

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | Symptom Checker: A Python Program | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | Team Members:NashrahZainabTayyaba |  | | | |
|  | | | | Date: 25-02-2024—From Office Wizards to Python Program—Sumbul |  | | | |
|  | | |  | | |  | | |

## What is a symptom checker program?

A symptom checker can be useful in daily life as a preliminary tool to understand potential health issues. It helps users assess symptoms, identify possible conditions, and decide whether professional medical advice is needed. However, it's essential to remember that it's not a substitute for professional diagnosis and should be used cautiously, as it may not always provide accurate results.

## How did we develop this code?

The development of the provided code involves creating a symptom checker program in Python. Here is a breakdown of how the code was developed:

1. **Class Design:**

- Two classes were designed: Symptom and SymptomChecker.

- The Symptom class represents individual symptoms, storing information such as name, associated conditions, actions, and medicines.

- The SymptomChecker class manages a list of predefined symptoms and provides a method (check\_symptoms) to match user-input symptoms with the predefined symptoms.

2. **Data Structure:**

- A list of Symptom objects is created within the SymptomChecker class to store information about various symptoms, conditions, actions, and medicines.

3. **Main Logic - check\_symptoms Method:**

- The check\_symptoms method in the SymptomChecker class is responsible for checking user-input symptoms.

- It iterates through the user's symptoms, matches them with predefined symptoms (ignoring case), and collects information about matched symptoms.

4. **User Interaction Loop:**

- The code uses a while loop to repeatedly ask the user to enter symptoms and check them using the SymptomChecker.

- The results are displayed, including possible conditions, recommended actions, and prescribed medicines.

5. **Input Processing:**

- User-input symptoms are obtained using the input function, converted to lowercase, and split into a list.

6. **Output Display:**

- The results of the symptom check are displayed to the user, including possible conditions, recommended actions, and prescribed medicines.

7. **Loop Control:**

- The loop continues until the user decides not to check symptoms again (another\_input != 'yes').

8. **Testing and Refinement:**

- The code is tested with different inputs to ensure it functions as intended.

- Adjustments are made based on testing, and the code is refined for clarity and correctness.

This development process involves a structured approach to class design, data structure implementation, and the creation of methods to handle specific functionality. The iterative testing and refinement ensure that the code meets the requirements and provides a user-friendly experience.

## Our Code:

class Symptom:

def \_\_init\_\_(self, name, conditions, actions, medicines):

self.name = name

self.conditions = conditions

self.actions = actions

self.medicines = medicines

class SymptomChecker:

def \_\_init\_\_(self):

self.symptoms = [

Symptom("cough", ["Common Cold", "Flu", "Bronchitis"], ["Stay hydrated", "Rest", "Over-the-counter cough syrup"], ["Acetaminophen", "Ibuprofen"]),

Symptom("fever", ["Flu", "COVID-19", "Pneumonia"], ["Rest", "Stay hydrated", "Take fever-reducing medication"], ["Acetaminophen", "Aspirin"]),

Symptom("hepatitis", ["Hepatitis A", "Hepatitis B", "Hepatitis C"], ["Rest", "Maintain hydration", "Consult a doctor"], ["Antiviral medication", "Vaccination"]),

Symptom("tb", ["Tuberculosis"], ["Complete antibiotic course", "Rest", "Isolation during contagious period"], ["Antibiotics"]),

Symptom("corona", ["COVID-19"], ["Isolate", "Seek medical attention", "Follow health guidelines"], ["Antiviral medication", "Vaccination (if available)"]),

Symptom("toothache", ["Tooth decay", "Gum disease", "Tooth abscess"], ["Rinse with warm saltwater", "Use pain relievers", "See a dentist"], ["Pain relievers", "Antibiotics (if infection)"]),

Symptom("headache", ["Tension headache", "Migraine", "Cluster headache"], ["Rest in a dark, quiet room", "Hydrate", "Over-the-counter pain relievers"], ["Prescription medications"]),

Symptom("migraine", ["Migraine"], ["Rest in a dark, quiet room", "Stay hydrated", "Use cold or warm compress"], ["Prescription migraine medications"]),

Symptom("low blood pressure", ["Hypotension"], ["Increase salt intake", "Stay hydrated", "Eat smaller, more frequent meals"], ["No specific medication (unless severe)"]),

Symptom("high blood pressure", ["Hypertension"], ["Adopt a healthy diet", "Exercise regularly", "Reduce sodium intake"], ["Antihypertensive medications"]),

Symptom("heat stroke", ["Heat exhaustion", "Dehydration", "Heat cramps", "Rhabdomyolysis", "Hyponatremia", "Cardiac events related to heat", "Renal failure related to heat"], ["Move to a cooler place", "Cool the body with cold compresses", "Rehydrate with water or electrolyte drinks"], ["Emergency medical attention"]),

]

def check\_symptoms(self, user\_symptoms):

symptom\_results = {}

for user\_symptom in user\_symptoms:

matched\_symptoms = [symptom for symptom in self.symptoms if

user\_symptom.lower() == symptom.name.lower() or user\_symptom.lower() in

map(str.lower, symptom.conditions)]

if matched\_symptoms:

for symptom in matched\_symptoms:

symptom\_results[symptom.name] = {

"conditions": symptom.conditions,

"actions": symptom.actions,

"medicines": symptom.medicines

}

return symptom\_results

while True:

symptom\_checker = SymptomChecker()

user\_symptoms = input("Welcome to the symptom checker!"'\n' "To provide you with accurate information, please be specific about your symptoms."'\n' "Remember, this tool is not a substitute for professional medical advice."'\n' "If you have a medical emergency, call your local emergency number immediately"'\n\n'"Enter your symptoms (comma-separated): ").lower().split(',')

symptom\_results = symptom\_checker.check\_symptoms(user\_symptoms)

if symptom\_results:

print("\nSymptom Checker Results:")

for symptom, info in symptom\_results.items():

print("\nFor " + symptom + ":")

print("Possible Conditions:")

for condition in info["conditions"]:

print("- " + condition)

print("Recommended Actions:")

for action in info["actions"]:

print("- " + action)

print("Prescribed Medicine:")

for medicine in info["medicines"]:

print("- " + medicine)

else:

print("\nNo matching conditions found for the provided symptoms.")

another\_input = input("Do you want to check symptoms again? (yes/no): ").lower()

if another\_input != 'yes':

break

## Review about the Course and Bano Qabil 2.0

This course covers a comprehensive range of Python concepts, from basic to advanced topics. The inclusion of practical examples and real-world applications makes the learning experience engaging and relevant. The course goes beyond just lectures. The inclusion of regular Q&A sessions ensures that students receive prompt assistance. One of the highlights of the course is the abundance of projects which provides valuable coding experience.