Hacettepe University

Department of Artificial Intelligence Engineering

BBM103 Assignment 2 Report

Bora Dere – 2220765021

20/11/2022

Contents

Analysis

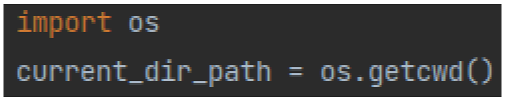
Cancer is one of the biggest threats to human life, out of question. And since there are lots of different kinds of cancer, it is difficult to speak in general about it. But one thing we all can agree on, early diagnosis and correct decision-making is fatally important. So, we need all the help that we can get. CDSS (Clinical Decision Support System) is one of them. A CDSS can be modified to satisfy different needs, which makes it flexible and that is one of the great advantages of it.

A CDSS basically does doctors' work and help them. It can list patient data, which are critically important since we are talking about human life. It can help in decision-making and eliminates the human error factor.

Design

Importing os library

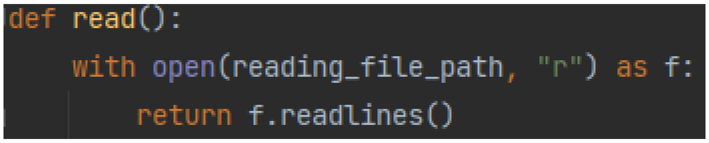
These 2 lines of code imports the os library (which is required in this assignment) and sets the *current_dir_path* as *os.getcwd()* (which returns a string representing the current working directory)[1]:



```
import os
current_dir_path = os.getcwd()
```

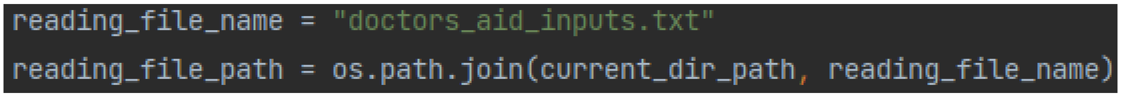
read function

Function *read* is defined to read the text input from the *doctors_aid_inputs.txt* and prepare it to use in the following functions. Definition:



```
def read():
    with open(reading_file_path, "r") as f:
        return f.readlines()
```

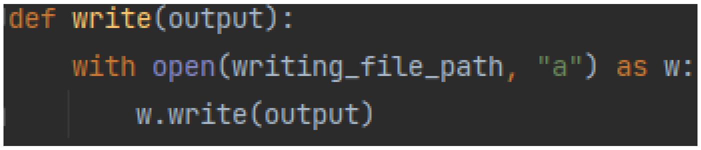
reading_file_path is the path of the input file and it is defined by using the os library in Python, as below:



```
reading_file_name = "doctors_aid_inputs.txt"
reading_file_path = os.path.join(current_dir_path, reading_file_name)
```

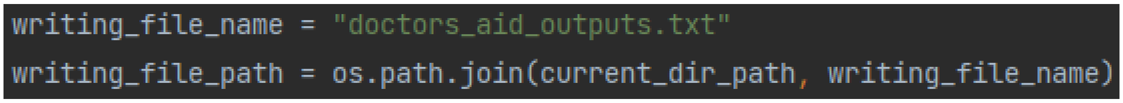
write function

Function *write* is defined to writing the output for each function to the output file.
Definition:



```
def write(output):
    with open(writing_file_path, "a") as w:
        w.write(output)
```

writing_file_path is the path of the output file and it is defined by using the os library in Python, as below:



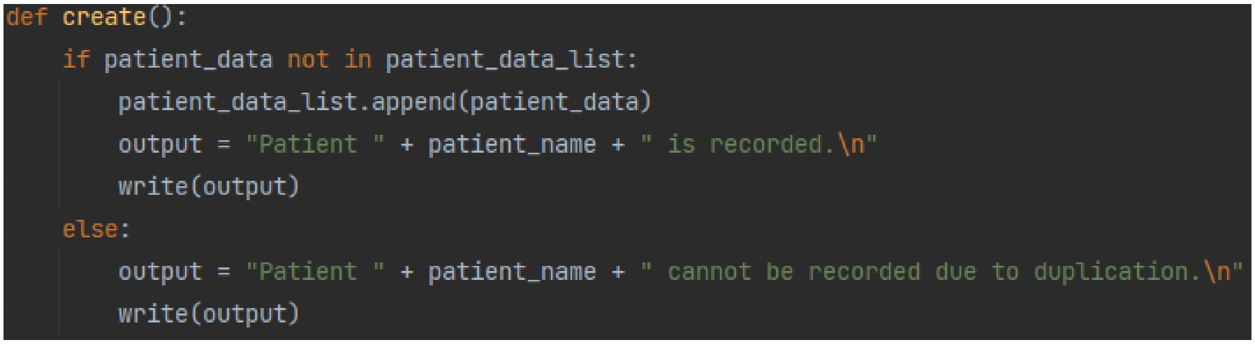
```
writing_file_name = "doctors_aid_outputs.txt"
writing_file_path = os.path.join(current_dir_path, writing_file_name)
```

What it does:

1. Take output from the function.
2. Open the output file to append the necessary output.
3. Write the output to the output file.

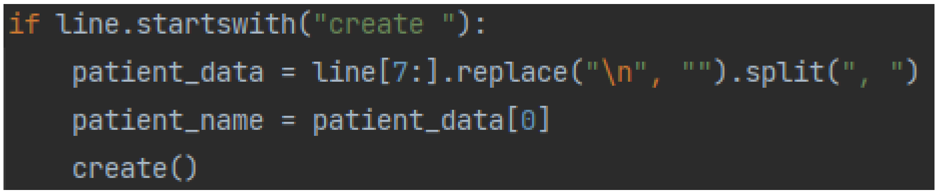
create function

Function *create* is defined to creating the patient and adding the patient data to *patient_data_list.* Definition:



```
def create():
    if patient_data not in patient_data_list:
        patient_data_list.append(patient_data)
        output = "Patient " + patient_name + " is recorded.\n"
        write(output)
    else:
        output = "Patient " + patient_name + " cannot be recorded due to duplication.\n"
        write(output)
```

patient_data is the complete data of a single patient. This function checks if the patient whom is tried to be added to the *patient_data_list* has been added before. If they are not added before, it adds them to the list. Otherwise, it states that they were already added to the list. *patient_name* here is defined in the driver code (which shall be explained furtherly):



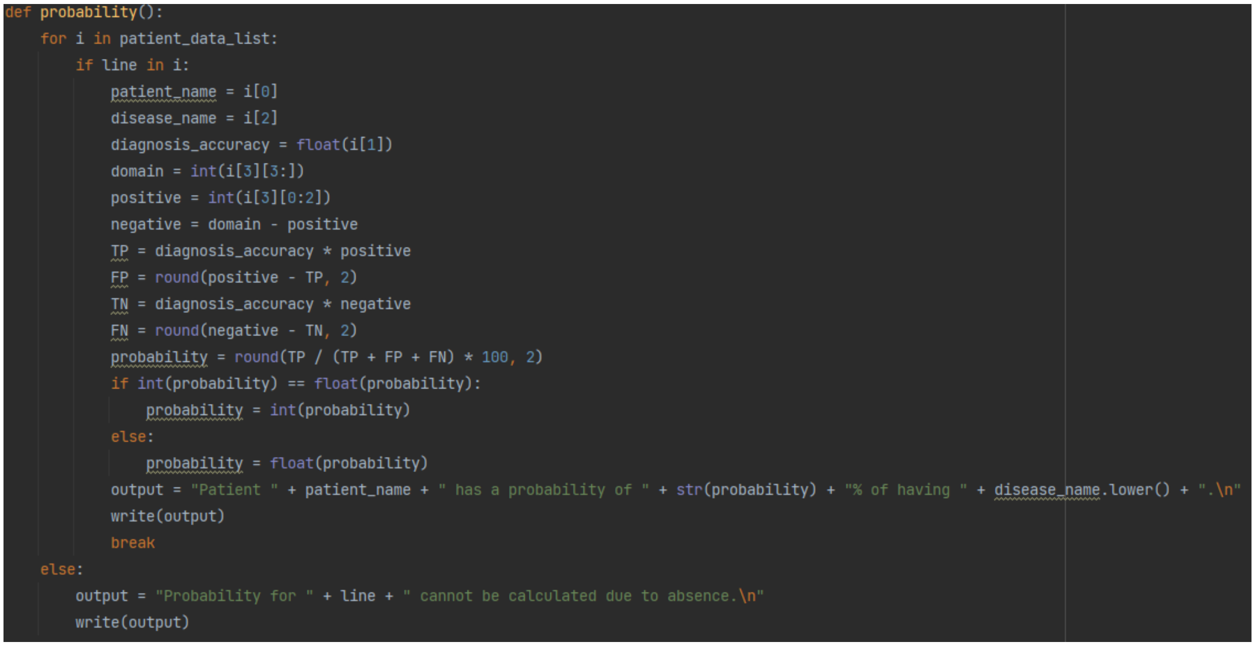
```
if line.startswith("create "):
    patient_data = line[7:].replace("\n", "").split(", ")
    patient_name = patient_data[0]
    create()
```

What it does:

1. Check if patient has been recorded before.
2. If yes, state that they will not be recorded again and exit the function.
3. If not, add the specified patient's data to the list and state that they have been added.
4. For both cases write the corresponding output.

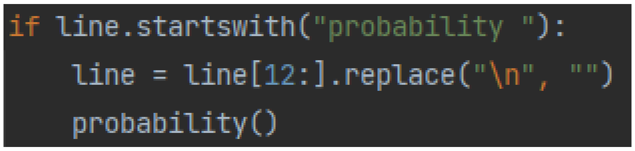
probability function

Function *probability* is defined to calculate the actual probability of the patient having the specified disease. It does this calculation based on the threat score method in error matrix.[2] Definition:



```
def probability():
    for i in patient_data_list:
        if line in i:
            patient_name = i[0]
            disease_name = i[2]
            diagnosis_accuracy = float(i[1])
            domain = int(i[3][3:])
            positive = int(i[3][0:2])
            negative = domain - positive
            TP = diagnosis_accuracy * positive
            FP = round(positive - TP, 2)
            TN = diagnosis_accuracy * negative
            FN = round(negative - TN, 2)
            probability = round(TP / (TP + FP + FN) * 100, 2)
            if int(probability) == float(probability):
                probability = int(probability)
            else:
                probability = float(probability)
            output = "Patient " + patient_name + " has a probability of " + str(probability) + "% of having " + disease_name.lower() + ".\n"
            write(output)
            break
    else:
        output = "Probability for " + line + " cannot be calculated due to absence.\n"
        write(output)
```

patient_name, disease_name, diagnosis_accuracy, domain, positive, negative, TP, FP, TN, FN and *probability* are all defined and calculated from the data in *patient_data_list* and necessary to complete the output files. *line* in the second output file is defined in the driver code (which shall be explained furtherly):



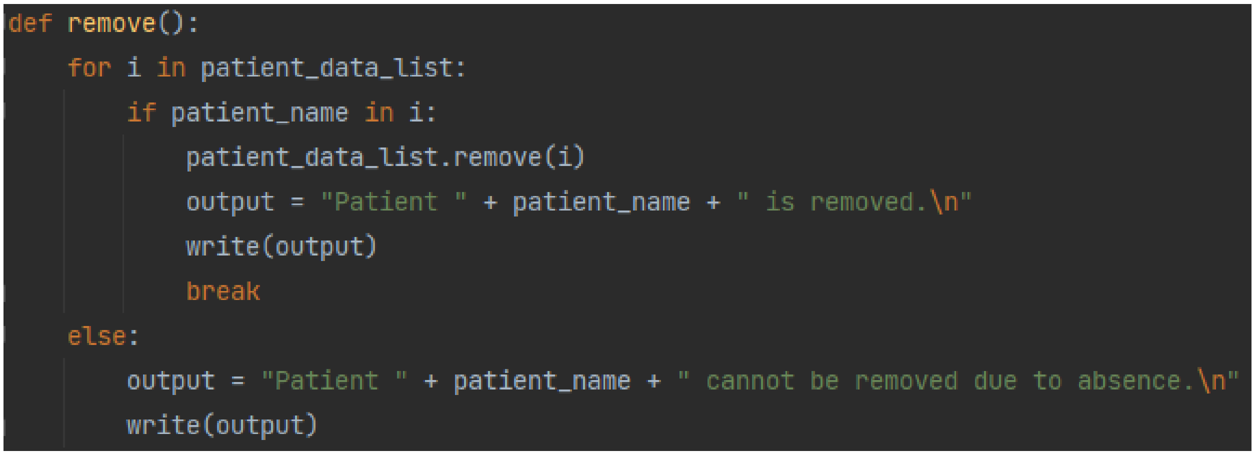
```
if line.startswith("probability "):
    line = line[12:].replace("\n", "")
    probability()
```

What it does:

1. Check if patient has been recorded before.
2. If yes, state that they will not be recorded again and exit the function.
3. If not, calculate and define *patient_name, disease_name, diagnosis_accuracy, domain, positive, negative, TP, FP, TN, FN* and *probability.*
4. For both cases write the corresponding output.

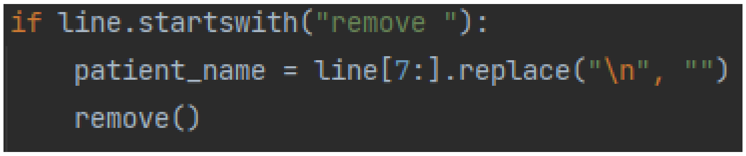
remove function

Function *remove* is defined to remove the specified patient’s data from the *patient_data_list*. Definition:



```
def remove():
    for i in patient_data_list:
        if patient_name in i:
            patient_data_list.remove(i)
            output = "Patient " + patient_name + " is removed.\n"
            write(output)
            break
    else:
        output = "Patient " + patient_name + " cannot be removed due to absence.\n"
        write(output)
```

patient_name is defined in the driver code (which shall be explained furtherly):



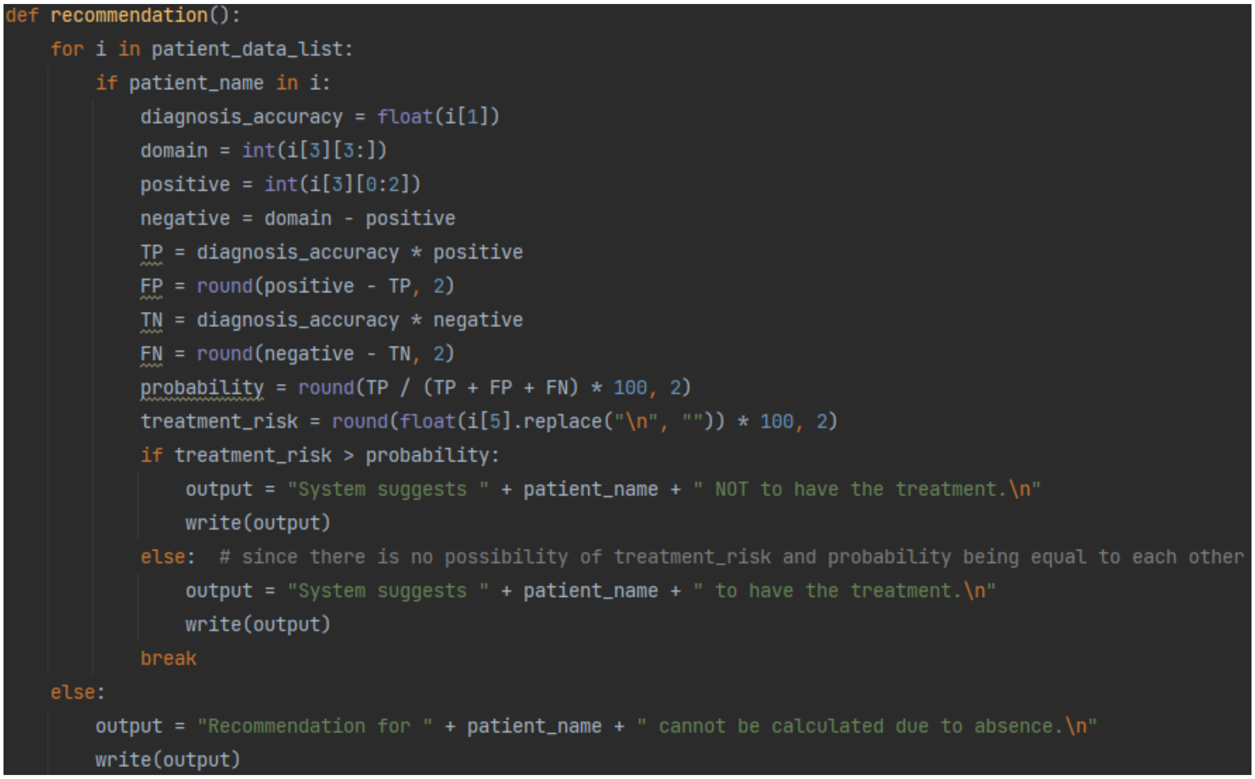
```
if line.startswith("remove "):
    patient_name = line[7:].replace("\n", "")
    remove()
```

What it does:

1. Check if patient has been recorded before.
2. If yes, remove the patient data from the list and state that they have been removed.
3. Break the loop (since I used for-else structure).
4. If patient data was not in the list, state that they will not be removed since they never existed.
5. For both cases write the corresponding output.

recommendation function

Function *recommendation* is defined to compare the treatment risk and actual probability of patient having the specified disease and deciding if they need to have the treatment. While doing the probability calculation, calling the *probability* function and using the value it returns is also an option but I wanted that calculation to be seen clearly under the *recommendation* function too. Definition:

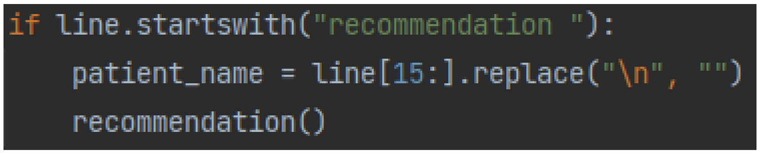


```python
def recommendation():
    for i in patient_data_list:
        if patient_name in i:
            diagnosis_accuracy = float(i[1])
            domain = int(i[3][3:])
            positive = int(i[3][0:2])
            negative = domain - positive
            TP = diagnosis_accuracy * positive
            FP = round(positive - TP, 2)
            TN = diagnosis_accuracy * negative
            FN = round(negative - TN, 2)
            probability = round(TP / (TP + FP + FN) * 100, 2)
            treatment_risk = round(float(i[5].replace("\n", "")) * 100, 2)
            if treatment_risk > probability:
                output = "System suggests " + patient_name + " NOT to have the treatment.\n"
                write(output)
            else:  # since there is no possibility of treatment_risk and probability being equal to each other
                output = "System suggests " + patient_name + " to have the treatment.\n"
                write(output)
            break
    else:
        output = "Recommendation for " + patient_name + " cannot be calculated due to absence.\n"
        write(output)
```

Sound explanation and reasoning

As you can see, I did the probability calculations here again instead of returning the *probability* function's result. Because I think being able to see those calculations at one sight where they are used is important. That way, someone wants to examine my code will not need to scroll up to see what my code does at background. It might look inefficient but the code would still do the same calculations even if I returned the value from *probability* function. So, there is actually not that much of a difference efficiency-wise.

Most of the variables are the same with the *probability* function. *treatment_risk* is new here and it is the risk of the treatment. *patient_name* is defined in the driver code (which shall be explained furtherly):



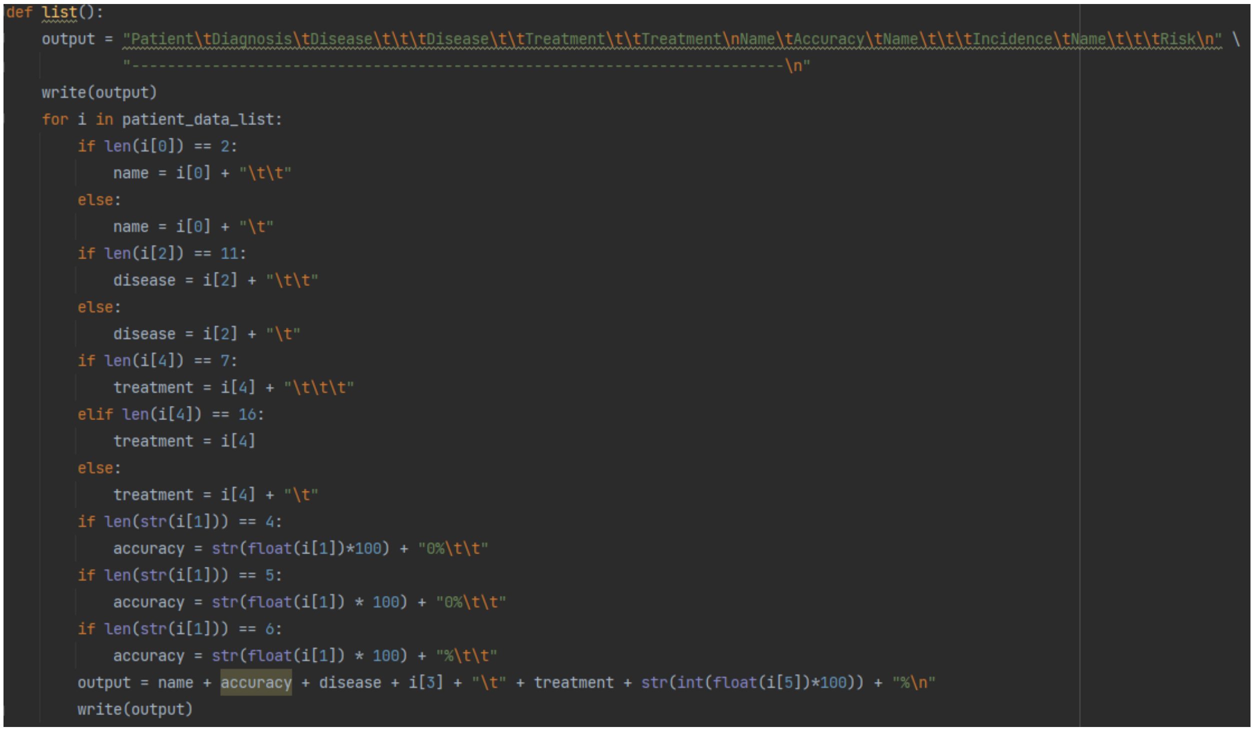
```
if line.startswith("recommendation "):
    patient_name = line[15:].replace("\n", "")
    recommendation()
```

What it does:

1. Check if the specified patient exists in *patient_data_list*.
2. If yes, calculate the *probability* and *treatment_risk* variables.
    a. If *treatment_risk* is greater than *probability* state that patient should not have the treatment.
    b. Else (since these two variables cannot be equal) state that patient should have the treatment.
    c. For both cases write the corresponding output.
    d. Break the loop (since I used for-else structure).
3. If not, state that system cannot make a recommendation since the specified patient does not exist in the list.
4. Write the corresponding output.

list function

Function *list* is defined to tabulate the current *patient_data_list* in the given format. It reads every element in *patient_data_list* and processes them to fit to the required format. Definition:



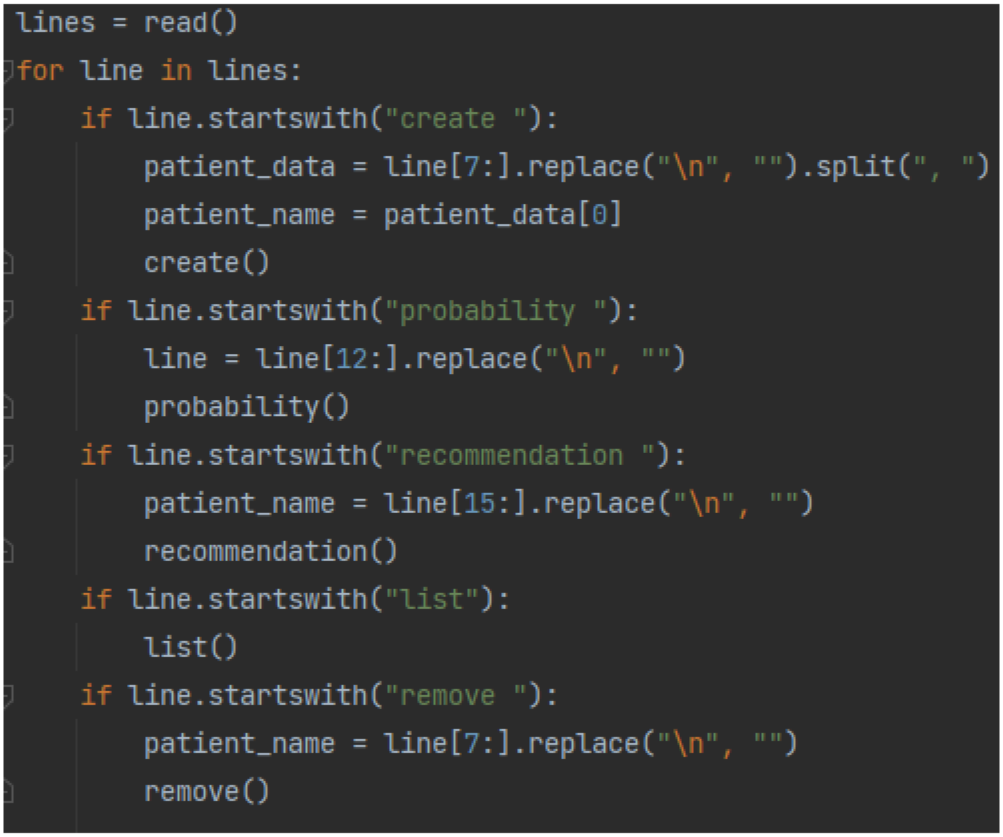
```
def list():
    output = "Patient\tDiagnosis\tDisease\t\t\tDisease\t\tTreatment\t\tTreatment\nName\tAccuracy\tName\t\t\tIncidence\tName\t\t\tRisk\n" \
             "-------------------------------------------------------------------------\n"
    write(output)
    for i in patient_data_list:
        if len(i[0]) == 2:
            name = i[0] + "\t\t"
        else:
            name = i[0] + "\t"
        if len(i[2]) == 11:
            disease = i[2] + "\t\t"
        else:
            disease = i[2] + "\t"
        if len(i[4]) == 7:
            treatment = i[4] + "\t\t\t"
        elif len(i[4]) == 16:
            treatment = i[4]
        else:
            treatment = i[4] + "\t"
        if len(str(i[1])) == 4:
            accuracy = str(float(i[1])*100) + "0%\t\t"
        if len(str(i[1])) == 5:
            accuracy = str(float(i[1]) * 100) + "0%\t\t"
        if len(str(i[1])) == 6:
            accuracy = str(float(i[1]) * 100) + "%\t\t"
        output = name + accuracy + disease + i[3] + "\t" + treatment + str(int(float(i[5])*100)) + "%\n"
        write(output)
```

What it does:

1. Create an output which should be written to the output file even though there is no patient record.
2. Write that obligatory output to create the main parts of the list.
3. Check i[0]'s (patient's name) length. It is an exception for patient Su.
4. If name's length is equal to 2 (patient Su) insert two tabs after it.
5. If not, insert 1 tab.
6. Check i[2]'s (patient's disease) length. It is an exception for Lung Cancer.
7. If disease's length is equal to 11 (Lung Cancer) insert two tabs after it.
8. If not, insert 1 tab.
9. Check i[4]'s (treatment's) length. It is used to make an exception for Surgery and Targeted Therapy.
10. If treatment's length is equal to 7 (Surgery) insert 3 tabs after it.
11. If treatment's length is equal to 16 (Targeted Therapy) insert nothing after it.
12. Else, insert 1 tab after it.
13. Check i[1]'s (accuracy's) length. There are 3 possibilities for it. Processes below this matter are made to fit the accuracy outputs to the format.
14. If it is equal to 4, add "0%\t\t" after it.
15. If it is equal to 5, add "0%\t\t" after it.
16. If it is equal to 6, add "%\t\t" after it.
17. Collect all the processed data in an output variable. Do the last formatting and adding processes.
18. Write the corresponding output.

Driver code

Driver code block checks the lines in the input file and calls the mentioned function.



```
lines = read()
for line in lines:
    if line.startswith("create "):
        patient_data = line[7:].replace("\n", "").split(", ")
        patient_name = patient_data[0]
        create()
    if line.startswith("probability "):
        line = line[12:].replace("\n", "")
        probability()
    if line.startswith("recommendation "):
        patient_name = line[15:].replace("\n", "")
        recommendation()
    if line.startswith("list"):
        list()
    if line.startswith("remove "):
        patient_name = line[7:].replace("\n", "")
        remove()
```

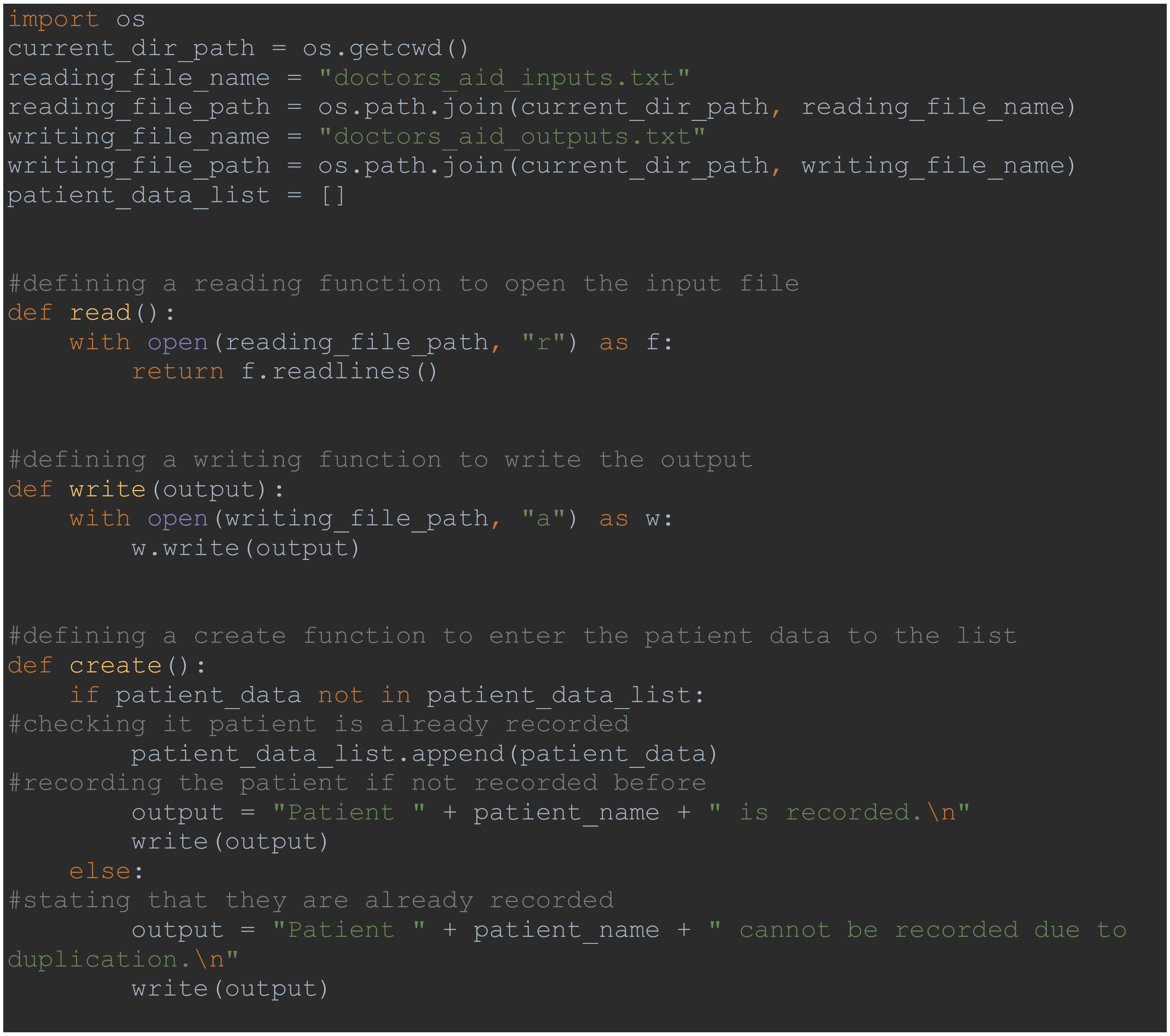
What it does:

1. Call the *read* function and assign its return as *lines*.
2. Check for the first word for each line.
3. Restrict the line content with respect to the first word of it if required.
4. For *create:*
    a. Define *patient_data* to record to the *patient_data_list* and *patient_name* from *patient_data*.
    b. Call the c*reate* function.
5. For *probability:*
    a. Define *line* to use in the *probability* function.
    b. Call the *probability* function.
6. For *recommendation:*
    a. Define *patient_name* from the line to use in the *recommendation* function.
    b. Call the *recommendation* function.
7. For *list:*
    a. Call the *list* function.
8. For *remove:*
    a. Define *patient_name* from the line to use in the *remove* function.
    b. Call the *remove* function.

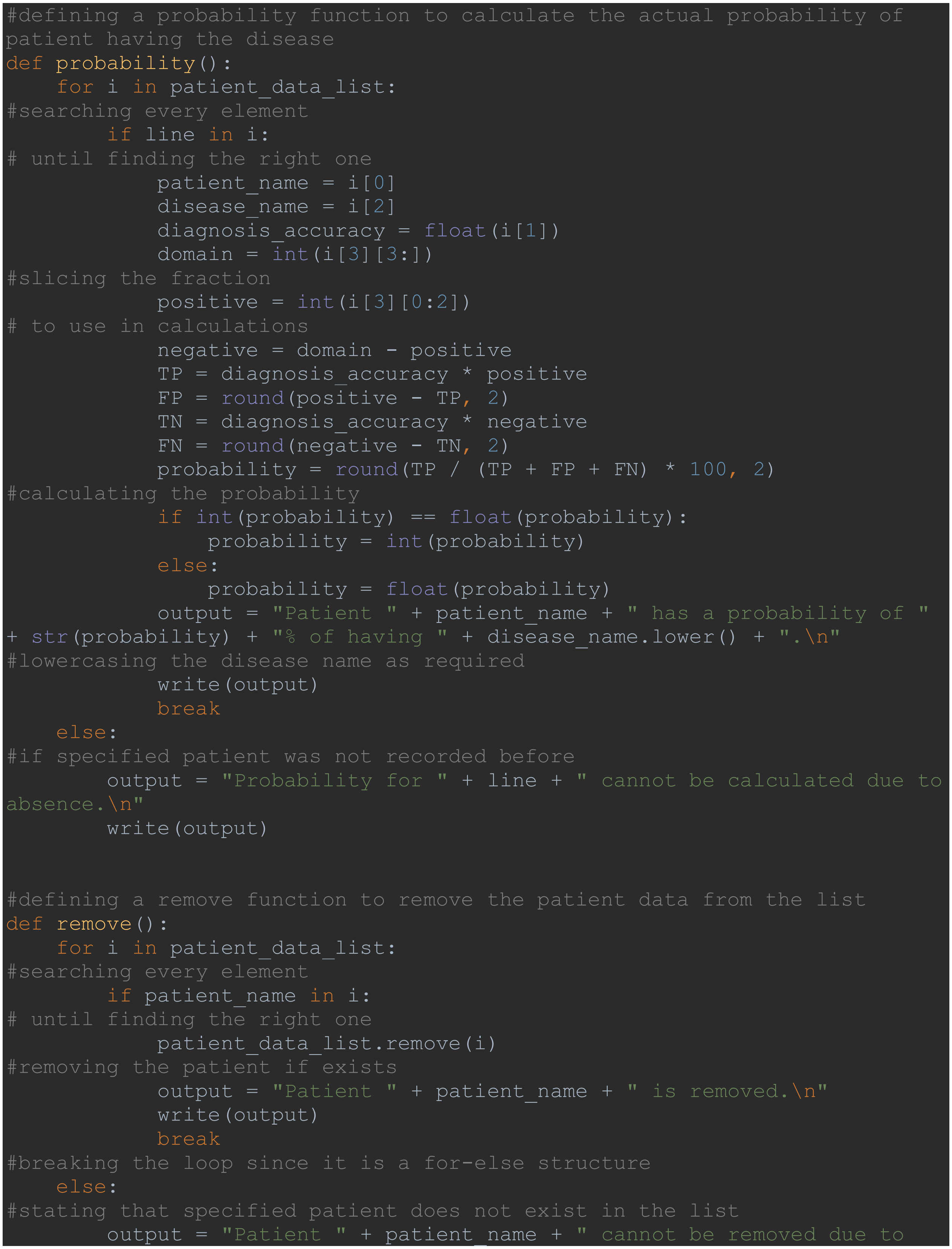
Programmer's Catalogue

This assignment was given at November 10th. At that day I could not do anything about it because when I received the mail from Piazza, I was at the meeting of ACM. Then we visited Anıtkabir as the ACM community to pay our respect to our great leader, Mustafa Kemal Atatürk. But from November 11th to today (20/11/2022) I have been trying, struggling, researching and doing everything that I can. My main step was at November 13th since it was the day that you answered our questions. Until that day, I actually had the same idea as now but I was not sure so I could not do anything about it.

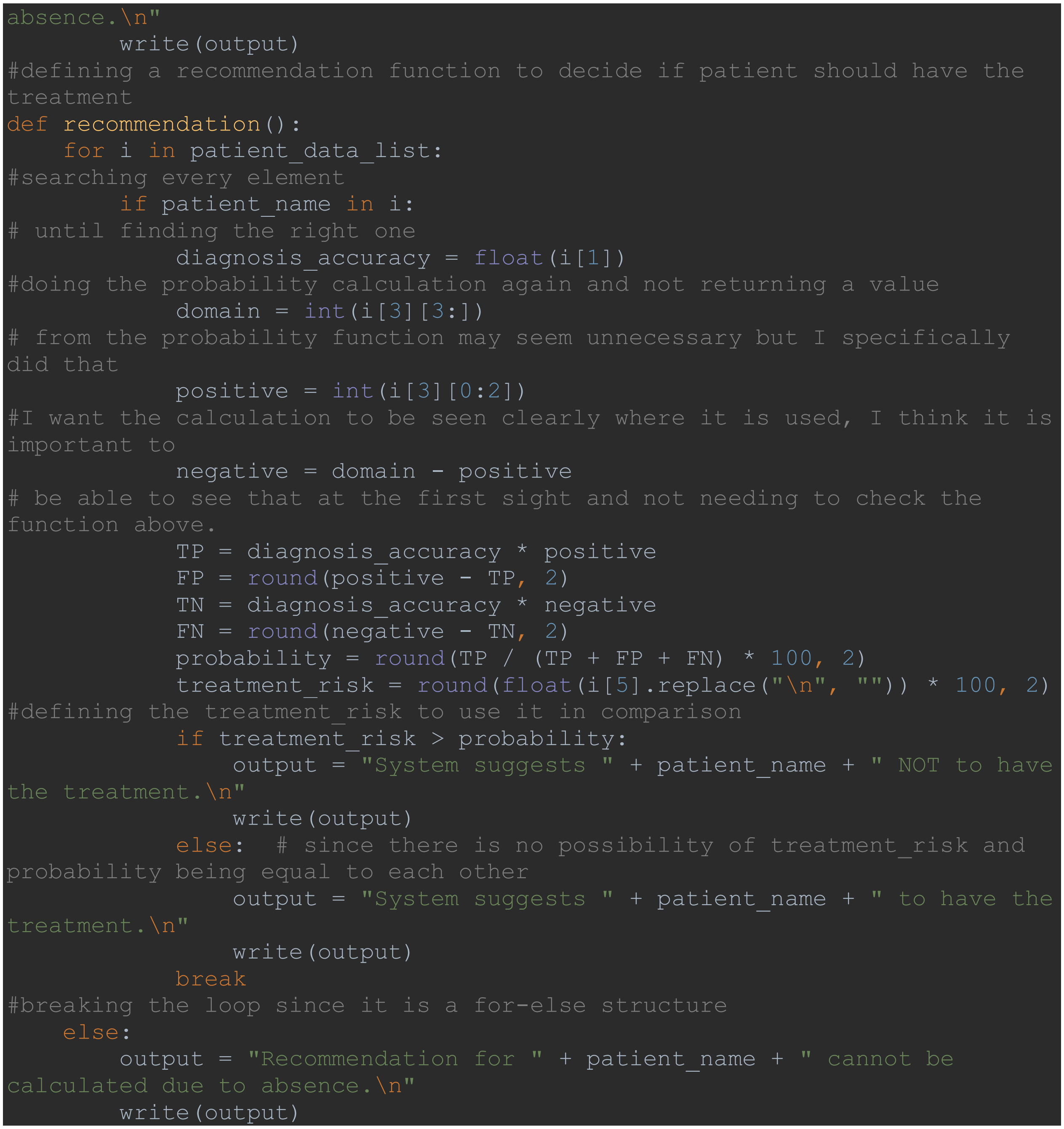
With the right input format and the patient data provided, code is usable for every possible scenario. Code itself can be found starting from the next page, part by part.



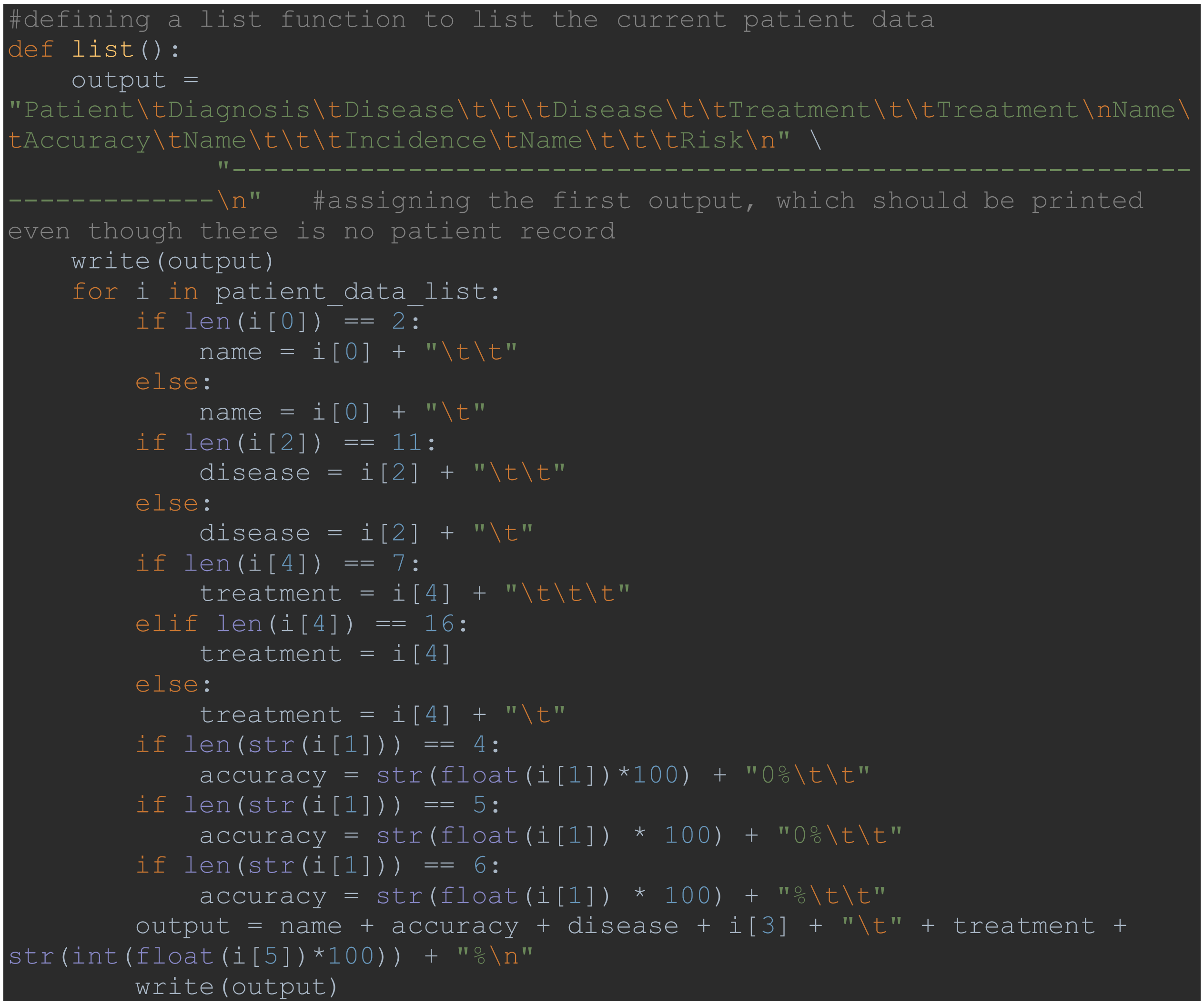
```
import os
current_dir_path = os.getcwd()
reading_file_name = "doctors_aid_inputs.txt"
reading_file_path = os.path.join(current_dir_path, reading_file_name)
writing_file_name = "doctors_aid_outputs.txt"
writing_file_path = os.path.join(current_dir_path, writing_file_name)
patient_data_list = []


#defining a reading function to open the input file
def read():
    with open(reading_file_path, "r") as f:
        return f.readlines()


#defining a writing function to write the output
def write(output):
    with open(writing_file_path, "a") as w:
        w.write(output)


#defining a create function to enter the patient data to the list
def create():
    if patient_data not in patient_data_list:
#checking it patient is already recorded
        patient_data_list.append(patient_data)
#recording the patient if not recorded before
        output = "Patient " + patient_name + " is recorded.\n"
        write(output)
    else:
#stating that they are already recorded
        output = "Patient " + patient_name + " cannot be recorded due to
duplication.\n"
        write(output)
```



```
#defining a probability function to calculate the actual probability of
patient having the disease
def probability():
    for i in patient_data_list:
#searching every element
        if line in i:
# until finding the right one
            patient_name = i[0]
            disease_name = i[2]
            diagnosis_accuracy = float(i[1])
            domain = int(i[3][3:])
#slicing the fraction
            positive = int(i[3][0:2])
# to use in calculations
            negative = domain - positive
            TP = diagnosis_accuracy * positive
            FP = round(positive - TP, 2)
            TN = diagnosis_accuracy * negative
            FN = round(negative - TN, 2)
            probability = round(TP / (TP + FP + FN) * 100, 2)
#calculating the probability
            if int(probability) == float(probability):
                probability = int(probability)
            else:
                probability = float(probability)
            output = "Patient " + patient_name + " has a probability of "
+ str(probability) + "% of having " + disease_name.lower() + ".\n"
#lowercasing the disease name as required
            write(output)
            break
    else:
#if specified patient was not recorded before
        output = "Probability for " + line + " cannot be calculated due to
absence.\n"
        write(output)


#defining a remove function to remove the patient data from the list
def remove():
    for i in patient_data_list:
#searching every element
        if patient_name in i:
# until finding the right one
            patient_data_list.remove(i)
#removing the patient if exists
            output = "Patient " + patient_name + " is removed.\n"
            write(output)
            break
#breaking the loop since it is a for-else structure
    else:
#stating that specified patient does not exist in the list
        output = "Patient " + patient_name + " cannot be removed due to
```



```
absence.\n"
        write(output)
#defining a recommendation function to decide if patient should have the
treatment
def recommendation():
    for i in patient_data_list:
#searching every element
        if patient_name in i:
# until finding the right one
            diagnosis_accuracy = float(i[1])
#doing the probability calculation again and not returning a value
            domain = int(i[3][3:])
# from the probability function may seem unnecessary but I specifically
did that
            positive = int(i[3][0:2])
#I want the calculation to be seen clearly where it is used, I think it is
important to
            negative = domain - positive
# be able to see that at the first sight and not needing to check the
function above.
            TP = diagnosis_accuracy * positive
            FP = round(positive - TP, 2)
            TN = diagnosis_accuracy * negative
            FN = round(negative - TN, 2)
            probability = round(TP / (TP + FP + FN) * 100, 2)
            treatment_risk = round(float(i[5].replace("\n", "")) * 100, 2)
#defining the treatment_risk to use it in comparison
            if treatment_risk > probability:
                output = "System suggests " + patient_name + " NOT to have
the treatment.\n"
                write(output)
            else:  # since there is no possibility of treatment_risk and
probability being equal to each other
                output = "System suggests " + patient_name + " to have the
treatment.\n"
                write(output)
            break
#breaking the loop since it is a for-else structure
    else:
        output = "Recommendation for " + patient_name + " cannot be
calculated due to absence.\n"
        write(output)
```



```
#defining a list function to list the current patient data
def list():
    output =
"Patient\tDiagnosis\tDisease\t\t\tDisease\t\tTreatment\t\tTreatment\nName\
tAccuracy\tName\t\t\tIncidence\tName\t\t\tRisk\n" \
             "------------------------------------------------------------
-------------\n"   #assigning the first output, which should be printed
even though there is no patient record
    write(output)
    for i in patient_data_list:
        if len(i[0]) == 2:
            name = i[0] + "\t\t"
        else:
            name = i[0] + "\t"
        if len(i[2]) == 11:
            disease = i[2] + "\t\t"
        else:
            disease = i[2] + "\t"
        if len(i[4]) == 7:
            treatment = i[4] + "\t\t\t"
        elif len(i[4]) == 16:
            treatment = i[4]
        else:
            treatment = i[4] + "\t"
        if len(str(i[1])) == 4:
            accuracy = str(float(i[1])*100) + "0%\t\t"
        if len(str(i[1])) == 5:
            accuracy = str(float(i[1]) * 100) + "0%\t\t"
        if len(str(i[1])) == 6:
            accuracy = str(float(i[1]) * 100) + "%\t\t"
        output = name + accuracy + disease + i[3] + "\t" + treatment +
str(int(float(i[5])*100)) + "%\n"
        write(output)
```

```
#Driver Code
lines = read()
#collecting all the data read with the read function
for line in lines:
#activating the correct function with respect to first word of the line
    if line.startswith("create "):
        patient_data = line[7:].replace("\n", "").split(", ")
#doing necessary slicing, splitting and replacing operations to obtain the
required format
        patient_name = patient_data[0]
        create()
    if line.startswith("probability "):
        line = line[12:].replace("\n", "")
#doing necessary slicing and replacing operations to obtain the required
format
        probability()
    if line.startswith("recommendation "):
        patient_name = line[15:].replace("\n", "")
#doing necessary slicing and replacing operations to obtain the required
format
        recommendation()
    if line.startswith("list"):
        list()
    if line.startswith("remove "):
        patient_name = line[7:].replace("\n", "")
#doing necessary slicing and replacing operations to obtain the required
format
        remove()
```

User Catalogue

To use the program, you must enter inputs to the *doctors_aid_inputs.txt* file with the correct format and patient data. You also must locate the *doctors_aid_inputs.txt and doctors_aid_outputs.txt* file to the same directory with the *Assignment2.py* file.

Usage of *create* function

You can use this function with all of the data provided in the *doctors_aid_inputs.txt* file.

Example: create Deniz, 0.9999, Lung Cancer, 40/100000, Radiotherapy, 0.50

Usage of *probability* function

You can use this function even without providing a patient entry before it. It handles that situation.

Example: probability Hayriye

Usage of *recommendation* function

You can use this function even without providing a patient entry before it. It handles that situation.

Example: recommendation Ateş

Usage of *list* function

You can use this function even without providing a patient entry before it but it would print an empty list. If you want a list including data, provide patient entry before it.

Example: list

Usage of remove function

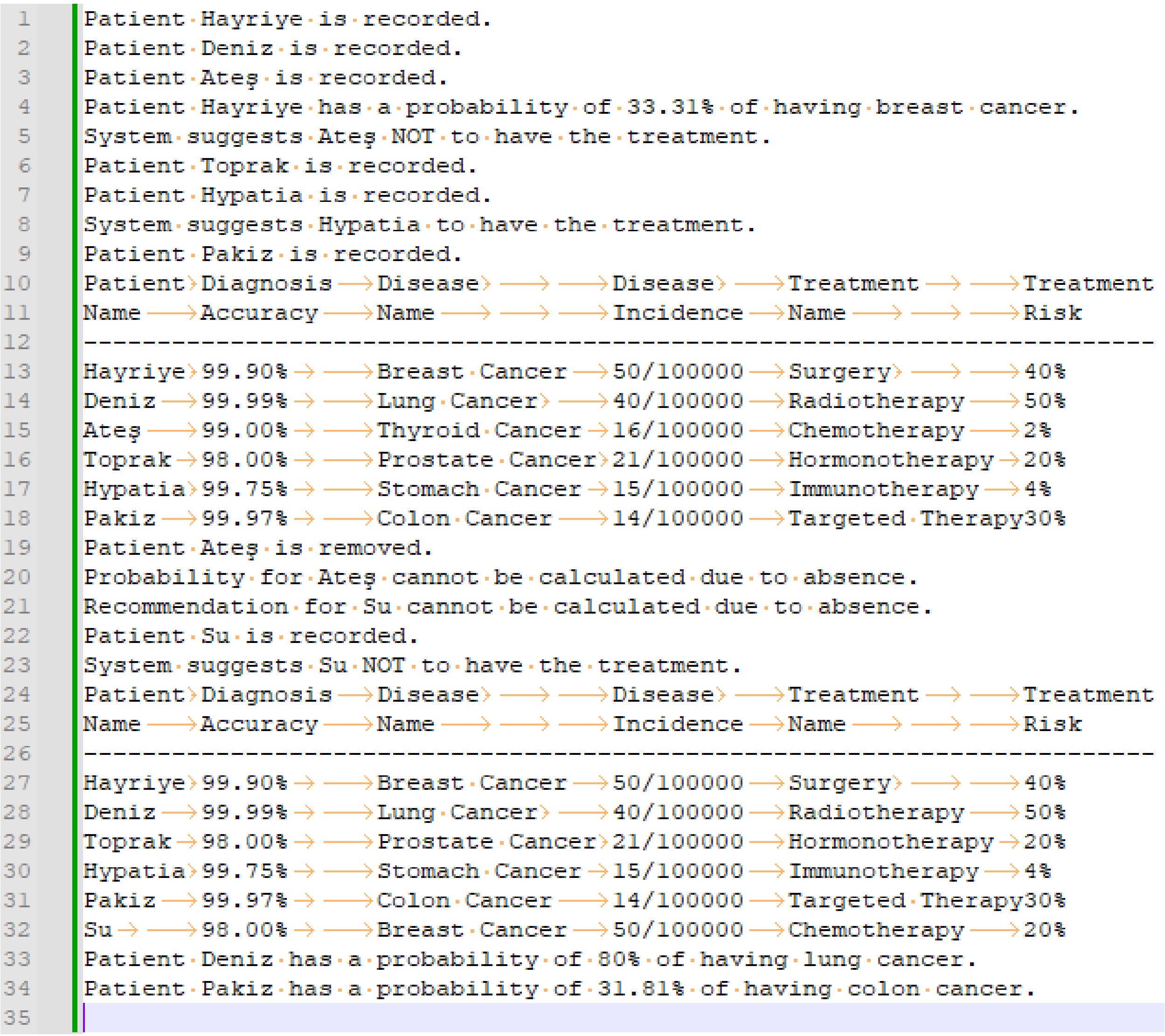
You can use this function even without providing a patient entry before it. It handles that situation.

Example: remove Hypatia

Restrictions:

If you enter *list* input without creating patients, it will print an empty list. Which should be fine as you declared in Piazza.

If you view the output text file in a text editor which takes 1 tab equal to 8 whitespaces, output of the *list* function may seem inordinate. You should use a text editor which takes 1 tab equal to 4 whitespaces and arranges the tab like Notepad++. With Notepad++, the output at the next page is achievable with the provided input file:



```
Patient Hayriye is recorded.
Patient Deniz is recorded.
Patient Ateş is recorded.
Patient Hayriye has a probability of 33.31% 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.
```

[1] https://docs.python.org/3/library/os.html#os.getcwd

[2] https://en.wikipedia.org/wiki/Confusion_matrix

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	…	…