|  |  |
| --- | --- |
| **programming for ai**  **MID-TERM PROJECT** | **HOSPITAL MANAGEMENT SYSTEM**  A report containing concepts of Python and Numpy  **SUBMITTED TO: Dr. MADIHA YOUSAF**  **SUBMITTED BY: HAMDAN SETHI(SP23-BAI-015) , AAINA BATOOL(SP23-BAI-001)** |

**HOSPITAL MANAGEMENT SYSTEM**

This report will explain how various Python programming concepts are utilized in the ***Hospital Management System***. These concepts include **variables, loops, conditional statements, arithmetic & Boolean** **operators, functions, classes, strings, tuples, lists, and dictionaries.** The system is designed to manage patients, doctors, and appointments, allowing users to add, view, and manipulate data in a hospital management context. Additionally, it provides aggregate analysis using numpy.

**INTRODUCTION:**

The **Hospital Management System** is a Python-based program that allows users to perform various operations related to patient and doctor management. The system includes features like:

1. Adding patients and doctors
2. Scheduling and viewing appointments
3. Saving and loading data to/from CSV files
4. Performing aggregate data analysis (average appointments per doctor) using numpy

The system includes the following main components:

**Patient Class:** Contains details such as patient ID, name, age, and disease.

**Doctor Class:** Stores doctor-related details, including specialization and years of experience.

**Appointment Class:** Holds appointment information, including which patient is meeting which doctor at what time.

**CSV Handling:** Allows the system to save and load data for persistence.

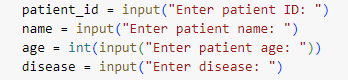
**Aggregate Analysis**: Uses numpy to calculate the average number of appointments per doctor.

**PYTHON CONCEPTS USED:**

1. **Variables:**

Variables store essential information such as patient ID, name, age, disease, doctor specialization, appointment dates, and more.

Variables are used throughout the program to store the data entered by the user or automatically generated:



Here patient\_id, name, age and disease are variabls that store the specific data. Here the input is from user.

1. **Loops:**

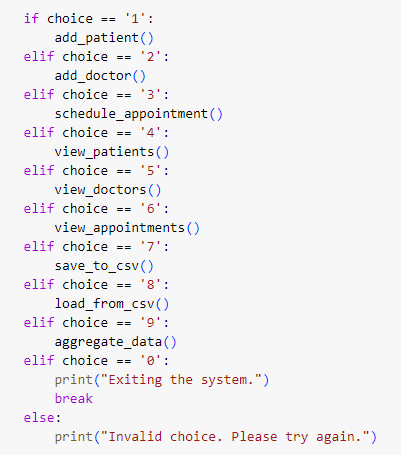
Loops are used to iterate over lists (e.g., lists of patients, doctors, or appointments). For example, when viewing patients:



The **for** loop iterates through the list of patients and prints their details.

1. **Conditional Statement:**

Conditionals are used for decision-making in the program, such as checking user input or verifying the existence of an appointment:



The if, elif, and else statements are used to navigate the menu based on user input.

1. **Arithmetic Boolean and Operators**

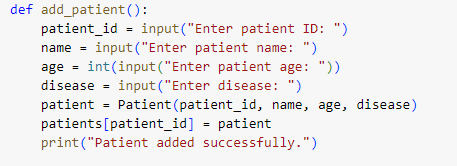
Arithmetic operators are used for calculations, such as determining the average number of appointments per doctor. Boolean operators are used in conditionals to compare values, like checking if a patient or doctor exists:



It calculates average number of appointments per doctor.

1. **Functions:**

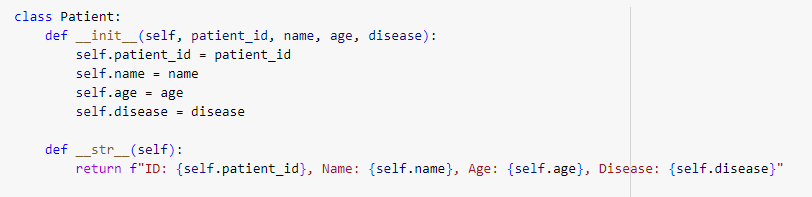
Functions are defined to perform specific tasks like adding patients, doctors, scheduling appointments, and performing analysis:



The function add\_patient() encapsulates the logic for adding a new patient to the system.

1. **Classes:**

Classes define the structure of patient, doctor, and appointment objects:



 The Patient class defines the attributes of a patient and includes a method (\_\_str\_\_) to display patient information.

1. **Strings:**

Strings are used extensively to display messages to the user and format output:



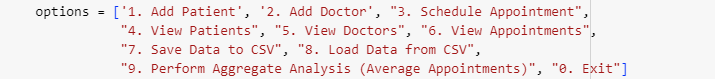
 The f-string is used to format and print the patient's details in a human-readable way.

1. **Tuples:**

Tuples are used to stre data that cant be changed. Here in this system, tuples are not used.

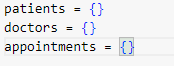
1. **Lists:**

List is used here in main.



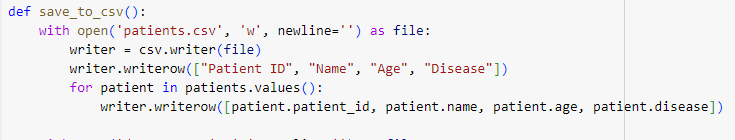
1. **Dictionaries:**

Dictionaries store key-value pairs. For example, tracking appointments by doctor:

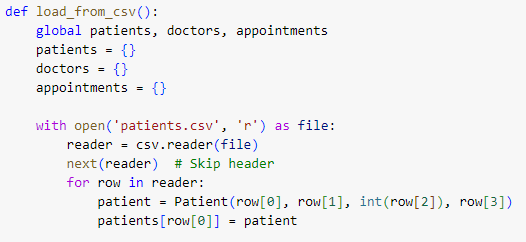


1. **SAVING AND LOADING DATA:**

The system can save and load data to and from CSV files.



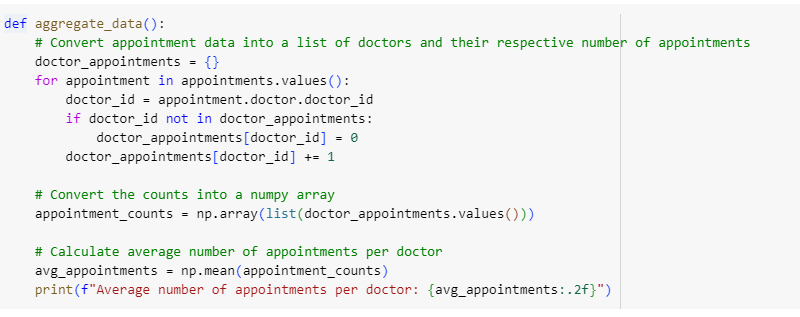
The system writes the patient, doctor, and appointment data to separate CSV files for persistent storage.



The system reads the CSV files and populates the in-memory dictionaries with the stored data.

1. **Aggregate Analysis using NumPy**

The aggregate\_data() function calculates the average number of appointments per doctor using NumPy.

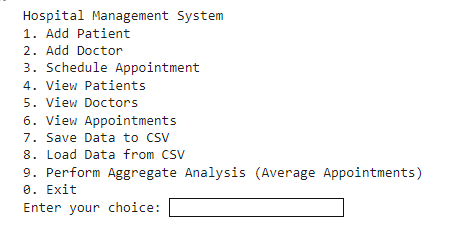


NumPy is used to create an array of appointment counts for each doctor.

The average number of appointments is calculated using np.mean().

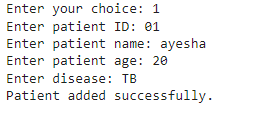
**SNIPPETS OF OUTPUT:**

Here is the general output when the code is run. Based on youre requirement, user enters the choice.



1. **ADD PATIENT:**

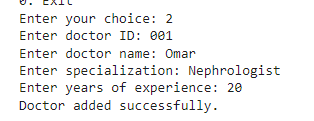
If choice 1 is entered, the patient credentials are asked from the user.



The patient is added successfully.

1. **ADD DOCTOR:**

If choice 2 is entered, the patient credentials are asked from the user.



The Dr. is added successfully.

1. **SCHEDULE APPOINTMENT:**

If choice 1 is entered, the patient credentials are asked from the user.

A white screen with black text

Description automatically generated

Appointment is scheduled successfully.

1. **VIEW PATIENTS:**

To view patients, choice 4 is entered and all the registered patients are shown.

**A black text on a white background

Description automatically generated**

1. **VIEW DOCTORS:**

To view patients, choice 5 is entered and all the registered patients are shown.

A black and white text

Description automatically generated

1. **SAVE DATA TO CSV:**

Data is saved to csv file.

****

1. **LOAD DATA FROM SCV:**

Data is loaded from csv file.

****

1. **PERFORM AGGREGATE ANALYSIS:**

By entering choice 9, aggregate function is applied and number of appointments per doctor is calculated.

****

1. **EXIT:**

****

**CONCLUSION:**

This report explains the Hospital Management System, which can manage patients, doctors, and appointments. It provides functionalities for saving and loading data to/from CSV files and applying NumPy for aggregate analysis. The system's design is modular and can be extended to include additional features such as search or filtering.