BBM 103:

INTRODUCTION TO PROGRAMMING LABORATORY I

FALL 2022

Assignment 2: Doctor's Aid

2210356056
Sinan DOĞAN

24.11.2022

Table of Contents

Analysis

This program; will include patient name, diagnostic accuracy, disease name, disease incidence, treatment name, and treatment risk. It will perform operations according to the input file.

- create a new patient,
- delete an existing patient,
- list all existing patients with their information,
- patient's probability of having the disease,
- patient's recommendation for a particular treatment

are what the program does.

These commands may be given to the program in different order. The outputs of the commands will be given in the output file.

Design

With read method in Python, the input file is read line by line, the lines are examined step by step, the necessary function is called, and the requirements is given to the function. At the end of all defined functions, the obtained information is written to the output file with write method.

- def create(patientName, diagnosisAccuracy, diseaseName, diseaseIncidence, treatmentName, treatmentRisk):

```
get name to patientName
    if name in patientList:
        print an error message
    else:
        add a new patient in patientList,
        print a successful message
```

This function will create a new patient. User have to enter: patient name, diagnosis accuracy, disease name, disease incidence, treatment name, treatment risk. The information will be kept in the patient list.

- def remove(name):

```
get name to patientName
    if name in patientList:
        delete the patient in patientList
        print a successful message
    else:
        print an error message
```

This function will delete an existing patient from the list. The user has to enter a name, if the entered name is not in the patient list, an error message will be given.

- def list():

```
calculate the longest row for all columns
calculate the indentation for all rows using the longest row
print the all rows with their indentation
```

This function will print the patients in the list in a table. If there is no patient in the list, it will give an empty table.

- def returnProbability(name):

```
get name to patientName
    if name in patientList:
        calculate the probability using the information of patient
        return the probability
```

This function will be used to calculate the actual fatality probability of a patient. The user has to enter a name, if the entered name is not in the patient list, an error message will be given.

- def probability(name):

```
get name to patientName
    if name in patientList:
        call the returnProbability(patientName) function
        print a message with probability
    else:
        print an error message
```

This function will be used to give a message about probability using the returnProbability function. The user has to enter a name, if the entered name is not in the patient list, an error message will be given.

- def recommendation(name):

```
get name to patientName
    if name in patientList:
        call the returnProbability(patientName) function
        if treatmentRisk > probability:
            print a message system suggest NOT to have the treatment
        else:
            print a message system suggest to have the treatment
    else:
        print an error message
```

This function will be used to give recommendation for a particular treatment. The user has to enter a name, if the entered name is not in the patient list, an error message will be given.

- def writeOutput(write):

```
get output message from function
print the output to the output file
```

This function gets the messages that should be written by the another function, and prints them to the output file.

- def readInput():

```
read the input file
analyze all lines
get the command and other requirements
call the function according to command
```

This function reads the input file. After analyzing all the lines, it calls the necessary functions.

Programmer's Catalogue

Function: create()

```
def create(patientName, diagnosisAccuracy, diseaseName, diseaseIncidence, treatmentName, treatmentRisk):
    duplication = False
    for i in patientList:
        if i[0] == patientName:
            duplication = True
    if duplication:
        writeOutput(f"Patient {patientName} cannot be recorded due to duplication.\n")
    else:
        patientList.append([patientName,
                            "{:.2f}".format(round(float(diagnosisAccuracy)*100, 2))+"%",
                            diseaseName,
                            diseaseIncidence,
                            treatmentName,
                            str(int(float(treatmentRisk)*100))+"%"])
        writeOutput(f"Patient {patientName} is recorded.\n")
```

This function adds a new patient to patientList with given information provided in the input file.

The first line of this function, duplication = False, to avoid patient duplication in patientList. If there is a patient same name in patientList, duplication become True and if statement give an error. If duplication is false, else statement runs and new patient is created by given information to the function, then successful message is given.

Function: remove()

```
def remove(name):
    absence = True
    for i in patientList:
        if i[0] == name:
            absence = False
            patientList.remove(i)
            writeOutput(f"Patient {name} is removed.\n")
    if absence:
        writeOutput(f"Patient {name} cannot be removed due to absence.\n")
```

This function removes the given patient from patientList.

The first check of this function is that, the given patient is in patientList or not. This is checked by “absence and if statement”, if there is a patient with given name in patientList, that patient is removed from patientList and “absence” become “False”. Otherwise, if there is no patient in patientList (absence = True), “if absence” statement runs and error message is given.

Function: list()

```
def list():
    maxColumn = []
    for i in range(6):
        columns = []
        for j in range(len(patientList)):
            columns.append(len(patientList[j][i]))
        maxColumn.append(max(columns))

    spaces = []
    for i in maxColumn:
        space = (i//4+1 if i % 4 != 0 else i//4)*4
        spaces.append(space)

    counter = 1
    for row in patientList:
        c = []
        for i in range(6):
            tabs = spaces[i]-len(row[i])
            c.append(tabs//4+1 if tabs % 4 != 0 else tabs//4)
        tab = "\t"
        if counter == 3:
            writeOutput(f"{'-'*c[0]*4}{'-'*c[1]*4}{'-'*c[2]*4}{'-'*c[3]*4}{'-'*c[4]*4}{'-'*9}\n")
        else:

writeOutput(f"{row[0]}{tab*c[0]}{row[1]}{tab*c[1]}{row[2]}{tab*c[2]}{row[3]}{tab*c[3]}{row[4]}{tab*c[4]}{row[5]}\n")
        counter += 1
```

This function provides us to the table of patientList.

The first step of this function is that, check all columns and rows and how many characters the item is, then the longest item is recorded for every column. The longest item is used for table indentation later.

The second step is, calculate the how many spaces have to be in one column for the table.

The third step is that, calculate the indentation for every item in one column, and create the table. Here, tab="\t" have to be written because f-string expression part cannot include a backslash. "counter" provides us that; when "counter = 2", "-" given the third line of the table as much as space of one row the table.

Function: returnProbability()

```
def returnProbability(name):
    for i in patientList:
        if i[0] == name:
            inTurkey = int(i[3].split("/")[0])
            accuracy = round(float(i[1][0:-1])/100,4)
            probability = round((inTurkey*accuracy) / ((inTurkey*accuracy)+(99950*(1-accuracy)))*100,2)
            return(probability)
```

This function returns to the patient's probability of having the disease. This calculation is made taking into account Confusion Matrix.

Function: probability()

```
def probability(name):
    absence = True
    for i in patientList:
        if i[0] == name:
            writeOutput(f"Patient {name} has a probability of {returnProbability(name)}% of having {i[2].lower()}.\n")
            absence = False
    if absence:
        writeOutput(f"Probability for {name} cannot be calculated due to absence.\n")
```

This function gives output to the patient's probability of having the disease, with uses returnProbability() function. Absentation is checked by "absence" and "if statement", if there is a patient with given name in patientList, output given and "absence" become "False". Otherwise, if there is no patient in patientList (absence = True), "if absence" statement runs and error message is given.

Function: recommendation()

```
def recommendation(name):
    absence = True
    for row in patientList:
        if row[0] == name:
            index = patientList.index(row)
            risk = int(patientList[index][5][:-1])
            if risk > returnProbability(name):
                writeOutput(f"System suggests {name} NOT to have the treatment.\n")
            else:
                writeOutput(f"System suggests {name} to have the treatment.\n")
            absence = False
            break
    if absence:
        writeOutput(f"Recommendation for {name} cannot be calculated due to absence.\n")
```

This function gives output to the patient's recommendation for a particular treatment. The recommendation calculation compares the patient's probability of having the disease and the treatment risk probability. Absentation is checked by "absence" and "if statement", if there is a patient with given name in patientList, output given and "absence" become "False". Otherwise, if there is no patient in patientList (absence = True), "if absence" statement runs and error message is given.

Function: writeOutput()

```
def writeOutput(write):
    with open("output.txt", "a", encoding='utf-8') as f:
        f.write(write)
```

This function is used when an output will write to the output file.

Function: readInput()

```
def readInput():
    with open("doctors_aid_inputs.txt", "r", encoding='utf-8') as inputFile:
        for line in inputFile:
            command = line.split()[0]
            if command == "list":
                list()
            else:
                sep = line.split(", ")
                patientName = sep[0].split()[1]
                if command == "create":
                    create(patientName, sep[1], sep[2], sep[3], sep[4], sep[5])
                elif command == "remove":
                    remove(patientName)
                elif command == "recommendation":
                    recommendation(patientName)
                elif command == "probability":
                    probability(patientName)
```

This function reads the input file given by a user. It separates the input file line by line, then it analyzes every line, then it senses the command and calls the essential function. If some data have to be given to essential function, data are given from the analyzed line.

Extras

```
patientList = [
    ["Patient", "Diagnosis", "Disease", "Disease","Treatment", "Treatment"],  # for first row of the list
    ["Name", "Accuracy", "Name", "Incidence", "Name", "Risk"],  # for second row of the list
    ["", "", "", "", "", ""]  # for third row of the list "-"
]
```

The first 3 items of patientList are given manually, this is used to create the first 3 rows of the table.

```
with open("doctors_aid_outputs.txt","w") as file:
    file.write("")
```

That command is used to clear doctors_aid_outputs.txt file.

User Catalogue

Program's User Manual & Tutorial

In this program, users have to write all command in an input file which name is “doctors_aid_inputs.txt”. Outputs are given in an output file which name is “doctors_aid_outputs.txt”

- Create a New Patient:

create *Patient Name*, *Diagnosis Accuracy*, *Disease Name*, Disease Incidence, Treatment Name, Treatment Risk

The “create” command adds a new patient to the system.

If trying to create a patient with already in the patient list, it gives an error message.

- Delete an Existing Patient:

remove *Patient Name*

The “remove” command removes an existing patient from the system.

If there is no such patient, it gives an error message.

- List All Patients with their Information:

list

The “list” command is used to list the complete information in the system.

If there is no such patient, it gives an error message.

- Patient’s Probability of having the Disease:

probability *Patient Name*

The “probability” command is used to calculate and help the actual fatality probability of a patient.

If there is no such patient, it gives an error message.

- A Patient’s Recommendation for a Particular Treatment:

recommendation *Patient Name*

The “recommendation” command is used to give a system recommendation to a patient about whether to have the treatment.

If there is no such patient, it gives an error message.

Restrictions on the Program

In this program, requirements of the commands (which is indicated italics and blue) must be entered completely. Other cases are not checked. You may get an error if you don't enter completely.

Evaluate Yourself / Guess Grading

Evaluation	Points	Evaluate Yourself / Guess Grading
Indented and Readable Codes	5	**5**
Using Meaningful Naming	5	**5**
Using Explanatory Comments	5	**5**
Efficiency (avoiding unnecessary actions)	5	**5**
Function Usage	25	**24**
Correctness	35	**35**
Report	20	**16**
There are several negative evaluations	…	…