Hacettepe University Department of Computer Engineering

BBM 103 Assignment 2 Report

Gülvera Yazılıtaş – 2210356111

23.11.2022

Index

1. Analysis

2. Design

3. Programmers Catalogue

4. Users Catalogue

1 Analysis

The assignment is to create a program name “ Doctor’s Aid ”. In this program data of patients are wanted to record and categorize their arguments of them. The data are given in an input.txt file and the outputs of the program should be written in an output.txt file.

1.1 Data :

The data of a patient have six categories:

1. Patient Name
2. Diagnosis Accuracy
3. Disease Name
4. Disease Incidence
5. Treatment Name
6. Treatment Risk

1.2 The user uses five commands in this program:

1. Create: To record a new patient to a list.
2. Remove: Delete a recorded patient from the list.
3. Probability: To get the patient's probability of having cancer.
4. Recommendation: Program recommends patient treatment if it is necessary.
5. List: To list all the patients recorded by then in a table form.

- The program should understand the commands and print the results on the output file after execution.

1.3 Possible problems when running the program;

- The possibility of a patient not being on the list should be taken into account. Otherwise, the program gives errors. For example, when it gets the remove command for patient x, it can not remove the patient if the patient is not recorded. Except for the 'create' and the 'list' commands, the absence possibility must be considered for all commands.
- In addition, while writing the record list, attention should be paid to the white spaces. Because the table must be readable and not confusing

2 Design

- **Reading lines and taking commands**

First of all, program should read lines in the input txt file and understand every line to take command as well. To do that the file needs to open then every line is read and put in a " lines " list. All these reading implementations are done in a reading function.
For every line in the list, the whole line is split from the white spaces, and they are put in another list. the first element of each line is read and assigned to a variable to take the command.

- **Writing to output file**

To write every message in the program generates a writing function can be made and used after every command execution. Understanding the commands
A control mechanism , namely if - elif, controls the command variable and finds the proper operation for it. For example, if the command is 'create', the create code part should be executed.

- **Recording a new patient**

First a main patient list , copy patient list (it is for printing the list)and name list for the patients is made. And function to create does these steps:
First control the name if it is in the name list , if it is then write to output file that it can not recorded due to duplication.
If it is not in the name list then add the name to nae list and add the name and the other information about patient to the main patient data list and copy patient list(percentage arguments should be arranged in this part . (i.e., 0,999-->%99.9)
Write ' patient is recorded.'

- **Removing a patient**

First program reads the name of the patient that is desired to remove. assign this name to variable. The remove function checks this name in the name list if it is not there then write an absence message.
if it is there then removes it from name list , patient list and copy patient list.
write the message that patient is removed.

- **Calculating probability**

First formula of having a cancer probability should be found. (video link)
And then the arguments that is needed for the formula should be taken from the patient list: Test accuracy and disease incidence.
after calculations, the result is written on the output.
For sure, the possibility of the absence of a patient should be considered with control methods.

- **Recommendation**

Arguments of treatment risk belongs to patient is found and assign a variable then the disease probability and treatment risk are compared. If reatment risk is greater than probability, then treatment do not be suggested. Vice versa it is suggested. Lastly, message is written on the file.

- **Printing patient list**

To do that copy patient list is used. Because its arguments were fixed in the previous steps. For ordered and proper lists' tabulate ' function can be used. Because it arranges the white spaces automatically.

3 Programmer's Catalogue

reading() function:

```
def reading():
    with open('doctors_aid_inputs.txt', 'r') as f:
        lines = f.readlines()  # I read all lines from input and put them
in a list for convenience.
        a = len(lines)
        for i in range(a):  # Using "for" for operate every single line and
command by order.
            line = lines[i]
            l1 = line.split(' ')
            cmd = l1[0]
```

In reading function command control part :

As it is seen every other function is called in the reading function. Reading function is designed as a distribution point for commands besides its main function.

```
if cmd == 'create':
    l1.pop(0)  # Now I take whole line except command and convert it to a
list named "l2".
    listtostr = ' '.join([str(elem) for elem in l1])
    l2 = listtostr.split(',')

    l3 = l2.copy()  # Here I arrange some percentage arguments and add
arranged form to copy_pl list.
    b = l2[1].strip().replace('.', '')
    if len(b) > 3:
        l3[1] = b[1:3] + '.' + b[3:] + '%'
    else:
        l3[1] = b[1:] + '%'
    c = l3[5].replace('\n', '')
    l3[5] = c[3:] + '%'
    copy_pl.append(l3)

    create(l2)  # Calling Create Function.
```

Before calling 'create function', in the if block copy patient list is made here.
Every percentage argument in patient data converts to a more readable form by adding '%' at and of the numbers.

```
elif cmd == 'remove':
    l1.pop(0)
    removename = l1[0]  # I take the name that will be removed.

    remove(removename)

elif cmd == 'probability':
    l1.pop(0)
    listtostr = ' '.join([str(elem) for elem in l1])
    l2 = listtostr.split(',')
```

```
    probname = l2[0]  # I take the name whose probability will be
calculated.

    probability(probname)

elif cmd == 'recommendation':
    l1.pop(0)
    listtostr = ' '.join([str(elem) for elem in l1])
    l2 = listtostr.split(',')
    recname = l2[0]  # I take the name that will be given recommendation.

    recommendation(recname)

else:  # Else is stand for our last command which is "list".
    table1 = 'Patient  Diagnosis   Disease          Disease      Treatment
Treatment'
    # I choose to write first line of the table by arranging the spaces
manually for convenience.
    writing(table1)

    list()  # Calling list function.
```

wrtiting() function :

```
def writing(messageforwriting):
    with open('doctors_aid_outputs.txt', 'a') as g:
        g.write('\n')
        g.write(messageforwriting)
```

The argument ‘messageforwriting’ must be in a string form. So every message shoul be written by taking into account this.

create() function:

```
def create(parameter):
    name = parameter[0]
    if name in namelist:
        message = str('Patient ' + name + ' cannot be recorded due to
duplication.')
    else:
        message = str('Patient ' + name + ' is recorded.')
        namelist.append(name)
        patientlist.append(parameter)

    writing(message)  # I use writing function in every function for
printing result

    return patientlist
```

remove() function :

```
def remove(name):
    name = name.strip()  # I use strip to get rid of blank spaces.
    if name in namelist:
        ind = namelist.index(name)  # I find the index of the name which is
required to remove.
        namelist.pop(ind)
        patientlist.pop(ind)
        copy_pl.pop(ind)
        remove_message = 'Patient ' + name + ' is removed.'
        writing(remove_message)
    else:
        remove_message = 'Patient '+name+'cannot be removed due to
absence.'
        writing(remove_message)
```

list() function:

```
def list():
    # To have an ordered table I take the second words of headers for
tabulate function.
    columns = ['Name', 'Accuracy', 'Name', 'Incedence', 'Name', 'Risk']
    message = tabulate(copy_pl, headers=columns)
    writing(message)
```

probability() function:

```
def probability(name):  # probability = disease_incidence/(1-
accuracy+disease_incidence)
    name = name.strip()
    if name in namelist:
        ind = namelist.index(name)
        accuracy = float(patientlist[ind][1])  # Taking accuracy argument.
        int_incedence = patientlist[ind][3].strip()
        n, d = map(int, int_incedence.split('/'))  # Taking disease
incidence argument.
        disease_inc = n / d
        prob = disease_inc/(1-accuracy+disease_inc)
        prob = (prob.__round__(4))*100
        prob = str(prob)
        cancer_type = patientlist[ind][2]
        message = 'Patient ' + name + ' has a probability of '+prob+' %
having '+cancer_type+'.'
        writing(message)
```

```
else:
    message = 'Probability for '+name+' cannot be calculated due to
absence.'
    writing(message)
```

recommendation() function :

```
def recommendation(name):  # if treatment risk > prob do not recommend
treatment
    name = name.strip()
    if name in namelist:
        ind = namelist.index(name)
        accuracy = float(patientlist[ind][1])
        int_incedence = patientlist[ind][3].strip()
        n, d = map(int, int_incedence.split('/'))
        disease_inc = n / d
        prob = disease_inc / (1 - accuracy + disease_inc)
        prob = (prob.__round__(4)) * 100  # I calculate the probability
again for comparing.
        treat_risk = float(patientlist[ind][5].replace('\n', ' '))*100  #
Treatment risk
        if treat_risk > prob:
            message = 'System suggests '+name+' NOT to have the treatment.'
            writing(message)
        else:
            message = 'System suggests '+name+' to have the treatment.'
            writing(message)
    else:
        message = 'Recommendation for '+name+' cannot be calculated due to
absence.'
        writing(message)
```

In the end, just calling reading() function once will be enough to run the program.

```
reading()  # Finially, I call just the reading function to operate all the
input and output.
```

4 User's Catalogue

Restrictons about the progmram and input file:

User should write create arguments in this format:

Command PatientName, DiagnosisAccuracy, DiseaseName, DiseaseIncidence, TreatmentName, TreatmentRisk

For example :

create Hayriye, 0.999, Breast Cancer, 50/100000, Surgery, 0.40

For list :

Write just 'list'

For other commands:

Command PatientName

To execute the program :

Be careful about the input and output file name it should be exactly this 'doctors_aid_inputs.txt' and 'doctors_aid_outputs.txt'

Go to the program and txt files directory in the command prompt.

Write python3 Assignment2.py

And the outputs of the inputs automatically are written.

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	…	…