Assignment 2: Doctor’s Aid

Name: Yusuf İpek	***SN:*** 222035048
BBM103	
Delivery Date: 27.11.2020	

Contents

Analysis

In the assignment, we have to write a program which helps the doctors. Doctors can see all patients in the list, calculate the actual having disease probability and decide about this thanks to this program. Also program calculates treatment risk and probability together and suggest for treatment, in this way doctors can make better decisions.

We have some informations about the patients in this program. The informations are name of patients, disease name and disease incidence, treatment risk and treatment name and diagnosis accuracy. We take them with create function and show them with list function also we use them for calculating actual probability and deciding suggestion for treatment.

The doctors can create and remove patients, list patients' information, calculate the actual patient's probability of having diesase and ask for treatment suggestion in the program.

Defined commands:

- create patient,
- remove patient,
- probability of patient,
- recommendation for patient,
- list patients.

Design

Lists

There is two defined lists in the program. One of them is for inputs and the other one is for patients informations.

Reading Input File

Get the input file which is in the same directory of program. After that, read line by line. For each line delete the end of the line string (\n) and add to a list.

Create Output File

Create a output file in the same directory. If the output file exists clear it.

Define Funcitons

Define the specified functions in the program.

Find the Commands

Look a list which has the lines of the input file. For each element of the list, serach for the commands. Compare to first element of the each list elements with defined functions and call the wanted funciton with remain of the list parameter

Write the Outputs

Write the outputs of the funcitons for each line's datas.

Programmer's Catalogue

Main Program

```
datas = []
patients = []
```

Two lists are defined in these lines. '*datas'* list is for input file's datas. '*patients'* list is for patients (which is recorded with create command)

```
# main program
readInputFile()  # read all lines of file and add to the "datas" array
createOutputFile()  # creat empty output file
for data in datas:
    command = data.split()[0]
    if(command == "list"):
        listPatients()
    elif (command == "create"):
        data = data.split(", ")
        data[0] = data[0].split(" ")[1]
        createNewPatient(data)
    else:
        data = data.split(" ")
        data.pop(0)
        if(command == "remove"):
            removePatient(data[0])
        elif(command == "probability"):
            probability(data[0])
        elif(command == "recommendation"):
            recommendation(data[0])
```

There is a main program's codes in the left. The program reads input file and creates output file. After reading, checks each line's first element and compare to functions' name. If line's first element is equal to one of the specified funcitons' name call this function.

For the create function, the program split data and remove command 'create' and call create function with data parameter.

For the list command just call the *list* function.

For the remove, probability and *recommendation* commands remove the command and call functions with parameter patient name.

File Functions

```
import os
```

Import os module for getting current directory of program file to reach input and output files.

```
# file functions
currentDirectory = os.getcwd()
outpuFileName = "doctors_aid_outputs.txt"
outputFilePath = os.path.join(currentDirectory, outpuFileName)


def readInputfile():
    fileName = "doctors_aid_inputs.txt"
    filePath = os.path.join(currentDirectory, fileName)
    global datas
    with open(filePath, "r") as _file:
        while True:
            data = _file.readline()
            if data == "":  # Last line check
                break
            else:
                # delete \n which is end of the line
                data = data.replace("\n", "")
                datas.append(data)


def createOutputFile():
    _file = open(outputFilePath, "w")  # write text to the file
    _file.write("")
    _file.close()


def writeFile(text, line=True):
    _file = open(outputFilePath, "a")  # append text to the file
    if(line):
        text += "\n"
    _file.write(text)
    _file.close()
```

First three lines for getting the current directory and set the output file path for *createOutputFile* function and *writeFileFunction.*

readInputFile Function
Get the input file with file name and current directory. Read each lines (until the readed line is equal to *“”*), remove end of the line character “*n*” and add to the “*datas*” list. In *datas* list, the program has commands and patients’ specified informations.(for example: *remove Deniz, list, probability Hayriye, create Hayriye, 0.999, Breast Cancer…*)

createOutputFile Function
If there is a output file which is named “doctors_aid_outputs.file” and in the program’s directory, clear the file. Otherwise, create a file.

writeFile Function
Paramaters: string *text*, bool *line*

Open the output file and append the parameter text to into the file. If ‘*line’* is True add text ‘\n’ for new line. The ‘*line*’ parameter default value is True.

Patients Functions

```
# datas funcitons
def checkPatient(patientName):
    for patient in patients:
        if(patient[0] == patientName):
            return True
    return False
```

chechkPatient Function
Parameter: *string patientName*

Return type: *bool*

This function checks the *patientName* in the *patients* list. If there is a match return *true* else return *false*. The program checks patient’s existence becasue avoiding duplication and absence situations.

```
def getIndexOfPatient(patientName):
    if checkPatient(patientName):
        for i in range(len(patients)):
            if(patients[i][0] == patientName):
                return i
    return -1
```

getIndexOfPatient Function

Parameter: *string patientName*

Return type: *int*

This function checks the patient's existence by using *checkPatients* function. If there is no patient in the *patients* list which is named *patientName* return -1, otherwise return patient's index of the patients list.

```
def createNewPatient(patient):
    patientName = patient[0]
    if(checkPatient(patientName)):
        writeFile("Patient {} cannot be recorded due to duplication."
        .format(patientName))
    else:
        patients.append(patient)
        writeFile("Patient {} is recorded.".format(patientName))
```

createNewPatient Function

Parameter: *list patient*

This function adds *patient* to *patients* list if the patient is not in the list already.

The function uses *checkPatient* function to check duplication by taking first element of the *patient* which is equal to patient name. If *patient* is not in the list, this function adds patient to list and write the output file that "*patient is recorded*". Otherwise write the output file "*can not recorded due to duplication.*".

```
def removePatient(patientName):
    index = getIndexOfPatient(patientName)
    if(index != -1):
        patients.pop(index)
        writeFile("Patient {} is removed.".format(patientName))
    else:
        writeFile("Patient {} cannot be removed due to absence."
        .format(patientName))
```

removePatient Function

Parameter: *string patientName*

This function call *getIndexOfPatient* function to learn index of patient which named *patientName*. If the patient is in the *patients* list, use pop function with its index to remove this paitent from the list.

```
def calculateProbability(index):
    _diseaseInsidence = patients[index][3].split("/")
    diseaseInsidence = int(_diseaseInsidence[0]) / int(_diseaseInsidence[1])
    diagnosisAccuracy = float(patients[index][1])
    probability =  (diseaseInsidence / ((1-diagnosisAccuracy) + diseaseInsidence))
    probability = str(probability)[:6]
    #for % i multiply the value with 100 and take 4 digits so i return 6 digits
    return probability
```

calculateProbability Function

Paramater: *int index*

Return: *string probability*

Actual probability = disase incidence / ((1-diagnosis accuary) + disease incidence)*

Actual probability = incidence / (wrong diagnosis + incidence)

Actual probability = incidence / all situations

This function calculates the actual having disease probability. Take the *index* of patient for getting patient's *diagnoses accuracy* for *disease incidence* values. Getting values and convert them to the *float*. Calculate *probability* by the formula*. In the end return just first 6 digits of the probability value.

```
def probability(patientName):
    index = getIndexOfPatient(patientName)
    if(index != -1):
        probability = calculateProbability(index)
        probability = str(float(probability) * 100)
        if(probability[-1] == '0'):
            probability = probability[:-2] # delete .0
        probability += "%"
        writeFile("Patient {} has a probability of {} of having {}."
        .format(patientName, probability,patients[index][2].lower()))
    else:
        writeFile("Probability for {} cannot be calculated due to absence."
        .format(patientName))
```

probability Function

Parameter: *string patientName*

This function calculate the probability with *calculateProbability* function and write the output file. Of course, if patient is exist in the *patients* list.

Firstly learn index of the patient with patientName and *getIndexOfPatient* function. Then check existance.

If patient exists call *calculateProbability* function, *calculateProbability* function returns value like this “0.3333” and this function convert this value to “33.33%” format by convert the value *float* and multiply it by 100 and delete last zeros (.0) and convert to *string* value. In the end add “%” character to value and write the output file *patient name*, *probability value*, *diesase name*.

If patient does not exist write the output file “*cannot calculated due to absence*”.

```
def recommendation(patientName):
    index = getIndexOfPatient(patientName)
    if(index != -1):
        _probability = float(calculateProbability(index))
        if(_probability > float(patients[index][5])):
            writeFile("System suggests {} to have the treatment."
            .format(patients[index][0]))
        else:
            writeFile("System suggests {} NOT to have the treatment."
            .format(patients[index][0]))
    else:
        writeFile("Recommendation for {} cannot be calculated due to absence."
        .format(patientName))
```

recommendation Function

Paramater: *string patientName*

This function calculate the probability with *calculateProbability* function and compare to treatment risk value. If the risk is greater than probability suggests to have treatment else doesn’t suggest to have treatment.

Firstly learn index of the patient with patientName and *getIndexOfPatient* function. Then check existance.

If patient exists call *calculateProbability* function. Then write suggest or not. Suggestion depends on comparison of the *probability* and *treatment risk* values.

If patient does not exist write the output file “*cannot calculated due to absence*”.

```
def listPatients():
    writeFile("Patient\tDiagnosis\tDisease\t\t\tDisease\t\tTreatment\t\tTreatment")
    writeFile("Name\tAccuracy\tName\t\t\tIncidence\tName\t\t\tRisk")
    writeFile("-------------------------------------------------------------------------------")
    for patient in patients:
        for i in range(len(patient)):
            tabCount = 0
            tabCount -= len(patient[i]) // 4 # change the tab count by lenght of the item
            if(i == 0 and patient[i] == "Su"):
                tabCount += 2 # 2*4 = 8 character
                writeFile(patient[i]+("\t"*tabCount), line=False)
            elif(i == 1):
                value = str(float(patient[1]) * 100)
                if(len(value)< 5): #4 numbers 1 .
                    value += "0" *(5 - len(value))
                writeFile(value + "%\t\t",line=False)
            elif(i == 2 and patient[i]):
                tabCount += 4
                writeFile(patient[i]+("\t"*tabCount), line=False)
            elif(i == 4):
                tabCount += 4
                writeFile(patient[i]+("\t"*tabCount), line=False)
            elif(i == len(patient) - 1):
                text = str(float(patient[i]) * 100)
                if(text.__contains__(".0")):
                    text = text[:-2]
                writeFile(text+ "%") #end of the line
            else:
                writeFile(patient[i]+"\t",line=False)
```

listPatients Functions
This function write a table which has patients informations.

First write the informations titles.

Then check for the each patients' each informations. Check for the specific values for the order.

And write the informations to the output file.

writeFile(text, *line*= False) for writing same line.

For the disease accuarcy, convert to value "0.xx" to "xx%" and add the "0" characters for complete 4 digits and write the output file.

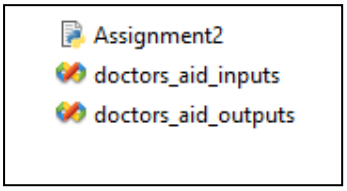
For the treatment risk, convert to value "0.xx" to "xx%" and remove the unnecessary zeros (.0). In the end just write the value and go to the next line

For the other informations calculates the needed tab counts. For patient names the total character count is 8 blanket and it is equal to 2 tabs but the names have characters too, so the program divides the lenght of the name by 4 and look just quotient and decrease the tab count by this quotient. Finally program has the needed tab count for the names. For writing multiply "\t" character by needed tab count. For the other informations calculation is same but the total character count is changing.

Total tab count is 2 for patient names.

Total tab count is 4 for disease names and treatment names

User Catalogue



1. Write the defined commands into the input txt file. (doctors_aid_inputs.txt)
2. Execute the python file. (Assignment2.py)
3. Python file writes the output of defined commands into the output file.
4. Check out the output file. (doctors_aid_outputs.txt)

You can use just specified functions. (*create, remove, probability, recommendation, list*)

1. Create function for creating new patient.

Usage: "*create {patient_name}, {diagnosis_accuracy}, {disease_name}, {disease_incidence}, {treatment_name}, {treatment_risk}"*

Example: "`create Yusuf, 0.999, Breast Cancer, 50/100000, Surgery, 0.40`"

2. Remove function for removing existing patient.

Usage: *"remove {patient_name}"*

Example: "`remove Yusuf`"

3. Probability function for calculate actual probability.

Usage: *"probability {patient_name}"*

Example: "`probability Yusuf`"

4. Recommendation function for system suggestion about treatment.

Usage: *"recommendation {patient_name}"*

Example: "`recommendation Yusuf`"

5. List function for list all patients and their informations.

Usage&Example: *"list"*

***Note**: The input file has to be in same directory with the python file and the input file's name has to be "doctors_aid_inputs".*

***Note:** You can see the input and output files samples in the next page.*

doctors_aid_inputs.txt

```
create Hayriye, 0.999, Breast Cancer, 50/100000, Surgery, 0.40
create Deniz, 0.9999, Lung Cancer, 40/100000, Radiotherapy, 0.50
create Ateş, 0.99, Thyroid Cancer, 16/100000, Chemotherapy, 0.02
probability Hayriye
recommendation Ateş
create Toprak, 0.98, Prostate Cancer, 21/100000, Hormonotherapy, 0.20
create Hypatia, 0.9975, Stomach Cancer, 15/100000, Immunotherapy, 0.04
recommendation Hypatia
create Pakiz, 0.9997, Colon Cancer, 14/100000, Targeted Therapy, 0.30
list
remove Ateş
probability Ateş
recommendation Su
create Su, 0.98, Breast Cancer, 50/100000, Chemotherapy, 0.20
recommendation Su
list
probability Deniz
probability Pakiz
```

Input file sample

doctors_aid_outputs.txt

```
Patient Hayriye is recorded.
Patient Deniz is recorded.
Patient Ateş is recorded.
Patient Hayriye has a probability of 33.33% of having breast cancer.
System suggests Ateş NOT to have the treatment.
Patient Toprak is recorded.
Patient Hypatia is recorded.
System suggests Hypatia to have the treatment.
Patient Pakiz is recorded.
Patient Diagnosis   Disease          Disease     Treatment       Treatment
Name    Accuracy    Name             Incidence   Name            Risk
-------------------------------------------------------------------------
Hayriye 99.90%      Breast Cancer    50/100000   Surgery         40%
Deniz   99.99%      Lung Cancer      40/100000   Radiotherapy    50%
Ateş    99.00%      Thyroid Cancer   16/100000   Chemotherapy    2%
Toprak  98.00%      Prostate Cancer  21/100000   Hormonotherapy  20%
Hypatia 99.75%      Stomach Cancer   15/100000   Immunotherapy   4%
Pakiz   99.97%      Colon Cancer     14/100000   Targeted Therapy30%
Patient Ateş is removed.
Probability for Ateş cannot be calculated due to absence.
Recommendation for Su cannot be calculated due to absence.
Patient Su is recorded.
System suggests Su NOT to have the treatment.
Patient Diagnosis   Disease          Disease     Treatment       Treatment
Name    Accuracy    Name             Incidence   Name            Risk
-------------------------------------------------------------------------
Hayriye 99.90%      Breast Cancer    50/100000   Surgery         40%
Deniz   99.99%      Lung Cancer      40/100000   Radiotherapy    50%
Toprak  98.00%      Prostate Cancer  21/100000   Hormonotherapy  20%
Hypatia 99.75%      Stomach Cancer   15/100000   Immunotherapy   4%
Pakiz   99.97%      Colon Cancer     14/100000   Targeted Therapy30%
Su      98.00%      Breast Cancer    50/100000   Chemotherapy    20%
Patient Deniz has a probability of 80% of having lung cancer.
Patient Pakiz has a probability of 31.81% of having colon cancer.
```

Output file sample

Grading Table

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	…	25
Correctness	35	…	35
Report	20	…	20
There are several negative evaluations	…	…	