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

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.

2. Desarrollo

2.1. **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.

2.2. **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.

2.3. **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).

2.4.**Capturas de Pantalla**

2.4.1. **Codificación de la clase Paciente. Archivo pacientes.py**

```python
# Clase Paciente que guarda los datos y las consultas médicas
class Paciente:
    def __init__(self, nombre, cedula, edad, tipo_de_sangre):
        self.nombre = nombre
        self.cedula = cedula
        self.edad = edad
        self.tipo_de_sangre = tipo_de_sangre
        self.consulta = []  # Lista de diccionarios con las consultas

    # Agrega una nueva consulta al paciente
    def agregar_consulta(self, datos_consulta):
        self.consulta.append(datos_consulta)

    # Muestra todos los datos del paciente y sus consultas
    def mostrar_datos(self):
        print(f"\nNombre: {self.nombre}")
        print(f"Cédula: {self.cedula}")
        print(f"Edad: {self.edad}")
        print(f"Tipo de Sangre: {self.tipo_de_sangre}")

        if not self.consulta:
            print("Consultas: No tiene consultas registradas.")
        else:
            print("Consultas:")
            for i, consulta in enumerate(self.consulta, 1):
                print(f"  Consulta #{i}:")
                print(f"    Fecha: {consulta['fecha']}")
                print(f"    Diagnóstico: {consulta['diagnostico']}")
                print(f"    Tratamiento: {consulta['tratamiento']}")
        print()  # Espacio extra al final
```

2.4.2. **Codificación del menú principal. Archivo main.py**

Programacion IV semana 1,2,3,4 > consultas_medicas > main.py > ...

```python
# Importamos las funciones necesarias desde el archivo funciones.py
from funciones import registrar_paciente, agregar_consulta_paciente, mostrar_paciente, mostrar_todos
# Función que muestra el menú principal
def menu():
    while True:
        print("----- SISTEMA DE GESTIÓN DE CONSULTAS MÉDICAS -----")
        print("1. Registrar nuevo paciente")
        print("2. Agregar consulta a un paciente")
        print("3. Mostrar datos de un paciente")
        print("4. Mostrar todos los pacientes")
        print("5. Salir")
        # Solicitamos una opción al usuario
        opcion = input("Seleccione una opción: ")
        # Llamamos a la función correspondiente según la opción elegida
        if opcion == "1":
            registrar_paciente()
        elif opcion == "2":
            agregar_consulta_paciente()
        elif opcion == "3":
            mostrar_paciente()
        elif opcion == "4":
            mostrar_todos()
        elif opcion == "5":
            print("Gracias por usar el sistema. ¡Hasta pronto!")
            break
        else:
            print("Opción no válida. Intente de nuevo.\n")
# Ejecutamos el menú solo si este archivo se ejecuta directamente
if __name__ == "__main__":
    menu()
```

2.4.3. **Menú principal mostrado correctamente**

```
----- SISTEMA DE GESTIÓN DE CONSULTAS MÉDICAS -----
1. Registrar nuevo paciente
2. Agregar consulta a un paciente
3. Mostrar datos de un paciente
4. Mostrar todos los pacientes
5. Salir
Seleccione una opción: 7
Opción no válida. Intente de nuevo.

----- SISTEMA DE GESTIÓN DE CONSULTAS MÉDICAS -----
1. Registrar nuevo paciente
2. Agregar consulta a un paciente
3. Mostrar datos de un paciente
4. Mostrar todos los pacientes
5. Salir
Seleccione una opción:
```

2.4.4. **Codificación del registro de un paciente. Archivo funciones.py**

```python
# Función para registrar un nuevo paciente
def registrar_paciente():
    nombre = input("Nombre del paciente: ")
    while nombre == "":
        print("El nombre no puede estar vacio. ")
        nombre = input("Ingrese el nombre del paciente: ")

    # Validamos que la cédula sea numérica, de 10 dígitos y única
    while True:
        cedula = input("Cedula: ")
        if len(cedula) == 10 and cedula.isdigit():
            if cedula in [paciente.cedula for paciente in lista_pacientes]:
                print("La cédula ya está registrada. Ingrese una cédula diferente.")
            else:
                break
        else:
            print("La cédula debe tener 10 dígitos y ser numérica.")

    # Validamos que la edad sea un número mayor que cero
    while True:
        try:
            edad = int(input("Edad: "))
            if edad <= 0:
                print("La edad tiene que ser mayor a 0.")
            else:
                break
        except ValueError:
            print("Edad invalida, ingrese de nuevo.")
```

```python

    # Menú para seleccionar el tipo de sangre
    while True:
        print("Tipo de sangre: \n1. A+\n2. A-\n3. B+\n4. B-\n5. AB+\n6. AB-\n7. O+\n8. O-")
        try:
            opcion = int(input("Ingrese el tipo de sangre: "))
            if 1 <= opcion <= 8:
                tipo_de_sangre = tipos_de_sangre[opcion - 1]
                break
            else:
                print("Opción inválida. Debe ser un número entre 1 y 8.")
        except ValueError:
            print("Tipo de sangre inválido.")

    # Creamos el paciente y lo agregamos a la lista
    paciente_variable = Paciente(nombre, cedula, edad, tipo_de_sangre)
    lista_pacientes.append(paciente_variable)
    print("Paciente registrado con éxito.\n")

```

2.4.5. **Registro exitoso de un paciente**

```
----- SISTEMA DE GESTIÓN DE CONSULTAS MÉDICAS -----
1. Registrar nuevo paciente
2. Agregar consulta a un paciente
3. Mostrar datos de un paciente
4. Mostrar todos los pacientes
5. Salir
Seleccione una opción: 1
Nombre del paciente: Jeremy Jácome
Cedula: sg14ha5r5y
La cédula debe tener 10 dígitos y ser numérica.
Cedula: 1756827711
Edad: 0
La edad tiene que ser mayor a 0.
Edad: YO
Edad invalida, ingrese de nuevo.
Edad: 18
Tipo de sangre:
1. A+
2. A-
3. B+
4. B-
5. AB+
6. AB-
7. O+
8. O-
Ingrese el tipo de sangre: 7
Paciente registrado con éxito.
```

2.4.6. **Codificación de agregar consulta médica. Archivo funciones.py**

```python
# Agregar una consulta a un paciente existente
def agregar_consulta_paciente():
    while True:
        cedula_busqueda = input("Ingrese la cédula del paciente: ")
        if len(cedula_busqueda) == 10 and cedula_busqueda.isdigit():
            break
        else:
            print("La cédula debe tener 10 dígitos y ser numérica.")

    paciente = buscar_paciente(cedula_busqueda)
    if paciente:
        # Ingresar detalles de la consulta
        fecha = input("Ingrese la fecha de la consulta: ")
        diagnostico = input("Ingrese el diagnóstico: ")
        tratamiento = input("Ingrese el tratamiento: ")

        # Creamos un diccionario con los datos de la consulta
        diccionario_consulta = {
            "fecha": fecha,
            "diagnostico": diagnostico,
            "tratamiento": tratamiento
        }

        # Agregamos la consulta al paciente
        paciente.agregar_consulta(diccionario_consulta)
        print("Consulta registrada con éxito.\n")
    else:
        print("Paciente no encontrado.\n")
```

2.4.7. **Ingreso de consulta médica**

```
----- SISTEMA DE GESTIÓN DE CONSULTAS MÉDICAS -----
1. Registrar nuevo paciente
2. Agregar consulta a un paciente
3. Mostrar datos de un paciente
4. Mostrar todos los pacientes
5. Salir
Seleccione una opción: 2
Ingrese la cédula del paciente: as78965478
La cédula debe tener 10 dígitos y ser numérica.
Ingrese la cédula del paciente: 5768946758
Paciente no encontrado.

----- SISTEMA DE GESTIÓN DE CONSULTAS MÉDICAS -----
1. Registrar nuevo paciente
2. Agregar consulta a un paciente
3. Mostrar datos de un paciente
4. Mostrar todos los pacientes
5. Salir
Seleccione una opción: 2
Ingrese la cédula del paciente: 1756827711
Ingrese la fecha de la consulta: 25/04/2025
Ingrese el diagnóstico: Gripe
Ingrese el tratamiento: Antigripal
Consulta registrada con éxito.
```

2.4.8. **Función para buscar un paciente. Archivo funciones.py**

```
57  # Buscar un paciente por su cédula
58  def buscar_paciente(cedula):
59      for paciente in lista_pacientes:
60          if paciente.cedula == cedula:
61              return paciente
62      return None
```

2.4.9. **Codificación de mostrar paciente. Archivo funciones.py**

```python
# Mostrar un paciente específico
def mostrar_paciente():
    while True:
        cedula_busqueda = input("Ingrese la cédula del paciente: ")
        if len(cedula_busqueda) == 10 and cedula_busqueda.isdigit():
            break
        else:
            print("La cédula debe tener 10 dígitos y ser numérica.")

    paciente = buscar_paciente(cedula_busqueda)
    if paciente:
        paciente.mostrar_datos()
    else:
        print("Paciente no encontrado.\n")
```

2.4.10. **Visualización de un paciente con sus consultas**

```
1. Registrar nuevo paciente
2. Agregar consulta a un paciente
3. Mostrar datos de un paciente
4. Mostrar todos los pacientes
5. Salir
Seleccione una opción: 4

Nombre: Jeremy Jácome
Cédula: 1756827711
Edad: 18
Tipo de Sangre: O+
Consultas:
  Consulta #1:
    Fecha: 25/04/2025
    Diagnóstico: Gripe
    Tratamiento: Antigripal


Nombre: Edison Meneses
Cédula: 1234567891
Edad: 37
Tipo de Sangre: O-
Consultas:
  Consulta #1:
    Fecha: 15/04/2025
    Diagnóstico: Tos
    Tratamiento: Mentas


Nombre: Monica Ormaza
Cédula: 0401023320
Edad: 50
Tipo de Sangre: A+
Consultas:
  Consulta #1:
    Fecha: 10/04/2025
    Diagnóstico: Dolor de cabeza
    Tratamiento: Iduprofeno
```

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

```python
# Mostrar todos los pacientes registrados
def mostrar_todos():
    if not lista_pacientes:
        print("No hay pacientes registrados.")
    else:
        for paciente in lista_pacientes:
            paciente.mostrar_datos()

```

2.4.12. **Lista de todos los pacientes registrados**

```
----- SISTEMA DE GESTIÓN DE CONSULTAS MÉDICAS -----
1. Registrar nuevo paciente
2. Agregar consulta a un paciente
3. Mostrar datos de un paciente
4. Mostrar todos los pacientes
5. Salir
Seleccione una opción: 4

Nombre: Jeremy Jácome
Cédula: 1756827711
Edad: 18
Tipo de Sangre: O+
Consultas:
  Consulta #1:
    Fecha: 25/04/2025
    Diagnóstico: Gripe
    Tratamiento: Antigripal


Nombre: Edison Meneses
Cédula: 1234567891
Edad: 37
Tipo de Sangre: O-
Consultas: No tiene consultas registradas.


Nombre: Monica Ormaza
Cédula: 0401023320
Edad: 50
Tipo de Sangre: A+
Consultas: No tiene consultas registradas.
```

3. 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

4. Anexos

4.1. **Fragmentos de código relevantes**

Fragmento 1: Validación del tipo de sangre

```
7   # Lista de tipos de sangre disponibles
8   tipos_de_sangre = ["A+", "A-", "B+", "B-", "AB+", "AB-", "O+", "O-"]
```

```
    # Menú para seleccionar el tipo de sangre
    while True:
        print("Tipo de sangre: \n1. A+\n2. A-\n3. B+\n4. B-\n5. AB+\n6. AB-\n7. O+\n8. O-")
        try:
            opcion = int(input("Ingrese el tipo de sangre: "))
            if 1 <= opcion <= 8:
                tipo_de_sangre = tipos_de_sangre[opcion - 1]
                break
            else:
                print("Opción inválida. Debe ser un número entre 1 y 8.")
        except ValueError:
            print("Tipo de sangre inválido.")

    # Creamos el paciente y lo agregamos a la lista
    paciente_variable = Paciente(nombre, cedula, edad, tipo_de_sangre)
```

Fragmento 2: Método mostrar_datos de la clase Paciente

```
    # Muestra todos los datos del paciente y sus consultas
    def mostrar_datos(self):
        print(f"\nNombre: {self.nombre}")
        print(f"Cédula: {self.cedula}")
        print(f"Edad: {self.edad}")
        print(f"Tipo de Sangre: {self.tipo_de_sangre}")

        if not self.consulta:
            print("Consultas: No tiene consultas registradas.")
        else:
            print("Consultas:")
            for i, consulta in enumerate(self.consulta, 1):
                print(f"  Consulta #{i}:")
                print(f"    Fecha: {consulta['fecha']}")
                print(f"    Diagnóstico: {consulta['diagnostico']}")
                print(f"    Tratamiento: {consulta['tratamiento']}")
        print()  # Espacio extra al final
```

4.2. **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.