

**Carrera:**

Ingeniería en sistemas de la información cuarto semestre

**Materia:**

Programación IV

**Docente:**

Edison Fernando Meneses Torres

**Estudiante:**

Jeremy Jácome

**Tema:**

Ejercicio de Programación Orientada a Objetos en Python

**Fecha:**

25 de abril de 2025

**Contenido**

[**1.** **Introducción** 2](#_Toc196510229)

[**2.** **Desarrollo** 2](#_Toc196510230)

[2.1. **Estructura del Proyecto** 2](#_Toc196510231)

[2.2. **Diseño de la Clase Paciente** 2](#_Toc196510232)

[2.3. **Funcionamiento del Menú y Funciones** 3](#_Toc196510233)

[2.4. **Capturas de Pantalla** 4](#_Toc196510234)

[**3.** **Conclusiones** 13](#_Toc196510235)

[**4.** **Anexos** 13](#_Toc196510236)

[4.1. **Fragmentos de código relevantes** 13](#_Toc196510237)

[4.2. **Observaciones adicionales** 14](#_Toc196510238)

**Sistema de Gestión de Consultas Médicas**

1. **Introducción**

El presente proyecto tiene como objetivo desarrollar un sistema básico de gestión de pacientes y sus consultas médicas, utilizando el lenguaje de programación Python. Este sistema permite registrar pacientes, agregar consultas médicas a cada paciente, y visualizar tanto los datos individuales como el listado completo de los pacientes registrados.

El sistema fue desarrollado con fines académicos para reforzar conocimientos en programación estructurada, manipulación de datos en listas y diccionarios, validación de entradas, y diseño de menús interactivos por consola. Representa un ejemplo práctico de cómo organizar y estructurar un proyecto real usando funciones y clases en Python.

1. **Desarrollo**
   1. **Estructura del Proyecto**

El proyecto está compuesto por los siguientes archivos:

* **main.py o pacientes.py**: Contiene el menú principal del sistema. Desde aquí se llama a las funciones necesarias para registrar pacientes, agregar consultas, y mostrar la información.
* **funciones.py**: Contiene todas las funciones que controlan la lógica del sistema. Aquí se implementan los procesos de validación, registro y búsqueda de pacientes, así como la gestión de consultas.
* **pacientes.py**: Define la clase Paciente, la cual contiene los atributos y métodos necesarios para representar y manejar la información de cada paciente y sus consultas médicas.
  1. **Diseño de la Clase Paciente**

La clase Paciente fue diseñada para almacenar los siguientes datos:

* Nombre del paciente
* Cédula (única)
* Edad
* Tipo de sangre
* Lista de consultas médicas (cada una es un diccionario con fecha, diagnóstico y tratamiento)

Además, incluye métodos para:

* Agregar una consulta nueva.
* Mostrar todos los datos del paciente y sus consultas, con formato amigable para el usuario.
  1. **Funcionamiento del Menú y Funciones**

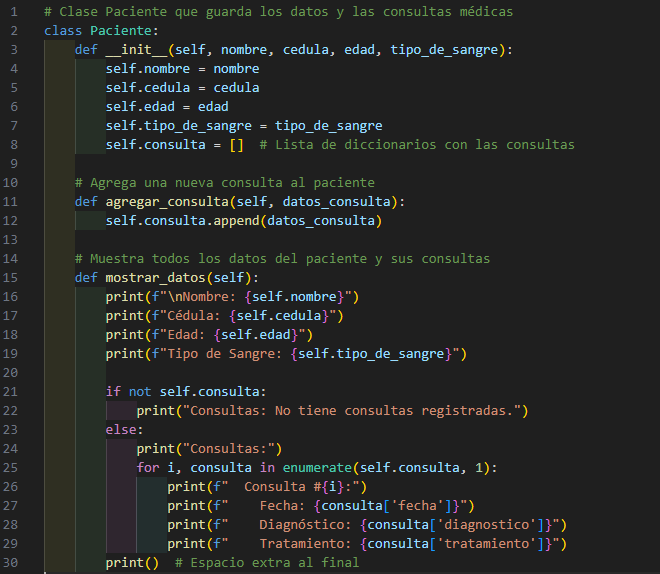
El menú principal ofrece al usuario cinco opciones:

1. Registrar nuevo paciente: Solicita los datos necesarios (nombre, cédula, edad, tipo de sangre) y los guarda si son válidos.
2. Agregar consulta a un paciente: Permite ingresar una nueva consulta médica a un paciente ya registrado, validando que la cédula exista.
3. Mostrar datos de un paciente: Busca un paciente por su cédula y muestra sus datos y todas sus consultas.
4. Mostrar todos los pacientes: Recorre la lista de pacientes y muestra sus datos completos.
5. Salir: Finaliza la ejecución del programa.

Cada función está diseñada para ser lo más intuitiva y clara posible, y maneja los errores comunes de entrada del usuario mediante bucles while y excepciones (try/except).

## **Capturas de Pantalla**

* + 1. **Codificación de la clase Paciente. Archivo pacientes.py**



* + 1. **Codificación del menú principal. Archivo main.py**

Texto

El contenido generado por IA puede ser incorrecto.

* + 1. **Menú principal mostrado correctamente**

Texto

El contenido generado por IA puede ser incorrecto.

* + 1. **Codificación del registro de un paciente. Archivo funciones.py**

Texto

El contenido generado por IA puede ser incorrecto. Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

* + 1. **Registro exitoso de un paciente**

Texto

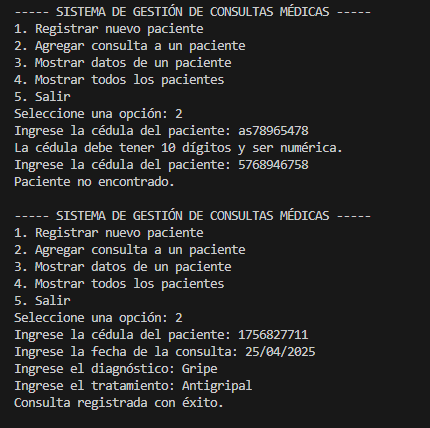
El contenido generado por IA puede ser incorrecto.

* + 1. **Codificación de agregar consulta médica. Archivo funciones.py**

Texto

El contenido generado por IA puede ser incorrecto.

* + 1. **Ingreso de consulta médica**

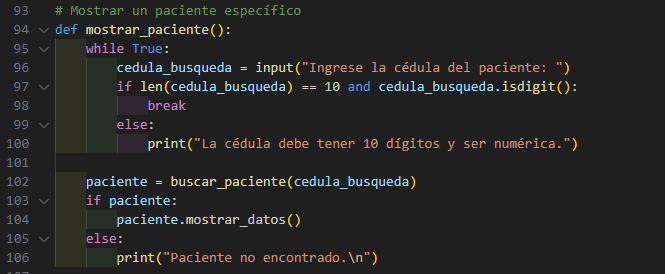


* + 1. **Función para buscar un paciente. Archivo funciones.py**

Texto

El contenido generado por IA puede ser incorrecto.

* + 1. **Codificación de mostrar paciente. Archivo funciones.py**



* + 1. **Visualización de un paciente con sus consultas**

Texto

El contenido generado por IA puede ser incorrecto.

* + 1. **Codificación de la función mostrar todos los pacientes. Archivo funciones.py**

Texto

El contenido generado por IA puede ser incorrecto.

* + 1. **Lista de todos los pacientes registrados**

Texto

El contenido generado por IA puede ser incorrecto.

1. **Conclusiones**

Durante el desarrollo de este proyecto, se consolidaron los conocimientos fundamentales de programación estructurada en Python, como:

* Uso de listas, diccionarios y clases
* Validación de datos del usuario
* Organización modular del código
* Manejo de estructuras de control como while, if y excepciones

Una de las partes más interesantes fue el diseño de la clase Paciente, ya que permitió encapsular la información de forma ordenada y facilitó la expansión del sistema. También fue retador lograr que las consultas se impriman con un formato legible y sin estructuras como llaves o corchetes.

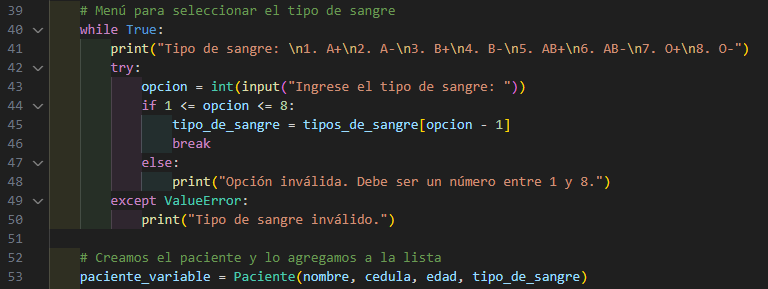
Como posibles mejoras futuras se puede considerar:

* Guardar y cargar los datos desde archivos JSON para mantener persistencia
* Añadir opción de eliminar o modificar pacientes
* Crear una interfaz gráfica con Tkinter
* Validar mejor el formato de fecha y agregar un historial ordenado cronológicamente

1. **Anexos**
   1. **Fragmentos de código relevantes**

**Fragmento 1: Validación del tipo de sangre**





**Fragmento 2: Método mostrar\_datos de la clase Paciente**

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

* 1. **Observaciones adicionales**
* El sistema fue probado con múltiples registros para asegurar la estabilidad del menú y la gestión de datos.
* El código fue estructurado en módulos para facilitar su mantenimiento, lectura y escalabilidad futura.