# Python Coursework

Logan Miller

# Task 1

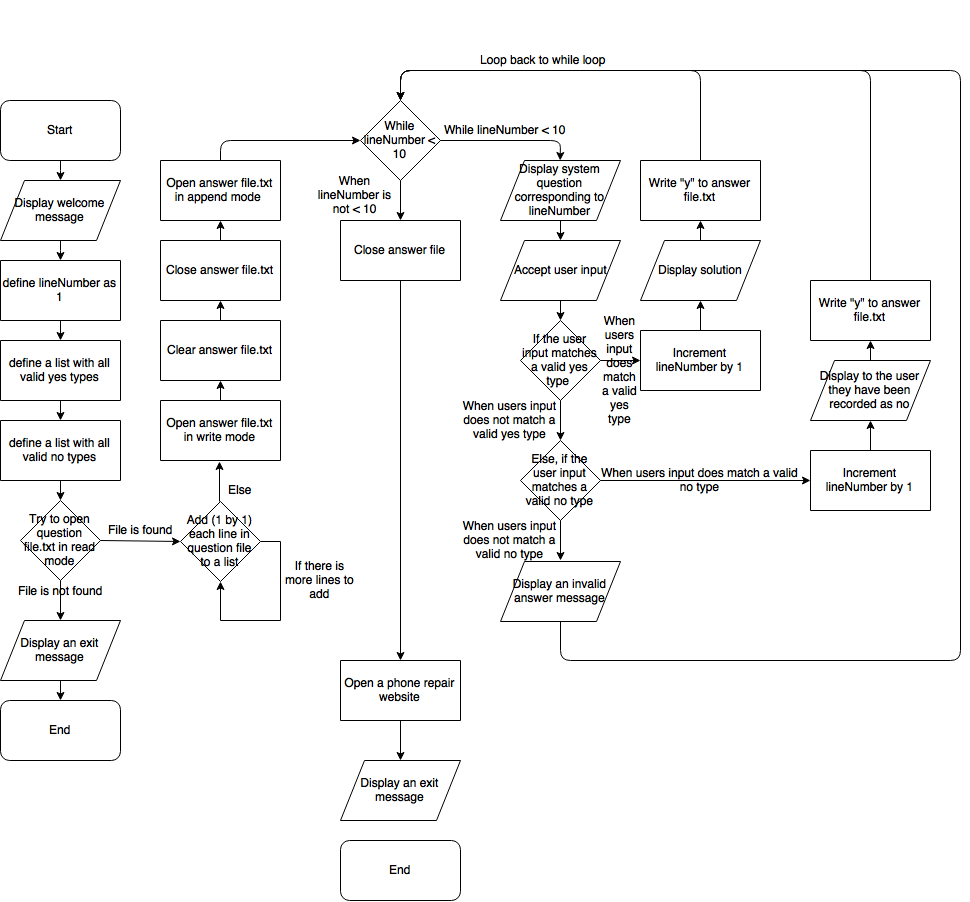
## Design (9marks)

I am going to design an automated troubleshooting program that will ask the user yes or no questions about problems common with phones. My troubleshooting program will then display a solution to the problem if the user confirms that they have the problem. This will then be followed up by a confirmation, asking if their phone is now fixed, and will either exit the program (if it is fixed) or continue asking questions (if the phone is not fixed).

### Success Criteria

* The system should introduce the program.
* The system should ask a question about the phone.
* The system should accept yes and no answers.
* The process should be repeated.
* The system should allow the user to exit if the problem was resolved.

### Flow Chart



### Validation

In my code, there will be validation needed when the user answers the program's questions. I plan to use a lookup check to validate this data, by looking up the user's input in 2 tuples which contain all the values the program will accept for a "yes” answer and a tuple which contain all the values the program will accept for a “no”. For the program to accept the input the input must match one item in either of the tuples, if it does not the user will be required to re-enter its answer.

### Pseudo-code

*Display a welcome message*

*Define lineNumber as a 0 integer value*

*Define yesTypes as a list of acceptable yes values*

*Define noTypes as a list of acceptable no values*

*Try to open question file.txt: If the file can be opened, open it*

*If the file cannot be opened, display an error message and close the program*

*Append every line of the question file to questionFileList*

*Open answer file.txt*

*Clear answer file.txt*

*Close answer file.txt*

*Re-open answer file.txt*

*While the lineNumber is less than 10 print the systems question:*

*Display a system question*

*Request a user input*

*Turn the user's input into all lower case*

*If the user's input matches an acceptable value in the yesTypes*

*Increment lineNumber by +1*

*Display a solution to the problem*

*Add a y to answer file.txt*

*Else, if the user's input matches an acceptable value in the noTypes*

*Increment lineNumber by +1*

*Display a message saying the user's response was recorded as no*

*Add an n to answer file.txt*

*Else:*

*Display an invalid response message and tell them to try again*

*Close answer file.txt*

*Open a phone repair website*

*Display an exit message and close the program*

### Identifying Data Types

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Type of variable | Why I used this variable type | What the variable will do | Definition |
| *userInputCaseSensitive* | String | Because the user input could consist of any combination of characters, by using a string the program will not crash if the user inputs an extreme response to the systems question. | Store the user’s input. | A finite sequence of characters, which can include any characters. |
| *userInput* | Because this variable is *userInputCaseSensitive* after it has gone through the *.lower()* method.  *.lower()* only works of strings. | Store the user’s input after all the letters have been converted to lower case |
| *enterToExit* | Because this variable is used to let the user exit the program by pressing enter. If they accidently enter a character before they press enter an error will not occur as a string accepts any character. | To allow the user to press enter when they wish to exit. |
| *lineNumber* | Integer | Because this variable is used as a counter to decide which line to read from the text files used in the program. The variable cannot be a float as you cannot get fractions of a line in *.txt* text files and it cannot be a string as the main while loop is dependent on if it is less than a certain integer. | Store the number for which line they program should read from the text file. | A number which is not a fraction but is a whole number and may or may not become negative. |
| *questionFileList* | List | Because there needs to be an item for each line in the relevant file and the list needs to be able to be updated for when I append each line to the list. | Store each line the relevant text file as an item in the list. | A sequence data type which contains items separated by commas, enclosed in square brackets. Each item can be a different data type. You can update the contents of the tuple (e.g. you can add items, remove items, edit items or re-order items) |
| *solutionFileList* |
| *noTypes* | Tuple | Because there needs to be multiple items which can be easily read, however, it does not need to be updated. | Store each string which the program recognises as a yes/no answer as items in the tuple. | A sequence data type which contains items separated by commas, enclosed in parentheses. Each item can be a different data type. You cannot update the contents of the tuple (e.g. you cannot add items, remove items, edit items or re-order items) |
| *yesTypes* |
| *questionFile* | File object | Because it is the easiest way to read data from an external location from the python file. | Store the data the program needs to read the relevant text file. | A TextIOWrapper which allows python to randomly access to text stored externally. |
| *solutionFile* |

**Test Strategy**

|  |  |  |
| --- | --- | --- |
| What I am testing | How I will test it | Expected outcome |
| That my program will respond correctly to an expected (valid) input. | By typing in “*no*” when answering the program’s question. | The program to interpret the user’s input as no to the question by not providing a solution and moving on to the next question. |
| That my program will respond correctly to extreme inputs, which are valid but at the edge of what the program should accept. | By typing in “*aye*” when answering the program’s question. | The program to interpret the user’s input as yes to the question by providing a solution and asking if my problem was resolved. |
| That my program will respond correctly to invalid, erroneous, data; which may be a mistake from the user. | By typing in “*1NC0GN1T0*” when answering the program’s question. | The program to interpret the user’s input as an invalid answer and allow them to answer the same question again. |
| That my program will allow me to exit once my problem is solved. | By typing in “*yes*” when the program asks me if my problem is solved. | The program to interpret the user’s input as yes to the question by allowing me to exit the program. |
| That my program will allow me to continue going through questions if my problem has not been solved by the solution provided. | By typing in “*no*” when the program asks me if my problem is solved. | The program to interpret the user’s input as no to the question by continuing on to the next question on if I have a certain problem with my phone. |
| That my program will shut down correctly when there is a missing text file. | By deleting “*question file.txt*” from the directory the program is in and then running the program. | The program to realise a key file is missing, display a message saying it will now shut down and then close itself. |
| That my program will go through all the questions | By answering “*no*” until all the questions have been answered. | The program to redirect the user to a phone repair website, display a message saying it will now shut down and then close itself. |

## Developed Solutions

|  |  |  |
| --- | --- | --- |
| Programming Technique | Code | Why I used it |
| Input | userInputCaseSensitive **=** input**(**"USER: "**)** | To allow the user to answer if they have a problem or not, this allows the user to communicate with the program. |
| Output | **print(**"Is your problem resolved?\n"**)** | To ensure the user understands what they are being asked – without this the user would find the program difficult to use without reading a manual. |
| Variable | lineNumber **=** 0 | To define which line of the text files needs to be outputted, it needs to change so the program can progress to different problems and solutions. |
| If statement | **if** userInput **in** yesTypes**:** | To run a different portion of code for if the user’s input is yes than if it was not yes. |
| While loop | **while** lineNumber **<** 11**:** | So that the program goes through correct amount of lines before ending the loop. |

## Testing: (9 marks)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| What I am testing | How I will test it | Expected outcome | Actual outcome | Changes made |
| That my program will respond correctly to an expected, valid, input. | By typing in “*no*” when answering the program’s question. | The program to interpret the user’s input as no to the question by not providing a solution and moving on to the next question. | The program said the answer was recorded as no and asked the next question. | None. |
| That my program will respond correctly to extreme inputs, which are valid but at the edge of what the program should accept. | By typing in “*aye*” when answering the program’s question. | The program to interpret the user’s input as yes to the question by providing a solution and asking if my problem was resolved. | The program provided a solution. | None. |
| That my program will respond correctly to invalid, erroneous, data; which may be a mistake from the user. | By typing in “*1NC0GN1T0*” when answering the program’s question. | The program to interpret the user’s input as an invalid answer and allow them to answer the same question again. | The program said the answer was invalid and repeated the question. | None. |
| That my program will allow me to exit once my problem is solved. | By typing in “*yes*” when the program asks me if my problem is solved. | The program to interpret the user’s input as yes to the question by allowing me to exit the program. | The program thanked me for using the program and allowed me to press enter to exit the program. | None. |
| That my program will allow me to continue going through questions if my problem has not been solved by the solution provided. | By typing in “*no*” when the program asks me if my problem is solved. | The program to interpret the user’s input as no to the question by continuing on to the next question on if I have a certain problem with my phone. | The program continued with the next question. | None. |
| That my program will shut down correctly when there is a missing text file. | By deleting “*question file.txt*” from the directory the program is in and then running the program. | The program to realise a key file is missing, display a message saying it will now shut down and then close itself. | The program displayed a custom error message saying there is a missing file and allowed me to press enter to exit the program. | None. |
| That my program will go through all the questions | By answering “*no*” until all the questions have been answered. | The program to redirect the user to a phone repair website, display a message saying it will now shut down and then close itself. | The program redirected me to a phone repair site, displayed an exit message and allowed me to press enter to exit the program. | None. |

### Criteria Check

|  |  |
| --- | --- |
| Criteria | How it has been met |
| The system should introduce the program. | With a “*print*” command. |
| The system should ask a question about the phone. | With a “*print*” command. |
| The system should accept yes and no answers. | With an “*input*” command. |
| The process should be repeated. | With a “*while*” loop. |
| The system should allow the user to exit if the problem was resolved. | By using “*if*” and “*elif*” commands, along with the OS module for specifically exiting the program. |

## Full code (18 marks)

phone-repair.py:



# Task 2

## Design (9 marks)

I will design an automated troubleshooting program. This program will ask the user what problem they are experiencing with their mobile phone; the user will then enter their issue. After this, my troubleshooting program will scan the user's input for keywords and suggest a solution based on keywords it has found in user input. After this, my program will ask if their phone is fixed and will either exit the program (if it is fixed) or continue asking questions (if the phone is not fixed).

### Success Criteria

* The system should introduce the program.
* The system should accept the user entering a problem with the phone.
* The system should scan the input and correctly identify keywords.
* The system should correctly link certain keywords to certain problems and provide a solution for the relevant problem.
* The system should allow the user to re-phrase the question if the correct solution was not provided .
* The system should allow the user to exit if the problem was resolved.

### Flow Chart

****

### Validation

Validation in my program will be required when the user confirms or denies whether their problem has been resolved. Their answer will only be accepted if their answer is in a tuple containing valid yes answers or a tuple containing valid no answers. If the input is not valid, the user will be required to re-enter its answer.

### Pseudo-code



### Identifying Data Types

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Type of variable | Why I used this variable type | What the variable will do | Definition |
| *userInputCaseSensitive* | String | Because the user input could consist of any combination of characters, by using a string the program will not crash if the user inputs an extreme response to the systems question. | Store the user’s input. | A finite sequence of characters, which can include any characters. |
| *userInput* | Because this variable is *userInputCaseSensitive* after it has gone through the *.lower()* method.  *.lower()* only works of strings. | Store the user’s input after all the letters have been converted to lower case |
| *enterToExit* | Because this variable is used to let the user exit the program by pressing enter. If they accidently enter a character before they press enter an error will not occur as a string accepts any character. | To allow the user to press enter when they wish to exit. |
| *questionFileList* | List | Because there needs to be an item for each line in the relevant file and the list needs to be able to be updated for when I append each line to the list. | Store each line the relevant text file as an item in the list. | A sequence data type which contains items separated by commas, enclosed in square brackets. Each item can be a different data type. You can update the contents of the tuple (e.g. you can add items, remove items, edit items or re-order items) |
| *solutionFileList* |
| *noTypes* | Tuple | Because there needs to be multiple items which can be easily read, however, it does not need to be updated. | Store each string which the program recognises as a yes/no answer as items in the tuple. | A sequence data type which contains items separated by commas, enclosed in parentheses. Each item can be a different data type. You cannot update the contents of the tuple (e.g. you cannot add items, remove items, edit items or re-order items) |
| *yesTypes* |
| *questionFile* | File object | Because it is the easiest way to read data from an external location from the python file. | Store the data the program needs to read the relevant text file. | A TextIOWrapper which allows python to randomly access to text stored externally. |
| *solutionFile* |

**Test Strategy**

|  |  |  |
| --- | --- | --- |
| What I am testing | How I will test it | Expected outcome |
| That my program will respond correctly when the user’s input contains a keyword. | By typing in “*my screen is frozen*” when entering in my input. | The program to detect the keyword “*frozen*” and provide a solution for if the screen if frozen. |
| That my program will respond correctly when the user’s input does not contain any keywords. | By typing in “*How long is a piece of string?*” when entering in my input. | The program to not detect any keywords, inform the user that it could not, and allow me to re-phrase the question. |
| That my program will respond correctly if the user’s input is blank. | By typing in nothing followed by enter when entering in my input. | The program to detect there is nothing in the input, inform me of what I am meant to type and allow me to try again. |
| That my program will allow me to exit once my problem is solved. | By typing in “*yes*” when the program asks me if my problem is solved. | The program to interpret the user’s input as yes to the question by allowing me to exit the program. |
| That my program will allow me to re-phrase my input if a solution is not found | By firstly typing in “*The phone’s display is chilled?”* and then (if it allows me) re-phrase my question to “*The phone’s display is frozen?”.* | The program to not detect any keywords, inform the user that it could not, and allow me to re-phrase the question and then detect the keyword “*frozen*” in the re-phrased question and provide a solution for if the screen if frozen. |
| That my program will shut down correctly when there is a missing text file. | By deleting “*key word file.txt*” from the directory the program is in and then running the program. | The program to realise a key file is missing, display a message saying it will now shut down and then close itself. |

## Developed Solutions

|  |  |  |
| --- | --- | --- |
| Programming Technique | Code | Why I used it |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| What I am testing | How I will test it | Expected outcome | Actual outcome | Changes made |
| That my program will respond correctly when the user’s input contains a keyword. | By typing in “*my screen is frozen*” when entering in my input. | The program to detect the keyword “*frozen*” and provide a solution for if the screen if frozen. | The program provided a solution for having a frozen screen. | None. |
| That my program will respond correctly when the user’s input does not contain any keywords. | By typing in “*How long is a piece of string?*” when entering in my input. | The program to not detect any keywords, inform the user that it could not, and allow me to re-phrase the question. | The program told me to rephrase my input as it could not detect any keywords. | None. |
| That my program will respond correctly if the user’s input is blank. | By typing in nothing followed by enter when entering in my input. | The program to detect there is nothing in the input, inform me of what I am meant to type and allow me to try again. | The program told me to rephrase my input as it could not detect any keywords. | I added an if statement saying if the input is blank then repeat the instructions said at the start of the program. |
| That my program will allow me to exit once my problem is solved. | By typing in “*yes*” when the program asks me if my problem is solved. | The program to interpret the user’s input as yes to the question by allowing me to exit the program. | The program thanked me for using the program and allowed me to press enter to exit the program. | None. |
| That my program will allow me to re-phrase my input if a solution is not found | By firstly typing in “*The phone’s display is chilled”* and then (if it allows me) re-phrase my question to “*The phone’s display is frozen”.* | The program to not detect any keywords, inform the user that it could not, and allow me to re-phrase the question and then detect the keyword “*frozen*” in the re-phrased question and provide a solution for if the screen if frozen. | The program first told me to re-phrase my question, allowed me to enter a new response and then provided a solution for having a frozen screen. | None. |

## Testing: (9 marks)

### Criteria Check

|  |  |
| --- | --- |
| Criteria | How it has been met |
| The system should introduce the program. | With a “*print*” command. |
| The system should accept the user entering a problem with the phone. | With an “*input*” command. |
| The system should scan the input and correctly identify keywords. | By using a “*for*” loop and “*.split()*” to separate a list of keywords and an “*if*” statement to detect any keywords. |
| The system should correctly link certain keywords to certain problems and provide a solution for the relevant problem. | By having an integer (“*repeats*”) be linked to both the key word and the solution (which is specific to the problem). |
| The system should allow the user to re-phrase the question if the correct solution was not provided. | By using “*print*” and “*input*” commands to ask and receive an answer to if their problem was solved, and an “*if*” statement to detect if they responded yes or no to the question. |

## Full code (18 marks)



# Task 3

## Design (9 marks)

I aim to design a troubleshooting program that will identify the device and then load a custom set of solutions specific to that device. The program should also be able to scan the user’s input for keywords related to a problem. Furthermore, my program will allocate the user a case number (that will theoretically be sent to a technician) if a solution cannot be found to the phone’s problem. In addition to this, I plan to make the program modular, to improve program upgradability and stability, and have two outputs for the user to improve user accessibility. These outputs will be text displayed on the screen and audio.

### Success Criteria

* The system should introduce the program.
* The system should find out the user’s device.
* The system should load a set of solutions relevant to the user’s device.
* The system should scan the input and correctly identify keywords.
* The system should correctly link certain keywords to certain problems and provide a solution for the relevant problem.
* The system should allocate the user a case number if a solution has not been found.
* The system should allow the user to exit if the problem was resolved.
* The system should provide both audio and visual outputs.
* The system should be modular.

### Flow Chart

phone.py



modules.py













### Validation

Validation is vital in many aspects of my task 3 code. It will be used in the “*yesOrNoExit()*” function when the user confirms or denies whether they want to exit the program. Their input will only be accepted if their answer is found in a tuple containing valid yes answers or a tuple containing valid no answers. The user will have to re-enter their answer if it is not found.

Moreover, in the “device()” function the user will be asked what operating system their device is running. Their answer is required to be either in “*iosDevice*”, “*androidDevice*” or “*windowsDevice*”. If their input is not in any of the aforementioned lists they will be asked if they wish to exit the program.

Finally, validation will be required when the user confirms or denies whether their problem has been resolved. If their answer is not located in “yesTypes” or “noTypes” they will be required to type in their input again.

### Pseudo-code

phone.py



modules.py

#Importing external modules

**import** os**,** webbrowser**,** winsound

FUNCTION startUp**():**

#Setting global variables

**global** audioDict**,** yesTypes**,** noTypes**,** keyWordFileList**,** solutionFileList**,** iosDevice**,** androidDevice**,** windowsDevice**,** devices

#Defining variables

yesTypes SET TO **(**"yes"**,** "yea"**,** "y"**,** "yep"**,** "aye"**,** "true"**)**

noTypes SET TO **(**"no"**,** "nah"**,** "n"**,** "nope"**,** "false"**)**

audioDict SET TO **{}**

keyWordFileList SET TO **[]**

solutionFileList SET TO **[]**

iosDevice SET TO **[**"ios"**,** "apple"**,**"iphone"**,** "ipad"**,** "ipod"**]**

androidDevice SET TO **[**"android"**,** "samsung"**,** "google"**,** "nexus"**,** "pixel"**]**

windowsDevice SET TO **[**"windows"**,** "microsoft"**,** "nokia"**,** "lumia"**,** "zune"**]**

devices SET TO **[**iosDevice**,** androidDevice**,** windowsDevice**]**

#Trying to open audio files

**try:**

#Repeat process for each audio file

ENDFOR

**for** fileNumber **in** range**(**10**):**

#Add android audio files to dictionary

audioDict**[**"android{0}"**.**format**(**fileNumber**)]** SET TO "audio/android" **+** str**(**fileNumber**)**

ENDFOR

#Add iOS audio files to dictionary

audioDict**[**"ios{0}"**.**format**(**fileNumber**)]** SET TO "audio/ios" **+** str**(**fileNumber**)**

ENDFOR

#Add windows audio files to dictionary

audioDict**[**"windows{0}"**.**format**(**fileNumber**)]** SET TO "audio/windows" **+** str**(**fileNumber**)**

ENDFOR

#Repeat process for each audio file

ENDFOR

ENDFOR

**for** fileNumber **in** range**(**12**):**

#Add generic audio files to dictionary

audioDict**[**"generic{0}"**.**format**(**fileNumber**)]** SET TO "audio/generic" **+** str**(**fileNumber**)**

ENDFOR

ENDFOR

#Terminating the program IF FileNotFoundError (a run time error) is found

ENDIF

**except** FileNotFoundError**:**

#Displaying exit message

OUTPUT "ERROR: The LoganBerry phone repair service is unable to open due to 1 OR more missing files\n"

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic0"**),**winsound**.**SND\_FILENAME**)**

#Running exitSequence

exitSequence**()**

#Displaying intro message

OUTPUT "Hello AND welcome to LoganBerry phone repair services, please note we only support devices running iOS, Android or Windows\n"

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic1"**),**winsound**.**SND\_FILENAME**)**

ENDFUNCTION

FUNCTION exitSequence**():**

#Setting global variables

**global** audioDict

#Allowing user to choose to exit

OUTPUT "Press enter to exit the program\n"

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic2"**),**winsound**.**SND\_FILENAME**)**

enterToExit SET TO input**()**

#Exiting program

os**.**\_exit**(**0**)**

ENDFUNCTION

FUNCTION yesOrNoExit**():**

#Setting global variables

**global** audioDict**,** yesTypes**,** noTypes

#Starting yes OR no loop

**while** **True:**

#Accepting answer

userInputCaseSensitive SET TO input**(**"USER: "**)**

#Converting answer to lower case

userInput SET TO userInputCaseSensitive**.**lower**()**

#If the user wants to exit

IF userInput **in** yesTypes**:**

#Displaying exit message

OUTPUT "Thank you for using our phone repair program\n"

ENDFOR

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic3"**),**winsound**.**SND\_FILENAME**)**

#Running exitSequence

exitSequence**()**

#Else, IF the user wants to try again

ENDIF

ELSEIF userInput **in** noTypes**:**

#Exiting function

RETURN

#Else yes OR no answer not recognised

ELSE**:**

OUTPUT "Answer not recognised, please answer either yes OR no\n"

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic4"**),**winsound**.**SND\_FILENAME**)**

ENDIF

ENDFUNCTION

ENDWHILE

FUNCTION device**():**

#Setting global variables

**global** audioDict**,** iosDevice**,** androidDevice**,** windowsDevice**,** devices**,** deviceModel**,** operatingSystem

#Asking for device model

ENDFOR

OUTPUT "Please tell us your device name/model (e.g. 'iPhone 5C 16GB Blue')\n"

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic5"**),**winsound**.**SND\_FILENAME**)**

#Accepting answer AND storing answer under variable, deviceModel

deviceModel SET TO input**(**"DEVICE NAME/MODEL: "**)**

#Starting main device loop

**while** **True:**

#Asking for operating system

ENDFOR

OUTPUT "Please tell us what operating system your phone has (either iOS, Android OR Windows)\n"

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic6"**),**winsound**.**SND\_FILENAME**)**

#Accepting answer

userInputCaseSensitive SET TO input**(**"USER: "**)**

#Converting answer to lower case

userInput SET TO userInputCaseSensitive**.**lower**()**

#Repeating process 3 times for the 3 operating systems / device types

ENDFOR

**for** deviceType **in** range**(**3**):**

#Repeating process 5 times for the 5 key words linked to each operating system / device type

ENDFOR

**for** keyWord **in** range**(**5**):**

#If the user's input equals the list in the devices list that corresponds with the deviceType integer AND the string in the devices[deviceType] list that corresponds with the keyWord integer

IF userInput IS EQUAL TO **((**devices**[**deviceType**])[**keyWord**]):**

#Define operatingSystem as the string that corresponds with the integer 0, which is found in the list that corresponds with the deviceType integer in the devices list

operatingSystem SET TO **((**devices**[**deviceType**])[**0**])**

#Exiting function

RETURN

ENDIF

ENDFOR

#Inform user that their input did not match any keywords AND asking IF they want to exit

ENDIF

ENDFOR

ENDFOR

OUTPUT "Operating system not supported or recognised, would you like to exit the program (IF you answer no we will allow you to re-type / re-phrase your previous answer)?\n"

ENDIF

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic7"**),**winsound**.**SND\_FILENAME**)**

#Running yesOrNoExit

yesOrNoExit**()**

ENDFUNCTION

ENDWHILE

FUNCTION managingFiles**():**

#Setting global variables

**global** audioDict**,** keyWordFileList**,** solutionFileList**,** operatingSystem

#Trying to open files in read mode

**try:**

keyWordFile SET TO open**(**"key word file.txt"**,** "r"**)**

#Opening solution file specific to one chosen by user

ENDIF

solutionFile SET TO open**(**operatingSystem **+** " solution file.txt"**,** "r"**)**

#Terminating the program IF FileNotFoundError (a run time error) is found

ENDIF

**except** FileNotFoundError**:**

#Displaying exit message

OUTPUT "ERROR: The LoganBerry phone repair service is unable to open due to 1 OR more missing files\n"

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic0"**),**winsound**.**SND\_FILENAME**)**

#Running exitSequence

exitSequence**()**

#Putting each line of the file as a separate item into a list

**for** lines **in** keyWordFile**:**

#.strip() used to remove the '\n' used to indicate a new line

keyWordFileList**.**append**(**lines**.**strip**())**

#Putting each line of the file as a separate item into a list

ENDFOR

**for** lines **in** solutionFile**:**

solutionFileList**.**append**(**lines**)**

ENDFUNCTION

ENDFOR

FUNCTION diagnosingProblem**():**

#Setting global variables

**global** audioDict**,** keyWordFileList**,** solutionFileList**,** operatingSystem**,** userInput

#Displaying instructions

OUTPUT "Please tell us what is wrong with your"**,** operatingSystem**,** "phone\n"

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic8"**),**winsound**.**SND\_FILENAME**)**

#Accepting problem

userInputCaseSensitive SET TO input**(**"USER: "**)**

#Converting answer to lower case

userInput SET TO userInputCaseSensitive**.**lower**()**

#Repeating process 10 times for 10 problems program can diagnose

ENDFOR

**for** repeats **in** range**(**10**):**

#Repeating process for each key word (separated by a comma) in the keyWordFileList that corresponds with the integer repeats

ENDFOR

**for** keyWords **in** keyWordFileList**[**repeats**].**split**(**","**):**

#If a key word is found

IF keyWords **in** userInput**:**

#Displaying solution the corresponds with the integer repeats (AND with the key word found)

OUTPUT solutionFileList**[**repeats**],** "\n"

#Playing above message

IF operatingSystem IS EQUAL TO "android"**:**

winsound**.**PlaySound**(**audioDict**.**get**(**"android" **+** str**(**repeats**)),**winsound**.**SND\_FILENAME**)**

ELSEIF operatingSystem IS EQUAL TO "ios"**:**

winsound**.**PlaySound**(**audioDict**.**get**(**"ios" **+** str**(**repeats**)),**winsound**.**SND\_FILENAME**)**

ELSEIF operatingSystem IS EQUAL TO "windows"**:**

winsound**.**PlaySound**(**audioDict**.**get**(**"windows" **+** str**(**repeats**)),**winsound**.**SND\_FILENAME**)**

#Checking IF the user has resolved the problem

ENDIF

ENDIF

OUTPUT "Is your problem resolved?\n"

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic9"**),**winsound**.**SND\_FILENAME**)**

#Running yesOrNoExit

yesOrNoExit**()**

#Exiting function

RETURN

ENDIF

ENDFOR

#Display message

ENDFOR

OUTPUT "Sorry, your problem was not recognised\n"

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic10"**),**winsound**.**SND\_FILENAME**)**

#Exiting function

RETURN

ENDFUNCTION

FUNCTION diagnosticFailed**():**

#Setting global variables

**global** audioDict**,** solutionFileList**,** deviceModel**,** operatingSystem**,** userInput

#Trying to assign this case the next case number in the cases.txt file

**try:**

#Defines caseNumber as integer of amount of lines in cases.txt

caseNumber SET TO len**(**open**(**"cases.txt"**).**readlines**())**

#Creating a case file (using write mode) IF FileNotFoundError (a run time error) is found

ENDIF

**except** FileNotFoundError**:**

cases SET TO open**(**"cases.txt"**,** "w"**)**

#Closing case file so it can be used in a different mode

ENDIF

cases**.**close**()**

#Opening case file in appending mode

cases SET TO open**(**"cases.txt"**,** "a"**)**

#Appending case details to file

cases**.**write**(**"CASE NUMBER: " **+** str**(**caseNumber**)** **+** " = DEVICE TYPE: " **+** operatingSystem **+** " = DEVICE MODEL:" **+** deviceModel **+** " = USER'S PROBLEM: " **+** userInput **+** "\n"**)**

#Closing case file

cases**.**close**()**

#Telling user they have a case number AND we are googling the problem for them

ENDFOR

OUTPUT "You have been assigned case number " **+** str**(**caseNumber**)** **+** ". A technician will be with you shortly\nWe will also show you google results related to your problem."

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic11"**),**winsound**.**SND\_FILENAME**)**

ENDFOR

webbrowser**.**open**(**"https://google.com/?q=" **+** operatingSystem **+** "+" **+** deviceModel **+** "+" **+** userInput**)**

#Running exitSequence

exitSequence**()**

### Identifying Data Types

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Type of variable | Why I used this variable type | What the variable will do | Definition |
| *userInputCaseSensitive* | String | Because the user input could consist of any combination of characters, by using a string the program will not crash if the user inputs an extreme response to the systems question. | Store the user’s input. | A finite sequence of characters, which can include any characters. |
| *deviceModel* | Store the device model. |
| *operatingSystem* | By using a string instead of assigning integers to each operating system, I can mention the operating system in outputs or file names without using if statements. | Store the user’s operating system name. |
| *userInput* | Because this variable is *userInputCaseSensitive* after it has gone through the *.lower()* method.  *.lower()* only works of strings. | Store the user’s input after all the letters have been converted to lower case |
| *enterToExit* | Because this variable is used to let the user exit the program by pressing enter. If they accidently enter a character before they press enter an error will not occur as a string accepts any character. | To allow the user to press enter when they wish to exit. |
| *caseNumber* | Integer | Apart from the case number needing to (obviously) be a number it must also be defined from the number of lines in cases.txt. The variable cannot be a float as you cannot get fractions of a line in a *.txt.* | Store the user’s case number. | A number which is not a fraction but is a whole number and may or may not become negative. |
| *keyWordFileList* | List | Because there needs to be an item for each line in the relevant file and the list needs to be able to be updated for when I append each line to the list. | Store each line the relevant text file as an item in the list. | A sequence data type which contains items separated by commas, enclosed in square brackets. Each item can be a different data type. You can update the contents of the tuple (e.g. you can add items, remove items, edit items or re-order items) |
| *solutionFileList* |
| *iosDevice* | Because there are multiple strings that are related to iOS devices. | Store strings that are related to the relevant type of device. |
| *androidDevice* | Because there are multiple strings that are related to android devices. |
| *windowsDevice* | Because there are multiple strings that are related to windows devices. |
| *devices* | It is the easiest way to store all the type of devices my program supports in one variable. | Store all the types of devices my program supports as individual items. |
| *noTypes* | Tuple | Because there needs to be multiple items which can be easily read, however, it does not need to be updated. | Store each string which the program recognises as a yes/no answer as items in the tuple. | A sequence data type which contains items separated by commas, enclosed in parentheses. Each item can be a different data type. You cannot update the contents of the tuple (e.g. you cannot add items, remove items, edit items or re-order items) |
| *yesTypes* |
| *keyWordFile* | File object | Because it is the easiest way to read data from an external location from the python file. | Store the data the program needs to read the relevant text file. | A TextIOWrapper which allows python to randomly access to text stored externally. |
| *solutionFile* |
| *cases* |
| *audioDict* | Dictionary | Because it is the easiest way to associate two separate items together in one variable (e.g. ‘*android solution 8*’ and ‘*audio\android8.wav*’) | Store the file path to all audio files used in the program. | An unordered hash table with each key having a value assigned to it. The key is unique within the dictionary, whereas the values may or may not be. |

**Test Strategy**

|  |  |  |
| --- | --- | --- |
| What I am testing | How I will test it | Expected outcome |
| That my program will allow me to select android as my operating system. | By typing “*android*” when asked what operating system I run. | The program to obtain its solutions from “*android solution file.txt*” |
| That my program will respond correctly when the user’s input contains a keyword. | By typing in “*my sd card does not work*” when entering in my problem. | The program to detect the keyword “*frozen*” and provide a solution for if the screen if frozen. |
| That my program will respond correctly when the user’s input does not contain any keywords. | By typing in “*Who is Incognito?*” when entering in my problem. | The program not to find a solution and allocate the user a case number, as well as Googling the issue for the user. |
| That my program will create a case file if one does not already exist. | By deleting “*cases.txt*” from the directory the program is in and then running the program. | The program to create cases.txt and allocate the user case number 0. |
| That my program will allow me to exit if the user’s operating system is not supported. | By typing “*ubuntu*” when asked what operating system I run and then “*yes*” when asked if I wish to exit. | The program to ask if I wish to exit the program and after I say “*yes*”, close itself. |
| That my program will allow me to exit once my problem is solved. | By typing in “*yes*” when the program asks me if my problem is solved. | The program to interpret the user’s input as yes to the question by allowing me to exit the program. |
| That my program will shut down correctly when there is a missing text file. | By deleting “*key word file.txt*” from the directory the program is in and then running the program. | The program to realise a key file is missing, display a message saying it will now shut down and then close itself. |

## Developed Solutions

|  |  |  |
| --- | --- | --- |
| Programming Technique | Code | Why I used it |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Testing: (9 marks)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| What I am testing | How I will test it | Expected outcome | Actual outcome | Changes made |
| That my program will allow me to select android as my operating system. | By typing “*android*” when asked what operating system I run. | The program to obtain its solutions from “*android solution file.txt*” | The program said “*Please tell us what is wrong with your android phone*” and loaded “*android solution file.txt*” | None. |
| That my program will respond correctly when the user’s input contains a keyword. | By typing in “*my sd card does not work*” when entering in my problem. | The program to detect the keyword “*sd*” and provide a solution for if the SD card is corrupt. | The program advised me to format my SD card on a PC. | None. |
| That my program will respond correctly when the user’s input does not contain any keywords. | By typing in “*Who is Incognito?*” when entering in my problem. | The program not to find a solution and allocate the user a case number, as well as Googling the issue for the user. | The program did not recognise my problem, allocated me a case number and opened my default web browser with the problem Googled. | None. |
| That my program will create a case file if one does not already exist. | By deleting “*cases.txt*” from the directory the program is in and then running the program. | The program to create “*cases.txt*” and allocate the user case number 0. | An UnboundLocalError message appeared. | I added a line which defines “*caseNumber*” as 0 if “*cases.txt*” does not exist |
| That my program will allow me to exit if the user’s operating system is not supported. | By typing “*ubuntu*” when asked what operating system I run and then “*yes*” when asked if I wish to exit. | The program to ask if I wish to exit the program and after I say “*yes*”, close itself. | The program asked if I would like to exit and after I said yes it closed itself. | None. |
| That my program will allow me to exit once my problem is solved. | By typing in “*yes*” when the program asks me if my problem is solved. | The program to interpret the user’s input as yes to the question by allowing me to exit the program. | The program closed itself. | None. |
| That my program will shut down correctly when there is a missing text file. | By deleting “*key word file.txt*” from the directory the program is in and then running the program. | The program to realise a key file is missing, display a message saying it will now shut down and then close itself. | After telling the program my operating system an error message was displayed and the program closed itself. | None. |

### Criteria Check

|  |  |
| --- | --- |
| Criteria | How it has been met |
| The system should introduce the program. | With a “*print*” command. |
| The system should find out the user’s device. | With an “*input*” command. |
| The system should load a set of solutions relevant to the user’s device. | With “*print*” and “*input*” commands, as well as “*if*” statements and “*for*” loops to find out the user’s operating system, followed by the “*open*” command to open a text file full of solutions for a specific operating system. |
| The system should scan the input and correctly identify keywords. | By using a “*for*” loop and “*.split()*” to separate a list of keywords and an “*if*” statement to detect any keywords. |
| The system should correctly link certain keywords to certain problems and provide a solution for the relevant problem. | By having an integer (“*repeats*”) be linked to both the key word and the solution (which is specific to the problem). |
| The system should allocate the user a case number if a solution has not been found. | With the “*len*”, “*open*” and “*.readlines()*” commands to set the case number as the amount of lines used in cases.txt; this prevents two or more user’s having the same case number. Additionally, if no cases.txt is found a simple “*caseNumber = 0*” command is used. |
| The system should allow the user to exit if the problem was resolved. | By using “*if*” and “*elif*” commands, along with the OS module for specifically exiting the program. |
| The system should provide both audio and visual outputs. | With the “*print*” command and the winsound module. |
| The system should be modular. | By using multiple functions in modules.py. The user launches the program from phones.py |

## Full code (18 marks)

phone.py



modules.py

#Importing external modules

**import** os**,** webbrowser**,** winsound

**def** startUp**():**

#Setting global variables

**global** audioDict**,** yesTypes**,** noTypes**,** keyWordFileList**,** solutionFileList**,** iosDevice**,** androidDevice**,** windowsDevice**,** devices

#Defining variables

yesTypes **=** **(**"yes"**,** "yea"**,** "y"**,** "yep"**,** "aye"**,** "true"**)**

noTypes **=** **(**"no"**,** "nah"**,** "n"**,** "nope"**,** "false"**)**

audioDict **=** **{}**

keyWordFileList **=** **[]**

solutionFileList **=** **[]**

iosDevice **=** **[**"ios"**,** "apple"**,**"iphone"**,** "ipad"**,** "ipod"**]**

androidDevice **=** **[**"android"**,** "samsung"**,** "google"**,** "nexus"**,** "pixel"**]**

windowsDevice **=** **[**"windows"**,** "microsoft"**,** "nokia"**,** "lumia"**,** "zune"**]**

devices **=** **[**iosDevice**,** androidDevice**,** windowsDevice**]**

#Trying to open audio files

**try:**

#Repeat process for each audio file

**for** fileNumber **in** range**(**10**):**

#Add android audio files to dictionary

audioDict**[**"android{0}"**.**format**(**fileNumber**)]** **=** "audio/android" **+** str**(**fileNumber**)**

#Add iOS audio files to dictionary

audioDict**[**"ios{0}"**.**format**(**fileNumber**)]** **=** "audio/ios" **+** str**(**fileNumber**)**

#Add windows audio files to dictionary

audioDict**[**"windows{0}"**.**format**(**fileNumber**)]** **=** "audio/windows" **+** str**(**fileNumber**)**

#Repeat process for each audio file

**for** fileNumber **in** range**(**12**):**

#Add generic audio files to dictionary

audioDict**[**"generic{0}"**.**format**(**fileNumber**)]** **=** "audio/generic" **+** str**(**fileNumber**)**

#Terminating the program if FileNotFoundError (a run time error) is found

**except** FileNotFoundError**:**

#Displaying exit message

**print(**"ERROR: The LoganBerry phone repair service is unable to open due to 1 or more missing files\n"**)**

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic0"**),**winsound**.**SND\_FILENAME**)**

#Running exitSequence

exitSequence**()**

#Displaying intro message

**print(**"Hello and welcome to LoganBerry phone repair services, please note we only support devices running iOS, Android or Windows\n"**)**

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic1"**),**winsound**.**SND\_FILENAME**)**

**def** exitSequence**():**

#Setting global variables

**global** audioDict

#Allowing user to choose to exit

**print(**"Press enter to exit the program\n"**)**

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic2"**),**winsound**.**SND\_FILENAME**)**

enterToExit **=** input**()**

#Exiting program

os**.**\_exit**(**0**)**

**def** yesOrNoExit**():**

#Setting global variables

**global** audioDict**,** yesTypes**,** noTypes

#Starting yes or no loop

**while** **True:**

#Accepting answer

userInputCaseSensitive **=** input**(**"USER: "**)**

#Converting answer to lower case

userInput **=** userInputCaseSensitive**.**lower**()**

#If the user wants to exit

**if** userInput **in** yesTypes**:**

#Displaying exit message

**print(**"Thank you for using our phone repair program\n"**)**

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic3"**),**winsound**.**SND\_FILENAME**)**

#Running exitSequence

exitSequence**()**

#Else, if the user wants to try again

**elif** userInput **in** noTypes**:**

#Exiting function

**return**

#Else yes or no answer not recognised

**else:**

**print(**"Answer not recognised, please answer either yes or no\n"**)**

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic4"**),**winsound**.**SND\_FILENAME**)**

**def** device**():**

#Setting global variables

**global** audioDict**,** iosDevice**,** androidDevice**,** windowsDevice**,** devices**,** deviceModel**,** operatingSystem

#Asking for device model

**print(**"Please tell us your device name/model (e.g. 'iPhone 5C 16GB Blue')\n"**)**

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic5"**),**winsound**.**SND\_FILENAME**)**

#Accepting answer and storing answer under variable, deviceModel

deviceModel **=** input**(**"DEVICE NAME/MODEL: "**)**

#Starting main device loop

**while** **True:**

#Asking for operating system

**print(**"Please tell us what operating system your phone has (either iOS, Android or Windows)\n"**)**

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic6"**),**winsound**.**SND\_FILENAME**)**

#Accepting answer

userInputCaseSensitive **=** input**(**"USER: "**)**

#Converting answer to lower case

userInput **=** userInputCaseSensitive**.**lower**()**

#Repeating process 3 times for the 3 operating systems / device types

**for** deviceType **in** range**(**3**):**

#Repeating process 5 times for the 5 key words linked to each operating system / device type

**for** keyWord **in** range**(**5**):**

#If the user's input equals the list in the devices list that corresponds with the deviceType integer and the string in the devices[deviceType] list that corresponds with the keyWord integer

**if** userInput **==** **((**devices**[**deviceType**])[**keyWord**]):**

#Define operatingSystem as the string that corresponds with the integer 0, which is found in the list that corresponds with the deviceType integer in the devices list

operatingSystem **=** **((**devices**[**deviceType**])[**0**])**

#Exiting function

**return**

#Inform user that their input did not match any keywords and asking if they want to exit

**print(**"Operating system not supported or recognised, would you like to exit the program (if you answer no we will allow you to re-type / re-phrase your previous answer)?\n"**)**

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic7"**),**winsound**.**SND\_FILENAME**)**

#Running yesOrNoExit

yesOrNoExit**()**

**def** managingFiles**():**

#Setting global variables

**global** audioDict**,** keyWordFileList**,** solutionFileList**,** operatingSystem

#Trying to open files in read mode

**try:**

keyWordFile **=** open**(**"key word file.txt"**,** "r"**)**

#Opening solution file specific to one chosen by user

solutionFile **=** open**(**operatingSystem **+** " solution file.txt"**,** "r"**)**

#Terminating the program if FileNotFoundError (a run time error) is found

**except** FileNotFoundError**:**

#Displaying exit message

**print(**"ERROR: The LoganBerry phone repair service is unable to open due to 1 or more missing files\n"**)**

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic0"**),**winsound**.**SND\_FILENAME**)**

#Running exitSequence

exitSequence**()**

#Putting each line of the file as a separate item into a list

**for** lines **in** keyWordFile**:**

#.strip() used to remove the '\n' used to indicate a new line

keyWordFileList**.**append**(**lines**.**strip**())**

#Putting each line of the file as a separate item into a list

**for** lines **in** solutionFile**:**

solutionFileList**.**append**(**lines**)**

**def** diagnosingProblem**():**

#Setting global variables

**global** audioDict**,** keyWordFileList**,** solutionFileList**,** operatingSystem**,** userInput

#Displaying instructions

**print(**"Please tell us what is wrong with your"**,** operatingSystem**,** "phone\n"**)**

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic8"**),**winsound**.**SND\_FILENAME**)**

#Accepting problem

userInputCaseSensitive **=** input**(**"USER: "**)**

#Converting answer to lower case

userInput **=** userInputCaseSensitive**.**lower**()**

#Repeating process 10 times for 10 problems program can diagnose

**for** repeats **in** range**(**10**):**

#Repeating process for each key word (separated by a comma) in the keyWordFileList that corresponds with the integer repeats

**for** keyWords **in** keyWordFileList**[**repeats**].**split**(**","**):**

#If a key word is found

**if** keyWords **in** userInput**:**

#Displaying solution the corresponds with the integer repeats (and with the key word found)

**print(**solutionFileList**[**repeats**],** "\n"**)**

#Playing above message

**if** operatingSystem **==** "android"**:**

winsound**.**PlaySound**(**audioDict**.**get**(**"android" **+** str**(**repeats**)),**winsound**.**SND\_FILENAME**)**

**elif** operatingSystem **==** "ios"**:**

winsound**.**PlaySound**(**audioDict**.**get**(**"ios" **+** str**(**repeats**)),**winsound**.**SND\_FILENAME**)**

**elif** operatingSystem **==** "windows"**:**

winsound**.**PlaySound**(**audioDict**.**get**(**"windows" **+** str**(**repeats**)),**winsound**.**SND\_FILENAME**)**

#Checking if the user has resolved the problem

**print(**"Is your problem resolved?\n"**)**

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic9"**),**winsound**.**SND\_FILENAME**)**

#Running yesOrNoExit

yesOrNoExit**()**

#Exiting function

**return**

#Display message

**print(**"Sorry, your problem was not recognised\n"**)**

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic10"**),**winsound**.**SND\_FILENAME**)**

#Exiting function

**return**

**def** diagnosticFailed**():**

#Setting global variables

**global** audioDict**,** solutionFileList**,** deviceModel**,** operatingSystem**,** userInput

#Trying to assign this case the next case number in the cases.txt file

**try:**

#Defines caseNumber as integer of amount of lines in cases.txt

caseNumber **=** len**(**open**(**"cases.txt"**).**readlines**())**

#Creating a case file (using write mode) if FileNotFoundError (a run time error) is found

**except** FileNotFoundError**:**

cases **=** open**(**"cases.txt"**,** "w"**)**

#Closing case file so it can be used in a different mode

cases**.**close**()**

#Defines caseNumber as 0

caseNumber **=** 0

#Opening case file in appending mode

cases **=** open**(**"cases.txt"**,** "a"**)**

#Appending case details to file

cases**.**write**(**"CASE NUMBER: " **+** str**(**caseNumber**)** **+** " | DEVICE TYPE: " **+** operatingSystem **+** " | DEVICE MODEL:" **+** deviceModel **+** " | USER'S PROBLEM: " **+** userInput **+** "\n"**)**

#Closing case file

cases**.**close**()**

#Telling user they have a case number and we are googling the problem for them

**print(**"You have been assigned case number " **+** str**(**caseNumber**)** **+** ". A technician will be with you shortly\nWe will also show you google results related to your problem."**)**

#Playing above message

winsound**.**PlaySound**(**audioDict**.**get**(**"generic11"**),**winsound**.**SND\_FILENAME**)**

#Opening manipulated Google URL containing key information about phone problem

webbrowser**.**open**(**"https://google.com/?q=" **+** operatingSystem **+** "+" **+** deviceModel **+** "+" **+** userInput**)**

#Running exitSequence

exitSequence**()**