del segundo programa:

Primero ingresamos los datos

```
1. Agregar Estudiantes
2. Simular evacuacion
3. Mostrar los estudiantes, en esta simulacion
4. Sali
Seleccione una opción: 1
Ingrese el nombre del estudiante: victor
Ingrese el aula del estudiante: 200
Ingrese el id del estudiante: 123456789
```

Mostramos los datos guardados

```
Seleccione una opción: 3
Los estudiantes que estan en espera son los siquientes


            El estudiante: victor
            Del Aula: 200
            Con su ID: 123456789
            ------------------------------------
```

Simulando la evacuación

```
Seleccione una opción: 2
Se simulara una evaciacion

               None
               El estudiante: victor
               Del Aula: 200
               Con su ID: 123456789
               None
               Ha sido correctamente evacuados, continuando con los siguientes

Se simulara una evaciacion


  Se simulara una evaciacion

                None
                El estudiante: sofia
                Del Aula: 201
                Con su ID: 0987654321
                None
                Ha sido correctamente evacuados, continuando con los siguientes


  ========= MENÚ EVACUACION =========
```

Ejecución del tercer programa:
Primero ingresamos los datos

```
         1. Ingreso de registro medico
         2. Comenzar a navegar entre registros
         3. Ver registro completo de estudiantes
         4. Salir

Ingrese una opcion a realizar: 1
Ingrese el nombre del estudiante: victor
Ingrese el motivo de la visita: Doleres en el cuerpo
Ingrese la fecha de la visita DD/MM/AA: 02/11/2025
Se ha guardado con exito
```

Al ingresar los datos podemos navegar entre los registros realizados ponemos usar los comandos que se mencionan para navegar entre ellos

```
  Ingrese una opcion a realizar: 2
  A continuacion, navegara por la paginas ingresadas
  Comando que se peuden usar: adelante; atras; inicio; fin;

  El primer registro en el que se encuentra es:
               Nombre del estudiante:jorge
               Motivo por la visita: Gripe
               Fecha de la visita : 13/02/2025


  Ingrese la accion que desea realizar:
                                                    Ln 31, Col 29   Spaces
```

```
Ingrese la accion que desea realizar: adelante
Se ha movido ha al reguistro del estudiante:
                        Nombre del estudiante:sofia
                        Motivo por la visita: Una lesion en el brazo
                        Fecha de la visita : 12/02/2025
```

```
Ingrese la accion que desea realizar: atras
Se ha movido ha al reguistro del estudiante:
                        Nombre del estudiante:jorge
                        Motivo por la visita: Gripe
                        Fecha de la visita : 13/02/2025
```

Ejecución del cuarto programa:
Primero ingresamos los datos

```
1. Agregar incidente
2. Eliminar incidente
3. Buscar indicente de los estudiantes
4. Mostrar todos los incidentes
5. Salir
Seleccione una opción: 1
Ingrese el nombre del estudiante: jorge
Ingrese el año que sucedio el incidente: 10/10/2025
Ingrese el id del estudiante: 0987654321
INgrese el tipo de incidente que paso: muy bullicioso dentro de las aulas
```

Mostramos todos datos ingresados

```
========= MENU EVACUACION =========
1. Agregar incidente
2. Eliminar incidente
3. Buscar indicente de los estudiantes
4. Mostrar todos los incidentes
5. Salir
Seleccione una opción: 4
Los estudiantes que estan dentro de la lista para la evacuacion son los siguinetes:

                Nombre del estudiante: jorge
                ID del estudiante: 0987654321
                Año del incidente: 10/10/2025
                Tipo de incidente: muy bullicioso dentro de las aulas
                ---------------------------------------------

Los estudiantes que estan dentro de la lista para la evacuacion son los siguinetes:

                Nombre del estudiante: victor
                ID del estudiante: 1234567890
                Año del incidente: peleas entre compañeros
                Tipo de incidente: falta de respectos
                ---------------------------------------------
```

Buscamos los datos que necesitamos mediante su nombre

```
========= MENÚ EVACUACION =========
1. Agregar incidente
2. Eliminar incidente
3. Buscar indicente de los estudiantes
4. Mostrar todos los incidentes
5. Salir
Seleccione una opción: 3
Ingrese el nombre del estudiante que quiero buscarvictor

                    Registro del estudiante buscado:
                    Nombre del estudiante:victor
                    Año del del reguistro :peleas entre compañeros
                    Tipo de incidente: peleas entre compañeros
                    Id del estudiante: 1234567890
```

Eliminamos datos de igual forma mediante su nombre

```
  ========= MENÚ EVACUACION =========
  1. Agregar incidente
  2. Eliminar incidente
  3. Buscar indicente de los estudiantes
  4. Mostrar todos los incidentes
  5. Salir
  Seleccione una opción: 2
  Ingrese el nombre del estudiante que quiere eliminar: jorgue
  La lista esta vacia, primeor ingrese valores :D
```

Menú principal

```
////Sistema de Gestión Escolar de Emergencia/////

        Bienvenidos a nuestro sistema de gestion
        en donde podremos realizar diferentes tipos de listados
        para los diferente tipos de problemas que puede suceder
        dentro de un colegio o escuela

        Las opciones que se pueden ralizar son los siguientes:
        1. Reportes medicos Estudiantes
        2. Evacuacion escolar ordenada
        3. Ver los registros medicos de los estudiantes
        4. Ver y almacenar bitacoras de incidentes escolares
        5. Salir

Ingrese una opcion que se desee realizar:
```