Hospital Chatbot System

Semester project

NAME
Muhammad Arslan Saleem

python-docx
Project Report

GR No: 228

Date: 15-Jan-26

Introduction

This Python program works as a basic Hospital Chatbot System. It is designed to take patient information, verify contact details, analyze symptoms, and suggest possible diseases with basic medicines and doctor advice. This program is mainly for learning purposes and shows how programming can be used to automate simple healthcare-related tasks.

What This Program Does (Program Function)

This program performs the following tasks step by step:
1. Takes patient personal information.
2. Validates contact number and email.
3. Takes symptoms from the patient.
4. Matches symptoms with stored disease data.
5. Displays possible disease, medicines, and doctor advice.
6. If no disease matches, advises consulting a doctor.

How the Program Works

Step 1: Importing Required Module

```
import re
```

The program imports the 're' module. This module is used to check whether the entered email address is valid or not by using regular expressions.

Step 2: Disease Database Creation

```
symptoms_info = {
  "Flu": {
    "symptoms": ["fever", "body pain", "cough", "sore throat"],
    "Recommanded_medicines": ["Paracetamol", "Vitamin C", "Cough Syrup"],
    "Docter_advice": "Drink warm water, rest well, avoid cold drinks and crowded places."
  },
  "Fever": {
    "symptoms": ["high temperature", "weakness", "chills"],
    "Recommanded_medicines": ["Panadol", "Brufan"],
    "Docter_advice": "Drink plenty of water, take rest, and monitor temperature regularly."
```

```
    },
    "Cold": {
        "symptoms": ["runny nose", "sneezing", "blocked nose"],
        "Recommanded_medicines": ["Antihistamines", "Nasal Spray"],
        "Docter_advice": "Stay hydrated, keep yourself warm, and take proper rest."
    },
    "Stress": {
        "symptoms": ["anxiety", "headache", "sleep problem", "tension"],
        "Recommanded_medicines": ["Doctor consultation recommended"],
        "Docter_advice": "Exercise daily , practice deep breathing , and maintain good sleep."
    }
}
```

A dictionary named 'symptoms_info' is created. This dictionary stores diseases as keys and their related information as values. Each disease contains:
- Symptoms list
- Recommended medicines
- Doctor advice

This acts as a small medical database for the program.

Step 3: Display Welcome Message

```
print("\nWelcome to the Hospital Reception\n")
```

A welcome message is displayed on the screen to inform the user that they are using the hospital chatbot system.

Step 4: Patient Basic Information Input

```
name = input("Enter patient name: ")
age = input("Enter patient age: ")
```

The program asks the patient to enter their name and age. This information is stored and helps identify the patient.

Step 5: Contact Number Validation

```
while True:
    number = input("Enter your 11-digit contact number: ")
    if number.isdigit() and len(number) == 11:
        break
    else:
        print("Invalid number. Please try again.")
```

The program asks the patient to enter an 11-digit contact number. A while loop is used to ensure the number is valid.

Conditions checked:
- The number must contain only digits.
- The number must be exactly 11 digits long.

If the number is incorrect, the program keeps asking until a valid number is entered.

Step 6: Email Address Validation

```
while True:
    email = input("Enter your email address: ")
    if re.match(r"^[\w\.-]+@[\w\.-]+\.\w+$", email):
        break
    else:
        print("Invalid email. Please try again.")
```

The program asks for an email address. Using regular expressions, the program checks whether the email format is correct. If the email is invalid, the user is asked to enter it again.

Step 7: Symptoms Input from Patient

```
user_symptoms = input("\nEnter your symptoms : ").lower()
```

```
user_symptoms_list = [s.strip() for s in user_symptoms.split(",")]
```

The patient enters symptoms separated by commas. The program converts the input into lowercase and splits it into a list. This ensures easy and accurate matching with stored symptoms.

Step 8: Symptom Matching and Disease Detection

```
for disease, data in symptoms_info.items():
    for symptom in user_symptoms_list:
        if symptom in data["symptoms"]:
            print(f"\n--- Possible Disease: {disease} ---")
            print("Recommended Medicines:", ", ".join(data["Recommanded_medicines"]))
            print("Doctor Advice:", data["Docter_advice"])
            found = True
            break
    if found:
        break
```

The program compares each entered symptom with the symptoms stored in the disease database. If even one symptom matches:
- The program identifies a possible disease.
- Displays the disease name.
- Shows recommended medicines.
- Displays doctor advice.

After finding a match, the program stops further checking.

Step 9: No Disease Match Condition

```
if not found:
    print("\nSorry! No matching disease found.")
    print("Please consult a doctor for proper diagnosis.")
```

If none of the entered symptoms match any disease in the database, the program shows a message advising the patient to consult a doctor for proper medical diagnosis.

OUTPUT:

```
Python 3.14.0 (tags/v3.14.0:ebf955d, Oct  7 2025, 10:15:03) [MSC v.1944 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.

= RESTART: C:\Users\Kali User\AppData\Local\Programs\Python\Python314\arslan.py

Welcome to the Hospital Reception

Enter patient name: Arslan
Enter patient age: 18
Enter your 11-digit contact number: 03123456789
Enter your email address: abc@gmail.com

Enter your symptoms : high temperature

--- Possible Disease: Fever ---
Recommended Medicines: Panadol, Brufan
Doctor Advice: Drink plenty of water, take rest, and monitor temperature regular
ly.
```

Conclusion

This Hospital Chatbot System program clearly demonstrates how Python can be used to collect user input, validate data, process information, and make decisions. It is a useful learning example for beginners and can be expanded into a real system by adding more diseases, symptoms, and advanced medical logic.