Hacettepe University

Department of Computer Engineering

Assignment 2: Doctor's Aid

Leen Said - 2220356194

24-11-2022

Introduction

In this section I talk about the problem of the assignment and the goal of this project.

In this assignment, we are asked to create a program that behaves like a database as well as return outputs based on data in this "database".

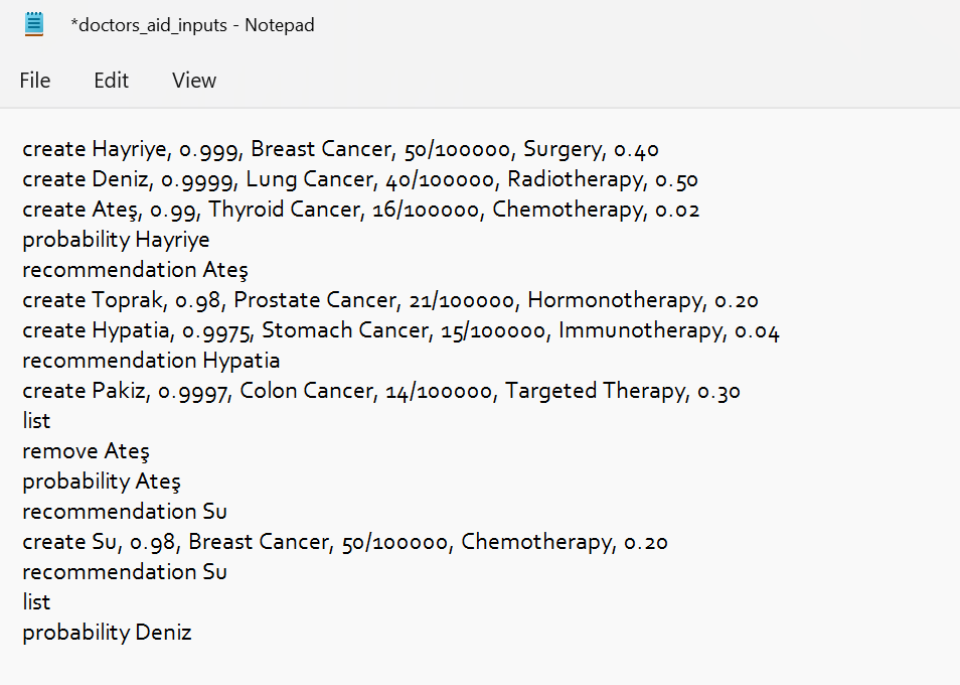
This assignment focuses on a probability-based decision system called Doctor's Aid for clinicians.

"*Worldwide, an estimated 19.3 million new cancer cases and almost 10.0 million cancer deaths occurred in 2020. Female breast cancer has surpassed lung cancer as the most commonly diagnosed cancer, with an estimated 2.3 million new cases (11.7%). However, after prevention, early detection and treatment are possible via mammographic screening for breast cancer. Still, it has limitations, such as overdiagnosis and overtreatment. Overdiagnosis is a critical issue in understanding the potential harms of screening for any cancer. In addition to causing harm through overtreatment, it labels a person as having cancer, with all of the psychological, financial, time, and opportunity costs that may entail, regardless of whether the cancer is treated or surveilled.*"

For this aid utility, patient name, diagnosis accuracy, patient's disease name, disease incidence, treatment name, and the treatment risk probability will be given in an input file named "doctors_aid_inputs.txt" . The program should read lines from the input text files and turn them into command lines, which will generate an output that will be written to another text file.

Data

In Assignment2 we are provided with the input text file “doctors_aid_inputs.txt” to test the working of the Doctor’s Aid for clinicians system.



Using the input text file given above, Doctor’s Aid for clinicians system is expected to generate and write the output to “doctors_aid_outputs.txt” as follows:

doctors_aid_outputs - Notepad

File Edit View

Patient Hayriye is recorded.
Patient Deniz is recorded.
Patient Ateş is recorded.
Patient Hayriye has a probability of 33.32% 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 Name	Diagnosis Accuracy	Disease Name	Disease Incidence	Treatment Name	Treatment 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 Therapy	30%

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 Name	Diagnosis Accuracy	Disease Name	Disease Incidence	Treatment Name	Treatment Risk
Hayriye	99.90%	Breast Cancer	50/100000	Surgery	40%
Deniz	99.99%	Lung Cancer	40/100000	Radiotherapy	50%

doctors_aid_outputs - Notepad

File Edit View

Patient Hypatia is recorded.
System suggests Hypatia to have the treatment.
Patient Pakiz is recorded.

Patient Name	Diagnosis Accuracy	Disease Name	Disease Incidence	Treatment Name	Treatment 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 Therapy	30%

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 Name	Diagnosis Accuracy	Disease Name	Disease Incidence	Treatment Name	Treatment 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 Therapy	30%
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.

Analysis

In this section, I talk about the analysis of the problem of the assignment.

Our program should be able to read lines from any input text file and use these lines as command lines. The program should recognize the command line using the initial word of every line. The program is going to use if statements to check the word at the beginning of every line. After recognizing the command line, the program is going to call previously defined functions to solve the problem.

The code should contain at least 8 functions to perform efficiently. These functions will be discussed in detail in the following section.

After calling the function, and obtaining the desired output, our program should be able to write the output to the desired text file.

Design

In this section, I will discuss the design of the code.

1. The code starts with importing the following modules:
    1.1. `from tabulate import tabulate`

        This is going to be helpful while turning our list into a table later

    1.2. `import copy`

        This is going to be helpful to take a deep copy of our list to which some modifications will be made without affecting the main list

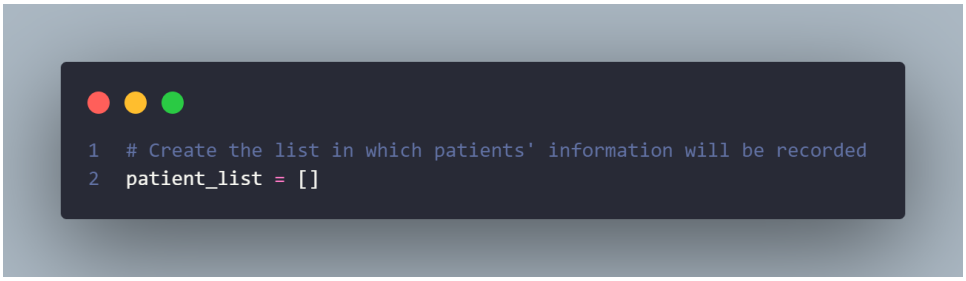
2. Function `read()`

```
# Function to read the lines from the input text file
def read():
    f0 = open("doctors_aid_inputs.txt", "r")
    read = f0.readlines()
    return read
```

3. Function `export()`

```
#  Function to write the output to the output text file
def export():
    f1 = open("outputs.txt", "a")
    return f1
```

4. Create empty list `patient_list = []`



```
# Create the list in which patients' information will be recorded
patient_list = []
```

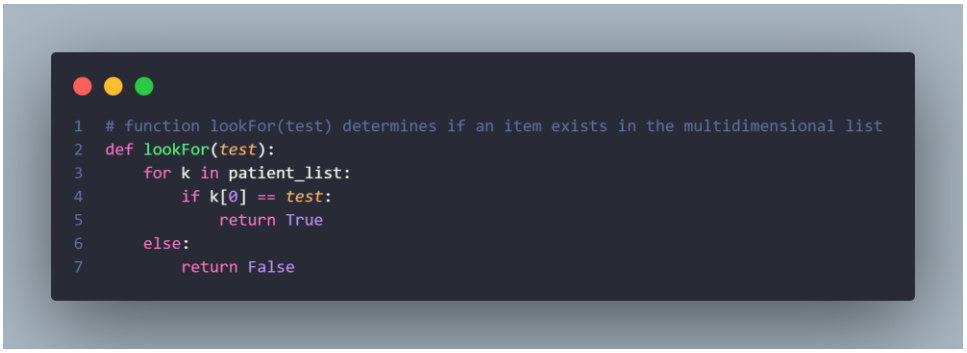
5. Function `create(x)`

```python
# create(x) function creates a new row in the list patient_list
# parameter x will take the line in input file that starts with "create"


def create(x):
    e = export()
    # split x where there exists a comma
    patient = x.split(",")
    # remove create from the string
    patient[0] = patient[0].replace("create ", "")
    # remove unnecessary white space
    patient[1] = patient[1].strip()
    patient[2] = patient[2].strip()
    patient[3] = patient[3].strip()
    patient[4] = patient[4].strip()
    patient[5] = patient[5].strip()
    patient = patient[0 : len(patient) + 1]
    t_f = patient in patient_list
    # if the patient's information already exists in the table
    if t_f:
        write_this = "Patient ", patient[0], " cannot be recorded due to duplication.\n"
        e.writelines(write_this)
    else:
        # append new record to patient_list
        patient_list.append(patient)
        write_this = "Patient ", patient[0], " is recorded.\n"
        e.writelines(write_this)

```

6. Function `lookFor(y)`



```python
# function lookFor(test) determines if an item exists in the multidimensional list
def lookFor(test):
    for k in patient_list:
        if k[0] == test:
            return True
    else:
        return False
```

7. Function `remove_item(item)`

```python
# function remove_item(item) removes a patient's record
def remove_item(item):
    e = export()
    for k in patient_list:
        if k[0] == item:
            # remove the row from the table
            patient_list.remove(k)
            write_this = "Patient ", item, " is removed.\n"
            e.writelines(write_this)

    # if patient's name doesn't already exist in the record
    else:
        write_this = "Patient ", item, " cannot be removed due to absence\n"
        e.writelines(write_this)

```

8. Function `convert_to_float(frac_str)`

```python
# function convert_to_float(frac_str) converts a string fraction to a float
def convert_to_float(frac_str):
    num, denom = frac_str.split("/")
    num = int(num)
    denom = int(denom)
    cal = num / denom
    return cal

```

9. Function `probability(y)`

```python
# function probability(y) calculates the probability using values from the table
def probability(y):
    e = export()
    # find if the patient's name exists in the table
    if lookFor(y):
        # find the index of the patient's record in the table (row index)
        for p in patient_list:
            if p[0] == y:
                index_prob = patient_list.index(p)  # this is the row index
                # incidence is found in patient_list row index = index_prob, column index = 3
                # call function convert_to_float() since incidence is a string fraction
                incidence = convert_to_float(patient_list[index_prob][3])
                # accuracy is found in patient_list row index = index_prob, column index = 1
                accuracy = float(patient_list[index_prob][1])
                # formula for probability
                prob = incidence / (1 - accuracy + incidence)
                # find probability as a percentage
                prob = round(prob * 100, 2)
                # find the type of cancer the patient has
                type_of_cancer = patient_list[index_prob][2]
                write_this = (
                    "Patient ",
                    y,
                    " has a probability of ",
                    str(prob),
                    "% of having ",
                    type_of_cancer,
                    ". \n",
                )
                e.writelines(write_this)
                return prob
    # if the patient's name cannot be found in the records
    else:
        write_this = "Probability for ", y, " cannot be calculated due to absence. \n"
        e.writelines(write_this)

```

10. Function `prob_for_recommendation(yy)`

```python
# This function does the same job as probability(y) function except printitng the output
def prob_for_recommendation(yy):
    e = export()
    # find if the patient's name exists in the table
    if lookFor(yy):
        # find the index of the patient's record in the table (row index)
        for p in patient_list:
            if p[0] == yy:
                index_prob = patient_list.index(p)  # this is the row index
                # incidence is found in patient_list row index = index_prob, column index = 3
                # call function convert_to_float() since incidence is a string fraction
                incidence = convert_to_float(patient_list[index_prob][3])
                # accuracy is found in patient_list row index = index_prob, column index = 1
                accuracy = float(patient_list[index_prob][1])
                # formula for probability
                prob = incidence / (1 - accuracy + incidence)
                # find probability as a percentage
                return prob
```

11. Function `recommendation(mm)`

```
# function recommendation(mm) provides treatment recommendation based on the patient's data
def recommendation(mm):
    e = export()
    # find if patient's name exists in the table
    if lookFor(mm):
        # find the index of the patient's record in the table (row index)
        for q in patient_list:
            if q[0] == mm:
                index_risk = patient_list.index(q)  # row index
                # risk is found in patient_list row index = index_risk, column index = 5
                risk = float(patient_list[index_risk][5])
                # call function probability(mm)
                proba = prob_for_recommendation(mm)
                if risk < proba:
                    write_this = "System suggests ", mm, " NOT to have the treatment\n"
                    e.writelines(write_this)
                else:

                    write_this = "System suggests ", mm, " to have the treatment\n"
                    e.writelines(write_this)

    # if patient's name does Not exist in the table
    else:
        write_this = (
            "Recommendation for ",
            mm,
            " cannot be calculated due to absence.\n",
        )
        e.writelines(write_this)

```

12. Function `list_t(z)`

```
# function list_t(z) turns list into a table


def list_t(z):
    # creating table structure & column headers
    q = (
        "Patient\tDiagnosis\tDisease\t\t\tDisease\t\tTreatment\t\tTreatment\n"
        "Name\tAccuracy Name\t\t\tIncidence Name\t\t\tRisk\n"
        "-------------------------------------------------------------------------------\n"
    )
    # make the columns in the table align with proper whitespaces
    for k in z:
        if len(k[0]) < 4:
            q += k[0] + "\t\t"
        elif len(k[0]) < 8:
            q += k[0] + "\t"
        else:
            q += k[0]

        if len(k[1]) < 4:
            q += k[1] + "\t\t\t"
        elif len(k[1]) < 8:
            q += k[1] + "\t\t"
        elif len(k[1]) < 12:
            q += k[1] + "\t"
        else:
            q += k[1]

        if len(k[2]) < 4:
            q += k[2] + "\t\t\t\t"
        elif len(k[2]) < 8:
            q += k[2] + "\t\t\t"
        elif len(k[2]) < 12:
            q += k[2] + "\t\t"
        elif len(k[2]) < 16:
            q += k[2] + "\t"
        else:
            q += k[2]

        q += k[3] + "\t"

        if len(k[4]) < 4:
            q += k[4] + "\t\t\t\t"
        elif len(k[4]) < 8:
            q += k[4] + "\t\t\t"
        elif len(k[4]) < 12:
            q += k[4] + "\t\t"
        elif len(k[4]) < 16:
            q += k[4] + "\t"
        else:
            q += k[4]

        q += k[5] + "\n"

    e = export()
    e.writelines(q)
```

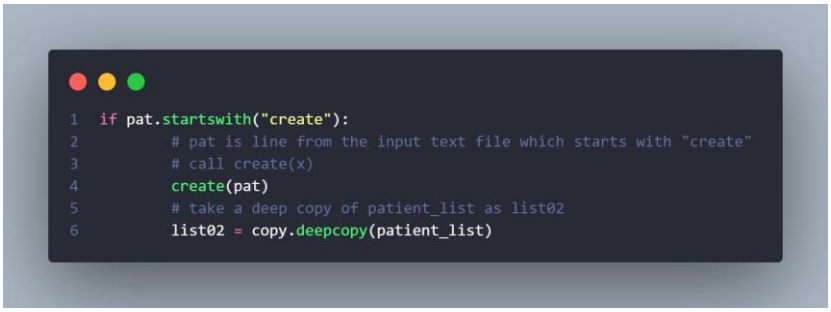
13. For loop to read the lines in input file :

```
for pat in read():
```

Call function `read()` to return read

14. if statements to check the initial word of every line

14.1. If first word = Create



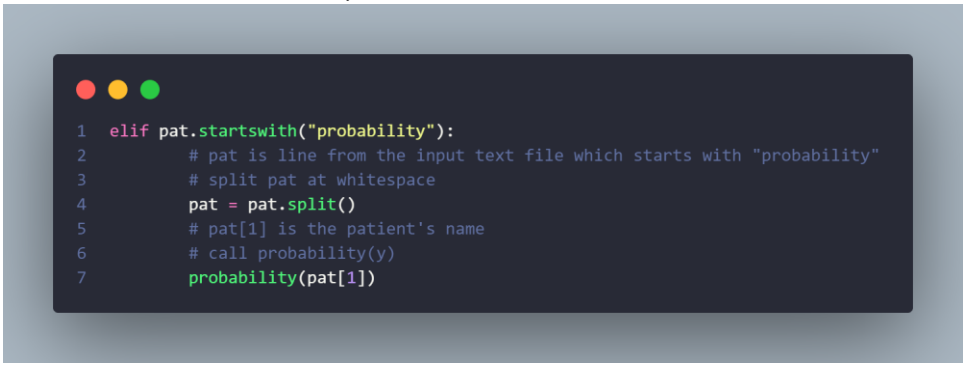
```
if pat.startswith("create"):
        # pat is line from the input text file which starts with "create"
        # call create(x)
        create(pat)
        # take a deep copy of patient_list as list02
        list02 = copy.deepcopy(patient_list)
```

14.2. If first word = Remove

```
elif pat.startswith("remove"):
        # pat is line from the input text file which starts with "remove"
        # split pat at whitespace
        pat = pat.split()
        # pat[1] is the patient's name
        # call remove_item(item)
        remove_item(pat[1])
        # take a deep copy of patient_list as list02
        list02 = copy.deepcopy(patient_list)

```

14.3. If first word = Probability



```
elif pat.startswith("probability"):
        # pat is line from the input text file which starts with "probability"
        # split pat at whitespace
        pat = pat.split()
        # pat[1] is the patient's name
        # call probability(y)
        probability(pat[1])
```

14.4. If first word = Recommendation

```
elif pat.startswith("recommendation"):
        # pat is line from the input text file which starts with "recommendation"
        # split pat at whitespace
        pat = pat.split()
        # pat[1] is the patient's name
        # call recommendation(mm)
        recommendation(pat[1])
```

14.5. If first word = list

```
elif pat.startswith("list"):
      for i in list02:
          for g in i:
              g = g.strip()
          i[1] = float(i[1]) * 100
          i[1] = str(i[1]) + "%"
          i[5] = float(i[5]) * 100
          i[5] = str(int(i[5])) + "%"
      # call list_t(z)
      list_t(list02)
```

For loop inside the if statement here is responsible to modify some of the values inside list02(patient_list deep copy)

Programmer's Catalogue

In this section, I will discuss the time it took me to analyze the problem of this assignment, design the code, as well as test and debug. I will also discuss the reusability of this code.

Time:

Analysis: I took the longest time to analyze the problem of the function and try to come up with a pseudocode to solve it. I had completely different ideas of how to approach the solution to this problem, but I decided to choose the most efficient one possible.

Design: Designing the code for this problem took me relatively the shortest time among other steps. It took me between 7-10 hours to write proper functions and arrange them together in the most efficient manner.

Testing and debugging: Testing and debugging the code took me a slightly longer time than designing the code since I ran into some logical errors, it took me between 12-14 hours to test and debug this code.

Reusability:

This code can be used to read any input file that contains command lines starting with:

"create" ---> creates a new record in the table

"remove" ---> remove any record in the table

"Probability " ---> calculates the probability of the patient having the disease

"Recommendation" ---> determines if the patient should take the treatment

"List" ---> displays the contents of the table

However, for this code to be reusable the text file must be changed in the read function.

User Catalogue

How to use Doctor's aids for clinicians:

1) ***Create a new record by typing a command line in the following manner:***

   create Patient name, Diagnosis Accuracy, Disease Name, Disease Incidence, Treatment Name, Treatment risk

2) ***Remove a record from the table by typing a command line in the following manner:***

   remove Patient name

3) ***Compute the probability of the patient having a particular disease by typing a command line in the following manner:***

   probability Patient name

4) ***Compute the system recommendation by typing a command line in the following manner:***

   recommendation Patient name

5) ***Display the list of the patients by typing a command line in the following manner:***

   list

Pay attention to the following:

1) ***This program is case-sensitive:***

   Make sure you type your commands in lowercase to get the desired output from the program.

2) ***Avoid making typos or spelling mistakes:***
   - Avoid making typos while writing the commands otherwise, the program will not be able to recognize the command and will fail to return an output
   - Avoid making spelling mistakes while writing the commands otherwise, the program will fail to locate the data of the patient's data in the record.

3) ***You will Not be able to able to compute the Probability or the Recommendation of a patient whose data doesn't exist in the records:***

   You will not be able to compute the probability or recommendation of a particular patient unless the patient's data is available in the records.