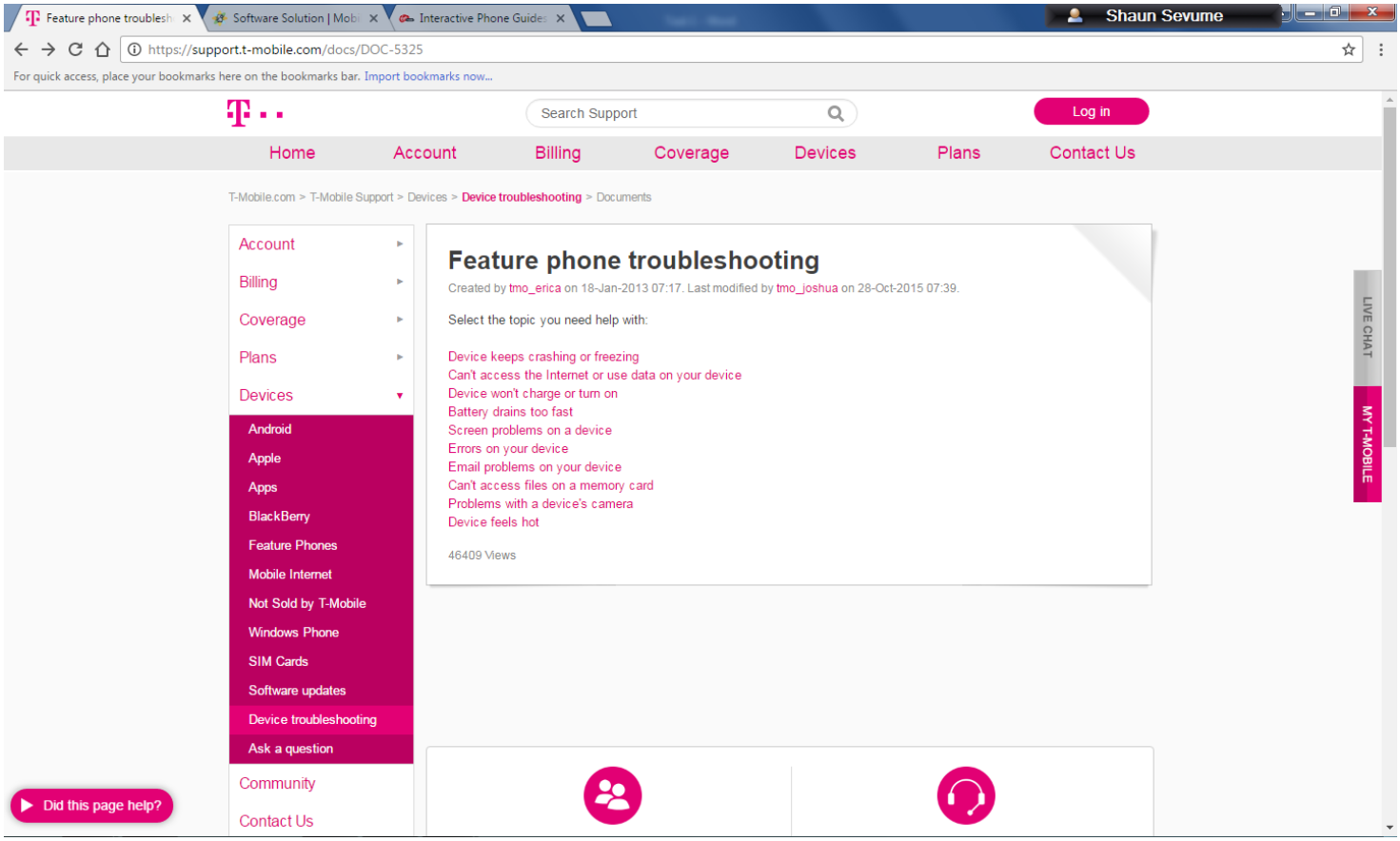
1. Analysis

I'm required to create a troubleshooting program for mobile phones that will be able to identify the user's device and load the correct troubleshooting program before analysing the problem to give them the appropriate help they need. If no solution is found, they will instead be given a case number and sent to a technician.

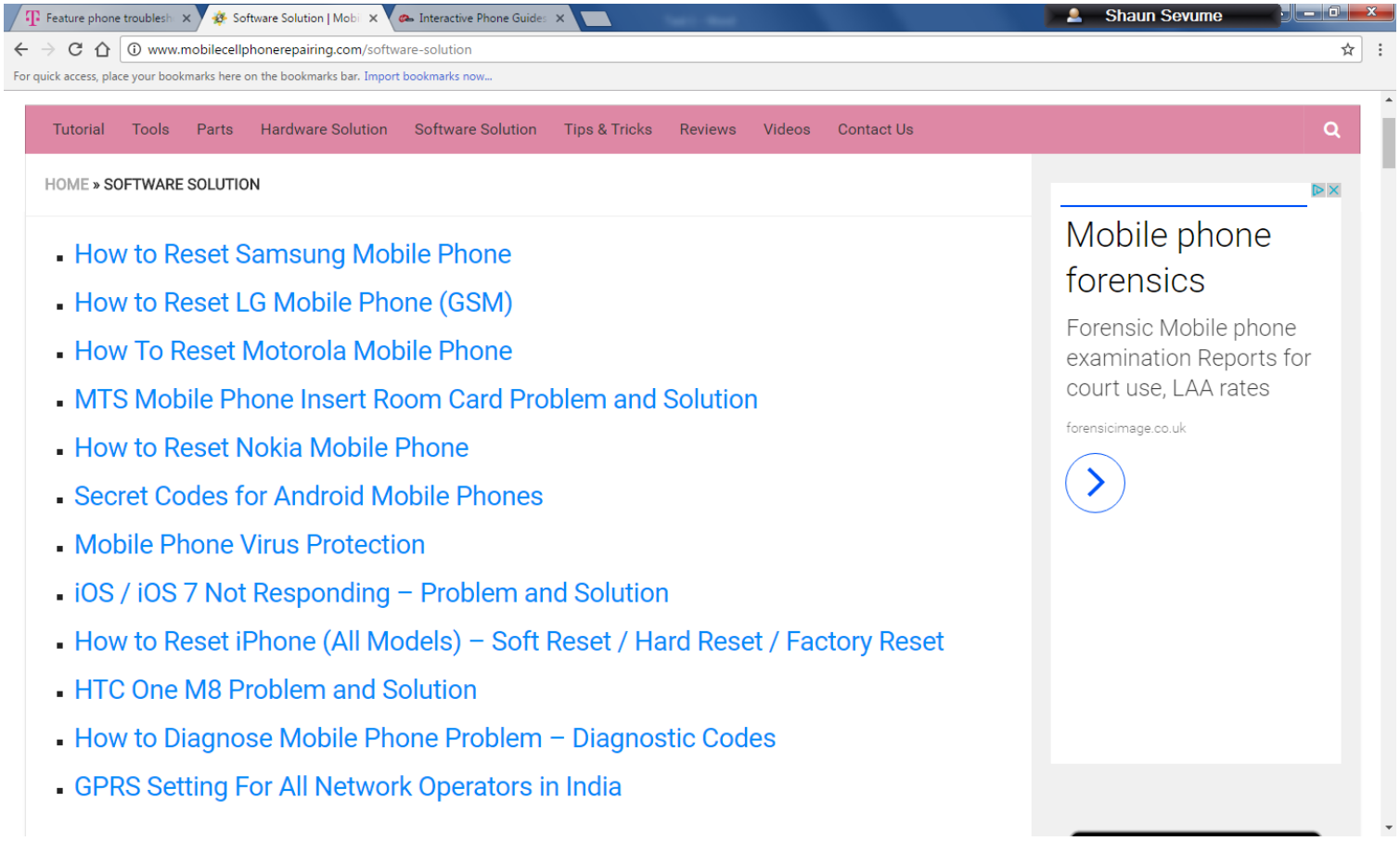
I did some research onto the current mobile troubleshooting programs online, and I found three, all of which were designed in a very similar fashion. In all of them, they required the user to click around to find what they were looking for.



The first website was T-Mobile, a popular mobile network. https://support.t-mobile.com/docs/DOC-5325

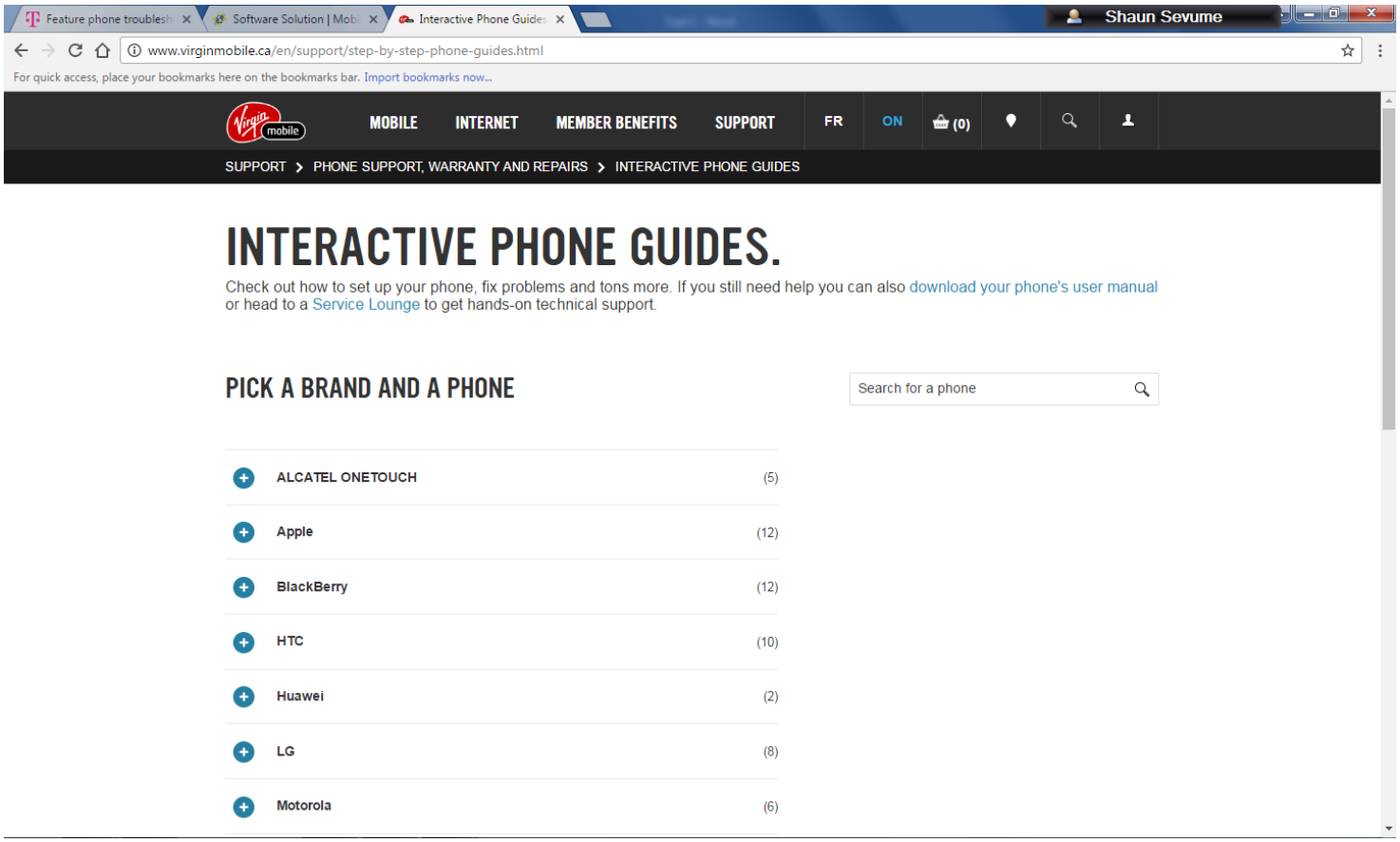
This site is a good example of what my program should be like. It has clear, broad problems and solutions. There is even an "Ask a question" tab and "Contact us", so T-Mobile are there to support you if you don't find your answer. Though not every problem is covered, it's easy to add to the list. There's also a search at the top to get around the site quickly. The only downside was the fact that solutions were not device specific, so instructions would be vague examples rather than clear direct ones for the user's particular device/OS.

The second website is Mobile phone cell repairing. It seemed to be a simple mobile troubleshooting site. http://www.mobilecellphonerepairing.com/software-solution



Similar to the T-Mobile website, this website had clear problems and solutions, and some were even device specific. A search bar and a “Contact us” tab also featured here, but at times user friendliness felt limited, and a lot of reading had to be done before making choices.

The third website was Virgin Mobile, another popular network provider.
http://www.virginmobile.ca/en/support/step-by-step-phone-guides.html



This particular website required you to scroll down until you found your device model, and then click it, which would present a list of options to choose from, which the user of course had to click on. You could search for the make and model, but not all instances were there present. I don't find this very user-friendly, because it requires a lot of input from the user, and they have to find their way around the site to get their answer.

As noted in tasks 1 and 2, these three were not actually programs, but rather a collection of hyperlinks on a site that contain information on specific subjects.

My program must be user friendly and easily understandable, targeted at people of a 12+ age. It must be able ask the user for their name, as well as other details such as e-mail address. It should also be able to generate a case number to give to the technician if no solution is found. It should also be able to store this data in a txt file. It must be able to run on most PC's, as python isn't a resource consuming program. Minimum requirements will be an Intel Pentium processor @500MHz and 256MB of RAM and an operating system of Windows XP. Other components needed include a mouse, a monitor and a keyboard as well as additional components found on a motherboard, such as a hard drive etc. It will be created in python.

Like with task 2, I couldn't find any similar programs to the one I was required to make on the internet. This meant that my program would be unique, set apart from the others by its own features, such as problem recording and case number allocation, as well as it's abilities to store the user's details and remember them for when a technician would come along to manually look at the problem. The trouble-shooters I did look at fitted the criteria for task 1, but not tasks 2 and 3.

Success Criteria

- Be able to request and store input data, such as the user's name or e-mail address
- Load the correct troubleshooting program based on the user's device
- Give an appropriate solution based on what it's identified and if no solution is found, store their details with a case number to refer to a technician.
- Maintain a sense of user-friendliness so that it stands out from other programs, (i.e, not just instantly requesting their details, but maybe greet them first.)
- Have all variables correctly defined in the code
- Make sure there are no errors, e.g. indentation error
- Solve display problems
- Solve a phone not being able to charge
- Solve a phone's touch screen problems
- Solve a phone's volume problems
- Solve phone brightness problems

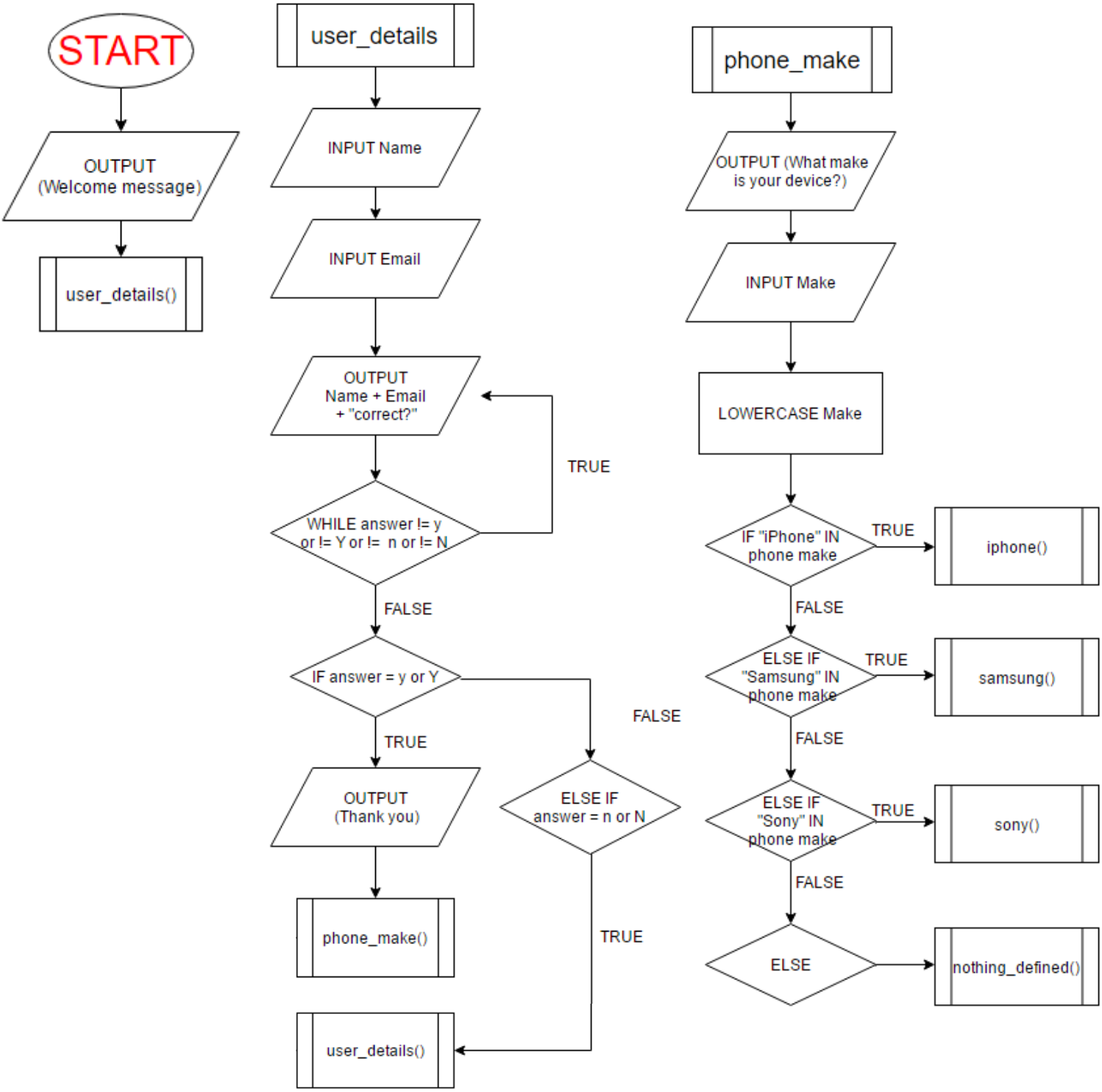
Test plan

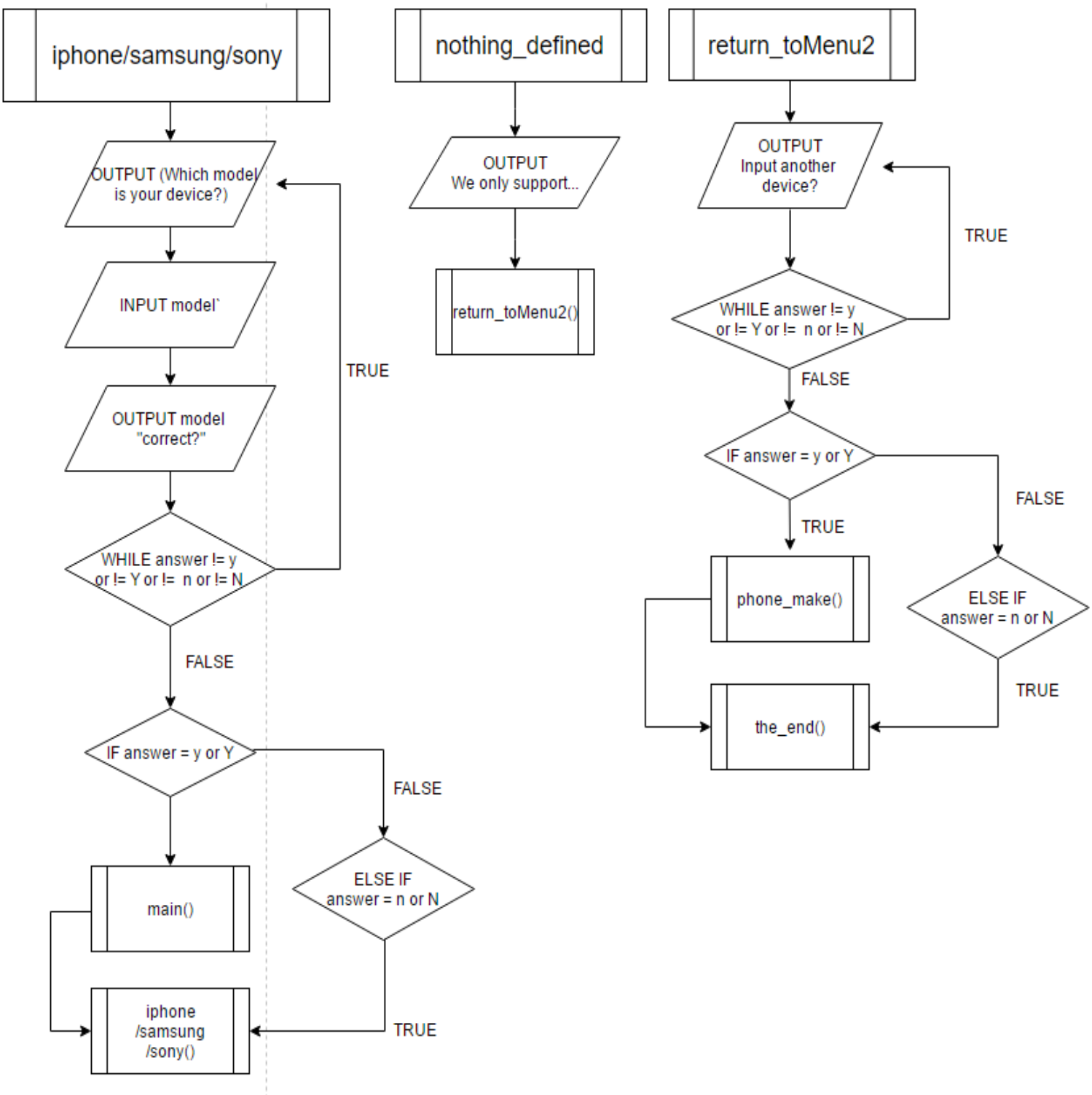
Test no.	Testing	Input (if applicable)	Expected output
1	Does it address the user by their name based on their input?	Shaun	Okay Shaun…
2	Does it wait the required time before carrying out the next function using time.sleep?	Shaun	(After the amount of time specified in the time.sleep function) Okay Shaun…
3	Does it correctly identify things like phone model?	iPhone	Which iPhone model?
4	Does it crash when something unexpected is inputted?	qwerty (anything random)	Error
5	Is my code correctly indented?	N/A	(Indentation) Error
6	Are variables correctly defined?	N/A	(Unidentified variable) Error
7	Is there an else statement in case the if statements are not met?	N/A	If you don't put else, there will be no output if no conditions are met.
8	Does it give appropriate outputs based on input?	Sony	What Sony model is it?

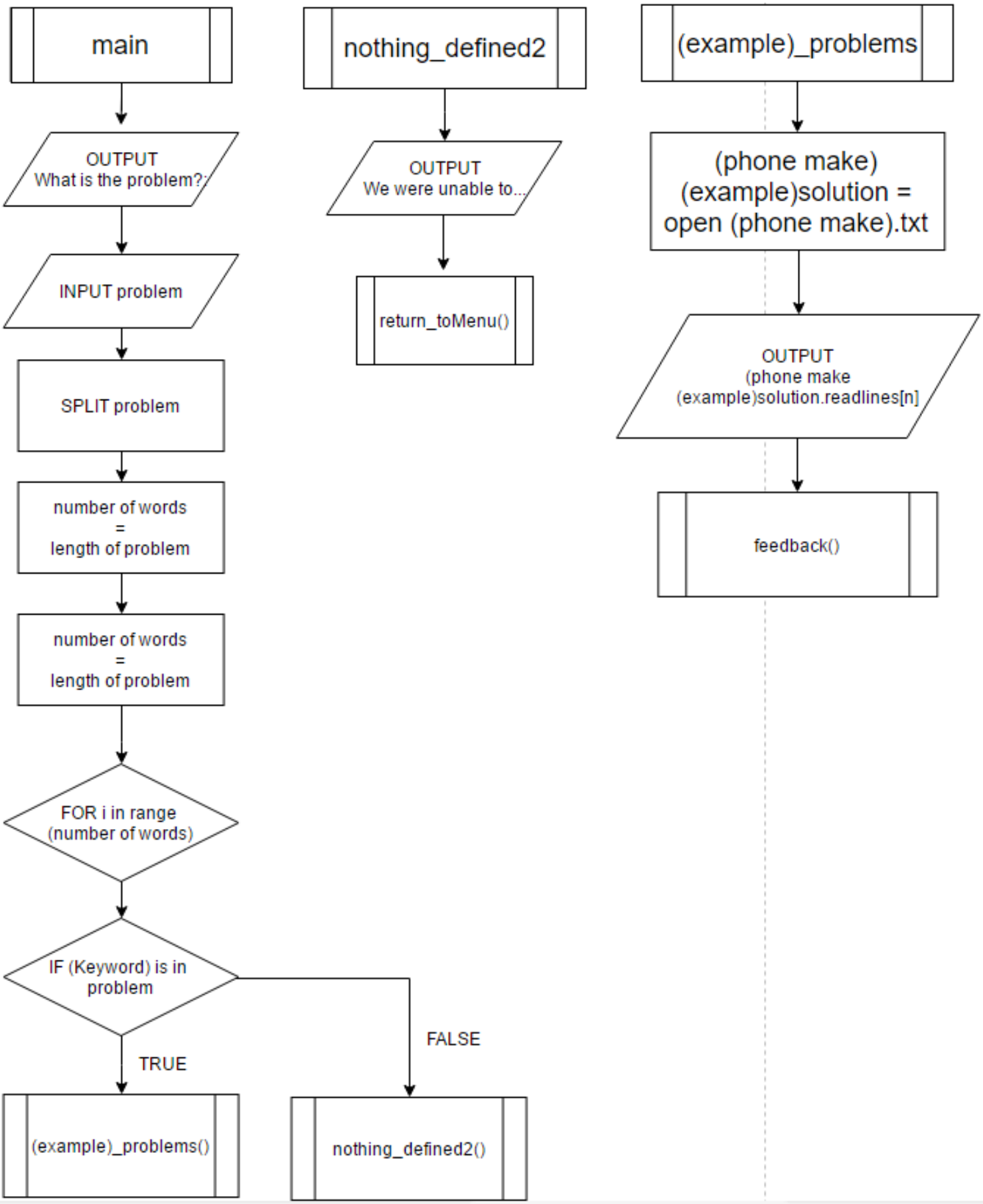
9	Does it record the user's details on a txt file if the issue wasn't solved?	(After being asked if their problem was solved) n/N	We have saved your details… Your case number is…
10	Does it keep on asking for an input for a specific variable before a condition is met? (Iteration)	(No keywords detected)	Try to enter something with some of these keywords. (A list of some keywords)

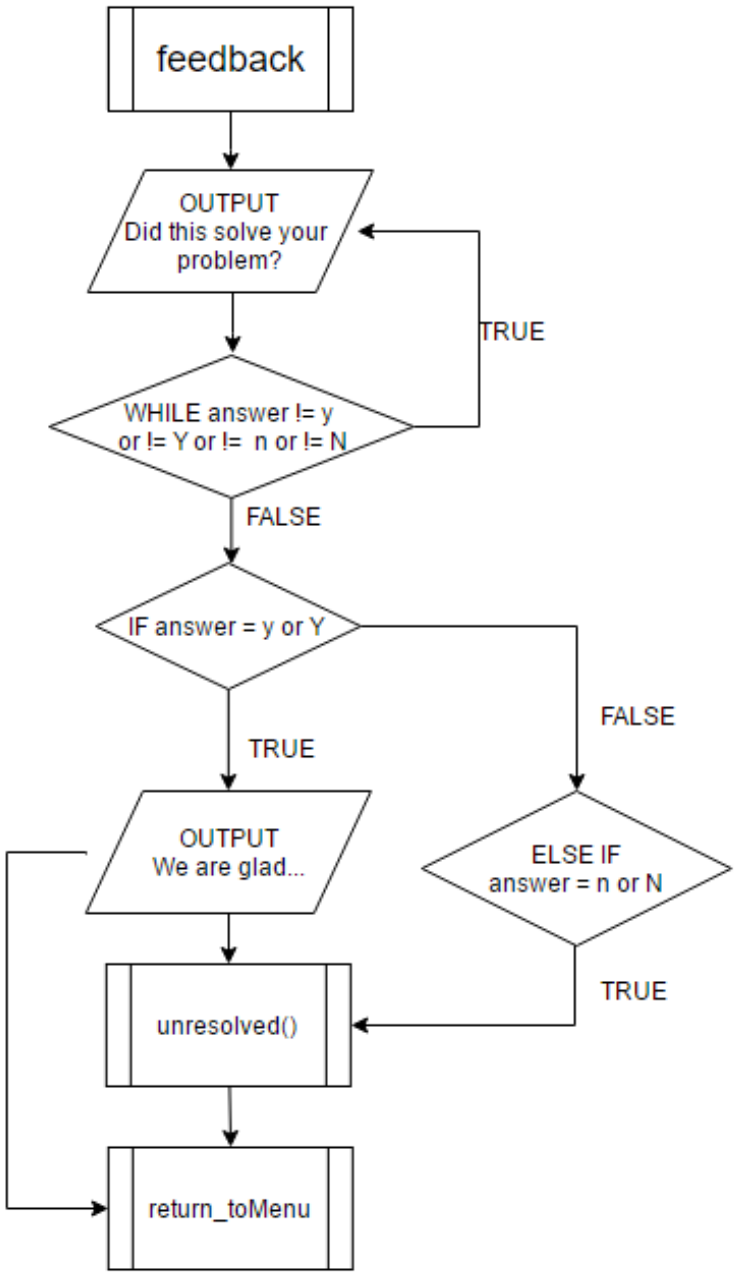
2. Design

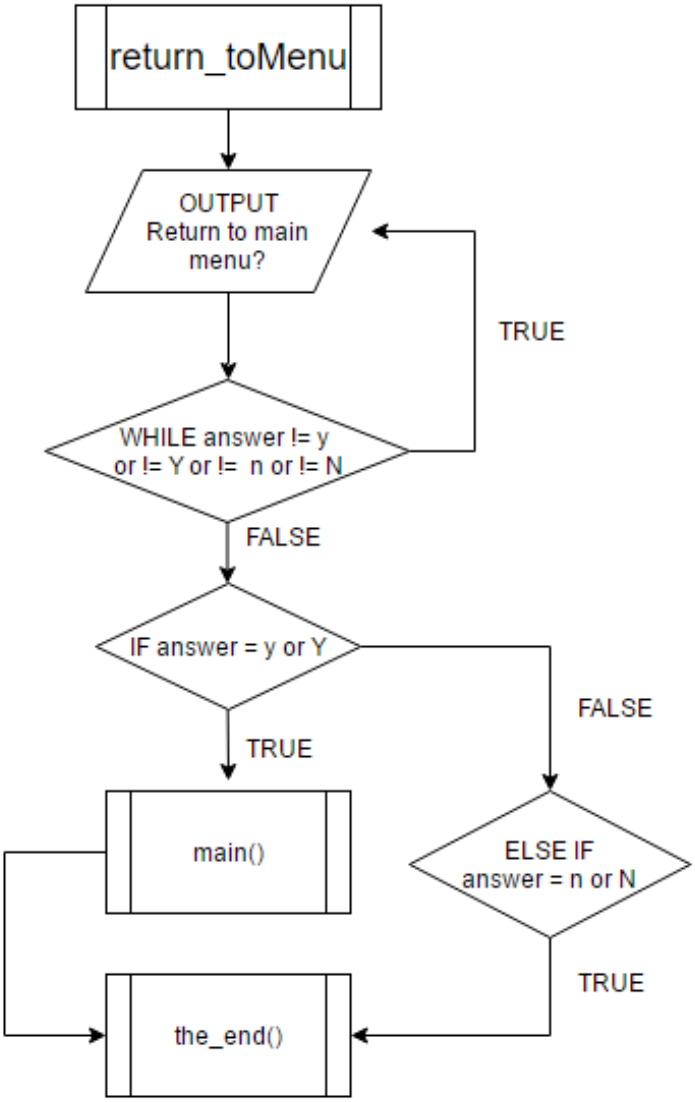
Flowchart

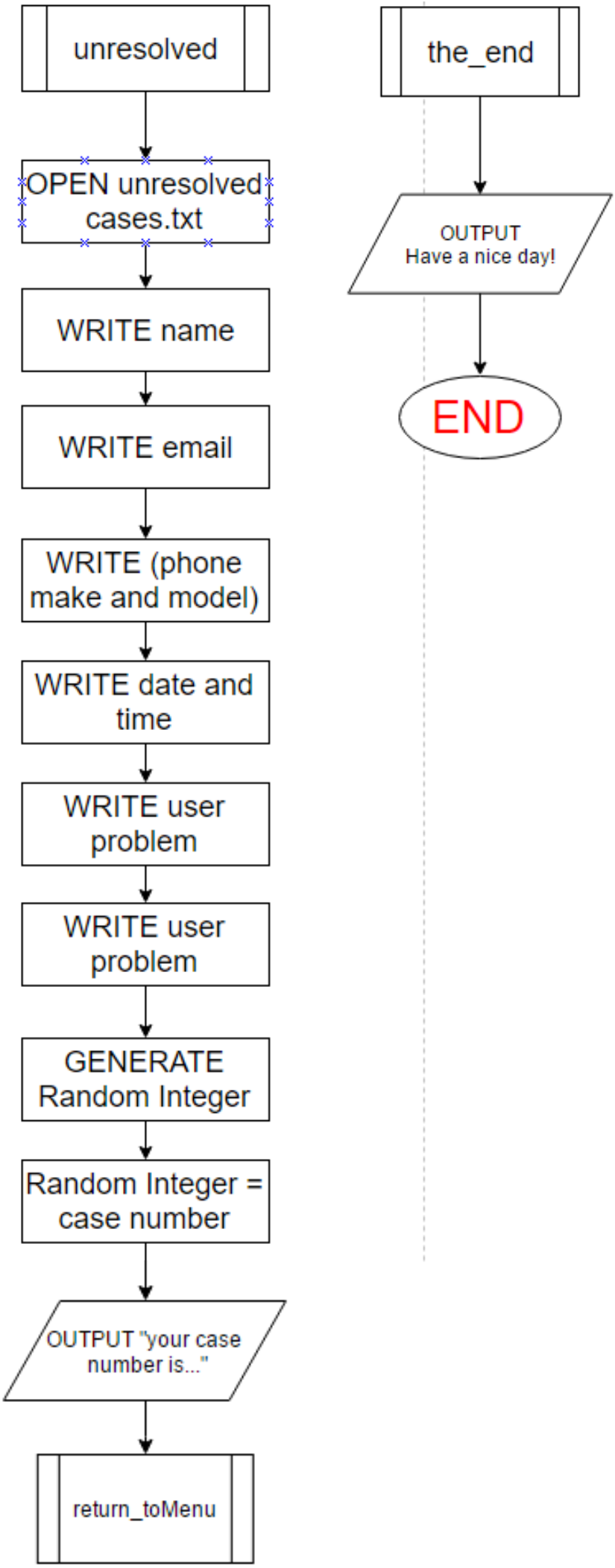



iphone/samsung/sony
OUTPUT (Which model is your device?)
INPUT model`
OUTPUT model "correct?"
WHILE answer != y or != Y or != n or != N
TRUE
FALSE
IF answer = y or Y
FALSE
ELSE IF answer = n or N
TRUE
main()
iphone /samsung /sony()
nothing_defined
OUTPUT We only support...
return_toMenu2()
return_toMenu2
OUTPUT Input another device?
TRUE
WHILE answer != y or != Y or != n or != N
FALSE
IF answer = y or Y
FALSE
TRUE
phone_make()
ELSE IF answer = n or N
TRUE
the_end()


main
OUTPUT
What is the problem?
INPUT problem
SPLIT problem
number of words
=
length of problem
number of words
=
length of problem
FOR i in range
(number of words)
IF (Keyword) is in
problem
TRUE
FALSE
(example)_problems()
nothing_defined2()
nothing_defined2
OUTPUT
We were unable to...
return_toMenu()
(example)_problems
(phone make)
(example)solution =
open (phone make).txt
OUTPUT
(phone make
(example)solution.readlines[n]
feedback()


feedback
OUTPUT
Did this solve your problem?
TRUE
WHILE answer != y or != Y or != n or != N
FALSE
IF answer = y or Y
FALSE
TRUE
OUTPUT
We are glad...
ELSE IF answer = n or N
TRUE
unresolved()
return_toMenu


return_toMenu
OUTPUT
Return to main menu?
TRUE
WHILE answer != y or != Y or != n or != N
FALSE
IF answer = y or Y
FALSE
TRUE
main()
ELSE IF answer = n or N
TRUE
the_end()


unresolved
OPEN unresolved cases.txt
WRITE name
WRITE email
WRITE (phone make and model)
WRITE date and time
WRITE user problem
WRITE user problem
GENERATE Random Integer
Random Integer = case number
OUTPUT "your case number is..."
return_toMenu
the_end
OUTPUT
Have a nice day!
END

Pseudocode

```
START

IMPORT time

IMPORT re

IMPORT random

IMPORT date and time

OUTPUT Welcome message


FUNCTION UNRESOLVED:

        OPEN unresolved cases.txt

        WRITE Name

        WRTE email

        WRITE Date and Time

        WRITE User Problem

        GENERATE Random Integer

        Random Integer = Case number

        WRITE Case number

        OUTPUT “your case number is” + Case number

        CALL NOTHING DEFINED


FUNCTION NOTHING DEFINED:

        OUTPUT "Remember, we only support iPhone, Sony and Samsung.”

        CALL PHONE MAKE


FUNCTIONED NOTHING DEFINED 2:

        OUTPUT No keywords found message

        CALL RETURN TO MENU
```

```
FUNCTION FEEDBACK:
        WHILE feedback ISN’T y, Y, OR n, N:
                OUTPUT “did we solve the problem?”
                INPUT feedback
                IF feedback = y or Y THEN
                        OUTPUT “We are glad…”
                        CALL RETURN TO MENU
                ELSE IF feedback = n or N THEN
                        OUTPUT “We are sorry…”
                        CALL UNRESOLVED
                END IF
        END WHILE

FUNCTION DISPLAY PROBLEMS:
        IF “iphone” in phone make THEN
                OPEN iphone.txt, READ
                OUTPUT iphone.txt line 0
                CALL FEEDBACK
        ELSE IF "samsung" in phone make THEN
                OPEN samsung.txt, READ
                OUTPUT samsung.txt line 0
                CALL FEEDBACK
        ELSE IF "sony" in phone make THEN
                OPEN sony.txt, READ
                OUTPUT sony.txt line 0
                CALL FEEDBACK
        END IF

FUNCTION BATTERY PROBLEMS:
        IF “iphone” in phone make THEN
                OPEN iphone.txt, READ
                OUTPUT iphone.txt line 1
                CALL FEEDBACK
        ELSE IF "samsung" in phone make THEN
                OPEN samsung.txt, READ
                OUTPUT samsung.txt line 1
```

```
            CALL FEEDBACK
        ELSE IF "sony" in phone make THEN
            OPEN sony.txt, READ
            OUTPUT sony.txt line 1
            CALL FEEDBACK
        END IF

FUNCTION TOUCH SCREEN PROBLEMS:
        IF “iphone” in phone make THEN
            OPEN iphone.txt, READ
            OUTPUT iphone.txt line 2
            CALL FEEDBACK
        ELSE IF "samsung" in phone make THEN
            OPEN samsung.txt, READ
            OUTPUT samsung.txt line 2
            CALL FEEDBACK
        ELSE IF "sony" in phone make THEN
            OPEN sony.txt, READ
            OUTPUT sony.txt line 2
            CALL FEEDBACK
        END IF

FUNCTION BRIGHTNESS PROBLEMS:
        IF “iphone” in phone make THEN
            OPEN iphone.txt, READ
            OUTPUT iphone.txt line 3
            CALL FEEDBACK
        ELSE IF "samsung" in phone make THEN
            OPEN samsung.txt, READ
            OUTPUT samsung.txt line 3
            CALL FEEDBACK
        ELSE IF "sony" in phone make THEN
            OPEN sony.txt, READ
            OUTPUT sony.txt line 3
            CALL FEEDBACK
        END IF
```

```
FUNCTION VOLUME PROBLEMS:
        IF “iphone” in phone make THEN
                OPEN iphone.txt, READ
                OUTPUT iphone.txt line 4
                CALL FEEDBACK
        ELSE IF "samsung" in phone make THEN
                OPEN samsung.txt, READ
                OUTPUT samsung.txt line 4
                CALL FEEDBACK
        ELSE IF "sony" in phone make THEN
                OPEN sony.txt, READ
                OUTPUT sony.txt line 4
                CALL FEEDBACK
        END IF

FUNCTION RETURN TO MENU:
        WHILE return ISN’T y, Y, n OR N:
                OUTPUT Return to main menu?
                INPUT return
                IF return = y OR Y THEN
                        CALL MAIN
                ELSE IF RETURN = n OR N THEN
                        CALL THE END
                END IF
        END WHILE
```

```
FUNCTION MAIN:
        OUTPUT “What is the problem with your device?"
        INPUT Problem
        LOWERCASE Problem
        SPLIT Problem
        Number of words = LENGTH OF Problem
        Counter = 0
        WHILE Counter != 999:
                FOR I IN THE RANGE of Number of words:
                        IF Problem = "blank" OR "turn" AND "on" OR “screen” THEN
                                CALL DISPLAY PROBLEMS
                        ELSE IF Problem = "charge" OR "battery" OR "dead" OR "charging" OR "die “THEN
                                CALL BATTERY PROBLEMS
                        ELSE IF Problem = "touch" AND “screen” OR “touch” THEN
                                CALL TOUCH SCREEN PROBLEMS
                        ELSE IF Problem = "dim" OR "bright" OR "brightness" OR “dark” THEN
                                CALL BRIGHTNESS PROBLEMS
                        ELSE IF Problem = "volume" OR "sound" OR "speaker" THEN
                                CALL VOLUME PROBLEMS
                        ELSE:
                                CALL NOTHING DEFINED 2
                        END IF
                END FOR
                Counter+1
                IF counter = 999 THEN
                        CALL Nothing Defined
                END IF
        END WHILE
```

```
FUNCTION SONY:
        WHILE confirmation ISN’T y, Y, OR n, N:
                OUTPUT “what sony model is it?”
                INPUT MODEL
                GLOBAL MODEL
                INPUT “Sony ” + MODEL + “,correct?”
                IF confirmation = y or Y THEN
                        CALL MAIN
                        BREAK
                ELSE IF confirmation = n or N THEN
                        CALL SONY
                        BREAK
                END IF
        END WHILE

FUNCTION SAMSUNG:
        WHILE confirmation ISN’T y, Y, OR n, N:
                OUTPUT “what samsung model is it?”
                INPUT MODEL
                GLOBAL MODEL
                INPUT “Samsung ” + MODEL + “,correct?”
                IF confirmation = y or Y THEN
                        CALL MAIN
                        BREAK
                ELSE IF confirmation = n or N THEN
                        CALL SAMSUNG
                        BREAK
                END IF
        END WHILE
```

```
FUNCTION IPHONE:
        WHILE confirmation ISN’T y, Y, OR n, N:
                OUTPUT “what iphone model is it?”
                INPUT MODEL
                GLOBAL MODEL
                INPUT “iPhone ” + MODEL + “,correct?”
                IF confirmation = y or Y THEN
                        CALL MAIN
                        BREAK
                ELSE IF confirmation = n or N THEN
                        CALL IPHONE
                        BREAK
                END IF
        END WHILE

FUNCTION PHONE MAKE:
        OUTPUT “what make is your device?”
        INPUT phone make
        LOWERCASE phone make
        GLOBAL phone make
        IF "iphone" in phone make THEN
                CALL IPHONE
        ELSE IF "samsung" in phone make THEN
                CALL SAMSUNG
        ELSE IF "sony" in phone make THEN
                CALL SONY
        ELSE:
                CALL NOTHING DEFINED
        END IF

FUNCTION USER DETAILS:
        OUTPUT “what’s your name?”
        INPUT Name
        GLOBAL Name
        OUTPUT “what’s your email address?”
        INPUT Email
```

```
        GLOBAL Email
        WHILE confirmation ISN'T y, Y, n OR N:
                OUTPUT "your name is (Name) and your email is (Email), correct?
                INPUT confirmation
                IF confirmation = y OR Y THEN
                        CALL PHONE MAKE
                        BREAK
                ELSE IF confirmation n OR N THEN
                        CALL USER DETAILS
                        BREAK
                END IF
        END WHILE

    CALL USER DETAILS
    END
```

Variable and Validation table

Variable Name	Type	Validation	Description
record	N/A	N/A	The variable used to record the user's details in the txt file.
feedback	String	y/Y or n/N	The user is asked if their problem has been resolved beforehand. The answer is then assigned to the variable feedback and an appropriate output will be displayed depending on if the user inputted y or n.
(phone model)_(problem)Solution	N/A	N/A	The name of this variable changes depending on the user's phone model and their problem. However, they all do the same thing. This variable is assigned to the 'open' function, to open the appropriate txt file. It is then used to read a specific line of the opened txt file, by adding .readlines[(Number)] after the variable.
return1	String	y/Y or n/N	The user is asked if they want to return to the main menu. Depending on the user input, it will either call another function that terminates the program or return to the main menu.
Validation	Integer	Numbers without decimal point in range (0-999)	This is a counter for one of the while loops. The program reads the user's input trying to look for specific keywords, however, the loop

			must be run several times before one is found. To make sure the loop doesn't run infinitely, this variable will keep track of how many times it has been run, by adding one to the variable before it repeats. When the counter reaches 999, an if statement will ensure that the program stops the loop and goes to another function that tells the user nothing was found.
problem	String	All characters	When the user is asked what their problem is, their input is assigned to this variable to be used later on in the code, such as when it needs to record the user's details, along with their problem.
problem_lower	String	All characters	After the variable 'problem' has been assigned to the user's input, it is then made lowercase by the .lower() function, and the new lowercase version is assigned to the problem_lower.
user_problem	String	All characters	When the problem_lower function has been assigned to something, it gets split with the .split function and the split version of the lowered version of the original input is assigned to the variable user_problem.
numberOfWords	Integer	Numbers without any decimal points above 0 (Based on how long the user's input is)	The len function is used to find out how many characters there are in the variable user_problem. That number is assigned to the variable 'numberOfWords'.
confirmation	String	y/Y or n/N	The user is asked to confirm their input. To confirm, they say y/Y and to re-enter whatever they were confirming, they say n/N.
model	String	All characters	When the user is asked to input their device model, their input is assigned to this variable.
make	String	All characters	When the user is asked to input the make of their device, their input is assigned to this variable.
make_lower	String	All characters	After the variable make has been assigned to the user's input, it is then made lowercase by the .lower() function, and the new lowercase version is assigned to the make_lower variable.
name	String	All characters	When the user is asked for their name, their input is assigned to this variable.
email	String	All characters	When the user is asked for their e-mail address, their input is assigned to this variable.

unresolved	Function	When feedback = n/N	A def function that is run when the user's problem cannot be solved. It opens a txt file to record their details, and allocates a case number to them, to be looked at by a technician. It then sends them to the return_toMenu() function.
nothing_defined	Function	When no valid models are detected in the variable 'make_lower'	It reminds the user that only iPhone, Samsung and Sony are supported before taking them to the return_toMenu
nothing_defined2	Function	When no keywords are detected in the variable user_problem, which is where the user inputs their problem.	It tells the user that no keywords could be identified and then takes them to the return_toMenu function.
feedback	Function	After a solution has been printed, this function is called.	The user is asked if their problem has been resolved beforehand. If the answer is yes they will be taken to the return_toMenu function. If the answer is no, then they will be taken to the unresolved function.
(example)_problems	Function	If the keywords relating to whatever 'example' may be are detected in the user's input, the appropriate function is run. For example, if the keyword relating to 'display' is detected, then the function display_problems will be called.	There is one def function for each possible solution. In each of the function, there is first an if statement to decide which txt file to open. If the user has an iphone, then "iphone.txt" will be opened, and so on. Once the appropriate file is opened, the appropriate line in accordance to the user's input will be read. For example, if the user's input was related to battery problems and they had a Samsung, then the txt file "Samsung.txt" would be opened, and the line that had the solutions related to the battery would be read and outputted by the program.
the_end	Function	If the user inputs n when asked if they want to return to the menu	This is the function that terminates the program, using the SystemExit command.
return_toMenu	Function	Called if the user answers "y/Y" when asked if their problem was solved, after their details have been recorded in a txt file or after the nothing_defined2 function is called.	Contains a while loop that keeps asking the user if they want to return to the menu until they enter y or n.

return_toMenu2	Function	Called after the nothing_defined function is called.	Contains a while loop that keeps asking the user if they want to input another device until they enter y or n.
main	Function	After the user confirms their phone model.	As the main part of the code, it contains the piece of code that asks the user what their problem is, before checking the input against pre-defined keywords and looking for a match. If any are found, the appropriate function will be called. If no keywords are found, then the nothing_defined2 function is called.
sony/samsung/iphone	Function	sony is called if they keyword “sony” is detected in the variable make_lower, which represents the phone make. In the same way, the samsung and iphone functions are called.	All 3 functions do the same thing. They ask the user what model their phone is, and once they confirm it, they get taken to the function main.
phone_make	Function	After the user confirms that their details are correct.	It asks them for their phone make, before converting the input into lowercase to match against the 3 pre-defined makes: iphone, samsung and sony, so that the user can be taken to the relevant function. If their input doesn’t include these models, they will instead be taken to the nothing_defined function.
user_details	Function	Called when the program is first run, and if the user inputs “n/N” when asked to confirm their details.	It asks the user for their name and email address, before assigning both of them to variables that are then made global. It then asks them to confirm their input. If their answer is yes, they will be taken to the phone_make function. If their answer is “n/N”, the function will be re-called so that they can re-enter their input.

3. Development

To start off the code, I used the import function to import time. This would allow me to use the time.sleep function that would come in handy later on in the program.

Task 3.py - G:/Task 3/Task 3.py (3.5.1)

File Edit Format Run Options Window Help

```
import time
```

Code:

This was the first def function I made in the code. I made it in charge of assigning and

```
def user_details():
    print("First off, what is your name?")
    name = input("Name: ")
    print("Okay, now what is your email address?")
    email = input("Email: ")
    print("So your name is " + name + " and your email is " + email + ", correct?")

    confirmation = 0

    while confirmation != "y" or confirmation != "Y" or confirmation != "n" or confirmation != "N":

        confirmation = input("Correct? (y/n)")

        if confirmation == "y" or confirmation == "Y":
            print("Thank you for sharing your details. They will be kept on our secure server.")
            phone_details()

        elif confirmation == "n" or confirmation == "N":
            user_details()

user_details()
```

storing the user's details to the variables name and email respectively. If the user wished to change these at a later point in the program, they would be referred back to here in the code. I used a similar while loop to those of task 2, that made sure the user could only progress if y/Y or n/N were entered.

Code:

When I was making this while loop, I discovered a slight problem. For the if statements inside the while loop, if they didn't have a function called in them, the loop would keep repeating. For example, Where it says phone_details(), python would call on that def function and go to that part of the code. However, if this was absent and the print function was there only, it would execute the print function, but also the loop again. To 'escape' the

loop, I tried to call another function in the if statements, so python was basically jumping

```
Welcome to the multicellular Troubeshooting program version 3.0! In addition
e model to tailor results to it, as well as giving you a refrerence number f
In order to make this work, we're going to need a few details from you.
First off, what is your name?
Name: Shaun
Okay, now what is your email address?
Email: SShaun304@gmail.com
So your name is Shaun and your email is SShaun304@gmail.com, correct?
Correct? (y/n)y
Thank you for sharing your details. They will be kept on our secure server.
Okay, what phone make your device?
Phone make: 6
Correct? (y/n)
```

around bits of the code.

However, this didn't work. Here was the output for the same code above.

As you can see, even though I had confirmed the details were correct, it went to the other DEF function for the phone make and model, but the loop was still going, as python was still asking if the details were correct. Then I remembered more effective way to exit a while loop. By using the break function. This was my code with the new function implemented.

And below is the output with this code, which had resolved the issue of the while loop not stopping.

```
def user_details():
    print("First off, what is your name?")
    name = input("Name: ")
    print("Okay, now what is your email address?")
    email = input("Email: ")
    print("So your name is " + name + " and your email is " + email + ", correct?")

    confirmation = 0

    while confirmation != "y" or confirmation != "Y" or confirmation != "n" or confirmation != "N":

        confirmation = input("Correct? (y/n)")

        if confirmation == "y" or confirmation == "Y":
            print("Thank you for sharing your details. They will be kept on our secure server.")
            phone_details()

        elif confirmation == "n" or confirmation == "N":
            user_details()
        break

user_details()
```

```
Welcome to the multicellular Troubeshooting program version 3.0! In addition
e model to tailor results to it, as well as giving you a refrerence number f
In order to make this work, we're going to need a few details from you.
First off, what is your name?
Name: Shaun
Okay, now what is your email address?
Email: SShaun304@gmail.com
So your name is Shaun and your email is SShaun304@gmail.com, correct?
Correct? (y/n)y
Thank you for sharing your details. They will be kept on our secure server.
Okay, what phone make your device?
Phone make: 6
>>>
```

Here in my code, I encountered another while loop problem. Here was the code.

```
if make == "iphone":
    model = 0

    while model != "3" or model != "4" or model != "5" or model != "6":
        print("What model is your device?(Enter the number ONLY, no letters)")
        model = input("Phone model: ")

        if model == "3":
            validation = 0

            while validation != "3" or validation != "3gs":
                validation = input("iPhone 3 or iPhone 3GS? ")
                validation = validation.lower()

                if validation == "3" or validation == "3gs":
                    main()
                    break


        elif model == "4":
            validation = 0

            while validation != "4" or validation != "4s":
                validation = input("iPhone 4 or iPhone 4S? ")
                validation = validation.lower()

                if validation == "4" or validation == "4s":
                    main()
                    break

        elif model == "5":
            validation = 0

            while validation != "5" or validation != "5c" or validation != "5s":
                validation = input("iPhone 5, iPhone 5C or iPhone 5S? ")
                validation = validation.lower()

                if validation == "5" or validation == "5c" or validation == "5s":
                    main()
                    break


        elif model == "6":
            print("iPhone 6, iPhone 6+, iPhone 6S or iPhone 6S+ ?")
            validation = 0

            while validation != "6" or validation != "6+" or validation != "6s" or validation != "6s+":
                validation = input("iPhone 6, iPhone 6+, iPhone 6S or iPhone 6S+ ? ")
                validation = validation.lower()

                if validation == "6" or validation == "6+" or validation == "6s" or validation == "6s+":
                    main()
                    break
```

The problem was that after identifying a make and model, it was supposed to go to another DEF function (which I had yet to work on so I just made it print “wurf” to let me know if it was working. Though it did print “wurf”, it also repeated the while loop question. Adding the break function here didn’t work. This was the output.

```
Welcome to the multicellular Troubeshooting program version 3.0! In additior
e model to tailor results to it, as well as giving you a refrerence number f
In order to make this work, we're going to need a few details from you.
First off, what is your name?
Name: Shaun
Okay, now what is your email address?
Email: d
So your name is Shaun and your email is d, correct?
Correct? (y/n)y
Thank you for sharing your details. They will be kept on our secure server.
Okay, what phone make your device?
Phone make: iphone
What model is your device?(Enter the number ONLY, no letters)
Phone model: f
What model is your device?(Enter the number ONLY, no letters)
Phone model: 4
iPhone 4 or iPhone 4S? 4
wurf
What model is your device?(Enter the number ONLY, no letters)
Phone model:
```

To fix this, I knew I would have to implement the break function to stop the loop from repeating, but there was nowhere to put it. There was already one in each of the IF statements and that didn't do anything because there was another while loop in it. If I put it at the end of the statements, then the while loop would only run once. So to make it more like the user_details() function (see above) I split the code up into more def functions. Here is the code.

```
def iphone3():
    validation = 0

    while validation != "3" or validation != "3gs":
        validation = input("iPhone 3 or iPhone 3GS? ")
        validation = validation.lower()

        if validation == "3" or validation == "3gs":
            main()
            break

def iphone4():
    validation = 0

    while validation != "4" or validation != "4s":
        validation = input("iPhone 4 or iPhone 4S? ")
        validation = validation.lower()

        if validation == "4" or validation == "4s":
            main()
            break

def iphone5():
    validation = 0

    while validation != "5" or validation != "5c" or validation != "5s":
        validation = input("iPhone 5, iPhone 5C or iPhone 5S? ")
        validation = validation.lower()

        if validation == "5" or validation == "5c" or validation == "5s":
            main()
            break

def iphone6():
    validation = 0

    while validation != "6" or validation != "6+" or validation != "6s" or validation != "6s+":
        validation = input("iPhone 6, iPhone 6+, iPhone 6S or iPhone 6S+ ? ")
        validation = validation.lower()

        if validation == "6" or validation == "6+" or validation == "6s" or validation == "6s+":
            main()
            break
```

```
def phone_make():
    print("Okay, what phone make your device?")
    make = input("Phone make: ")
    make = make.lower()

    if make == "iphone":
        model = 0
        while model != "3" or model != "4" or model != "5" or model != "6":
            print("What model is your device?(Enter the number ONLY, no letters)")
            model = input("Phone model: ")

            if model == "3":
                iphone3()
                break

            elif model == "4":
                iphone4()
                break

            elif model == "5":
                iphone5()
                break

            elif model == "6":
                iphone6()
                break
```

As you can see, instead of executing the while loop that asks the user to specify the model of their phone, python instead calls on a function which has the exact same purpose. The only difference is that the break function can be used to stop the first while loop about the variable model, so it isn't unnecessarily repeated. The output shows that this solution worked, since there was no more repeated loop.

```
Welcome to the multicellular Troubeshooting program version 3.0! In addition t
e model to tailor results to it, as well as giving you a refrerence number fo
In order to make this work, we're going to need a few details from you.
First off, what is your name?
Name: Shaun
Okay, now what is your email address?
Email: erhergh
So your name is Shaun and your email is erhergh, correct?
Correct? (y/n)y
Thank you for sharing your details. They will be kept on our secure server.
Okay, what phone make your device?
Phone make: iphone
What model is your device?(Enter the number ONLY, no letters)
Phone model: 5
iPhone 5, iPhone 5C or iPhone 5S? 5s
wurf
```

I noticed that if you enter spaces before your inputs, python won't recognize it. For example, if I entered Samsung with a space in front, it won't recognize the word Samsung Here was the code at the time.

```
def phone_make():
    print("Okay, what phone make your device?")
    make = input("Phone make: ")
    make = make.lower()

    if make == "iphone":
        model = 0
        while model != "3" or model != "4" or model != "5" or model != "6":
            print("What model is your device?(Enter the number ONLY, no letters)")
            model = input("Phone model: ")

            if model == "3":
                iphone3()
                break

            elif model == "4":
                iphone4()
                break

            elif model == "5":
                iphone5()
                break

            elif model == "6":
                iphone6()
                break

    elif make == "samsung":
        samsung()
```

```python
def samsung():
    confirmation = 0
    while confirmation != "y" or confirmation != "Y" or confirmation != "n" or confirmation != "N":
        model = input("Which model of Samsung is it? (e.g Samsung Galaxy S7)")
        confirmation = input("Samsung " + model + ",correct?")

        if confirmation == "y" or confirmation == "Y":
            main()
            break

        elif confirmation == "n" or confirmation == "N":
            samsung()
```

And then the output.

```
------------------------ RESTART: G:\Task 3\Task 3.py ------------------------
Welcome to the multicellular Troubeshooting program version 3.0! In addition to
e model to tailor results to it, as well as giving you a refrerence number for a
d that we only support iPhones, Samsungs and Sonys at the moment.
In order to make this work, we're going to need a few details from you.
First off, what is your name?
Name: Shaun
Okay, now what is your email address?
Email: s
So your name is Shaun and your email is s, correct?
Correct? (y/n)y
Thank you for sharing your details. They will be kept on our secure server.
Okay, what phone make your device?
Phone make:     SAMSUNG
>>> |
```

To fix this, I recycled the structure of my code from task 2, using a For loop and a variable that acted like a counter to ensure the loop ran for a sufficient amount of time before ending. This was the code.

```python
def phone_make():
    validation = 0
    print("Okay, what phone make is your device?")
    make = input("Phone make: ")
    make_lower = make.lower()
    phone_make = make_lower.split()
    numberOfWords = len(phone_make)
    while validation != 999:
        for i in range(numberOfWords):

            if phone_make[i] == "iphone":
                model = 0
                while model != "3" or model != "4" or model != "5" or model != "6":
                    print("What model is your device?(Enter the number ONLY, no letters)")
                    model = input("Phone model: ")

                    if model == "3":
                        iphone3()
                        break

                    elif model == "4":
                        iphone4()
                        break

                    elif model == "5":
                        iphone5()
                        break

                    elif model == "6":
                        iphone6()
                        break

            elif phone_make[i] == "samsung":
                samsung()

        validation +=1

        if validation == 999:
            print("Remember, we only support iPhone, Sony and Samsung.")
            phone_make()
```

And this was the output. The input was exactly the same as the previous time where it failed to identify “SAMSUNG”, but the alteration of the code resolved this issue.

```
Okay, what phone make your device?
Phone make:     SAMSUNG
Which model of Samsung is it? (e.g Samsung Galaxy S7)S7
Samsung S7,correct?n
Which model of Samsung is it? (e.g Samsung Galaxy S7)Galaxy S7
Samsung Galaxy S7,correct?y
wurf
```

When the user was asked to input their phone make, if they entered something invalid, the program returned this error.

```
Okay, what phone make is your device? (No numbers!)
Phone make: sf
Remember, we only support iPhone, Sony and Samsung.
Traceback (most recent call last):
  File "E:\Task 3\Task 3.py", line 133, in <module>
    user_details()
  File "E:\Task 3\Task 3.py", line 126, in user_details
    phone_make()
  File "E:\Task 3\Task 3.py", line 107, in phone_make
    phone_make()
TypeError: 'list' object is not callable
```

This was the code at the time.

```
def phone_make():
    validation = 0
    print("Okay, what phone make is your device? (No numbers!)")
    make = input("Phone make: ")
    make_lower = make.lower()
    phone_make = make_lower.split()
    numberOfWords = len(phone_make)
    while validation != 999:
        for i in range(numberOfWords):

            if phone_make[i] == "iphone":
                model = 0
                while model != "3" or model != "4" or model != "5" or model != "6":
                    print("What model is your device?(Enter the number ONLY, no letters)")
                    model = input("Phone model: ")

                    if model == "3":
                        iphone3()
                        break

                    elif model == "4":
                        iphone4()
                        break

                    elif model == "5":
                        iphone5()
                        break

                    elif model == "6":
                        iphone6()
                        break

            elif phone_make[i] == "samsung":
                samsung()

        validation += 1

        if validation == 999:
            print("Remember, we only support iPhone, Sony and Samsung.")
            phone_make()
```

I had had a similar problem in task 2, so I looked back to this piece of code from it.

```
def main_menu():
    Validation = 0 #This is a counter for the loop
    problem = input("Go ahead. What seems to be your problem? ")
    problem_lower = problem.lower()
    user_problem = problem_lower.split()
    numberOfWords = len(user_problem)
    while Validation != 999:
        for i in range(numberOfWords):
            if user_problem[i] == "display" or user_problem[i] == "blank" or user_problem[i] == "turn" and "on" or user_problem[i] == "screen":
                display_problems()

            elif user_problem[i] == "charge" or user_problem[i] == "battery" or user_problem[i] == "dead" or user_problem[i] == "charging" or user_problem[i] == "die":
                battery_problems()

            elif user_problem[i] == "touch" and "screen" or user_problem[i] == "touch":
                touchScreen_problems()

            elif user_problem[i] == "volume" or user_problem[i] == "sound" or user_problem[i] == "speaker":
                volume_problems()

            elif user_problem[i] == "call" or user_problem[i] == "ring" or user_problem[i] == "calls":
                calling_problems()

            elif user_problem[i] == "text" or user_problem[i] == "message" or user_problem[i] == "SMS":
                texting_problems()

            elif user_problem[i] == "internet" or user_problem[i] == "connect" or user_problem[i] == "wi" and "fi"  or user_problem[i] == "web":
                internet_problems()

            elif user_problem[i] == "dim" or user_problem[i] == "bright" or user_problem[i] == "brightness":
                dim_problems()

            elif user_problem[i] == "mic" or user_problem[i] == "microphone":
                mic_problems()

        Validation +=1

        if Validation == 999:
            nothingDefined()
```

```
def nothingDefined():
    print("Try and enter something that's maybe a little shorter, such as
    main_menu()
```

I implemented the use of the nothing_defined() function in my task 3 code, calling that function instead of re-calling the phone_make() function. This was the code I fixed.

```
def phone_make():
    validation = 0
    print("Okay, what phone make is your device? (No numbers, just the brand.)")
    make = input("Phone make: ")
    make_lower = make.lower()
    phone_make = make_lower.split()
    numberOfWords = len(phone_make)
    while validation != 999:
        for i in range(numberOfWords):

            if phone_make[i] == "iphone":
                model = 0
                while model != "3" or model != "4" or model != "5" or model != "6":
                    print("What model is your device?(Enter the number ONLY, no letters)")
                    model = input("Phone model: ")

                    if model == "3":
                        iphone3()
                        break

                    elif model == "4":
                        iphone4()
                        break

                    elif model == "5":
                        iphone5()
                        break

                    elif model == "6":
                        iphone6()
                        break

            elif phone_make[i] == "samsung":
                samsung()

        validation += 1

        if validation == 999:
            nothing_defined()
```

```
def nothing_defined():
    print("Remember, we only support iPhone, Sony and Samsung.")
    phone_make()
```

And the output.

```
Okay, what phone make is your device? (No numbers!)
Phone make: dhet
Remember, we only support iPhone, Sony and Samsung.
Okay, what phone make is your device? (No numbers!)
Phone make: yjtujkyi,l
Remember, we only support iPhone, Sony and Samsung.
Okay, what phone make is your device? (No numbers!)
Phone make: gjktyj
Remember, we only support iPhone, Sony and Samsung.
Okay, what phone make is your device? (No numbers!)
```

A big problem I encountered was this.

```
What model is your iPhone?(Enter the number ONLY, no letters)
Phone model: 5
iPhone 5, iPhone 5C or iPhone 5S? 5C
what is the problem
What model is your iPhone?(Enter the number ONLY, no letters)
Phone model:
```

After identifying the user's model, it was supposed to ask them the problem. And it did this, however it wouldn't stop asking for the model even though it had already been given. This was the piece of code that kept repeating.

```
def iphone():
    model = 0
    while model != "3" or model != "4" or model != "5" or model != "6":
        print("What model is your iPhone?(Enter the number ONLY, no letters)")

        model = input("Phone model: ")

        if model == "3":
            iphone3()
            break

        elif model == "4":
            iphone4()
            break

        elif model == "5":
            iphone5()
            break

        elif model == "6":
            iphone6()
            break
```

I made a number of changes to my code to attempt to solve this problem. The first one was adding the break functions as shown in the picture, which didn't work. I tried moving this section of code below the functions in the if statements (iphone3, iphone4 etc), but that didn't work either. Instead, I got rid of the whole while loop, replacing them with simple if and else statements.

Replaced code:

```
def iphone():
    model = 0

    print("What model is your iPhone?(Enter the number ONLY, no letters)")########

    model = input("Phone model: ")

    if model == "3":
        iphone3()


    elif model == "4":
        iphone4()


    elif model == "5":
        iphone5()


    elif model == "6":
        iphone6()

    else:
        iphone()
```

```
def phone_make():
    validation = 0
    print("Okay, what phone make is your device? (No numbers, just the brand.)")
    make = input("Phone make: ")
    make_lower = make.lower()
    phone_make = make_lower.split()
    numberOfWords = len(phone_make)
    while validation != 999:
        for i in range(numberOfWords):

            if phone_make[i] == "iphone":
                iphone()
                break

            elif phone_make[i] == "samsung":
                samsung()
                break

        validation += 1

        if validation == 999:
            nothing_defined()
            break
```

However, the problem still wasn't rectified. So I looked at the part of the code that was supposed call the iphone() function, which was this.

```
def phone_make():
    validation = 0
    print("Okay, what phone make is your device? (No numbers, just the brand.)")
    make = input("Phone make: ")
    make_lower = make.lower()

    if make_lower == "iphone":
        iphone()

    elif make_lower == "samsung":
        samsung()

    else:
        nothing_defined()
```

I decided to get rid of the while loop, just to see if that could be causing the problem. This was the edited code.

```
def phone_make():
    validation = 0
    print("Okay, what phone make is your device? (No numbers, just the brand.)")
    make = input("Phone make: ")
    make_lower = make.lower()

    if "iphone" in make_lower:
        iphone()

    elif "samsung" in make_lower:
        samsung()

    else:
        nothing_defined()
```

To further improve it before I even tested it, I made one small change to the if functions, to make up for the while loop not being there to identify the keywords.

By using the IN function, the program would look for the specific keyword "iphone" and "samsung" in the user's input, meaning it wouldn't matter if they entered spaces or other words along with the keywords. Case didn't matter due to the .lower() function making everything lowercase before it was identified.

The next thing to do was add the third and final device model; Sony. This was pretty straightforward, as it followed the same structure as the Samsung def function.

Sony function:

```
def sony():
    confirmation = 0
    while confirmation != "y" or confirmation != "Y" or confirmation != "n" or confirmation != "N":
        model = input("Which model of Sony is it? (e.g Xperia Z3, Xperia M4) " )
        confirmation = input("Sony " + model + ",correct? (y/n) ")

        if confirmation == "y" or confirmation == "Y":
            main()
            break

        elif confirmation == "n" or confirmation == "N":
            sony()
            break
```

Though I had made 3 algorithms for the 3 different device type, python had no way of identifying which one the user had picked. This had been explicitly mentioned as a requirement for task 3. There were a number of things I tried to get python to identify the device the user had picked. First off, I tried to create a variable called user_device and assign “iphone”, “sony”, or “Samsung” to it, depending on what the user picked. This was the code with the variable added.

```
def phone_make():
    validation = 0
    print("Okay, what phone make is your device? (No numbers, just the brand.)")
    make = input("Phone make: ")
    make_lower = make.lower()

    if "iphone" in make_lower:
        user_device == "iphone"
        iphone()

    elif "samsung" in make_lower:
        user_device == "samsung"
        samsung()

    elif "sony" in make_lower:
        user_device == "sony"
        sony()

    else:
        nothing_defined()
```

Then, the part of the code that relied on the user_device variable to identify the user device.

```
def main():
    if user_device == "iphone":
        print("got an iphone")

    elif user_device == "sony":
        print("sony user")
```

However, the output kept on telling me that the user_device variable wasn't defined.

Even though I tried declaring the variable at the start of the program and assigning the value 0 to it just as a placeholder, the program progressed further, but the main() function refused to run. As demonstrated in the picture below, after the last line before the program ended, it was supposed to identify the user's device as either iPhone or Sony (I only used 2 to test if it would work first).

```
Name: Shaun
Okay, now what is your email address?
Email: email
So your name is Shaun and your email is email, correct?
Correct? (y/n)y
Thank you for sharing your details. They will be kept on our secure server.
Okay, what phone make is your device? (No numbers, just the brand.)
Phone make: sony
Which model of Sony is it? (e.g Xperia Z3, Xperia M4) xperia z4
Sony xperia z4,correct? (y/n) y
>>>
```

When I edited the code in the main() function to see what the user_device variable had been assigned, the output was "0", meaning that "iphone" or "sony" was never assigned to the variable, which is why the value stayed as 0, so the main() function couldn't run, since none of the if statements were true and there was no else statement, forcing python to just terminate, with nothing further to read. This was the code I edited. All I did was add a print function, to see what was assigned to the user_device variable.

```
def main():
    print(user_device)

    if user_device == "iphone":
        print("got an iphone")

    elif user_device == "sony":
        print("sony user")
```

```
First off, what is your name?
Name: Shaun
Okay, now what is your email address?
Email: someone@example.com
So your name is Shaun and your email is someone@example.com, correct?
Correct? (y/n)y
Thank you for sharing your details. They will be kept on our secure server.
Okay, what phone make is your device? (No numbers, just the brand.)
Phone make: iphyo
Remember, we only support iPhone, Sony and Samsung.
Okay, what phone make is your device? (No numbers, just the brand.)
Phone make: iphone
Traceback (most recent call last):
  File "H:\Task 3\Task 3 - Copy.py", line 155, in <module>
    user_details()
  File "H:\Task 3\Task 3 - Copy.py", line 148, in user_details
    phone_make()
  File "H:\Task 3\Task 3 - Copy.py", line 131, in phone_make
    nothing_defined()
  File "H:\Task 3\Task 3 - Copy.py", line 9, in nothing_defined
    phone_make()
  File "H:\Task 3\Task 3 - Copy.py", line 119, in phone_make
    user_device == "iphone"
NameError: name 'user_device' is not defined
```

And the output to go with this code:

```
def phone_make():
    user_iphone = False
    user_sony = False
    user_samsung = False
    validation = 0
    print("Okay, what phone make is your device? (No numbers, just the brand.)")
    make = input("Phone make: ")
    make_lower = make.lower()

    if "iphone" in make_lower:
        user_iphone = True
        iphone()

    elif "samsung" in make_lower:
        user_samsung = True
        samsung()

    elif "sony" in make_lower:
        user_sony = True
        sony()
```

```
def main():
    if user_iphone = True:
        print("got an iphone")

    elif user_sony = True:
        print("got a sony")
```

Instead, I tried using the true and false statements. Here is how I integrated them into my code.

```
Name: Shaun
Okay, now what is your email address?
Email: something
So your name is Shaun and your email is something, correct?
Correct? (y/n)y
Thank you for sharing your details. They will be kept on our secure server.
Okay, what phone make is your device? (No numbers, just the brand.)
Phone make: iphone
What model is your iPhone?(Enter the number ONLY, no letters)
Phone model: 5
iPhone 5, iPhone 5C or iPhone 5S? 5S
0
>>>
```

This was the output when python ran the print(user_device) function.

The way in which it worked was that when python identified the user's device based on their input, it would change a variable (based on the device) to true. For example, if the user had an iphone, then the user_iphone variable would be changed to true and then in the main() function, it would print an appropriate response based on what was true, since all 3 variables start out as being false. At this point, it was just to test if python could successfully identify the user's device, keeping it assigned to a variable for later use. This was the output.

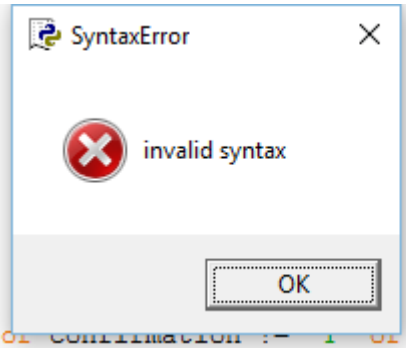
```
First off, what is your name?
Name: Shaun
Okay, now what is your email address?
Email: s
So your name is Shaun and your email is s, correct?
Correct? (y/n)y
Thank you for sharing your details. They will be kept on our secure server.
Okay, what phone make is your device? (No numbers, just the brand.)
Phone make: sony
Which model of Sony is it? (e.g Xperia Z3, Xperia M4) xperia j3
Sony xperia j3,correct? (y/n) y
Traceback (most recent call last):
  File "H:\Task 3\Task 3 - Copy.py", line 159, in <module>
    user_details()
  File "H:\Task 3\Task 3 - Copy.py", line 152, in user_details
    phone_make()
  File "H:\Task 3\Task 3 - Copy.py", line 132, in phone_make
    sony()
  File "H:\Task 3\Task 3 - Copy.py", line 26, in sony
    main()
  File "H:\Task 3\Task 3 - Copy.py", line 13, in main
    if user_iphone == True:
NameError: name 'user_iphone' is not defined
>>>
```

Straight away, I got the syntax error, for only adding one = instead of 2 where it was highlighted.

```
def main():
    if user_iphone = True:
        print("got an iphone")

    elif user_sony = True:
        print("got a sony")

def sony():
    confirmation = 0
    while confirmation != "y" or confirmation := 1 or confirmati
```



This is the amended code.

```
def main():
    if user_iphone == True:
        print("got an iphone")

    elif user_sony == True:
        print("got a sony")
```

Like with the user_device variable, the variable that was supposed to become true (in this case, user_iphone) wasn't defined, so it returned this error. This was going the same way as with the user_device variable, so I deleted it from the code, and it went back to being like this:

```
First off, what is your name?
Name: y
Okay, now what is your email address?
Email: y
So your name is y and your email is y, correct?
Correct? (y/n)y
Thank you for sharing your details. They will be kept on our secure server.
Okay, what phone make is your device? (No numbers, just the brand.)
Phone make: s
Remember, we only support iPhone, Sony and Samsung.
Okay, what phone make is your device? (No numbers, just the brand.)
Phone make: sony
Which model of Sony is it? (e.g Xperia Z3, Xperia M4) x
Sony x,correct? (y/n) y
Traceback (most recent call last):
  File "H:\Task 3\Task 3 - Copy.py", line 152, in <module>
    user_details()
  File "H:\Task 3\Task 3 - Copy.py", line 145, in user_details
    phone_make()
  File "H:\Task 3\Task 3 - Copy.py", line 128, in phone_make
    nothing_defined()
  File "H:\Task 3\Task 3 - Copy.py", line 9, in nothing_defined
    phone_make()
  File "H:\Task 3\Task 3 - Copy.py", line 125, in phone_make
    sony()
  File "H:\Task 3\Task 3 - Copy.py", line 25, in sony
    main()
  File "H:\Task 3\Task 3 - Copy.py", line 12, in main
    if "iphone" in make_lower:
NameError: name 'make_lower' is not defined
>>>
```

```
def phone_make():
    validation = 0
    print("Okay, what phone make is your device? (No numbers, just the brand.)")
    make = input("Phone make: ")
    make_lower = make.lower()

    if "iphone" in make_lower:
        iphone()

    elif "samsung" in make_lower:
        samsung()

    elif "sony" in make_lower:
        sony()

    else:
        nothing_defined()
```

By looking at *how* exactly the program identified the user's input, I came up with another possible solution. I kept this part of the code exactly the same. What I did change however, was this.

```
def main():
    if "iphone" in make_lower:
        print("You have an iphone")

    elif "sony" in make_lower:
        print("You have a sony")
```

The variable make_lower was the lowercase version of the variable make, which would be the user's input in response to what type of device they had. Python just looked for the keywords "iphone", "samsung" and "sony" in the input. Since the value of the make_lower function wouldn't change unless the user entered something else, the make_lower function could be used again in later parts of the code. So in the screenshot above, all I did was tell python to look for the keywords in the variable make_lower again, to determine the output. So if it found "iphone" in the make_lower variable, it would give a reponse related to an iphone. However, I got this error when I ran the code.

Then I remembered that in order to use the same variable in a different part of the code, you had to make it a global variable. However, I didn't know how to do that, so I did some research and found this website.

Using global variables in a function other than the one that created them

1630

If I create a global variable in one function, how can I use that variable in another function? Do I need to store the global variable in a local variable of the function which needs its access?

python global-variables scope

share improve this question

387

edited Sep 22 '14 at 12:58
igaurav
1,303 • 1 • 10 • 22

asked Jan 8 '09 at 5:45
user46646
25.9k • 33 • 59 • 74

add a comment

16 Answers

active oldest votes

2328

You can use a global variable in other functions by declaring it as `global` in each function that assigns to it:

```
globvar = 0

def set_globvar_to_one():
    global globvar    # Needed to modify global copy of globvar
    globvar = 1

def print_globvar():
    print globvar     # No need for global declaration to read value of globvar

set_globvar_to_one()
print_globvar()       # Prints 1
```

This was an example of how to make a variable globalized. So I implemented this into my own code, with the make_lower function.

```
def phone_make():
    validation = 0
    print("Okay, what phone make is your device? (No numbers, just the brand.)")
    make = input("Phone make: ")
    make_lower = make.lower()
    global make_lower

    if "iphone" in make_lower:
        iphone()

    elif "samsung" in make_lower:
        samsung()

    elif "sony" in make_lower:
        sony()

    else:
        nothing_defined()
```

The newly implemented 'global' keyword

This was the output.

```
Type "copyright", "credits" or "license()" for more information.
>>>
Warning (from warnings module):
  File "H:\Task 3\Task 3 - Copy.py", line 117
    global make_lower
SyntaxWarning: name 'make_lower' is assigned to before global declaration
>>>
==================== RESTART: H:\Task 3\Task 3 - Copy.py ====================
Welcome to the multicellular Troubeshooting program version 3.0! In additio
e model to tailor results to it, as well as giving you a refrerence number
d that we only support iPhones, Samsungs and Sonys at the moment.
In order to make this work, we're going to need a few details from you.
First off, what is your name?
Name: Shaun
Okay, now what is your email address?
Email: email@address.com
So your name is Shaun and your email is email@address.com, correct?
Correct? (y/n)y
Thank you for sharing your details. They will be kept on our secure server.
Okay, what phone make is your device? (No numbers, just the brand.)
Phone make: sony
Which model of Sony is it? (e.g Xperia Z3, Xperia M4) xperia m4
Sony xperia m4,correct? (y/n) y
You have a sony
>>>
```

Although I got a warning from python about the variable, the code was successful in identifying that I had a Sony, which meant that I could use this to get python to give answers specific to the user's device, which could be re-identified at any time due to the global variable.

When making the part of code to store the user's details and allocate them a number, I referred back to this piece of code, made in a previous lesson.

```
import random
from datetime import datetime

def dis():
    f= open('dis.txt','r')
    a = f.read()
    print(a)

def mem():
    f= open('mem.txt','r')
    b = f.read()
    print(b)

def main():

    while True:
        problem = input("\nWhat is the problem? You can write something like 'Display' or 'Memory' ")
        if problem=="display" or problem =="Display":
            dis()
        elif problem=="Memory" or problem=="memory":
            mem()
        else:
            print("We cannot deal with your case")
            f = open('text1.txt','a')
            time = datetime.now()
            time = str(time)
            f.write('\n' + time)
            name = input("What is your name? ")
            f.write('\n' + name)
            email = input("What is your e-mail address? ")
            f.write('\n' + email)
            f.write('\n' +problem)
            case = random.randint(0,999)
            case = str(case)
            f.write('\n' + case)
            f.close()
            print("Your case number is " + case + ". We hope to get back to you shortly.")
            raise SystemExit
main()
```

This code opened a txt file if the problem wasn't able to be solved, and recorded their details, their problem and assigned them a case number – which is exactly what I needed to do in my task 3. So I modified it and integrated this into my code. This is what it looked like.

```
def unresolved():
    record = open('unresolved cases.txt','a')
    record.write("\nName " + name)
    record.write('\nEmail: ' + email)
    date_time = datetime.now()
    date_time = str(date_time)
    record.write('\nDate + Time: ' + date_time)
    record.write('\nUser Problem: ' + problem)
    case = random.randint(0,999)
    case = str(case)
    record.write('\nCase No: ' + case)
    record.close()
    print("We are sorry we couldn't help you solve your :
    return_toMenu()
```

I changed a few variable names, so that they weren't meaningless. For example, "f" was changed to "record", since the program was recording their details. Time was changed to date time, since it provided the date as well as the time. The variables 'problem' and 'case' remained unchanged. Since I had already used the variable name 'problem' for the user's input when asked what their problem was, I made it a global variable.

```
def main():
    """
    print(make_lower + " " + model)
    """
    Validation = 0 #This is a counter for the loop
    problem = input("What is the problem with your device? ")
    global problem
```

I also added the ability for the program to tell the user's device model, by using the IN keyword with the make_lower variable.

```
record.write('\nEmail: ' + email) #(Following up from the previou
if "iphone" in make_lower: #IF statements are used here to decide
    record.write('\nDevice: iPhone '+ model)
elif "samsung" in make_lower:
    record.write('\nDevice: Samsung '+ model)
elif "sony" in make_lower:
    record.write('\nDevice: Sony '+ model)
```

Again, I also implemented the use of the .lower function in my while loops, like how I had done in task 1 and 2.

Before:

```
record.write('\nEmail: ' + email) #(Following up from the previou
if "iphone" in make_lower: #IF statements are used here to decide
    record.write('\nDevice: iPhone '+ model)
elif "samsung" in make_lower:
    record.write('\nDevice: Samsung '+ model)
elif "sony" in make_lower:
    record.write('\nDevice: Sony '+ model)
```

After:

```
while feedback != "y" or feedback != "n": #The while loop here means that unti

    feedback = input("Did this solve your problem? (y/n) ")#This is the questi
    feedback_lower = feedback.lower()

    if feedback_lower == "y":
        print("We are glad to have helped you solve your issue," + name + "!")
        return_toMenu()

    elif feedback_lower == "n": #After the first IF statement, each subsequent
        unresolved()
```

I also worked the \n function into to my code to make the overall appearance look neater. This is what it looked like before I tidied it up:

```
Welcome to the multicellular Troubeshooting program version 3.0! In addition to version 2.0's feature of identifying keywords, we can now identify your phone model to tailor re
sults to it, as well as giving you a refrerence number for a technician if we are still unable to solve your problem! Please bear in mind that we only support iPhones, Samsungs
and Sonys at the moment.
In order to make this work, we're going to need a few details from you.
First off, what is your name?
Name: Shaun
Okay, now what is your email address?
Email: someone@example.com
So your name is Shaun and your email is someone@example.com, correct?
Correct? (y/n) y
Thank you for sharing your details. They will be kept on our secure server.
Okay, what phone make is your device? (No numbers, just the brand.)
Phone make: Sony
Which model of Sony is it? (e.g Xperia Z3, Xperia M4) Xperia M4 Aqua
Sony Xperia M4 Aqua,correct? (y/n) y
What is the problem with your device? The volume is a bit weird

Make sure you turn up your volume. This can be done by pressing the volume keys located on the side of your device (usually the right). The one on top increases volume, whilst
the one on the bottom decreases volume. You can also access a volume slider by going to settings and looking for 'sound and notification'. Also, make sure the device isn't on s
ilent or vibration. Make sure it's upwards and try again. If the problem persists, this could mean a problem with your speakers. Hard falls and water damage can affect the spea
kers, but it's best to contact Sony and have a specialist look at it.
Did this solve your problem? (y/n) y
We are glad to have helped you solve your issue,Shaun!
Return to main menu?(y/n) n
Have a nice day!
>>>
```

And afterwards – a significant improvement in appearance.:

```
================= RESTART: Z:\Shaun Sevume Task 3\Task 3.py =================
Welcome to the multicellular Troubeshooting program version 3.0! In addition to version 2.0's feature of identifying keywords, we can now identify your phone model to tailor
results to it, as well as giving you a refrerence number for a technician if we are still unable to solve your problem! Please bear in mind that we only support iPhones, Samsun
gs and Sonys at the moment.

In order to make this work, we're going to need a few details from you.

First off, what is your name?
Name: Shaun

Okay, now what is your email address?
Email: SShaun304:gmail.com

So your name is Shaun and your email is SShaun304:gmail.com, correct? (y/n) y

Thank you for sharing your details. They will be kept on our secure server.

Okay, what phone make is your device? (No numbers, just the brand.)
Phone make: Sibt

Remember, we only support iPhone, Sony and Samsung.

Would you like to input another device?(y/n) y

Okay, what phone make is your device? (No numbers, just the brand.)
Phone make: Sony

Which model of Sony is it? (e.g Xperia Z3, Xperia M4) xperia
Sony xperia,correct? (y/n) y

What is the problem with your device? volume broke af


Make sure you turn up your volume. This can be done by pressing the volume keys located on the side of your device (usually the right). The one on top increases volume, whilst
the one on the bottom decreases volume. You can also access a volume slider by going to settings and looking for 'sound and notification'. Also, make sure the device isn't on s
ilent or vibration. Make sure it's upwards and try again. If the problem persists, this could mean a problem with your speakers. Hard falls and water damage can affect the spea
kers, but it's best to contact Sony and have a specialist look at it.

Did this solve your problem? (y/n) y

We are glad to have helped you solve your issue,Shaun!

Return to main menu?(y/n) n

Have a nice day!
>>>
```

Whilst cleaning up my program and code appearance, I noticed how I had already attempted to do some sort of cleaning beforehand.

```
if "iphone" in make_lower:
    iphone_displaySolution = open("iphone.txt", "r")
    time.sleep(1)
    print(" ")
    print(iphone_displaySolution.readlines()[0]) #Si
    feedback()
```

I had used print(“ ”) to insert a blank space. Well instead, I replaced this with the \n function, which meant more efficiency, since it was now all on one line and I only needed one print command instead of two. This is what it looked like after.

```
if "iphone" in make_lower:
    iphone_displaySolution = open("iphone.txt", "r") #H
    time.sleep(1)
    print("\n" + iphone_displaySolution.readlines()[0])
    feedback()
```

Techniques Used

Technique/Keyword	What does it do?	How is it used in my code?
def	Used to define a function, or a piece of code. You can use this to split your code into blocks so that instead of repeating code over and over again, you simply call the appropriate function which will have the code written in. It means you only have to write the code once, and makes it easier to identify which part of your code has a problem (should any arise).	It is used to split the code into blocks, making it easier to identify problems and not needing to repeat code. For example, the the_end() function is used to exit the program. It only has to be called instead of being repeated.
Import	Enables the use of specific techniques, depending on what you choose to import	I used import time so I can use the time.sleep technique. I also used import re for regular expressions, import random, so I could generate a random integer and import datetime so that I could record the system’s date and time in the txt file.
Time	Imported with the import function, which is the system time.	I use it with the time.sleep function, where you can specify how long you want the program to be idle for before executing the next line.

Print	Outputs whatever is specified in the speech marks and brackets.	I use the print function on various occasions, mostly to display instructions to the user.
Input	Assigns the user's input to a variable.	An example of when I have used this is when the user is asked their name. What they enter is assigned to the variable called name, made possible because of the input keyword.
While	Used to create a WHILE loop, where a piece of code is constantly repeated until the condition for the loop is either made true or false (depending on how you structured the loop.	I used a WHILE loop when the program was asking the user if their problem was solved or not. The condition was the input not being y or n, and whilst this is true, the while loop is run. Once the condition becomes false, the loop is broken, and the program can proceed.
Break	Used to break a while loop, or to stop it.	I used the break function with various while loops in my code, such as when the program required confirmation of the user's device make. Once a valid input (y/Y or n/N) was detected, the loop was broken, not being run anymore.
For	The exact same as a while loop, except you specify how many times the loop is run.	When identifying keywords, I used a for loop, and using the range technique/keyword, I specified for the for loop to run the same amount of times as the amount of words the user inputted.
Range	Used to specify a range of numbers. For example, if you entered 0, 5 in the brackets that follow the keyword, then the range would be from 0-4 but not including 5.	I used it with my for loop to specify how many times it should run. Since there was a variable assigned to the number of words the user had inputted called numberOfWords, I specified that as a range, since it was technically still an integer, because of its numerical value.

Len	Used to get the lengh/number of items in an object. For example “hello world” would have a value of 11, due to there being 11 characters (spaces included).	I used it to get the length of the variable user problem, which was the lowercased and split version of the user’s input, when asked what their problem was. The length would then be assigned to the variable numberOfWords
Str	Used to convert an integer to a string. An integer is a whole number above 0 and a string is anything inside parentheses/brackets.	I used it when the code generated a random integer to assign to the user. Since it was in integer form, it had to be converted to string form, to be able to be recorded in the txt file.
Or	This is usually (but not limited to) with if statement. It gives python an alternative condition to satisfy that applies to the same variable.	An example of when I used it is with the variable called feedback. The condition was whether the user’s input was y OR n. The or function makes sure that both conditions are checked before the program carries on running.
If /elif	This is a statement that has a condition, which needs to be satisfied first. The condition could be a variable being equal to something, or the user’s input matching a specified keyword. Once the condition is satisfied, the rest of the if statement is run. If the condition isn’t satisfied, then it will usually run the next if statement (if there are any), run the else statement (if present) or just terminate if nothing is specified for what the program has to do should the statement not be satisfied.	An example of me using an if statement is when I put ‘IF’ return1 = y. This means that if the variable called return has a value of y (based on user input), then whatever was specified to do if the condition was true is done. In this case, it returns to the main menu.

Raise	The raise keyword forces an exception to occur. There are many different exceptions, but the one I forced to occur was SystemExit.	I used it to raise the SystemExit exception, which terminated the program, when the user was done.
SystemExit	It is an exception, that is used to call the sys.exit() function – which exits/terminates the program.	Using the raise function, I raise this exception which calls the sys.exit() function, which terminates the program.
Open	This command tells python to open another file. What file to open is specified in the brackets that follow.	After the user input their problem and the problem is identified, the program opens the appropriate txt file depending on what phone the user had. It can be either “iphone.txt”, “samsung.txt” or “sony.txt, where all the solutions are written.
“r”/“a”/”w”	A command used with the ‘open’ technique to tell the program what to do after the txt file is opened. In this case, r stands for read, so the program knows that it has to read the text file. I also used “a” which means amend, or to edit the txt file. I used “w”, but in the form of .write, which of course means write	I used “a” (amend) when the program had to record the user’s details, because it edited the text file and added on the user’s details. I used “r” (read) when displaying the solutions to the user. I used .write when recording the user’s details, assigning it to the variable record.
.ranint	Used to generate a random number between a specified range.	I used it to generate a random number, and I specified between 0-999.
.readlines	Used with the open technique/keyword. After a txt file is open, you can specify what lines the program needs to read (and output), in case you only	Because all my solutions are in three txt files, once one of them is opened, I need to use this function to specify what lines of the txt file to read, so that the appropriate solution is outputted.

	want a specific part to be outputted.	
.lower	Converts user's input into lowercase by adding .lower to a variable that is assigned to the user's input.	I used it to make the user's input lowercase when they were asked what their problem was, so that it was the same case as they keywords I had added to identify the problem, making it easier for them to be identified as keywords.
.split	Splits the user's input by spaces so that it can be checked word by word until a keyword is matched, or until the counter reaches 999.	I used it after I used the .lower function, to split the user's input so that it could be checked word by word either until a match was found with one of the keywords, or unless the counter that controlled how many times the for loop ran reached its limit, 999.
Variables	Variables are objects with no value. They can be called literally anything (apart from reserved keywords such as import) and they can be assigned to things such as numbers, letters, different character and even user inputs. You can manipulate variables to make python perform certain things. For example, if you define a variable as 1, you could make an if statement to perhaps print something if that variable was equal to 1.	They are used all over my code. For example, the user's name is assigned to the variable 'name', the user's problem is assigned to the variable 'problem' and so on.
+	Used for adding things. If it's adding integers, then a new answer will be printed. However, if it's adding strings, then it will print them together.	When printing instructions to the user, the program first outputs "okay" followed by the user's name. in the code, I used print("So your name is" + name), which would print "So your name is" plus whatever had been assigned to the variable name, done by the user's input.

=/==/!=	Used to make something the same (equal) as something. If you are directly assigning a variable to a string value, then you use 2 equals signs instead of one. An exclamation mark and an equals sign is the same as saying “not equal to”.	When declaring variables, such as name, I used the input function, so that the user’s input would be equal (=) to the variable name. So if the input was ‘Shaun’ then name would = Shaun. I used != in my while loops. The condition was that while the user’s input != (was not equal to) “y/Y” or “n/N”, the question would be repeatedly asked.
#	Used to make annotations in the code.	I used it to annotate my code. Annotated text appears in red.
Camel case	A style of naming variables, whereby the first word is lowercase, and the second word begins with a capital. For example: userName is in camel case.	I used it when naming some of my variables, such as return_toMenu.

Development Review

I was required to create a troubleshooting program for mobile phones that would be able to identify the user's device and load the correct solutions accordingly, recognizing keywords in their input to direct them to the correct solution. To meet these requirements, I used a range of different functions. For example, to identify the user's device, I used the IN function. When the user inputted their device model, python would run 3 if statements to find out if any of the valid devices were there (one if statement per device). By using the line of code, IF "(device model)" IN (variable assigned to user input, which was in this case make_lower), my program would only carry out the necessary instructions once something had been identified, which was in this case, going to another function to get the user to specify their model.

```
print("\nOkay, what phone make is your device? (No numbers, just the brand.)")#See line 11.
make = input("Phone make: ")#See line 48.
make_lower = make.lower()#See line 33.
global make_lower#See line 190.

if "iphone" in make_lower:#See line 18
    iphone()#See line 33.

elif "samsung" in make_lower:#See line 20.
    samsung()#See line 33.

elif "sony" in make_lower:#See line 20.
    sony()#See line 33.

else: #The ELSE statement is only ran if none of the IF statements were satisfied.
    nothing_defined()#See line 33.
```

If the user's problem wasn't able to be solved, then the program should've stored their details in a txt file and given them a reference number, to see a technician. To solve this problem, I used many different functions. First, I used the global function, so that when the user entered their details and it was assigned to a variable, that variable could be called on from any part of the code.

```
name = input("Name: ")#See line 48.
global name#See line 190.
print("\nOkay, now what is your email address?")#See line 11.
email = input("Email: ")#See line 48.
global email#See line 190.
```

I also used the .write function, which enabled the program to open and write the details on a specific txt file.

```
record.write("\nName " + name)#.write is added to the declared variable (record), which tells python to write something in the txt file, specified in the brackets.
record.write('\nEmail: ' + email)#(Following up from the previous point) For example, here, the code is told to write "Email: " plus whatever the user entered as their email
if "iphone" in make_lower:#IF statements are used here to decide whether to write iphone, samsung or sony in the file. The IN function is used to identify keywords in the va
    record.write('\nDevice: iPhone '+ model)#See line 16.
elif "samsung" in make_lower:#After the first IF statement, each subsequent one after that must be ELIF (Else if) instead, as an alternative.
    record.write('\nDevice: Samsung '+ model)#See line 16.
elif "sony" in make_lower:#See line 20.
    record.write('\nDevice: Sony '+ model)#See line 16.
date_time = datetime.now() #The function datetime.now() retrieves the current date and time. This is then assigned to a variable.
date_time = str(date_time) #Since the date and time comes in integer form, it must first be converted to a string before it can be written in the txt file
record.write('\nDate + Time: ' + date_time) #After converting date and time to a string, it can be recorded.
record.write('\nUser Problem: ' + problem) #Here, the problem the user had is recorded. This is what they inputted when they got asked what their problem was.
case = random.randint(0,999) #The .randint function generates a random number, with a range you can specify in the brackets. The generated number is assigned to the variable
case = str(case) #Like with the date and time, it comes in integer form by default, so must be changed into string form to be written.
record.write('\nCase No: ' + case)#Since the random integer is now in string format, it can be recorded in the txt file.
```

I used the .ranint function, to generate a random case number to give the user, and the datetime.now() function to get the current date and time when recording the details. This was all put into a def function for easy access.

```
    date_time = datetime.now() #The function datetime.now() retrieves the current date and time. This is then assigned to a variable.
    date_time = str(date_time) #Since the date and time comes in integer form, it must first be converted to a string before it can be written in the txt file

case = random.randint(0,999) #The .randint function generates a random number, with a range you can specify in the brackets. The generated number is assigned to the variabl
case = str(case) #Like with the date and time, it comes in integer form by default, so must be changed into string form to be written.
```

(Below is the whole thing in a def function.)

```
def unresolved(): #This is where a function is defined, to avoid repetition in the code. This way, you simply have to call this function when needed, instead of rewriting it.
    record = open('unresolved cases.txt','a')#A variable is declared here and assigned to the function that opens the unresolved cases.txt file.
    record.write("\nName " + name)#.write is added to the declared variable (record), which tells python to write something in the txt file, specified in the brackets.
    record.write('\nEmail: ' + email)#(Following up from the previous point) For example, here, the code is told to write "Email: " plus whatever the user entered as their emai
    if "iphone" in make_lower:#IF statements are used here to decide whether to write iphone, samsung or sony in the file. The IN function is used to identify keywords in the va
        record.write('\nDevice: iPhone '+ model)#See line 16.
    elif "samsung" in make_lower:#After the first IF statement, each subsequent one after that must be ELIF (Else if) instead, as an alternative.
        record.write('\nDevice: Samsung '+ model)#See line 16.
    elif "sony" in make_lower:#See line 20.
        record.write('\nDevice: Sony '+ model)#See line 16.
    date_time = datetime.now() #The function datetime.now() retrieves the current date and time. This is then assigned to a variable.
    date_time = str(date_time) #Since the date and time comes in integer form, it must first be converted to a string before it can be written in the txt file
    record.write('\nDate + Time: ' + date_time) #After converting date and time to a string, it can be recorded.
    record.write('\nUser Problem: ' + problem) #Here, the problem the user had is recorded. This is what they inputted when they got asked what their problem was.
    case = random.randint(0,999) #The .randint function generates a random number, with a range you can specify in the brackets. The generated number is assigned to the variabl
    case = str(case) #Like with the date and time, it comes in integer form by default, so must be changed into string form to be written.
    record.write('\nCase No: ' + case)#Since the random integer is now in string format, it can be recorded in the txt file.
    record.close() #After all the details have been recorded, the txt file gets closed with the .close() function.
    print("\nWe are sorry we couldn't help you solve your issue. We have saved your details to be looked at by a technician. Your case number is " + case + ".")#Outputs an apol
    return_toMenu() #Here, a fuction is called, which is in another part of the code.
```

The program was supposed to be user friendly and easily understandable, which I think I achieved, because there were only simple questions asked that didn't need too much thinking, kept relatively short too. One of the requirements for the program was to ask the user's name and greet them by outputting their name. To solve this, I used the input() keyword, which allows the user to input a response to a question or an instruction given to them. Their input is then assigned to the variable called name, so every time the program has to address them by their name, we can use the variable to output whatever they inputted as their name.

```
name = input("Name: ")#See line 48.
```

To solve the keyword identification, I used a for loop, but because of the issue of it only running once, I had to put the for loop in the while loop, to force it to run 999 times, like in task 2. I used the .lower() function to make the user's input lowercase, to match the case of the code and make keyword identification easier. It any of the keywords were detected in the user's input, made lowercase for easier recognition by python, they would be taken to the appropriate place. It also had to account for variation in user input, which it did because it only picked out keywords, so it wouldn't matter what was entered as long as one keyword was entered. If they entered 'my phone wont charge' for example, then the keyword 'charge' would be identified.

```
    while Validation != 999: #As long as the counter is below 999, this loop is run.
        for i in range(numberOfWords): #'i' can represent any number. The range function specifies what range to look in, which is in this case the amount of words in the user's
            if user_problem[i] == "display" or user_problem[i] == "blank" or user_problem[i] == "turn" and "on" or user_problem[i] == "screen":#If any of these keywords are dete
                display_problems()#Calls the display_problems() function on line 35.

            elif user_problem[i] == "charge" or user_problem[i] == "battery" or user_problem[i] == "dead" or user_problem[i] == "charging" or user_problem[i] == "die":#See line
                battery_problems()#Calls the battery_problems() function on line 41.

            elif user_problem[i] == "touch" and "screen" or user_problem[i] == "touch":#See line 117.
                touchScreen_problems()#Calls the touchScreen_problems() function on line 47.

            elif user_problem[i] == "volume" or user_problem[i] == "sound" or user_problem[i] == "speaker":#See line 117.
                volume_problems()#Calls the volume_problems() function on line 53.

            elif user_problem[i] == "call" or user_problem[i] == "ring" or user_problem[i] == "calls":#See line 117.
                calling_problems()#Calls the calling_problems() function on line 59.

            elif user_problem[i] == "text" or user_problem[i] == "message" or user_problem[i] == "SMS":#See line 117.
                texting_problems()#Calls the texting_problems() function on line 65.

            elif user_problem[i] == "internet" or user_problem[i] == "connect" or user_problem[i] == "wi" and "fi"  or user_problem[i] == "web":#See line 117.
                internet_problems()#Calls the internet_problems() function on line 71.

            elif user_problem[i] == "dim" or user_problem[i] == "bright" or user_problem[i] == "brightness":#See line 117.
                dim_problems()#Calls the dim_problems() function on line 78.

            elif user_problem[i] == "mic" or user_problem[i] == "microphone":#See line 117.
                mic_problems()#Calls the mic_problems() function on line 84.

        Validation +=1 #Once all the if statements are checked, it adds one to the counter and runs the loop again, this time searching for keywords one space further along.

        if Validation == 999: #This if statement is run once the for loop has run 999 times (which is very quick to a computer). By this stage, we can be sure no keywords have b
            nothingDefined()#Calls the nothinDefined() function on line 15
```

```
Go ahead. What seems to be your problem? Do not enter multiple problems at once! my phone wont charge

Try using another cable and power adapter that you know works on another phone, to see if the problem lies with your phone or your charger. If your phone charges after this, th
en it was your charger that was the problem. If the phone isn't charging, this could mean there's something wrong with the charging port. check to see if there's any dust or th
ings obstructing it, and make sure the charger fits in properly(not hanging loosley) and your battery works too. There is most likley a problem with your charging port. You wil
l have to take your phone to the manufacturer to get them to sort it out.
```

I used def functions, so that I could simply call them when needed. I had a def function for each solution, so if someone had battery problems, the def function for battery problems

```
def battery_problems():#See line 15.
    time.sleep(1)#See line 8.
    batterySolution = open("solutions.txt", "r")#See line 36.
    print("\n" + batterySolution.readlines()[1])#See line 38.
    feedback()#See line 39.
```

would be called, and the txt file would be open with the 'open' and 'r' keywords, and then the specific line would be read out with the .readlines() function that contained solutions for battery problems.

I also added def functions that were there for navigation around the code to improve efficiency, so that it didn't have to be restarted all the time. For example, there was a function that contained a while loop which asked the user whether they wanted to return to the menu or not. If they chose yes, then the function for the main menu would be called, defined at the part of the code where the user had to input their problem. If, however they chose no, then another function would be called, one in charge of terminating the program.

```
Return to main menu?(y/n) n
```

I called it the_end() and all it did was raise the exception SystemExit and basically terminate the program. Making this function avoided me having to input this command on every part of my code where I wanted it to terminate.

```
def the_end():#See line 15.
    time.sleep(1)#See line 8.
    print("\nHave a nice day!")
    raise SystemExit #This terminates the program properly, so it isn't left running with nothing to display and nothing to be inputted.
```

```
Return to main menu?(y/n) n

Have a nice day!
>>>
```

Code

```
#Centre number: 10854
#Candidate number: 7116
#Name: Shaun Sevume

import time #This is necessary so that I can use the time.sleep function.
import re #Allows the code to open, read and write to txt files.
import random #Necessary for the .randint function to work. It generates a random number.
from datetime import datetime #Date and time is imported so that it can be recorded with the user's details, should they fail to find the solution.
print("Welcome to the multicellular Troubeshooting program version 3.0! In addition to version 2.0's feature of identifying keywords, we can now identify your phone model to ta
time.sleep(2) #This makes the program wait before executing the next line. How long it waits depends on the number specified in the brackets, which is in seconds.
print("\nIn order to make this work, we're going to need a few details from you.")#The PRINT function outputs whatever is in the brackets. This is the text the user would see.
time.sleep(1)#See line 10.

def unresolved(): #This is where a function is defined, to avoid repetition in the code. This way, you simply have to call this function when needed, instead of rewriting it.
    record = open('unresolved cases.txt','a')#A variable is declared here and assigned to the function that opens the unresolved cases.txt file.
    record.write("\nName " + name)#.write is added to the declared variable (record), which tells python to write something in the txt file, specified in the brackets.
    record.write('\nEmail: ' + email)#(Following up from the previous point) For example, here, the code is told to write "Email: " plus whatever the user entered as their emai
    if "iphone" in make_lower:#IF statements are used here to decide whether to write iphone, samsung or sony in the file. The IN function is used to identify keywords in the v
        record.write('\nDevice: iPhone '+ model)#See line 16.
    elif "samsung" in make_lower:#After the first IF statement, each subsequent one after that must be ELIF (Else if) instead, as an alternative.
        record.write('\nDevice: Samsung '+ model)#See line 16.
    elif "sony" in make_lower:#See line 20.
        record.write('\nDevice: Sony '+ model)#See line 16.
    date_time = datetime.now() #The function datetime.now() retrieves the current date and time. This is then assigned to a variable.
    date_time = str(date_time) #Since the date and time comes in integer form, it must first be converted to a string before it can be written in the txt file
    record.write('\nDate + Time: ' + date_time) #After converting date and time to a string, it can be recorded.
    record.write('\nUser Problem: ' + problem) #Here, the problem the user had is recorded. This is what they inputted when they got asked what their problem was.
    case = random.randint(0,999) #The .randint function generates a random number, with a range you can specify in the brackets. The generated number is assigned to the variabl
    case = str(case) #Like with the date and time, it comes in integer form by default, so must be changed into string form to be written.
    record.write('\nCase No: ' + case)#Since the random integer is now in string format, it can be recorded in the txt file.
    record.close() #After all the details have been recorded, the txt file gets closed with the .close() function.
    print("\nWe are sorry we couldn't help you solve your issue. We have saved your details to be looked at by a technician. Your case number is " + case + ".")#Outputs an apol
    return_toMenu() #Here, a fuction is called, which is in another part of the code.

def nothing_defined():#See line 14.
    print("\nRemember, we currently only support iPhone, Sony and Samsung.")#See line 11.
    return_toMenu2() #See line 33.

def nothing_defined2():#See line 14.
    print("\nSorry, but we were unable to identify any keywords. Please note that we only have a limited amount of solutions right now, so we can only solve so many \nproblems.'
    return_toMenu()#See line 33.

def feedback():#See line 14.
    feedback = 0 #The variable you're using in the while loop must be declared prior to the loop, or else you get an error saying that the variable hasn't been declared. For th

    while feedback != "y" or feedback != "n": #The while loop here means that until the answer is y or n, it will keep repeating.

        feedback = input("\nDid this solve your problem? (y/n) ")#This is the question that will be repeated until the while loop is broken. In this case, it has to become fals
        feedback_lower = feedback.lower()#See line 33.

        if feedback_lower == "y":#See line 18
            print("\nWe are glad to have helped you solve your issue," + name + "!")#See line 11.
            return_toMenu()#See line 33.

        elif feedback_lower == "n": #See line 20.
            unresolved()#See line 33.

def display_problems():#See line 14.
    if "iphone" in make_lower:#See line 18
        iphone_displaySolution = open("iphone.txt", "r") #Here it opens the txt file where the solutions are, but is only asked to read it this time.
        time.sleep(1)#See line 10.
        print("\n" + iphone_displaySolution.readlines()[0]) #Since it is all in one file, this specifies what lines to read, which will contain the appropriate solution.
        feedback() #See line 33.

    elif "samsung" in make_lower:#See line 20.
        samsung_displaySolution = open("samsung.txt", "r")#See line 60.
        time.sleep(1)#See line 10.
        print("\n" + samsung_displaySolution.readlines()[0]) #The .readlines() function makes my code more efficient because I only have to use 1 text file for all the solution
        feedback()#See line 33.

    elif "sony" in make_lower:#See line 20.
        sony_displaySolution = open("sony.txt", "r")#See line 60.
        time.sleep(1)#See line 10.
        print("\n" + sony_displaySolution.readlines()[0])#See line 68
        feedback()#See line 33.

def battery_problems():#See line 14.
    if "iphone" in make_lower:#See line 18
        iphone_batterySolution = open("iphone.txt", "r")#See line 60.
        time.sleep(1)#See line 10.
        print("\n" + iphone_batterySolution.readlines()[1])#See line 68
        feedback()#See line 33.

    elif "samsung" in make_lower:#See line 20.
        samsung_batterySolution = open("samsung.txt", "r")#See line 60.
        time.sleep(1)#See line 10.
        print("\n" + samsung_batterySolution.readlines()[1])#See line 68
        feedback()#See line 33.

    elif "sony" in make_lower:#See line 20.
        sony_batterySolution = open("sony.txt", "r")#See line 60.
        time.sleep(1)#See line 10.
        print("\n" + sony_batterySolution.readlines()[1])#See line 68
        feedback()#See line 33.

def touchScreen_problems():#See line 14.
    if "iphone" in make_lower:#See line 18
        iphone_touchScreenSolution = open("iphone.txt", "r")#See line 60.
        time.sleep(1)#See line 10.
        print("\n" + iphone_touchScreenSolution.readlines()[2])#See line 68
        feedback()#See line 33.

    elif "samsung" in make_lower:#See line 20.
        samsung_touchScreenSolution = open("samsung.txt", "r")#See line 60.
        time.sleep(1)#See line 10.
        print("\n" + samsung_touchScreenSolution.readlines()[2])#See line 68
        feedback()#See line 33.

    elif "sony" in make_lower:#See line 20.
        sony_touchScreenSolution = open("sony.txt", "r")#See line 60.
        time.sleep(1)#See line 10.
        print("\n" + sony_touchScreenSolution.readlines()[2])#See line 68
        feedback()#See line 33.
```

```
def dim_problems():#See line 14.
    if "iphone" in make_lower:#See line 18
        iphone_dimSolution = open("iphone.txt", "r")#See line 60.
        time.sleep(1)#See line 10.
        print("\n" + iphone_dimSolution.readlines()[3])#See line 68
        feedback()#See line 33.

    elif "samsung" in make_lower:#See line 20.
        samsung_dimSolution = open("samsung.txt", "r")#See line 60.
        time.sleep(1)#See line 10.
        print("\n" + samsung_dimSolution.readlines()[3])#See line 68
        feedback()#See line 33.

    elif "sony" in make_lower:#See line 20.
        sony_dimSolution = open("sony.txt", "r")#See line 60.
        time.sleep(1)#See line 10.
        print("\n" + sony_dimSolution.readlines()[3])#See line 68
        feedback()

def volume_problems():#See line 14.
    if "iphone" in make_lower:#See line 18
        iphone_volumeSolution = open("iphone.txt", "r")#See line 60.
        time.sleep(1)#See line 10.
        print("\n" + iphone_volumeSolution.readlines()[4])#See line 68
        feedback()#See line 33.

    elif "samsung" in make_lower:#See line 20.
        samsung_volumeSolution = open("samsung.txt", "r")#See line 60.
        time.sleep(1)#See line 10.
        print("\n" + samsung_volumeSolution.readlines()[4])#See line 68
        feedback()#See line 33.

    elif "sony" in make_lower:#See line 20.
        sony_volumeSolution = open("sony.txt", "r")#See line 60.
        time.sleep(1)#See line 10.
        print("\n" + sony_volumeSolution.readlines()[4])#See line 68
        feedback()#See line 33.

def the_end():#See line 14.
    time.sleep(1)#See line 10.
    print("\nHave a nice day!")#See line 11.
    raise SystemExit #This terminates the program properly, so it isn't left running with nothing to display and nothing to be inputted.

def return_toMenu():#See line 14.
    time.sleep(1)#See line 10.
    return1 = 0 #Before putting a variable in a while loop, this was necessary so that it was at least defined beforehand to prevent variable errors.

    while return1 != "y" or return1 != "n":#See line 46
        return1 = input("\nReturn to main menu?(y/n) ") #The input function means that the program will allow the user to input a response, which the program will 'react' to.
        return1_lower = return1.lower()

        if return1_lower == "y":#See line 18
            main()#See line 33.

        elif return1_lower == "n":#See line 20.
            the_end()#See line 33.

def return_toMenu2():#See line 14.
    time.sleep(1)#See line 10.
    return1 = 0#See line 160.

    while return1 != "y" or return1 != "n":#See line 46

        return1 = input("\nWould you like to input another phone make?(y/n) ")#See line 48.
        return1_lower = return1.lower()#See line 33.

        if return1_lower == "y":#See line 18
            phone_make()#See line 33.

        elif return1_lower == "n":#See line 20.
            device_details()#See line 33.
def main():#See line 14.
    Validation = 0 #This is a counter for the loop
    problem = input("\nWhat is the problem with your device? ")
    global problem #Here, the variable is made global, meaning it can be used elsewhere in the code. However, variables such as problem_lower are local variables, because they l
    problem_lower = problem.lower() #The .lower function converts the user's input into lowercase to match the case of the code. It is then assigned to a new variabke
    user_problem = problem_lower.split() #The .split function splits the words in each variable by spaces, to better identify keywords.
    numberOfWords = len(user_problem) #The len function counts how many characters are in something.
    while Validation != 999: #As long as the counter is below 999, this loop is run.
        for i in range(numberOfWords): #'i' can represent any number. The range function specifies what range to look in, which is in this case the amount of words in the user's
            if user_problem[i] == "display" or user_problem[i] == "blank" or user_problem[i] == "turn" and "on" or user_problem[i] == "screen": #If any of the keywords are found
                display_problems()#See line 33.

            elif user_problem[i] == "charge" or user_problem[i] == "battery" or user_problem[i] == "dead" or user_problem[i] == "charging" or user_problem[i] == "die":#See line
                battery_problems()#See line 33.

            elif user_problem[i] == "touch" and "screen" or user_problem[i] == "touch":#See line 20.
                touchScreen_problems()#See line 33.

            elif user_problem[i] == "dim" or user_problem[i] == "bright" or user_problem[i] == "brightness" or user_problem[i] == "dark":#See line 20.
                dim_problems()#See line 33.

            elif user_problem[i] == "volume" or user_problem[i] == "sound" or user_problem[i] == "speaker":#See line 20.
                volume_problems()#See line 33.

        Validation +=1 #Once all the if statements are checked, it adds one to the counter and runs the loop again, this time searching for keywords one space further along.
        if Validation == 999: #This if statement is run once the for loop has run 999 times (which is very quick to a computer). By this stage, we can be sure no keywords have b
            nothing_defined2()#See line 33.

def sony():#See line 14.
    confirmation = 0#See line 160.

    while confirmation != "y" or confirmation != "n":#See line 46

        model = input("\nWhich model of Sony is it? (e.g Xperia Z3, Xperia M4) " )#See line 48.
        global model#See line 190.
        confirmation = input("Sony " + model + ",correct? (y/n) ")#See line 48.
        confirmation_lower = confirmation.lower()#See line 33.

        if confirmation_lower == "y":#See line 18
            main()#See line 33.
            break #The break function is another way to stop a while loop. It tells python to 'break' (out of) the loop.

        elif confirmation_lower == "n":#See line 20.
            sony()#See line 33.
            break#See line 228.
```

```
def samsung():#See line 14.
    confirmation = 0#See line 160.

    while confirmation != "y" or confirmation != "n":#See line 46

        model = input("\nWhich model of Samsung is it? (e.g Galaxy S7, Galaxy Note) ")#See line 48.
        global model#See line 190.
        confirmation = input("Samsung " + model + ",correct? (y/n) ")#See line 48.
        confirmation_lower = confirmation.lower()#See line 33.

        if confirmation_lower == "y":#See line 18
            main()#See line 33.
            break#See line 228.

        elif confirmation_lower == "n":#See line 20.
            samsung()#See line 33.
            break#See line 228.

def iphone():#See line 14.
    confirmation = 0#See line 160.


    while confirmation != "y" or confirmation != "n":#See line 46

        model = input("\nWhich model of iPhone is it? (e.g 5C, 6S+) " )#See line 48.
        global model#See line 190.
        confirmation = input("iPhone " + model + ",correct? (y/n) ")#See line 48.
        confirmation_lower = confirmation.lower()#See line 33.

        if confirmation_lower == "y":#See line 18
            main()#See line 33.
            break#See line 228.

        elif confirmation_lower == "n":#See line 20.
            iphone()#See line 33.
            break#See line 228.

def phone_make():#See line 14.
    validation = 0#See line 160.
    print("\nOkay, what phone make is your device? (No numbers, just the brand.)")#See line 11.
    make = input("Phone make: ")#See line 48.
    make_lower = make.lower()#See line 33.
    global make_lower#See line 190.

    if "iphone" in make_lower:#See line 18
        iphone()#See line 33.

    elif "samsung" in make_lower:#See line 20.
        samsung()#See line 33.
```

```
    elif "sony" in make_lower:#See line 20.
        sony()#See line 33.

    else: #The ELSE statement is only ran if none of the IF statements were satisfied.
        nothing_defined()#See line 33.

def user_details():#See line 14.
    print("\nFirst off, what is your name?")#See line 11.
    name = input("Name: ")#See line 48.
    global name#See line 190.
    print("\nOkay, now what is your email address?")#See line 11.
    email = input("Email: ")#See line 48.
    global email#See line 190.

    confirmation = 0#See line 160.

    while confirmation != "y" or confirmation != "n":#See line 46

        confirmation = input("\nSo your name is " + name + " and your email is " + email + ", correct? (y/n) ") #By using + and a variable, it will print out whatever has been a
        confirmation_lower = confirmation.lower()#See line 33.

        if confirmation_lower == "y":#See line 18
            print("\nThank you for sharing your details. They will be kept on our secure server.")#See line 11.
            phone_make()#See line 33.
            break#See line 228.

        elif confirmation_lower == "n":#See line 20.
            user_details()#See line 33.
            break#See line 228.

user_details() #This is important to starting the code. Although all the def functions have been defined by this point, none have been referenced, and so calling one here start
```

4.Testing

Test no.	Testing	Code to be tested	Input (if applicable)	Expected output	Actual

1	Does it address the user by their name based on their input?	`name = input("Name: ")`	Shaun	So your name is Shaun…	`First off, what is your name?`
					`Name: Shaun`
					`Okay, now what is your email a`
					`Email: SShaun304@gmail.com`
					`So your name is Shaun and your`

2	Does it wait the required time before carrying out the next function using time.sleep?	`print("Welcome t`
		`time.sleep(2)`
		`print("In order`
			N/A	(After 2 seconds) In order to…	`Welcome to the multicellular`
					`e model to tailor results to`
					`d that we only support iPhone`
					`In order to make this work, w`
					(Printed after 2 seconds)

3	Does it correctly identify things like phone make?	`elif "sony" in make_lower:` `sony()`	Sony	Which model of Sony…	`Phone make: Sony` `Which model of Sony`

4	Does it crash when something unexpected is inputted?	while feedback != "y" or feedback != "Y" or feedback != "n" or feedback != "N":
		
		feedback = input("Did this solve your problem? (y/n) ")
			qwerty (anything random)	Did this solve your problem? (y/n)	`Did this solve your problem? (y/n) maybe`
					`Did this solve your problem? (y/n) i don't know`
					`Did this solve your problem? (y/n) ...`

5	Is my code correctly indented?	`def user_details():` `print("First off,`	N/A	First off, what is your name?	`First off, what is your name?`

6	Are variables correctly defined?	`make = input("Phone make: ")`
		`make_lower = make.lower()`
		`global make_lower`
		
		`if "iphone" in make_lower:`
		`iphone()`
			iPhone	Which model of iPhone…	`Phone make: iPhone`
					`Which model of iPhone i`

7	Is there backup line of code in case the if statements are not met to prevent program termination?	`if "iphone" in make_lower:`
		`iphone()`
		
		`elif "samsung" in make_lower:`
		`samsung()`
		
		`elif "sony" in make_lower:`
		`sony()`
		
		`else:`
		`nothing_defined()`
			N/A	Remember, we only support iPhone, Sony and Samsung.	`Okay, what phone make is your device? (No numbers,`
					`Phone make: HTC`
					`Remember, we only support iPhone, Sony and Samsung.`

8	Does it give appropriate outputs based on input?	`elif "samsung" in make_lower:` `samsung()`	Samsung	Which model of Samsung…	`Phone make: samsung` `Which model of Samsung`

9	Does it record the user's details on a txt file if the issue wasn't solved?	`record = open('unresolved cases.txt','a')`
		`record.write("\nName " + name)`
		`record.write('\nEmail: ' + email)`
		`if "iphone" in make_lower:`
		`  record.write('\nDevice: iPhone '+ model)`
		`elif "samsung" in make_lower:`
		`  record.write('\nDevice: Samsung '+ model)`
		`elif "sony" in make_lower:`
		`  record.write('\nDevice: Sony '+ model)`
		`date_time = datetime.now()`
		`date_time = str(date_time)`
		`record.write('\nDate + Time: ' + date_time)`
		`record.write('\nUser Problem: ' + problem)`
		`case = random.randint(0,999)`
		`case = str(case)`
		`record.write('\nCase No: ' + case)`
		`record.close()`
			(After being asked if their issue was solved) n/N	We have saved your details… Your case number is…
				
				(Details stored in the txt file)
					We are sorry we couldn't help you solve your issue. We have saved your details to be looked at by a technician. Your case number is 369.
					
					Name Shaun
					Email: SShaun304@gmail.com
					Device: Sony Xperia
					Date + Time: 2016-09-18 21:11:13.575692
					User Problem: The volume is too low
					Case No: 369

10	Does it keep on asking for an input for a specific variable before a condition is met? (Iteration)	while return1 != "y" or return1 != "Y" or return1 != "n" or return1 != "N": return1 = input("Return to main menu?(y/n) ")	(Anything that isn't y/Y or n/N)	Return to main menu?(y/n)	Return to main menu?(y/n) abc Return to main menu?(y/n) 123 Return to main menu?(y/n) ...

Testing Review

Testing the final part of my code was very successful. All ten tests ran perfectly, because they had been tested prior in the development section, where all the improvements were made. I also got a few of my friends to test out my code, which ran smoothly as well. I did however, add one small thing to my code during testing. When doing test no.7, I found out that after it told the user that the program only supported iPhone, Samsung and Sony, It would Simply call the phone_make() function that asked them to enter their device model. The user wasn't given any option to quit, as they might've not wanted to continue to use the program if it didn't have support for my model. To fix this, I gave the user an option to quit if their device model wasn't specified. By making a def function called return_toMenu2() it had the exact same layout as the original return_toMenu() function, but if the user entered y/Y, it took them back to phone_make() where they could enter another model, and if they entered n/N, then it went to the_end() to terminate the program. Below is a screenshot of the unresolved cases.txt file that stored the details of the users that didn't get a solution.

```
Name wunton master
Email: noodleshop@gmail.com
Date + Time: 2016-09-13 10:58:02.661835
User Problem: my phone  = brightness
Case No: 419
Name Goat Russell Westbrook
Email: Russwest44@gmail.com
Date + Time: 2016-09-13 14:57:48.128674
User Problem: battery is not being able to charge for ma phone u gret me blud
Case No: 333
Name Shaun
Email: email
Date + Time: 2016-09-17 18:30:38.861876
User Problem: battery is dead
Case No: 317
```

The fact that there are user details there means that the code ran smoothly, so that it could get to the point where recording details was required. I also got other people to test it, who didn't know how the code worked, and it worked. It also gave me a chance to see what kind of inputs other users give.

5.Evaluation

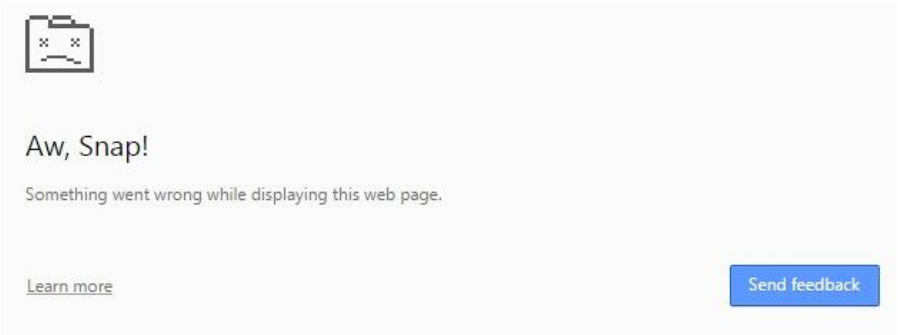
I was required to create a troubleshooting program for mobile phones identified the user's device and load the correct troubleshooting program before analysing the problem to give them the appropriate help they need. If no solution was found, they would instead be given a case number and sent to a technician. My solution does solve this problem, and also satisfies everything in the analysis section. This is proved by the results in the testing section, which all indicate that the code passed the tests, which were specifically based off of the success criteria and test plan. I think that my solution is efficient, but it could be better. For example, the use of def functions makes the code easier to read and identify where mistakes could be made. The use of the .readlines()[] function makes sure that I only have to use one txt file instead of many. However, I just think that the code could've been shorter, since a lot of it was repetitive, especially in the parts where it had to open the specific txt file to read the solution. My pseudocode and flowchart represent my code, and I didn't have to make any changes because I did agile programming instead of the waterfall lifecycle, because I found it easier to make the flowchart and pseudocode after the code had been made. To make my code more efficient, I had to make changes such as adding more functions like the extra return to menu function. Most of the time, I found myself not making changes, but rather adding things, because I found that moving around the code was the main problem, and often the user was denied chances to re-enter things if they entered something wrong, so I had to keep modifying the code to adapt to it. My pseudocode and flowchart represent my code, and I didn't have to make any changes because I did agile programming instead of the waterfall lifecycle, as I found it easier to make the flowchart and pseudocode after the code had been made, meaning that I was basically designing the code as I went along.

Limitations

I did face a few technical issues during my development, though it was only minor ones, nothing too major. I had to do my work on different devices, both pc's and laptops. I had to transfer my work between them using my USB, and make a copy each time I started my work on it. In other words, I always had a backup on each computer, just in case my USB was lost. However, there were some cases where I maybe saved a file in the wrong location by accident, so I had to work on another part of the controlled assessment until I could go back and sync the changes into one file. Most of the pc's I used ran flawlessly when I was doing my work, never once crashing or anything else. However, there were a few pc's that were sometimes slow, often freezing – especially when I tried to browse the net for advice. I also got an internal error to do with python one time when I was on the school computers, but restarting python solved the problem. There is a screenshot below.

```
IDLE internal error in runcode()
Traceback (most recent call last):
  File "C:\Program Files (x86)\Python 3.5\lib\idlelib\rpc.py", line 236, in asyncqueue
    if threading.current_thread() != self.sockthread:
AttributeError: 'MyRPCClient' object has no attribute 'sockthread'
```

There was also a network error I encountered when creating my flowchart, that meant I had to reload the web page and re-open the flowchart again. Fortunately, due to auto save, nothing was lost. The error is below.



I think that maybe more RAM or a faster processor would've easily solved the hardware issues, however it wasn't with all of the school computers, just some. Though these issues were all setbacks, none of them were actually serious enough to affect the solution I produced, and I only had to deal with these issues half of the time, if I was unlucky enough to get a bad computer. In terms of the actual program, it is limited in the amount of solutions it has, and the amount of devices it can actually provide solutions for. The fact that the while loop counter only has a limit of 999 is technically a limitation, because whilst it's highly unlikely for the user's input to exceed 999, if it ever does, the code won't account for any words from the 1000th onwards. Even though I can easily change the limit of words to let's say 9 trillion, there's always the possibility of exceeding the limit by one. Therefore, I would need to find a more efficient way of forcing the for loop to run.

Further Development

To improve my code, I would research harder to find an even more efficient solution, one that uses less lines and is easy to add to. To go even further, I would even add a GUI, so that it wasn't all text based, but it was actually interactive, perhaps even with pictures and animated text to make the program look smoother overall. I would also make a web-based version, and even a downloadable version, for offline use, and possibly even a mobile application, as well as support for other operating systems, so that it can be accessed on virtually any device. I would also make my program easy to add a new device to, or even able to add a new device to the code by itself if possible. All of this will be done with research onto existing codes and how they are structured, so I can get ideas on how to structure mine. I would also of course expand the code from a demonstration version to a full version, adding support for more devices and more solutions as well, once I figure out a more efficient way of doing so. In addition, instead of manually typing solutions into the code, perhaps I could make the code redirect the user to their phone manufacturer's home/ troubleshooting page, so that they weren't left completely in the dark.

6.Conclusion

I was required to create a troubleshooting program for mobile phones identified the user's device and load the correct troubleshooting program before analysing the problem to give them the appropriate help they need. If no solution was found, they would instead be given a case number and sent to a technician. When I started making my code, little research happened, because I either knew how I wanted to do things already, or I had an example saved (taken from various lessons) that I could always refer to. During the later parts of my code development, I began to use the internet more, to find out more about things such as global variables, and to also find out some phone problems specific to the model, so I could create solutions off of them. My previous tasks also helped me a lot, because some of the research done in task 2 helped me here, so I didn't have to do it again. Bits of the code from task 2 were also useful in my task 3 code, with some variable names and some structure components being recycled and revamped to fit task 3 better. My program would be able to help a user with very general problems, such as brightness and volume, and only those with an iPhone, Samsung or Sony. Though there were only five solutions for each device (a total of fifteen), I could always add more solutions and devices later on; however, I wasn't required to because it mentioned in the controlled assessment material that it was a demonstration version of the system only, and need not deal with all potential problems. I think that for a demonstration, the program would be quite helpful anyway, since it has support for the biggest phone brands with the most common problems, but of course can always be edited to accommodate for a wider range of devices and problems.

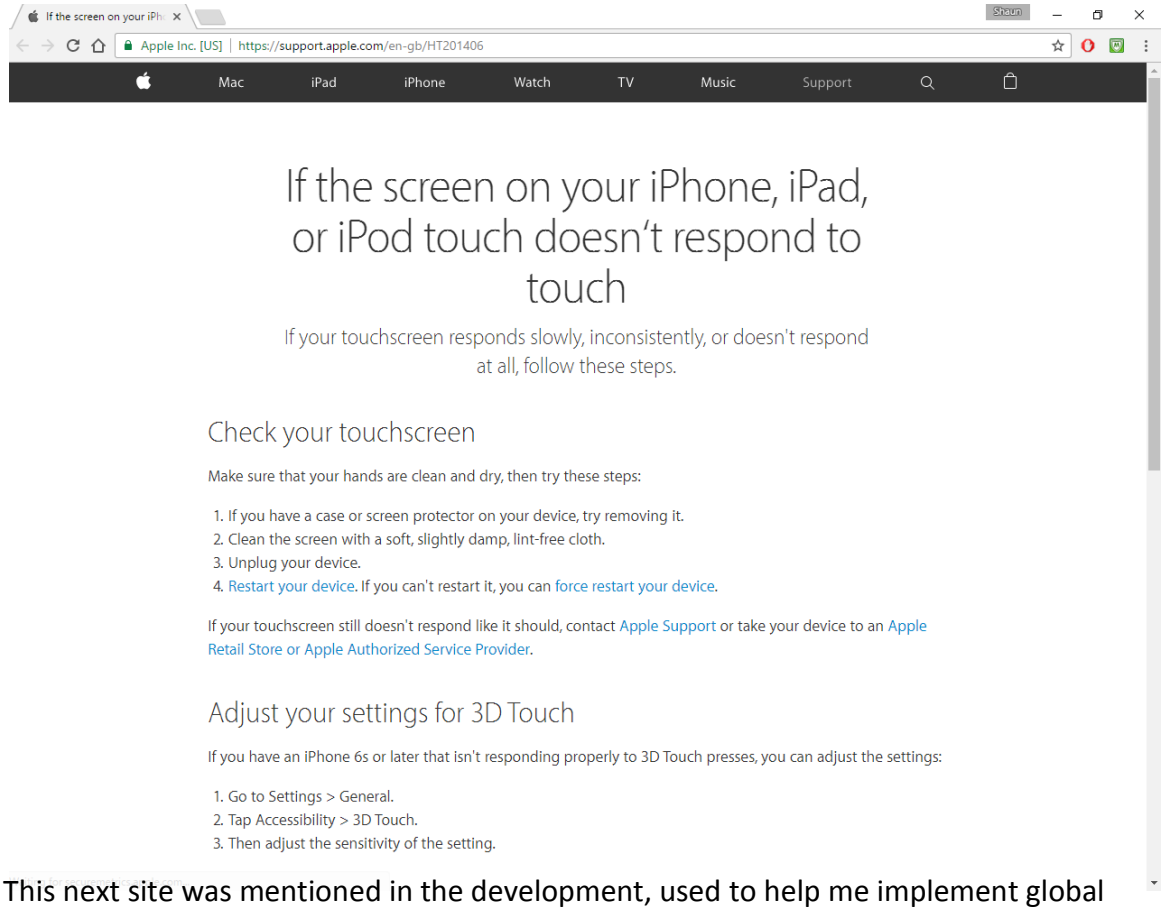
7.References

There are a few sites I visited to aid me in my task 3. These sites were to help me come up with solutions for iPhones. I only needed help with these because I didn't know much about troubleshooting iPhones, since I hadn't owned one for quite a while. Since I had a Sony, I knew a lot about troubleshooting those, so no research was required. I had also previously owned a Samsung, so like the Sony, there wasn't much need for research. In addition, Sony and Samsung both use the same OS (android), so troubleshooting would be pretty similar on both devices.

Sites:

http://www.howtogeek.com/216839/what-to-do-when-your-iphone-or-ipad-won%E2%80%99t-turn-on/

https://support.apple.com/en-gb/HT201406



If the screen on your iPhone, iPad, or iPod touch doesn't respond to touch

If your touchscreen responds slowly, inconsistently, or doesn't respond at all, follow these steps.

Check your touchscreen

Make sure that your hands are clean and dry, then try these steps:

1. If you have a case or screen protector on your device, try removing it.
2. Clean the screen with a soft, slightly damp, lint-free cloth.
3. Unplug your device.
4. Restart your device. If you can't restart it, you can force restart your device.

If your touchscreen still doesn't respond like it should, contact Apple Support or take your device to an Apple Retail Store or Apple Authorized Service Provider.

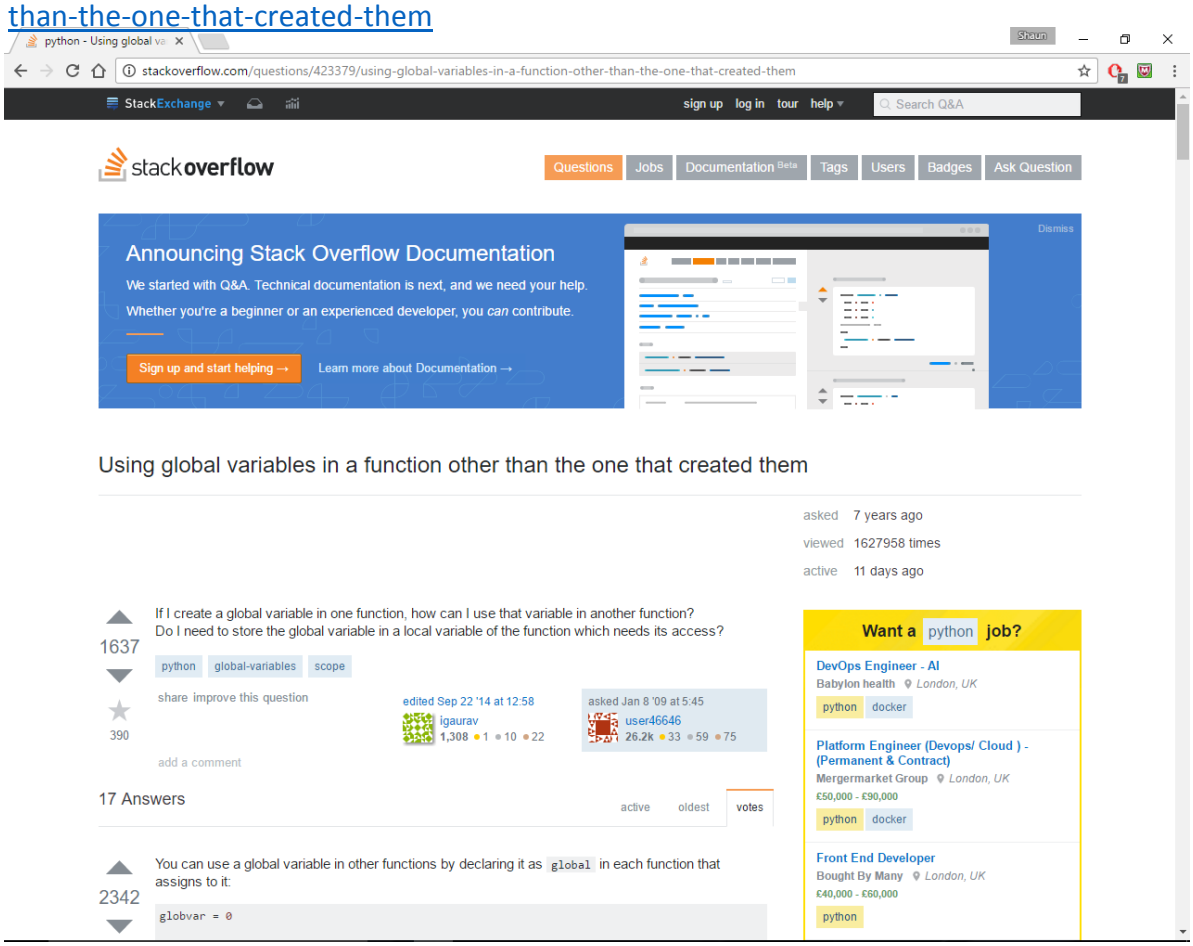
Adjust your settings for 3D Touch

If you have an iPhone 6s or later that isn't responding properly to 3D Touch presses, you can adjust the settings:

1. Go to Settings > General.
2. Tap Accessibility > 3D Touch.
3. Then adjust the sensitivity of the setting.

This next site was mentioned in the development, used to help me implement global variables in my code.

http://stackoverflow.com/questions/423379/using-global-variables-in-a-function-other-than-the-one-that-created-them



Announcing Stack Overflow Documentation

We started with Q&A. Technical documentation is next, and we need your help.

Whether you're a beginner or an experienced developer, you can contribute.

Sign up and start helping → Learn more about Documentation →

Using global variables in a function other than the one that created them

asked 7 years ago
viewed 1627958 times
active 11 days ago

1637

If I create a global variable in one function, how can I use that variable in another function? Do I need to store the global variable in a local variable of the function which needs its access?

python global-variables scope

share improve this question

edited Sep 22 '14 at 12:58 igaurav 1,308 1 10 22

asked Jan 8 '09 at 5:45 user46646 26.2k 33 59 75

add a comment

390

17 Answers

active oldest votes

2342

You can use a global variable in other functions by declaring it as `global` in each function that assigns to it:

```
globvar = 0
```

Want a python job?

DevOps Engineer - AI
Babylon health London, UK
python docker

Platform Engineer (Devops/ Cloud) - (Permanent & Contract)
Mergermarket Group London, UK
£50,000 - £90,000
python docker

Front End Developer
Bought By Many London, UK
£40,000 - £60,000
python