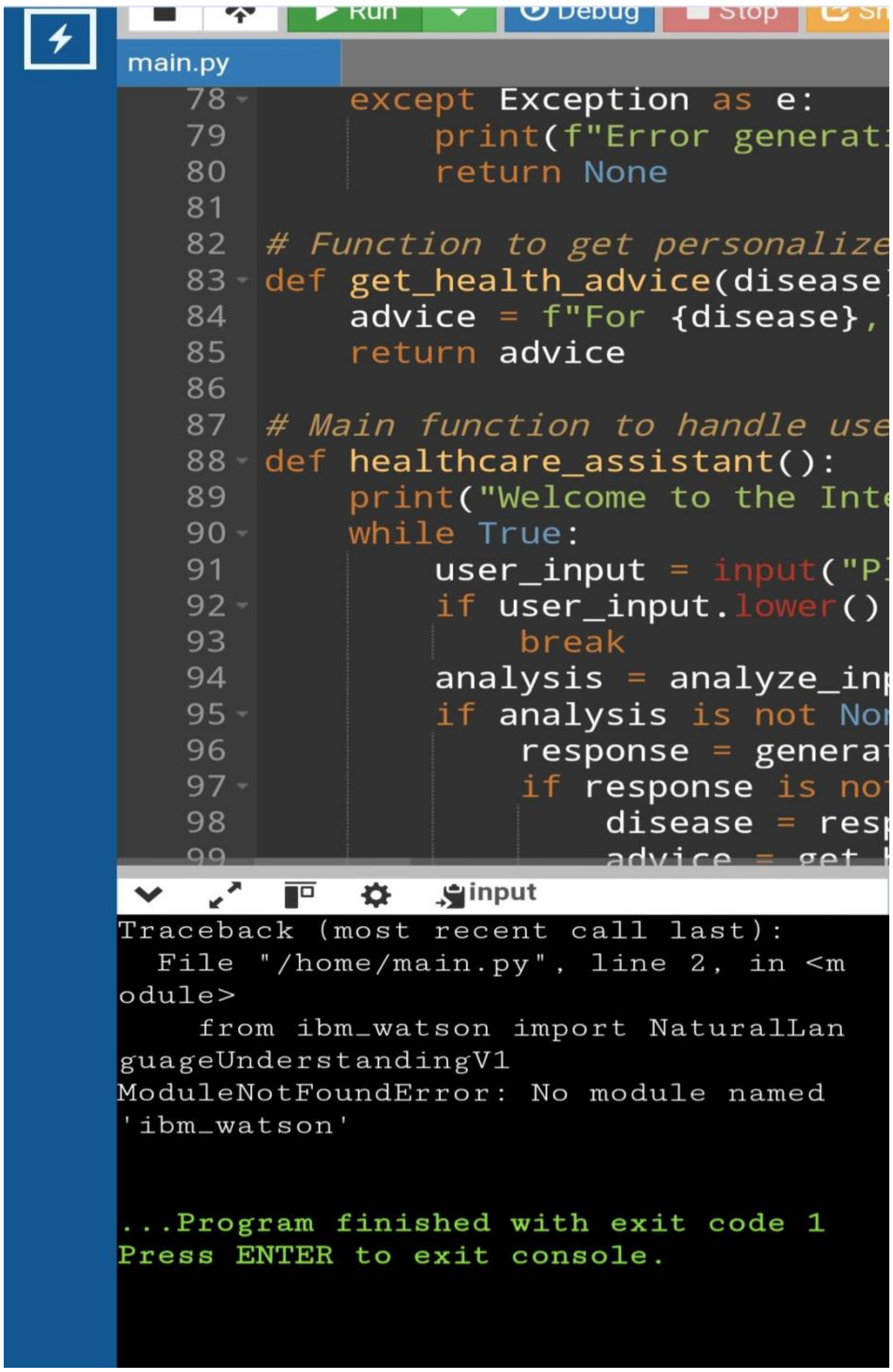
```
import pandas as pd

from ibm_watson import NaturalLanguageUnderstandingV1

from ibm_cloud_sdk_core.authenticators import IAMAuthenticator

from ibm_watson.natural_language_understanding_v1 import Features, EntitiesOptions,
KeywordsOptions

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score


# IBM Granite and NLU Credentials

IBM_NLU_API_KEY =gfgjIBM

IBM_NLU_URL = "https://api.us-south.natural-language-
understanding.watson.cloud.ibm.com"




# Initialize IBM NLU

authenticator = IAMAuthenticator(apikey=IBM_NLU_API_KEY)

natural_language_understanding = NaturalLanguageUnderstandingV1(

  version='2022-04-07',

  authenticator=authenticator

)

natural_language_understanding.set_service_url(IBM_NLU_URL)


# Sample dataset for training a model

data = {

  "symptoms": [
        "headache, fever, fatigue",
        "cough, sore throat, runny nose",
        "nausea, vomiting, diarrhea",
        "chest pain, shortness of breath",
        "abdominal pain, bloating",
    ],
    "diseases": [
        "flu",
        "common cold",
        "gastroenteritis",
        "heart disease",
        "irritable bowel syndrome",
    ]
}

df = pd.DataFrame(data)

# Vectorize the symptoms using TF-IDF
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(df['symptoms'])
y = df['diseases']

# Split the data into training and testing sets
X_train, X_test, y_train, X_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a random forest classifier
```

```
clf = RandomForestClassifier(n_estimators=100)
clf.fit(X_train, y_train)

# Evaluate the model
if X_test.size > 0:
    y_pred = clf.predict(X_test)
    print("Model Accuracy:", accuracy_score(y_test, y_pred))

# Function to analyze user input
def analyze_input(text):
    try:
        response = natural_language_understanding.analyze(
            text=text,
            features=Features(entities=EntitiesOptions(), keywords=KeywordsOptions())
        ).get_result()
        return response
    except Exception as e:
        print(f"Error analyzing input: {e}")
        return None

# Function to generate response based on analysis
def generate_response(analysis):
    try:
        symptom_vector = vectorizer.transform([analysis['text']])
        prediction = clf.predict(symptom_vector)
        return f"Based on your symptoms, you might have {prediction[0]}. Please consult a doctor for a proper diagnosis."
    except Exception as e:
        print(f"Error generating response: {e}")
        return None


# Function to get personalized health advice
def get_health_advice(disease):
    advice = f"For {disease}, you can try the following: rest, hydration, and over-the-counter medication. However, please consult a doctor for personalized advice."
    return advice


# Main function to handle user interaction
def healthcare_assistant():
    print("Welcome to the Intelligent Healthcare Assistant!")
    while True:
        user_input = input("Please describe your symptoms or type 'exit' to quit: ")
        if user_input.lower() == 'exit':
            break
        analysis = analyze_input(user_input)
        if analysis is not None and 'text' in analysis:
            response = generate_response(analysis)
            if response is not None:
                disease = response.split("you might have ")[1].split(".")[0]
                advice = get_health_advice(disease)
                print(f"{disease}: {advice}")
        else:
            print("Error: Unable to analyze input")


if __name__ == "__main__":
    healthcare_assistant()
```

Output:



```
    except Exception as e:
        print(f"Error generat
        return None

# Function to get personalize
def get_health_advice(disease
    advice = f"For {disease},
    return advice

# Main function to handle use
def healthcare_assistant():
    print("Welcome to the Inte
    while True:
        user_input = input("P
        if user_input.lower()
            break
        analysis = analyze_in
        if analysis is not No
            response = genera
            if response is no
                disease = resp
                advice = get_
```

```
Traceback (most recent call last):
  File "/home/main.py", line 2, in <m
odule>
    from ibm_watson import NaturalLan
guageUnderstandingV1
ModuleNotFoundError: No module named
'ibm_watson'


...Program finished with exit code 1
Press ENTER to exit console.
```