Hacettepe University
Computer Engineering Department

BBM103 Assignment 2 Report

Samed Ökçeci - 2210356015

Course Instructor: Fuat Akal

Teaching Assistant: Hayriye Çelikbilek

27.11.2022

Analysis

We have two lists:

- doctors_aid_inputs.txt
- doctors_aid_outputs.txt

We get doctors_aid_inputs.txt as input for our program. We write outputs to doctors_aid_outputs.txt file.

We should create an empty list to store data of our patients. We should make python read doctors_aid_inputs.txt file,and get sentences as inputs. Every line has different commands. Program should seperate command from values, call the proper function depending on the command, and execute codes. Then program should iterate to next input sentence. This process should continue until all inputs end.

There are seven mandatory functions:

1. Recommendation Function
2. Disease Probability Function
3. Listing Function
4. Creating New Patient Function
5. Removing Patient Function
6. Reading Input File Function
7. Saving The Output File Function

All functions have different outputs for different inputs, so write function should write outputs to doctors_aid_outputs.txt line by line.

Design

Main Function, main();

I coded a main function, which first defines patientsList in other words it is the main list in which we put data of patients. Then gets the inputs as a whole, puts them into for loop to execute commands line by line. In the beginning of the for loop sentence passes into line_splitter() function which returns all the values in a list. Since when the first value is command (create, remove,list etc.), I defined a variable named command and put this value to inside. Then I created a control flow to decide which function to call. When a if statement becomes True, there is a code block under the statement which executes proper function, and passes all the values except ‘command’ value. After all the codes are executed, function returns particular string. I passed this string into save_output() function to write into doctors_aid_outputs.txt.

Check Index Function, check_index();

It is one of the most important function of my program. It uses patient names as identifiers, checks if name exists inside patientsList list. If it exists, it returns integer named ‘index’ (For example: 0, 4, 3). If it doesn’t exist, it returns nothing (None). I used this function in functions, in which I do operations with patientsList.

Line Splitter Function, line_splitter();

This function is for splitting input lines into seperate elements in a list. First checks if line has ‘\n’. If it has, replaces ‘\n’ with ‘’, so basically deletes it. Then splits depending on the word counts. It returns splitted list.

Create Function, create();

It calls check_index function to check if patients exists. If check_index returns ‘None’, create() appends patient values into patientList. If it returns ‘index’, create() does nothing.
Expected outputs:

- “Patient Hayriye is recorded.”

- “Patient Hayriye cannot be recorded due to duplication.”

Remove Function, remove();

It calls check_index function to check if patients exists. If check_index returns ‘None’, remove() does returns output[2]. If it returns ‘index’, remove() uses index as identifier and pops from patientsList, output[1] .

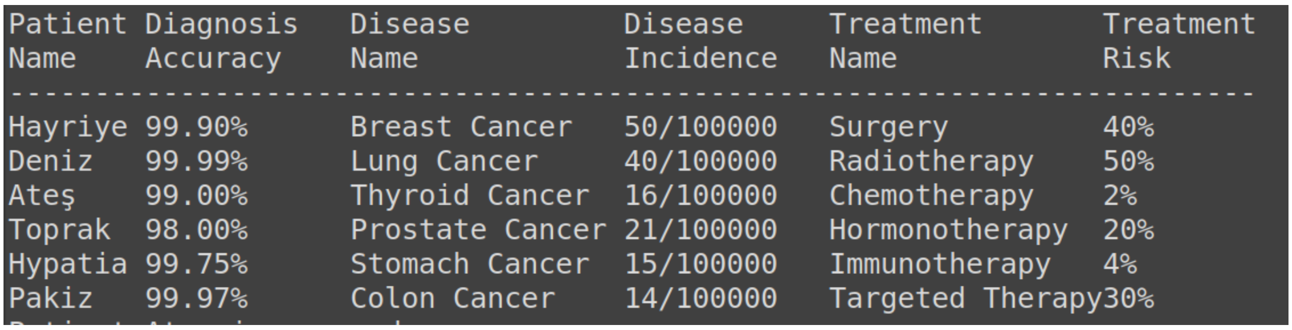
Expected outputs:

1. “Patient Deniz is removed.”
2. “Patient Deniz cannot be removed due to absence.”

List Function, list();

This function lists every patient inside patientList. Output depends on the content of the patientsList list. That means output can vary depending on the content of patientsList list.

Expected output:



```
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%
```

Probability Function;

I seperated this function to three different functions: probability_checker, probability_calculator, probability_percent_converter. So I have three functions instead one. I did this on perpous because it looks useful, better, readable and easy to understand. Also I can use this functions with other functions.

1-) probability_checker; When ‘command is probability’ program calls probability_checker, and passes all the values except command. First it checks if the patient exists in patientsList. If the person doesn’t exist, returns output[2]. If exists takes the values of the patient from patientsList. Then calls function probability_calculator. The output of probability_calculator is a decimal number. For example: 0.8232, 0.213, 0.13. Then passes the output of probability_calculator to probability_percent_converter. When probability_percent_converter executes commands returns a string percent value. The output of this percent value looks like 33.32%, 12.23%. Finally, returns output[1].

Expected output:

1. "Patient Hayriye has a probability of 33.32% of having breast cancer."
2. "Probability for Hayriye cannot be calculated due to absence."

2-)probability_calculator; This function takes needed values from patientsList, then computes 'Bayes Theorem' on this values. Returns a probability value. Probability value is a decimal number.

Expected output:

- 0.80, 0.3332 ...

3-)probability_percent_converter; Takes the output of probability_calculator, and converts it to percentage. The output of this percent value looks like 33.32%, 12.23%.

Recommend function, recommendation();

It calls check_index function to check if patients exists. If check_index returns 'None' returns output[1]. If it returns an 'index', takes patients treatment risk from patientsList, and compares with illness probability. If treatment risk is lower than doctor recommends to have the treatment, returns output[2]. Else doctor recommends not to have the treatment, returns output[3].

Expected outputs:

1. "Recommendation for Hayriye cannot be calculated due to absence."
2. "System suggests Hayriye to have the treatment."
3. "System suggests Hayriye NOT to have the treatment."

Open Input File Function, open_input_file;

This function opens 'doctors_aid_inputs.txt' with option 'r'. Returns file with readlines function.

Open Output File Function, open_output_file;

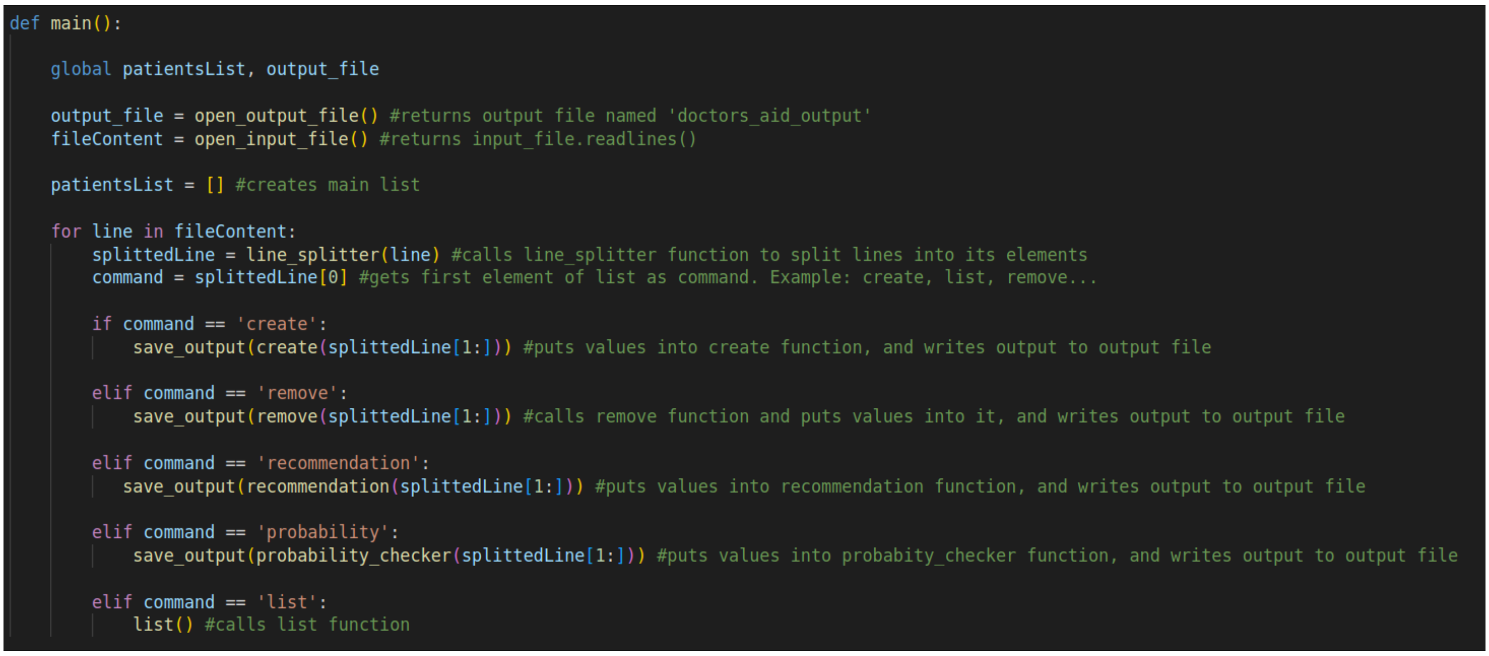
This function opens 'doctors_aid_outputs.txt' with option 'w', and returns open function.

Output Saving Function, save_output;

In takes a string parameter, and writes it into the output file.

Programmer's Catalogue

main() function;



```python
def main():

    global patientsList, output_file

    output_file = open_output_file() #returns output file named 'doctors_aid_output'
    fileContent = open_input_file() #returns input_file.readlines()

    patientsList = [] #creates main list

    for line in fileContent:
        splittedLine = line_splitter(line) #calls line_splitter function to split lines into its elements
        command = splittedLine[0] #gets first element of list as command. Example: create, list, remove...

        if command == 'create':
            save_output(create(splittedLine[1:])) #puts values into create function, and writes output to output file

        elif command == 'remove':
            save_output(remove(splittedLine[1:])) #calls remove function and puts values into it, and writes output to output file

        elif command == 'recommendation':
           save_output(recommendation(splittedLine[1:])) #puts values into recommendation function, and writes output to output file

        elif command == 'probability':
            save_output(probability_checker(splittedLine[1:])) #puts values into probabity_checker function, and writes output to output file

        elif command == 'list':
            list() #calls list function
```

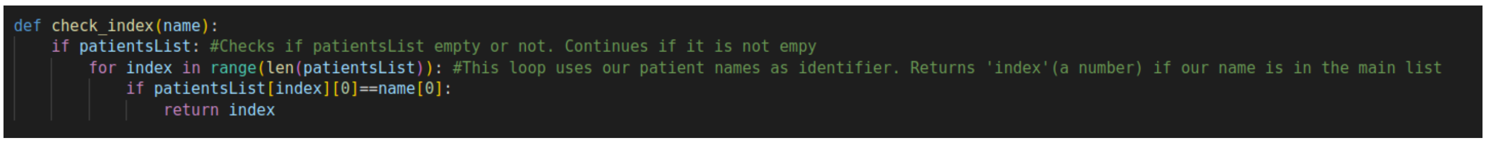
output_file: Contains the content of the doctors_aid_outputs.txt file

fileContent: Contains the content of the doctors_aid_inputs.txt file.

patientsList: It is a list to store patients and their values. It is accessible from everywhere of the program.

splittedLine: contains whole input sentence as list. For example: ["create", "Hayriye", "0.999", "Breast Cancer", "50/100000", "Surgery", "0.40"]
command: takes first argument of splittedLine (index 0).

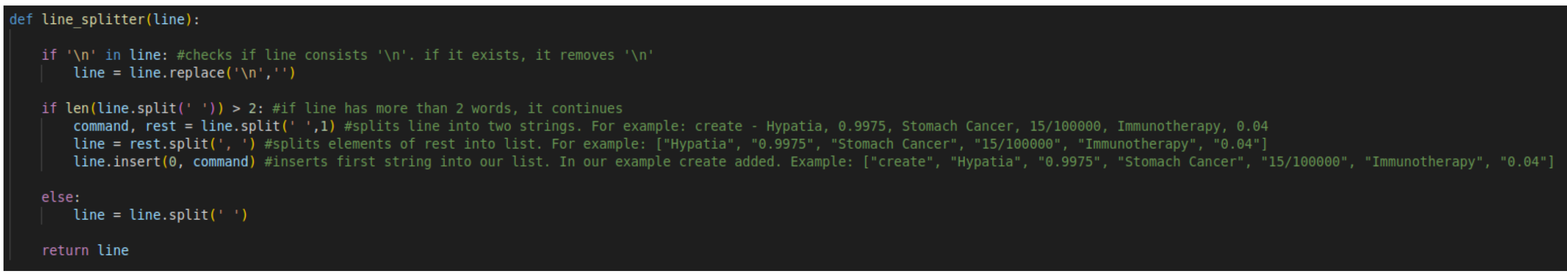
check_index()



```python
def check_index(name):
    if patientsList: #Checks if patientsList empty or not. Continues if it is not empty
        for index in range(len(patientsList)): #This loop uses our patient names as identifier. Returns 'index'(a number) if our name is in the main list
            if patientsList[index][0]==name[0]:
                return index
```

For loop loops depending on the length of patientsList. For every loop, it checks if my name[0] is in the list.

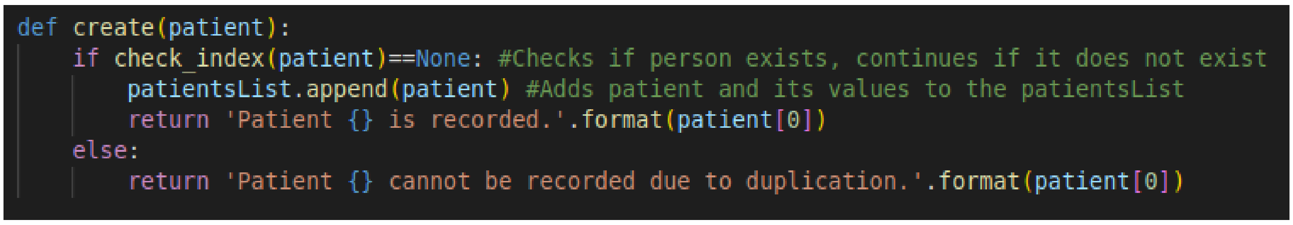
line_splitter()



```python
def line_splitter(line):

    if '\n' in line: #checks if line consists '\n'. if it exists, it removes '\n'
        line = line.replace('\n','')

    if len(line.split(' ')) > 2: #if line has more than 2 words, it continues
        command, rest = line.split(' ',1) #splits line into two strings. For example: create - Hypatia, 0.9975, Stomach Cancer, 15/100000, Immunotherapy, 0.04
        line = rest.split(', ') #splits elements of rest into list. For example: ["Hypatia", "0.9975", "Stomach Cancer", "15/100000", "Immunotherapy", "0.04"]
        line.insert(0, command) #inserts first string into our list. In our example create added. Example: ["create", "Hypatia", "0.9975", "Stomach Cancer", "15/100000", "Immunotherapy", "0.04"]

    else:
        line = line.split(' ')

    return line
```

Second if statement checks if my line has more than two arguments. If it has more than two arguments, it seperates the line depending on first space. Then it seperates second line depending on the ‘, ‘ argument. Then combines all two of them.

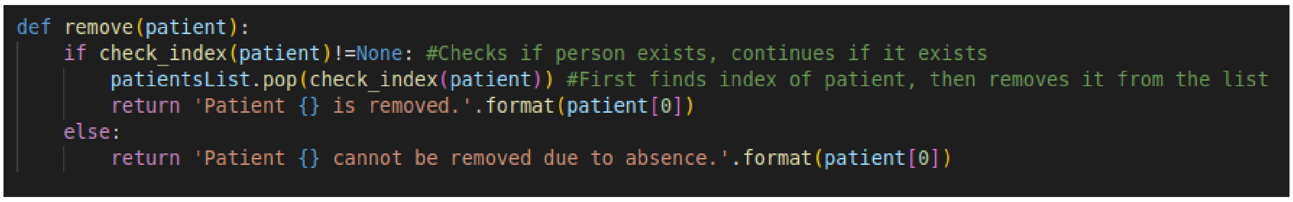
create()



```python
def create(patient):
    if check_index(patient)==None: #Checks if person exists, continues if it does not exist
        patientsList.append(patient) #Adds patient and its values to the patientsList
        return 'Patient {} is recorded.'.format(patient[0])
    else:
        return 'Patient {} cannot be recorded due to duplication.'.format(patient[0])
```

Example of the patient variable: [“Hayriye”, “0.999”, “Breast Cancer”, “50/100000”, “Surgery”, “0.40”]

remove()

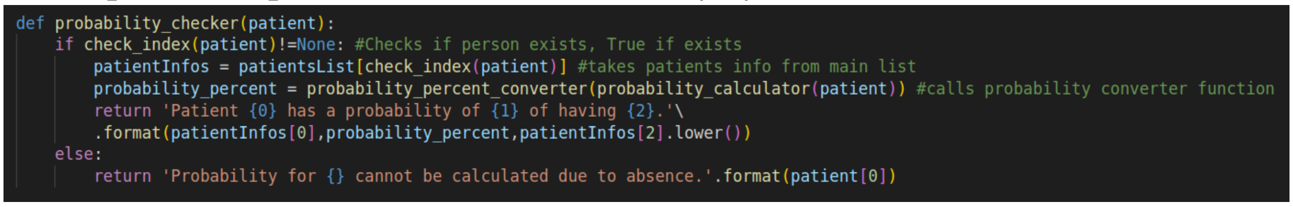


```python
def remove(patient):
    if check_index(patient)!=None: #Checks if person exists, continues if it exists
        patientsList.pop(check_index(patient)) #First finds index of patient, then removes it from the list
        return 'Patient {} is removed.'.format(patient[0])
    else:
        return 'Patient {} cannot be removed due to absence.'.format(patient[0])
```

Example of the patient variable: [“Hayriye”, “0.999”, “Breast Cancer”, “50/100000”, “Surgery”, “0.40”]

probability();

1.probability_checker()
Example of the patientInfos variable: [“Hayriye”, “0.999”, “Breast Cancer”,

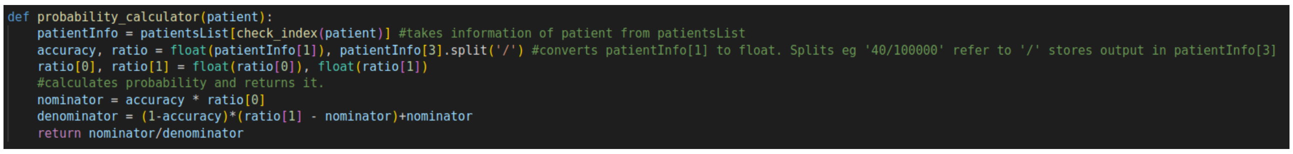


```python
def probability_checker(patient):
    if check_index(patient)!=None: #Checks if person exists, True if exists
        patientInfos = patientsList[check_index(patient)] #takes patients info from main list
        probability_percent = probability_percent_converter(probability_calculator(patient)) #calls probability converter function
        return 'Patient {0} has a probability of {1} of having {2}.'\
        .format(patientInfos[0],probability_percent,patientInfos[2].lower())
    else:
        return 'Probability for {} cannot be calculated due to absence.'.format(patient[0])
```

“50/100000”, “Surgery”, “0.40”]

Example of probability_percent variable: “32.22%”

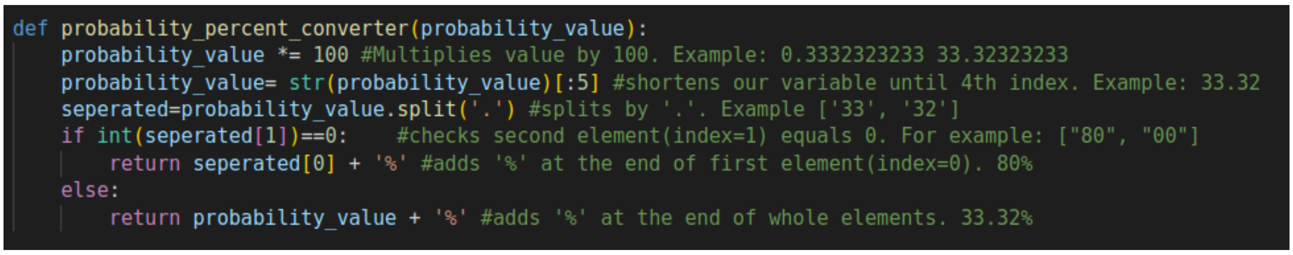
2.probability_calculator()



```python
def probability_calculator(patient):
    patientInfo = patientsList[check_index(patient)] #takes information of patient from patientsList
    accuracy, ratio = float(patientInfo[1]), patientInfo[3].split('/') #converts patientInfo[1] to float. Splits eg '40/100000' refer to '/' stores output in patientInfo[3]
    ratio[0], ratio[1] = float(ratio[0]), float(ratio[1])
    #calculates probability and returns it.
    nominator = accuracy * ratio[0]
    denominator = (1-accuracy)*(ratio[1] - nominator)+nominator
    return nominator/denominator
```

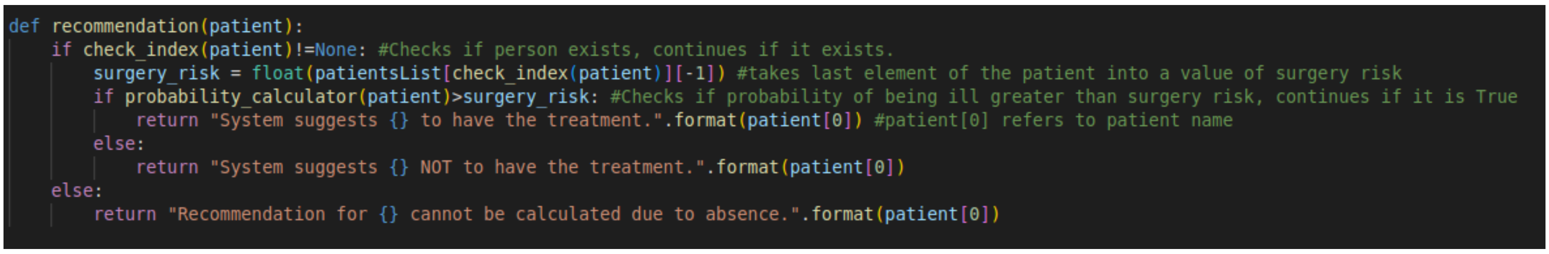
It computes Bayes Theorem.

3.probability_percent_converter()



```python
def probability_percent_converter(probability_value):
    probability_value *= 100 #Multiplies value by 100. Example: 0.3332323233 33.32323233
    probability_value= str(probability_value)[:5] #shortens our variable until 4th index. Example: 33.32
    seperated=probability_value.split('.') #splits by '.'. Example ['33', '32']
    if int(seperated[1])==0:    #checks second element(index=1) equals 0. For example: ["80", "00"]
        return seperated[0] + '%' #adds '%' at the end of first element(index=0). 80%
    else:
        return probability_value + '%' #adds '%' at the end of whole elements. 33.32%
```

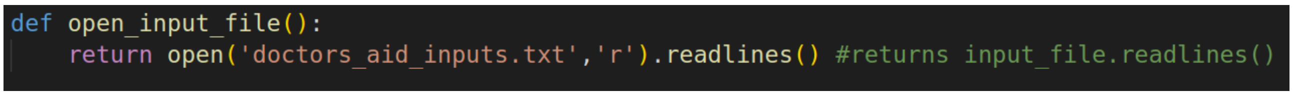
recommendation()



```python
def recommendation(patient):
    if check_index(patient)!=None: #Checks if person exists, continues if it exists.
        surgery_risk = float(patientsList[check_index(patient)][-1]) #takes last element of the patient into a value of surgery risk
        if probability_calculator(patient)>surgery_risk: #Checks if probability of being ill greater than surgery risk, continues if it is True
            return "System suggests {} to have the treatment.".format(patient[0]) #patient[0] refers to patient name
        else:
            return "System suggests {} NOT to have the treatment.".format(patient[0])
    else:
        return "Recommendation for {} cannot be calculated due to absence.".format(patient[0])
```

surgery_risk: contains last argument of the intended patient.
probability_calculator(patient): returns probability value of the intended patient.

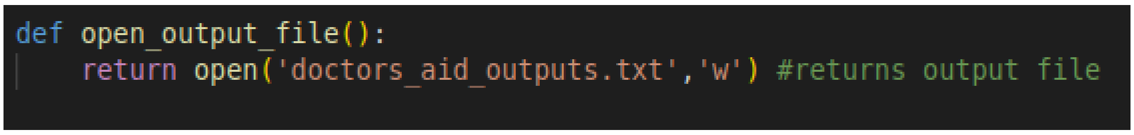
open_input_file()



```python
def open_input_file():
    return open('doctors_aid_inputs.txt','r').readlines() #returns input_file.readlines()
```

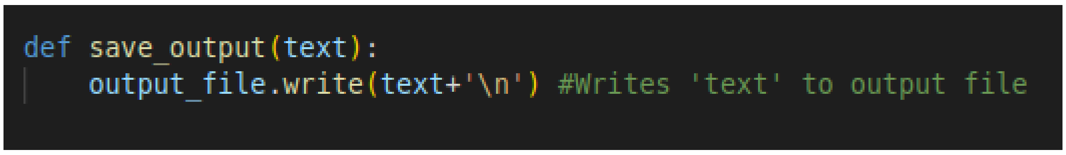
---

open_output_file()



```python
def open_output_file():
    return open('doctors_aid_outputs.txt','w') #returns output file
```

save_output()



```python
def save_output(text):
    output_file.write(text+'\n') #Writes 'text' to output file
```

User Catalogue

In order to run program and get outputs, you have to write commands inside the ‘doctors_aid_inputs.txt’ file. First you have to create a patient to get patients intended outputs. That means there won’t be proper outputs(wanted outputs) for uncreated patients. There are five basic commands:

1. Create
2. Remove
3. Recommendation
4. Probability
5. List

1. Create;

Usage: create <patient_name>, <diagnosis_accuracy>, <disease_name>, <disease_incidence>, <treatment_name>, <treatment_risk>

Examples: create Deniz, 0.9999, Lung Cancer, 40/100000, Radiotherapy, 0.50
create Ateş, 0.99, Thyroid Cancer, 16/100000, Chemotherapy, 0.02

Important Note: Between create and <patient_name> you have to put only one ‘space’, for others put ‘, ‘ (a comma and a space).

<patient_name>: Name of patient. Examples: Hayriye

<diagnosis_accuracy>: Accuracy of diagnosis. Examples: 0.8892, 0.992

<disease_name>: string, disease name. Examples: Breast Cancer, Colon Cancer

<disease_incidence>: How many people has diagnosed with disease in how many people. Examples: 20/30000, 50/100000 that means 50 people diagnosed cancer out of 100000 people.

<treatment_name>: Name of treatment. Examples: Hormonotherapy, Immunotherapy.

<treatment_risk>: Risk of treatment. Examples: 0.20, 0.23, 0.42

2. Remove;

Usage: remove <patient_name>
Example: remove Deniz
Only a ‘space between remove and <patient_name>.

3. Recommendation;

Usage: recommendation <patient_name>
Example: recommendation Hayriye
Only a ‘space between recommendation and <patient_name>.

4. Probability;

Usage: probability <patient_name>
Example: probability Hayriye
Only a ‘space between probability and <patient_name>.

5. List;
Usage: list
Example: list

Evaluation

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	…	…