Problem identification

Using Python I am going to be making a holiday comparison program which will search through keywords in a description given by the user to try and find the perfect holiday based on past reviews left by other users. I came up with the problem as I realised there was nowhere which allowed for this as previous holiday comparison websites work based off the user already knowing where they wish to go on holiday. I want to create a program which allows to user to come to a conclusion of where they want to go on holiday based off what they are looking for in a holiday. What makes this solvable is the ability for a computer to search through a program text much faster than a human could. This means the computer can do a human holiday advisors job but faster and more automated.

The program needs to be written well as to make it be able to make it clear what the user must do and user computing principles of iteration and selection to compare and contrast the multiple different options for a holiday to be able to offer them to a user.

Stakeholders

This program helps people who don't know where they want to go on holiday but know what kind of things they want to get done. For example, someone may want a good place to go hiking that's also close to a beach and my program will search through holiday destinations to find which location would be best based on the given description. A company that could use a program like this would be Trivago as they already have a search program inbuilt into their website and they could use a program like this because it would bring in more customers that are unsure of what they are doing. Helping Trivago compete with businesses based off human holiday advisors by making the search for their ideal holiday much simpler than ever before.

This program could be used both on the Trivago website and also be later adapted to the Trivago mobile app to make the program accessible to as many users as possible.

The other main stakeholder would be the User as they will ultimately be the people after a holiday so the program needs to be suited towards making their experience the best possible. That is why throughout testing I am going to try and keep the users as close to the development as possible.

Potential Issues

Some issues that may come in the program are that this program is based on user opinions so it may differ for each person and if the program is not user friendly in the UI it could become very clustered and drive customers away. The way I'm going to try and solve these problems is that for each person's opinion I'm going to compare it to other user reviews and find the closest ones to what is being searched for by finding an average over all reviews. To make the UI not clustered I'm going to try and keep everything very simple for the user and use ideas of information hiding to keep all of the unnecessary data from the user. The simpler the program for the user the more people that will want to use the program so the more beneficial it will be for the end company.

There are limitations to this program as with any other. The limitations of this program have to do with the allotted time the amount of available data usable by the program. To compare the descriptions of the user to previous reviews I have to have a backlog of these reviews. This program is also opinion based which causes a problem as everyone's opinions differ and certain interpretations of the given description may mean different things to different people. If I have

enough time at the end of the project, I'll make the program learn from ratings that way over time that way it will improve as more people use it.

Research

Primarily my research has been on Trivago and other holiday comparison websites so I can see how they find their "best deals" and the layout of their search functions.

For the object oriented approach, I'm looking to take I'm choosing to write my code in python, I have also considered using java. In the end I chose python as this is the language I am more confident with and am familiar with what needs to be done.

Other research I intend on doing is looking up other comparison codes to see how other people have compared two items in their codes. My code intends on using text searching and list comparison algorithms to compare the input options the user selects to a list of country attributes and place attributes desirable by the user. The reasons I have selected python to work in is there is a lot of in built list comparison available. I can use cmp(list1, list2) or I could pop particular attributes from the list to compare and search for it in the selected options.

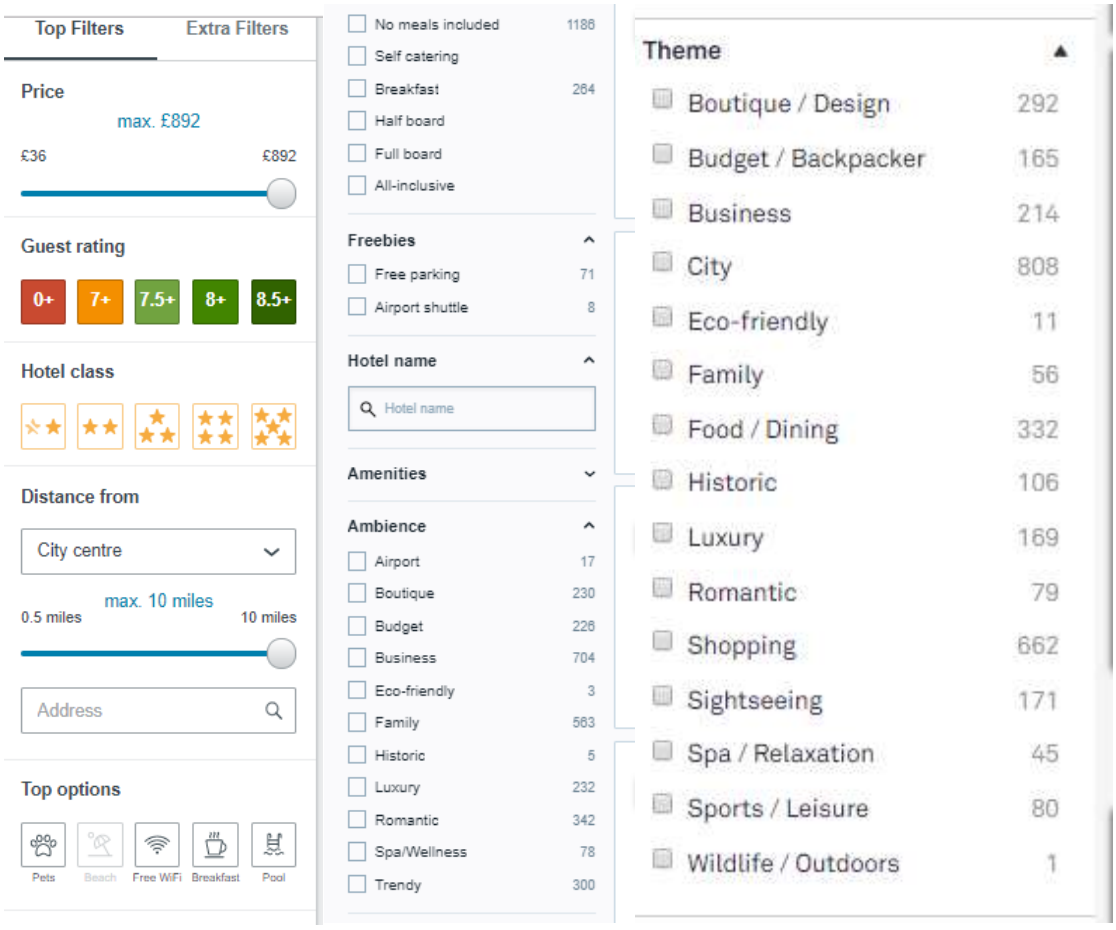
In order to create the attributes for each place for the comparison beforehand ill look at attributes listed on holiday comparison websites already and from that create a list to be compared to. The sample list I'm going to be using to compare to as proof of concept will be 20 items long in order to adequately meet testing requirements. I will be selecting each of these destinations by looking at what the top 15 destinations are and selecting the last 5 by hand. This is because some of the items will need to be similar to each other in order to try and fool the algorithm. My testing will try and find as many different scenarios as possible and I'll try and get members of the public to test my code in order to try and find scenarios which I may not have thought about. In asking members of the public I will be looking for people between the ages of 20 and 60 as that is the age of most people using this kind of program. The target age range is as wide as this is a holiday booking system and people of all ages go on holiday. The reasons I haven't included younger and older age groups than the selected range is that older people are not as technology capable and tend to not travel as much and younger people need their parents to travel with them. Any Questions I ask will also be to people of that same age range.

After considering how I should start and approach the project I decided to ask some members of the public how they thought the code should work. The response I got showed that people don't want to have to write a lot when looking for holidays so this gave me the idea of giving the user options for what they want out of a holiday. At the start of the code the user will be given the choice of writing a description or picking set choices this way they can choose how much they want to write the description should end up being more accurate as more accurate information is given so more can be interpreted. The way this would look is once you start the code the interface will ask what you want out of your holiday with examples and sliders showing things like distance from beaches, big cities or other thing people could look for in a holiday. To find out what people are looking for in a holiday I will research other holiday comparison sites to see what features they list with each of their reviews of holiday locations.

The requirements for the system will be python as this is what I'm creating the code on, in the real world and once the code would be transferred to the Trivago website it may have to be interpreted into java script and given more visual aid than already used as on the website it needs to fit the style of the rest of the website.

For my code to be successful I need the user to select from a list of options of what they can have in a holiday and for it to select a destination based on this. To test weather my code is accurately selecting destinations I will ask the user at the end for feedback and to rate the selected destination. This way I can improve on the code over time. It may be difficult for the code to distinguish between two destinations from time to time as sometimes they may be too similar. This isn't something that I can easily fix as this is down to user opinion. My solution for this will be to offer both location names and ask further questions to determine their preference.

I looked at three different holiday comparison websites and found the different features they offer when selecting a holiday, the three I looked at were Trivago, kayak and travel supermarket.



These websites offered ideas for features that my user can select in deciding what they want in their holiday. Not all of the above options are applicable for what I intend on doing for finding out what holiday the user is after so when creating my code, I will select only the appropriate options and ask the user to select from those.

From what can be seen in the research conducted I found that all the comparison websites required me to input a known destination and the features offered seemed to be based of the style of location picked. This led me to pick some features from each different websites and come up with some features of my own that may not have been applicable to the holiday comparison programs available.

I then researched some comparison code examples but couldn't find any full codes so used tutorials point to give an example of list comparison. The method of comparison ill be using will be comparing the indexes of the list in order to give a score where the first index gives the highest score. This

method will work as if the first and second item get compared the first item will give a higher multiplier than the second item so lists only comparing the last items will only give the lowest scores.

System Requirements

The system requirements are that you must be able to run python and TKinter for the gui programming for the menu. In terms of other hardware only a keyboard and mouse are required the mouse will be used in the gui programming.

Once the program is complete and sold to Trivago there will be no need for the user to have python as it will be attached to their website most likely via importing cgitb and using that to convert the python to the webpage. A similar process will be used to the app however this will be determined by what language the app will be written in.

Success Criteria

To ensure the program is a success it needs to be able to distinguish between similar cities and pick up words from the user description given. It needs to feel user friendly and accessible enough so even the least tech savvy people can understand what needs to be done. To ensure that happens I will keep in close connection with users so any wording changes or design changes can be made immediately.

For the program to be an improvement on Trivagos interface they already have it needs to add something that doesn't already exist. The city finder doesn't already exist and should help customers find where they want to go much easier.

For this project I will not be using AI this is because of the lack of available data to me so for this project to be successful the program needs to do all the already mentioned criteria whilst proving that if AI was used it would be possible for a more accurate and sustainable program to be created. Whilst my program should be successful in its own right. This will be because im also only searching through 20 cities.

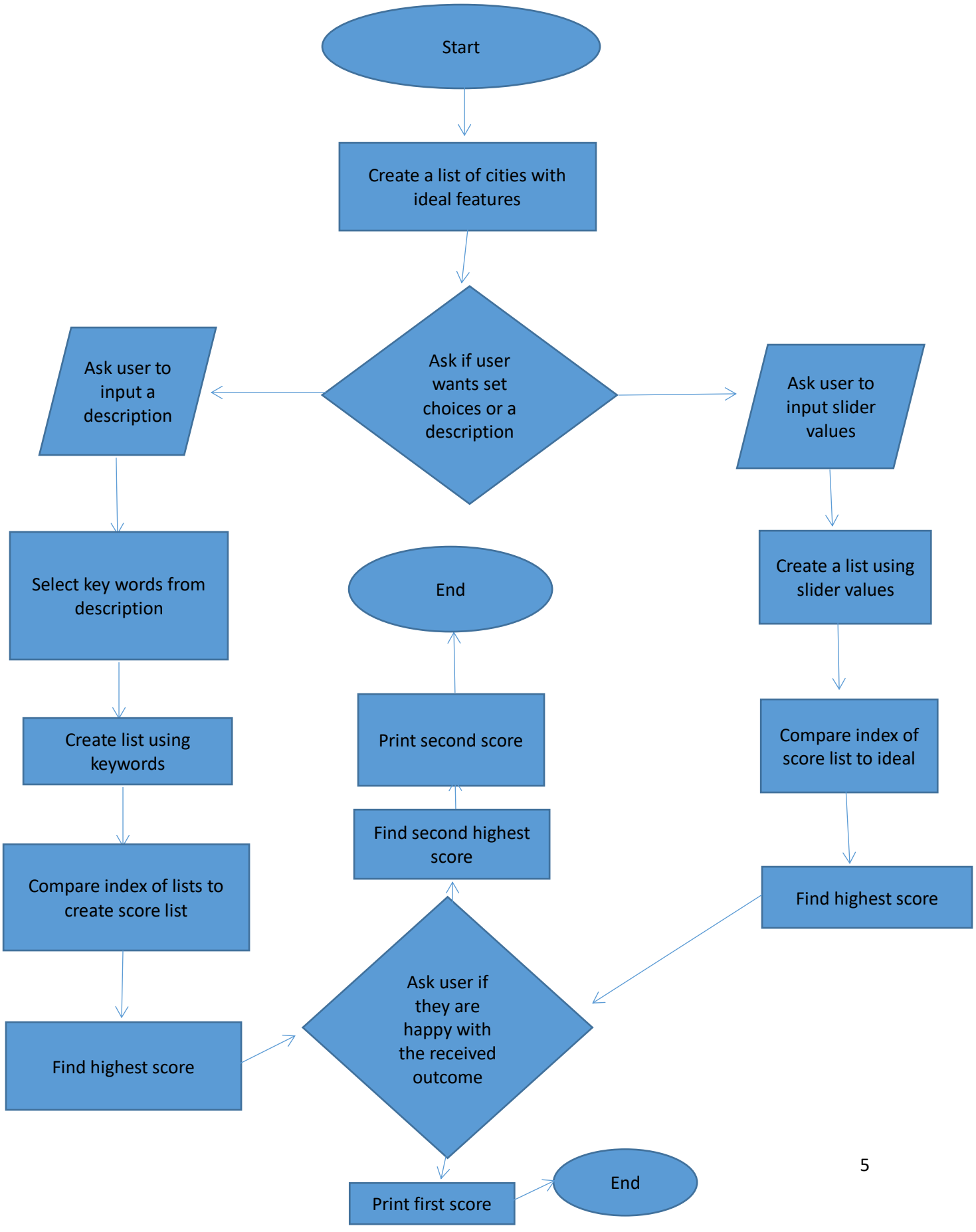
Design

For my design I opted to use both a flowchart and sudo code in order to really understand and plan the structure of the code I then plan on designing the menus to be used in the code that the user will be using to select their choices for their holidays.

The program can be seen to be programmed in two separate parts being the choices section and the description section. The choices part will be designed and created using GUI programming to make an interface for the menu to appear on. This is the method I have chosen as it feels like it will be the most user friendly option. This is important as user friendliness is a key factor in the designing of this program. This is a usability feature that will help the user visualise what they are selecting for their holiday.

These two parts can also be further broken down into smaller parts with the different steps including separating the user answers into lists to then be compared with the lists created for each city. This will allow for a computational solution as lists can quickly be compared by a computer and can also be easily ordered. There are many different ways to manipulate lists such as stacks or queues and these different ways will each provide a different method for me to use the data.

Flowchart


Start
Create a list of cities with ideal features
Ask if user wants set choices or a description
Ask user to input a description
Ask user to input slider values
Select key words from description
Create list using keywords
Compare index of lists to create score list
Find highest score
Create a list using slider values
Compare index of score list to ideal
Find highest score
Ask user if they are happy with the received outcome
Find second highest score
Print second score
End
Print first score
End

Sudo Code

Create lists (Most important, second most, third, least)

Choice = input (“Do you want to write a description or use a menu selection”)

If Choice is description:

    Description = input (“write a description with no spelling errors”)

    Key_words = Select key words in Description

    Key_list (Keyword 1, keyword 2, ect.)

    For each option in key_list:

        Inverse key list

        Inverse created list

        For each item in key list:

            Score =Keylist index * created list index

        Select list with highest score

        Is (Highest score list) a destination you want to go to:

            If yes:

                END

            If no:

                Select next highest score

                Return until answer is yes

Else:

    Select options from set menu

    Order these options based user preferences

    Keylist(option1,option2,option3)

    For each option in key_list:

        Inverse key list

        Inverse created list

        For each item in key list:

            Score =Keylist index * created list index

    Select list with highest score

Is (Highest score list) a destination you want to go to:

If yes:

END

If no:

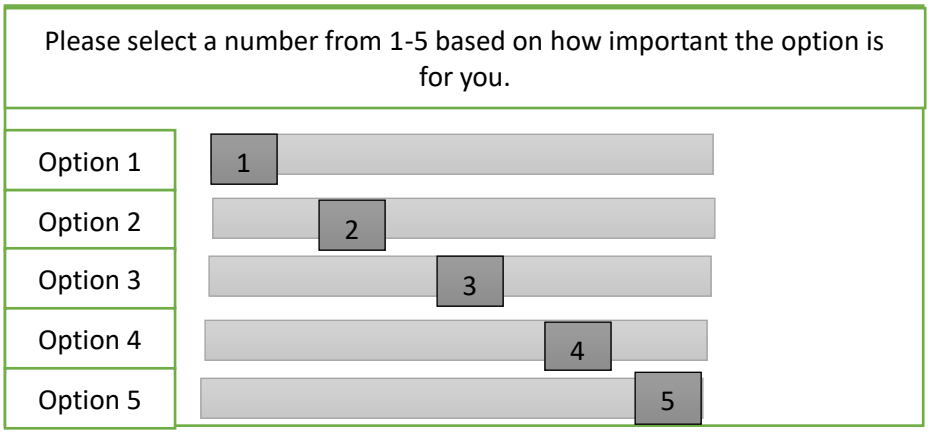
Select next highest score

Return until answer is yes

END

Menu Design

This is the menu design for the menu which will be available if you select to select the features.



This menu is a usability feature to be added to the program in order to help the user visualise what they are actually doing. This is going to be added both for those users that do not wish to write a lot and those users that may get confused or stuck using the descriptions option.

Test Table (Description)

Test Number	Input	Desired output	Reasoning
1	Historic being the only feature selected	Bangkok, London, Prague, Barcelona, Amsterdam, Moscow	This test shows that multiple cities can be detected these cities should be the selected ones as they all have historic as the first attribute and in this test none of the other attributes matter.
2	Historic, Shopping, Nightlife, Transport	Bangkok	This test shows the features can select a specific city.
3	Historic being only feature selected with the user selecting no after options given	Same cities as test 1 but after no selected the next cities will be Budapest, Vienna, Paris, New York	This shows the saying no also works. The cities selected are all the cities with historic in place number two.
4	Historic, Cheap , Food, Health	Budapest	This should be the outcome given as there are only 2 attributes in the wrong place. This test shows that the attributes don't have to be perfect for one city to be selected they just have to be close.
5	No keywords given	Should give user option to select from all of the attributes	I will be making sure that if the list of attributes picked up is equal to 0 then all attributes will be selected from.
6	Wrong word put in when selecting	Should ask user to input one of the options again	If the input isn't equal to the picked up on attributes it should ask the user to input the attributes again.
7	Not putting in the words either choices or description	Should ask the user to input either choices or description	If the input isn't equal to either choices or description the user should be asked again.

Test Table (Menu)

Test Number	Input	Desired output	Reasoning
1	Historic bar pulled to 5 stars	Bangkok	This section is different to the description section as even though I have only selected on attribute to 5 stars because all attributes are still accounted for the printed cities are not just those with historic In position one but instead is the one with historic in position on and the other attributes listed in order which in this case is Bangkok
2	Culture set to 5 shopping set to 4 Food and historic on 3 and nightlife set to 2	Hong kong and Istanbul	With these attributes the scores for Hong kong and Istanbul should be similar as there is only one difference between the two.
3	None of the scores changed	Bangkok	Because this will be creating a list with all the attributes in order this should give Bangkok as Bangkok has the first 4 attributes in order.
4	Each bar tested individually	Should show the city that has the current bar set to 5 in position one with the next attributes being in the order of historic shopping… ect	This test will show that each bar has an effect on the program individually and show the cities selected are calculated.
5	Saying no with each test	Should give the second option	This shows that a second choice is given and if multiple second choices aren't given for any of the tests that shows that the code isn't functioning correctly.

Software Development and testing Iteration 1

The software development methodology that I have decided to follow is the RAD system life cycle. I will be using 3 different iterations. These iterations will each contain a different aspect that users will point out during the testing of my program. This will work as I am only one person and will not think of every feature that users may wish to include.
I have chosen the RAD system life cycle as this should provide the ability to improve on what I have the most as with each time I reach the end of the cycle I will receive user feedback

To begin before coding I had to select cities. To do that what I did was select the top 15 most visited cities and then selected 3 more that I considered similar to another city on the list and then 2 which I considered completely different. After this I had to create model attributes for these cities so I could compare each city. A problem with this was this was mostly based off my opinion or experiences other people have had that I researched. This would be different for everyone and the only way to make this more accurate would be over time I could gather reviews and use AI to slowly make the searching algorithm better and use more people's opinions than just mine. The attributes I had come up with were:

Bangkok ('Historic', 'Shopping', 'Nightlife', 'Transport')
Budapest ('Cheap', 'Historic', 'Food', 'Health')
London ('Historic', 'Shopping', 'Nightlife', 'Culture', 'Health')
Vienna ('Food', 'Historic', 'Transport', 'Culture', 'Health')
Paris ('Food', 'Historic', 'Shopping', 'Nightlife')
Prague ('Historic', 'Cheap', 'Culture', 'Family', 'Food')
Dubai ('Shopping', 'Health', 'Nightlife', 'Food')
Singapore ('Health', 'Shopping', 'Food', 'Nightlife')
Tokyo ('Shopping', 'Food', 'Culture', 'Family', 'Historic')
Seoul ('Culture', 'Food', 'Historic', 'Nightlife')
New York ('Shopping', 'Historic', 'Culture', 'Nightlife')
Kuala Lumpur ('Nightlife', 'Cheap', 'Transport', 'Shopping', 'Food')
Hong Kong ('Culture', 'Shopping', 'Food', 'Nightlife')
Istanbul ('Culture', 'Shopping', 'Historic', 'Nightlife')
Barcelona ('Historic', 'Food', 'Culture', 'Nightlife')
Amsterdam ('Historic', 'Culture', 'Nightlife', 'Food')
Milan ('Food', 'Shopping', 'Culture', 'Nightlife')
Osaka ('Food', 'Culture', 'Shopping', 'Historic')
Moscow ('Historic', 'Nightlife', 'Health', 'Culture')
Johannesburg ('Health', 'Shopping', 'Historic', 'Nightlife')

I ordered each of these attributes in the lists to make what I considered the most important attribute first and the least important last this is so I can actually distinguish between the different cities otherwise these just become words that mean nothing to a searching algorithm.

This was following this step in the flow chart:

To convert this list into python code to create a list in python what I did is this:

```
Bangkok = ['HISTORIC', 'SHOPPING', 'NIGHTLIFE', 'TRANSPORT']
Budapest = ['AFFORDABILITY', 'HISTORIC', 'FOOD', 'HEALTH']
London = ['HISTORIC', 'SHOPPING', 'NIGHTLIFE', 'CULTURE', 'HEALTH']
Vienna = ['FOOD', 'HISTORIC', 'TRANSPORT', 'CULTURE', 'HEALTH']
Paris = ['FOOD', 'HISTORIC', 'SHOPPING', 'NIGHTLIFE']
Prague = ['HISTORIC', 'CHEAP', 'CULTURE', 'FAMILY', 'FOOD']
Dubai = ['SHOPPING', 'HEALTH', 'NIGHTLIFE', 'FOOD']
Singapore = ['HEALTH', 'SHOPPING', 'FOOD', 'NIGHTLIFE']
Tokyo = ['SHOPPING', 'FOOD', 'CULTURE', 'FAMILY', 'HISTORIC']
Seoul = ['CULTURE', 'FOOD', 'HISTORIC', 'NIGHTLIFE']
New_York = ['SHOPPING', 'HISTORIC', 'CULTURE', 'NIGHTLIFE']
Kuala_Lumpur = ['NIGHTLIFE', 'CHEAP', 'TRANSPORT', 'SHOPPING', 'FOOD']
Hong_Kong = ['CULTURE', 'SHOPPING', 'FOOD', 'NIGHTLIFE']
Istanbul = ['CULTURE', 'SHOPPING', 'HISTORIC', 'NIGHTLIFE']
Barcelona = ['HISTORIC', 'FOOD', 'CULTURE', 'NIGHTLIFE']
Amsterdam = ['HISTORIC', 'CULTURE', 'NIGHTLIFE', 'FOOD']
Milan = ['FOOD', 'SHOPPING', 'CULTURE', 'NIGHTLIFE']
Osaka = ['FOOD', 'CULTURE', 'SHOPPING', 'HISTORIC']
Moscow = ['HISTORIC', 'NIGHTLIFE', 'HEALTH', 'CULTURE']
Johannesburg = ['HEALTH', 'SHOPPING', 'HISTORIC', 'NIGHTLIFE']
```

The first thing the user must do is decide whether or not they wish to write a description or choose the attributes from a menu.

As followed in the flowchart I created:

To implement this choice I just put in a simple if statement as if they do not put choices or description an error will occur:

```
print("This is a holiday comparison code which will choose a holiday destination city around the world based on either an input description or a list of set choices")
    Choice = input("do you wish to write a description or choose from a list of set choices (input description or choices to make the decision):")
    if Choice == "description" or Choice.title() == ("Description"):
    elif Choice == "choices" or Choice.title() == ("Choices"):
        print("nothing")
    else:
        print("The choice you have input is not valid please try again")
```

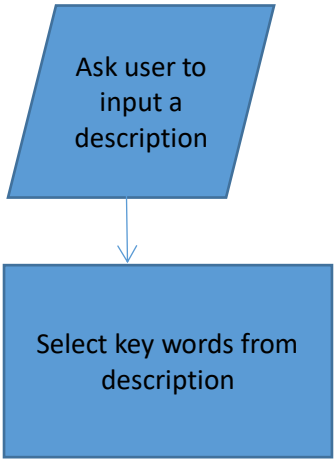
The else statement is in place in case either description or choices are not input.
This made me think I should create a function to restart the program if the correct answers are not input. I just called my function “Restart”:

```
def Restart():
    Choice = input("do you wish to write a description or choose from a list of set choices (input description or choices to make the decision):")
    if Choice == "description" or Choice.title() == ("Description"):
    elif Choice == "choices" or Choice.title() == ("Choices"):
        print("nothing")
    else:
        print("The choice you have input is not valid please try again")
        Restart()
Restart()
```

I placed the restart after the else statement as the program should return to the top if description or choices are not input and will ask the user again whether or not they want to go down the description path or the choices path.
I decided I would start coding the description part first and move on to the selected section after as I thought that part would be easier.
As I was not using proper AI modules because I do not have the data available to utilise it I decided to find words that would then link to specific attributes I could then add to this list of words later and as I get other people to test it they will definitely come up with more words I didn’t think of as without more resources it would be hard to come up with every word that could be used to describe an attribute.

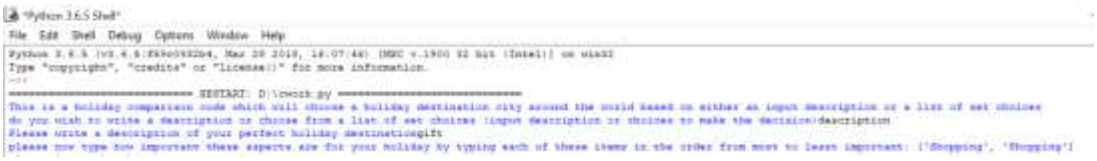
```
if Choice == "description" or Choice.title() == ("Description"):
    Description = input("Please write a description of your perfect holiday destination")
    Historic = ("history" or "sightseeing" or "old" or "ancient" or "past" or "tour" or "historic" or "historical" or "landmark")
    Shopping = ("retail" or "shopping" or "expenditure" or "gift" or "merchandise" or "purchase" or "market")
    Nightlife = ("nightlife" or "clubs" or "clubbing" or "party" or "rave" or "dj" or "music" or "club" or "bar" or "bars" or "pub" or "drink")
    Transport = ("manoeuvre" or "move" or "travel" or "transport" or "bus" or "train" or "tram" or "plane" or "airport" or "station" or "coach" or "bike" or "bicycle" or "boat"
    Cheap = ("cheap" or "budget" or "inexpensive" or "value" or "price" or "economical" or "affordable" or "sale" or "bargin" or "economy")
    Food = ("dining" or "cuisine" or "eat" or "food" or "meat" or "fish" or "taste" or "menu")
    Culture = ("culture" or "people" or "religion" or "social" or "ethnic" or "tourism" or "society")
    Health = ("health" or "recreation" or "life" or "environment" or "sport" or "park" or "cycling" or "walking" or "walk" or "leisure" or "park" or "beach" or "animal" or "wi
    Family = ("family" or "child" or "kid" or "park" or "roller" or "friendly" or "young")
```

I then had to make a list of all the attributes the user has found based on the words they used in the description and also ask the user what their description of their perfect holiday is as followed by the flow chart:



When testing if this had worked I discovered that nothing was being picked up.
Another method I tried to use to solve the problem was this:

When testing this:



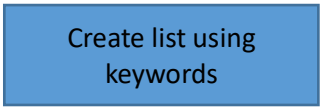
For 1-word shopping should not have appeared twice this was also strange because only one word was mentioned and the position of gift in the keywords list for shopping isn't in the second position either.
This made me then change my method for how I searched though both lists.
I then found a solution that functioned the way I wanted it to:

This solution searches through each index of the word list and compares them with the description the user has input appending each value to a list called order. The not in order part just makes sure that if the user uses 2 works for one attribute that it does not end up in the list twice.
The next thing I had to do was make sure that the list was in order so that it could be compared with the ordered lists of each of the cities.
If the user had put in no picked up words I decided to ask the user to order each attribute individually I did this by checking if the length of the attributes at the end was 0 then give all attributes.

To do this I just asked the user to order each attribute based on importance to their ideal holiday. This was simply done by getting the user to type each item out individually in order of importance and then creating a list out of the order said. Another way this could have been done would be to judge how often each keyword came up and in what order however without AI this would have been an ineffective method so I opted for asking the user directly.

The list “NewList” is created so that the user’s attributes can be placed into a list at the end.
We then check the length of the list Order as this is how many attributes they have so we can make sure they put in all of them. I made the list Check to make sure that the user had actually put in an attribute name making both comparisons in the .upper form to ensure they were checking the same word. I then compare each item in “UserList” by finding the word after the space than in the correct order append all of the information we have to “NewList”. The function again will restart that particular sequence if the user inputs an unexpected response.

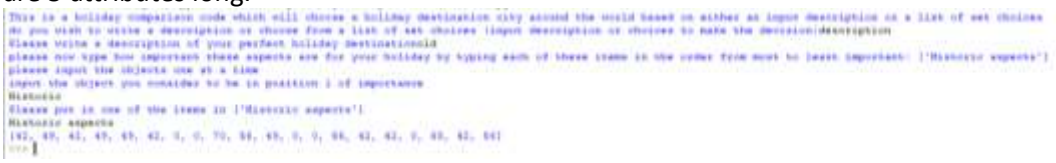
This is shown in my flow chart as this step:



Following my flowchart the next step I had to do was to compare the indexes of both list to create a score:

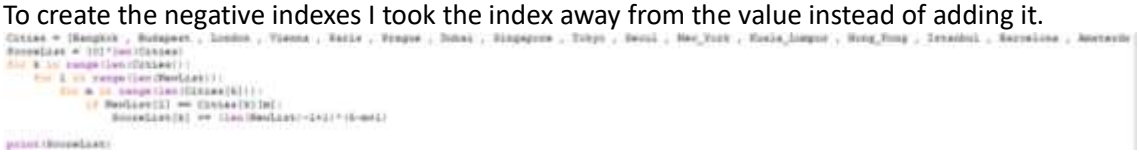
First what I did is I turned the cities into a list and then created a list of score 0 for the number of cities in the list. I then iterated through each city and then each attribute in the created ideal list. The next thing I did was check that the item I was looking at was in both lists otherwise the score should be 0 as it's not in either list. Then to compare the two lists I got the length of the lists and added the index and 1 because the first index is 0 and I multiplied both lists together and this creates a score for each item in the cities list.

The reason I have the value of 5+m for the cities indexes is because the biggest lists in the cities table are 5 attributes long.

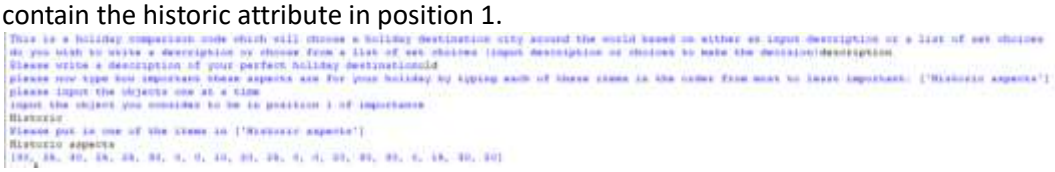


After testing I noticed a mistake. I tested with one attribute being historic. This is something I said Bangkok has as a number 1 attribute so it should be first however, the result I got doesn't show it as first. Through this I realised that I had to compare the negative indexes of each one so that the first values have the highest value and the next ones all decrease by one. This would also help solve the problem of longer lists having higher scores as the last value isn't worth much to the score and so can almost be disregarded. I believe cities with more attributes should come up a little bit more often as these will tend to be more popular locations anyway. For this reason I don't think the tiny extra score received will be a problem.

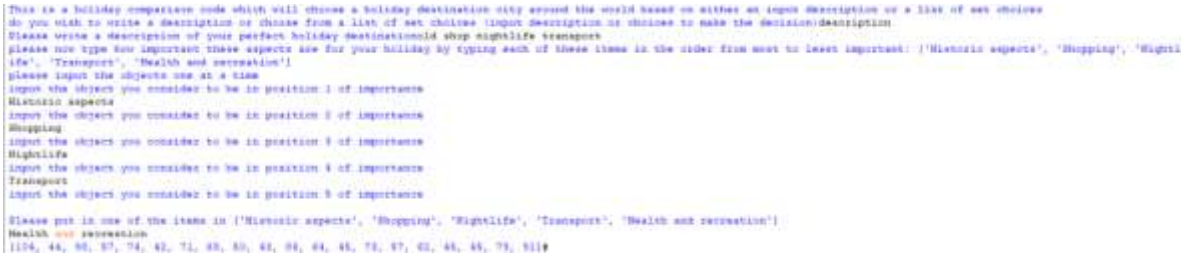
To create the negative indexes I took the index away from the value instead of adding it.



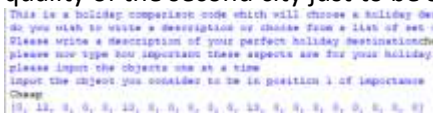
The result printed worked for the one attribute test as all the highest values are all of the cities that contain the historic attribute in position 1.



I tested the same code for multiple attributes to see if I would get the same result and I got the result I expected. The first test was for all of the qualities that my first city had so It should have been the highest score.



The second test I did was for if it was always making the first score highest so I tested with the first quality of the second city just to be sure.



And lastly I tested it with all the qualities of the second city to see if It would return that as the highest score.

After all of these test were successful I decided to move on to the next flowchart point which was:

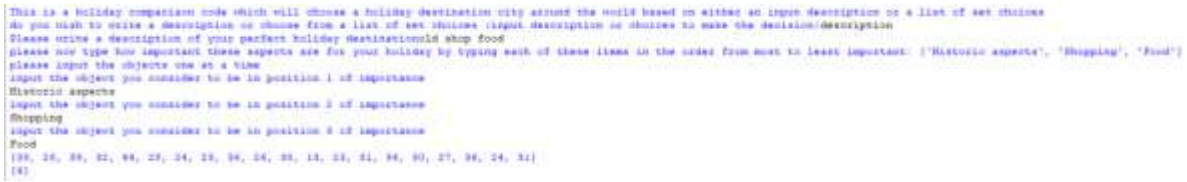
My method for doing this was to use the iter() function to iterate through my score list to make sure I got every highest value and then use the max() function to look for every maximum value of the already iterated score list.

I then created a variable called o which I made equal to -1 so that the range of the for loop starts on 0 as this is the first index of the score list. This is because each time I cycle through my for loop I am adding a value of 1 and -1+1 is equal to 0.

I then check if the Current city being iterated through has the same value as the maximum value from the score list and if it does I append the index. This means that I now have a list made of just the indexes of all of the highest value cities.

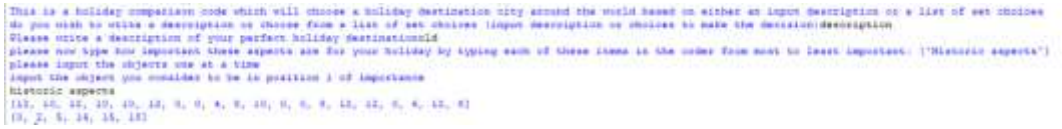
```
IterScore = iter(ScoreList)
City = max(ScoreList)

o = -1


Options = []
for n in range(0, len(ScoreList)):
    NextItem = next(IterScore)

    o = o+1
    if NextItem == City:

        Options.append(o)
```

To test if this worked I just printed the value of the table options. I could then check the value options which what I expected the outcome to be.



This first test was to see if the code worked for 1 value. The outcome of the value 4 shows the highest value is in the 5th position or the 4th index.

We checked if this same test worked again for different values and again it displayed the highest value.

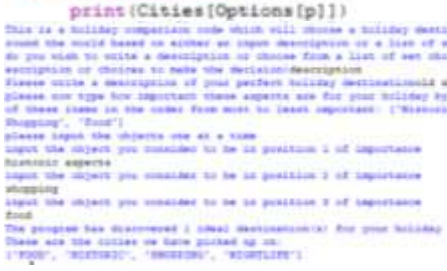


This test involved checking if it was discovering all the indexes of all the highest numbers. The outcome shows all the cities with value of 12 and by their index.

This last test was just to ensure that it wasn't just lucky that we got all of these values.

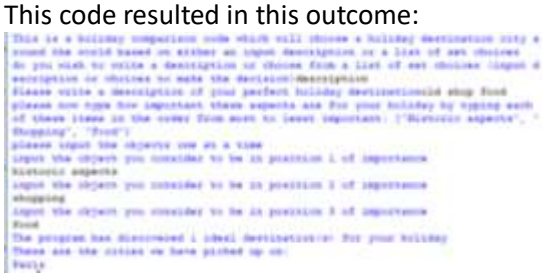
With these indexes what we can do is then print each city with these values.
To do this the first method we tested was to just create a for loop which prints all the values for the length of how many indexes there were.

```
print("The program has discovered", len(Options), "ideal destination(s)
print("These are the cities we have picked up on:")
for p in range(len(Options)):
    print(Cities[Options[p]])
```

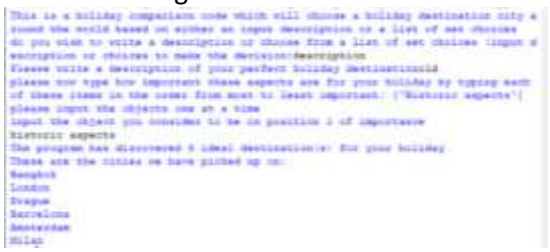


The results for the test were as that it printed the lists and not the city names. I then thought that what I would have to do is create a list of cities where the cities are strings and not lists.

This code resulted in this outcome:

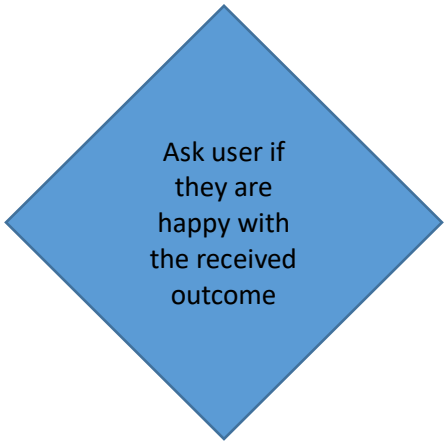


Seen as the result was a city name I checked with my previous testing values as I have used those attributes before to see if the city name was the correct one.
All of this being correct lead me test if it worked for multiple cities these where the results:



This proved that my code was working as intended.

Following my flowchart my next task was to give the user a second highest option for a city if they weren't happy with the first.



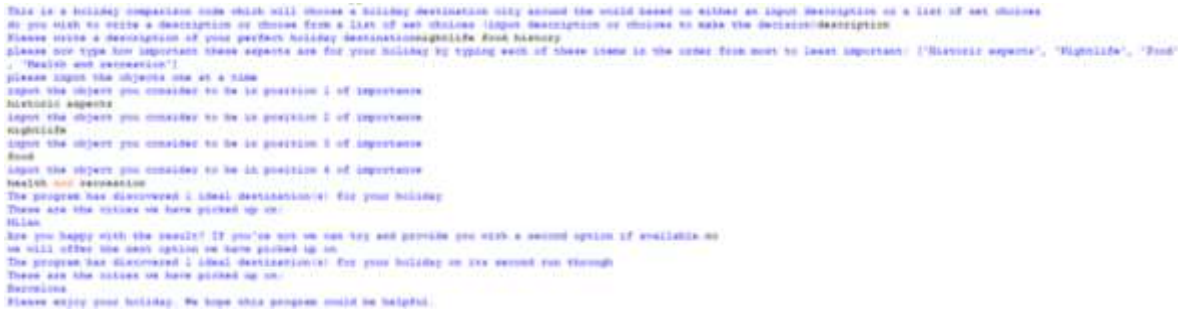
Thinking about how to approach this I thought I should order the score list and find the second value. The problem I thought with this was there may be multiple highest values so it is not just the second index. Thus I had to add the number of the first index numbers that existed.
The code I produced looked as follows:

The line City = sorted(ScoreList, …..) is used because what needs to be done first is to sort the score list from biggest to smallest because this isn't directly possible in python what must be done is you have to make the list from smallest to biggest and reverse it. Using the key int as a way for python to know what is actually being sorted as you can either sort numbers or letters.

The first few lines of this code are just asking the user weather they wish to have the second option. From the IterScore = iter (ScoreList) these lines were very similar to when we I first selected the city. The only difference was the variable r is not equal to -1 as its trying to find the second index so we add the variable FirstNumber because using the count () function we discovered how many of the first number existed.
We then printed each value in options and after all of these options are printed we can then exit the code as both solutions have been offered to the user.
When testing if this worked I added what the complete score list is so I could see what the second highest option should have been each time. These were the test results:

I used the same test words as I did for the first choices. This first test was to see if multiple second choices would be printed. Comparing this to the score list I could see that this was as intended.

This also test if multiple outcomes can be printed and it all works as I want it to.



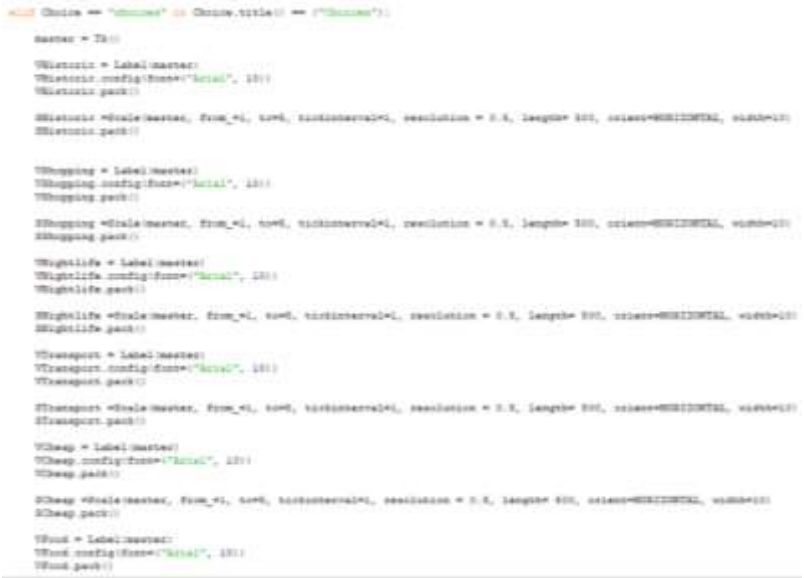
This test proved that only one value could be printed and the value it printed is the intended one.

This test was too see if one word would produce the correct result.

Seen as all my tests were successful I decided to move onto the second part of my code which is the menu choices.

For this I will also follow my flowchart however to start I have to use my menu design so create a blank template.

At the beginning of the program we are importing the module.

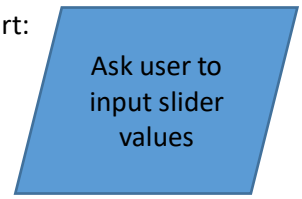


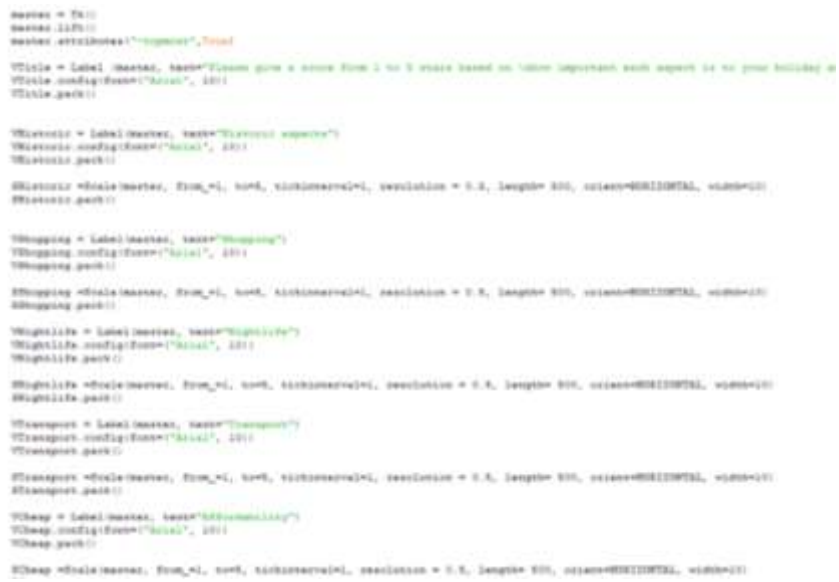
I used the pack method to implement each slider to the same window I could have also used a grid method however I opted for not using the grid as the sliders should be perfectly central on the window and packing them is the easiest way to do this.
I chose to make my slider go from 1-5 with intervals of 0.5 to represent a star system with half stars so that the program is more accurate.

This was the outcome of the program:

From this I decided to add labels so that I could see what each of the sliders meant and introduce the menu to the users.

This also follows the first step of the flowchart:





In labelling each slider I also decided to bring the window to the front of the user's computer screen as it can be irritating for the user to constantly have to click through to it.

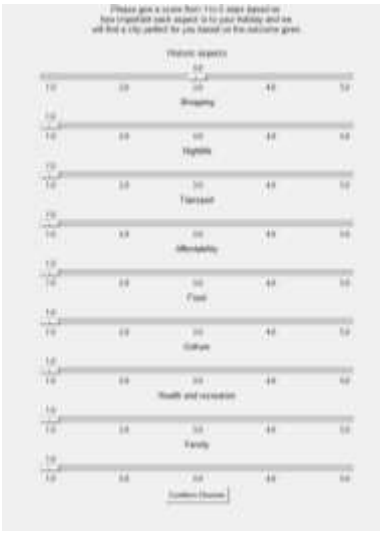
To then confirm this value I decided to create a button labelled confirm choice.

```
def get():
    HHistoric.get()
    HShopping.get()
    HNightlife.get()
    HTransport.get()
    HCheap.get()
    HFood.get()
    HCulture.get()
    HHealth.get()
    HFamily.get()
    master.destroy()
    return HHistoric, HShopping, HNightlife, HTransport, HCheap, HFood, HCulture, HHealth, HFamily

button = Button(master, text="Confirm Choices", command=get)
button.pack()
```

This was the method I tried to get the values. The problem I encountered with this was nothing and the values of each attribute were not a 1-5 number.

This prompted me to do some further reading on tkinter and the solution I found was to run my code as the function with the button. This would mean I could have a very similar code as I have for my description part. The reason this would work is that the issue I was having before was how I could make it so I knew what the current value of the slider is as once the window is closed these sliders get random values. My method for doing this involved running the code in the button before the button closed the code.

To start with I had to test that this method would actually work. To test this I wanted to see if I changed a value on the slider and then clicked confirm choices if the value output would be the value I put on the slider.



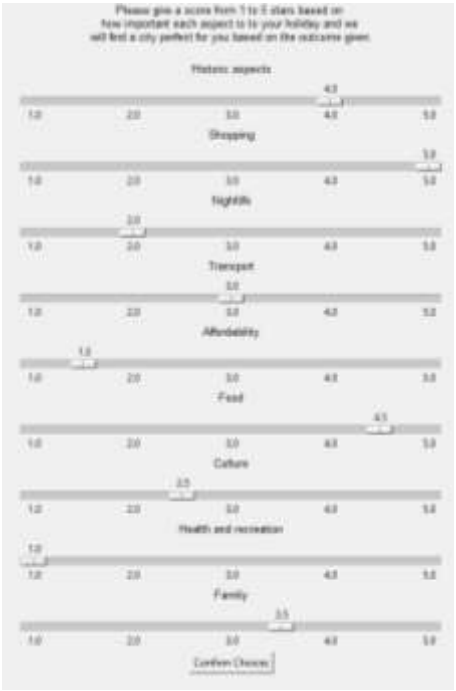
This proves that the values obtained are correct so I can now manipulate them.
The first step was to create an ordered list based off the highest numbers in the list. To do this I needed to create a list that changed based on the changes made in the first list.

Create a list using slider values

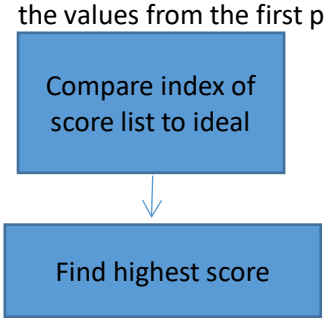
This was the code I used to do that.
The inspiration for how I did that came from a user on stack overflow who created this code:

```
list1 = [3,2,4,1, 1]
list2 = ['three', 'two', 'four', 'one', 'one2']

sortedRes = sorted(zip(list1, list2), key=lambda x: x[0]) # use 0 or 1 depending on what you
>>> [(1, 'one'), (1, 'one2'), (2, 'two'), (3, 'three'), (4, 'four')]
```

```
This is a holiday comparison code which will choose a holiday destination city around the world based on either an i
nput description or a list of set choices
do you wish to write a description or choose from a list of set choices (input description or choices to make the de
cision):choices
['SHOPPING', 'FOOD', 'HISTORIC', 'FAMILY', 'TRANSPORT', 'CULTURE', 'NIGHTLIFE', 'CHEAP', 'HEALTH']
```

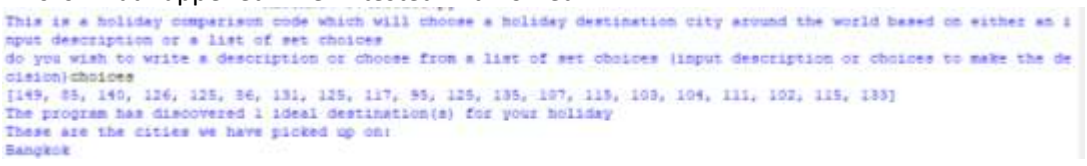


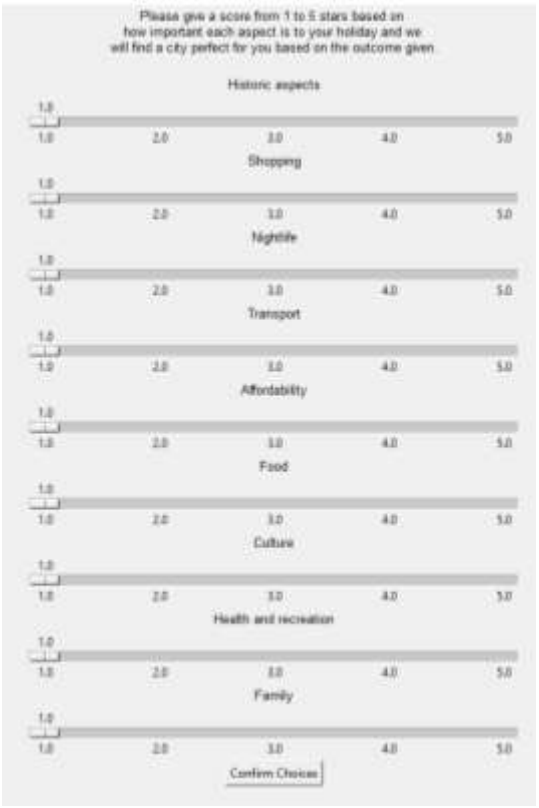
From the result you can see that the list created was ordered based off the values I input. This then meant I could use the code I created from the first part for finding a score and comparing and just adapt it to fit the new numbers we are working with. This will work as both parts are trying to reach the same end goal and we now have our menu values in the same format as when we manipulated the values from the first part.



This is the code we will be using for the finding of the city. As you can see it is exactly the same other than we are working with different variable names.

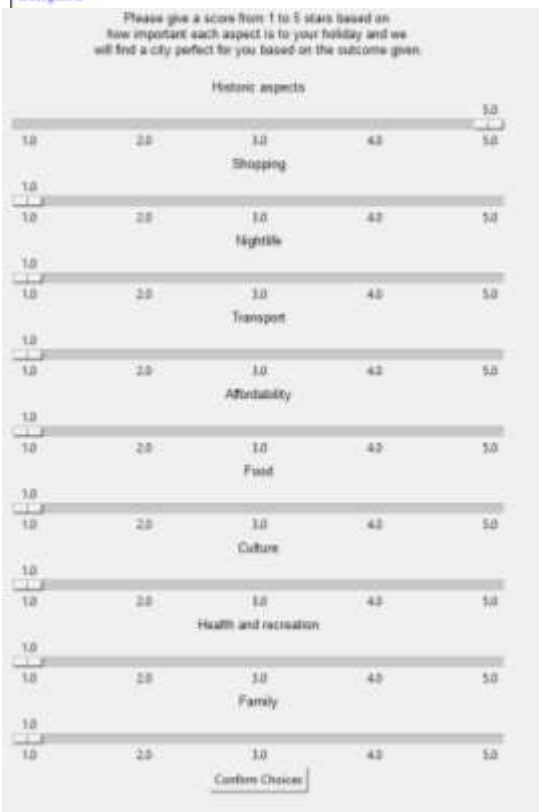
This is what happened when I tested if it worked:





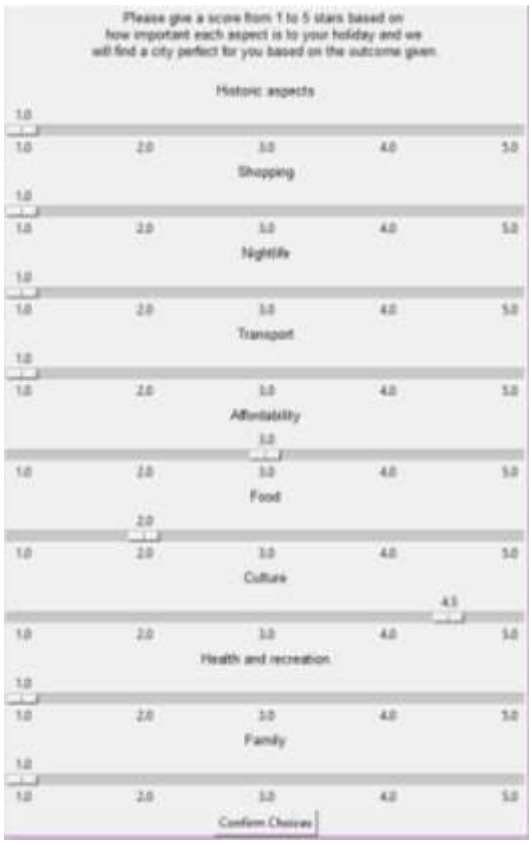
I kept the score on to see if the city the program was providing was the correct one.
In this case the city provided matched the input data.

```
This is a holiday comparison code which will choose a holiday destination city around the world based on either an
nput description or a list of set choices
do you wish to write a description or choose from a list of set choices (input description or choices to make the
cision) choices
[155, 100, 149, 137, 133, 104, 121, 117, 117, 104, 131, 122, 99, 119, 118, 118, 103, 107, 128, 137]
The program has discovered 1 ideal destination(s) for your holiday
These are the cities we have picked up on:
Bangkok
```



The reason putting historic to the top and leaving all the other attributes at 1 doesn't just give historic as the score is because this code makes a list out of all of the sliders this is why making a description is more accurate for the user but this method is faster for them. This is why I give the user a choice at the beginning.

```
This is a holiday comparison code which will choose a holiday destination city around the world based on either an i
nput description or a list of set choices
do you wish to write a description or choose from a list of set choices (input description or choices to make the de
cision)choices
[95, 115, 110, 132, 107, 131, 95, 95, 130, 131, 111, 137, 137, 121, 119, 118, 133, 134, 91, 79]
The program has discovered 2 ideal destination(s) for your holiday
These are the cities we have picked up on:
Kuala_Lumpur
Hong_Kong
```

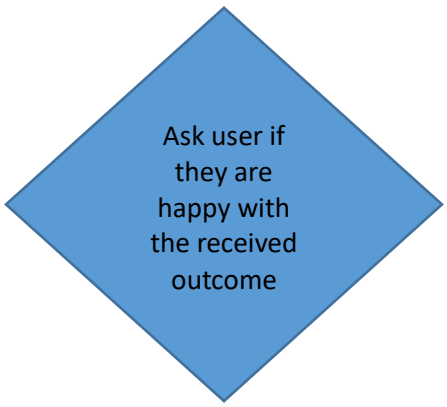


This test here shows you can get 2 results as when two scores are equal it still prints 2 results.

```
This is a holiday comparison code which will choose a holiday destination city around the world based on either an i
nput description or a list of set choices
do you wish to write a description or choose from a list of set choices (input description or choices to make the de
cision)choices
[88, 101, 89, 114, 76, 118, 100, 109, 119, 103, 72, 97, 103, 83, 91, 89, 101, 100, 92, 89]
The program has discovered 1 ideal destination(s) for your holiday
These are the cities we have picked up on:
Tokyo
```

This was a test to show that it is not just giving the same cities each time and the different attributes are effecting the chosen city.

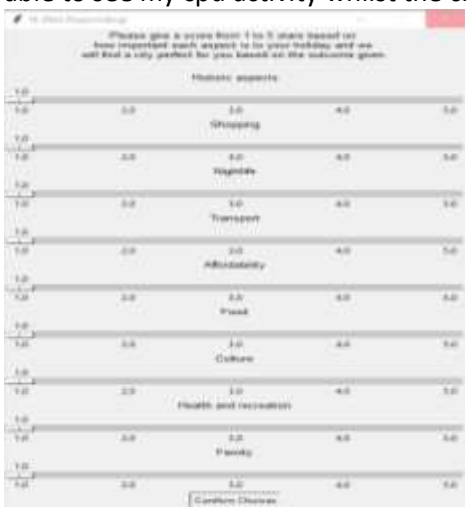
With all of these tests being successful we could then move on to asking the user if they liked their first choice or not and then giving them a second choice if they said no as we did with the first part.



To do this I took my code from the first part and changed it to fit the variables used in the second.

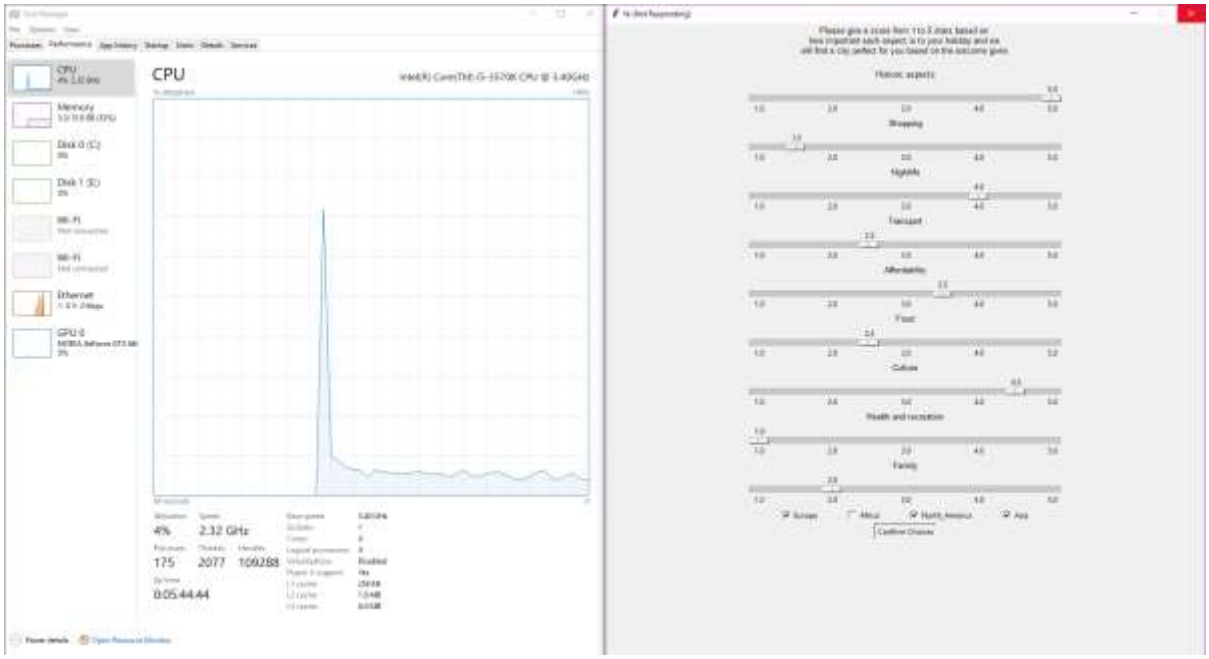
```
def Ask():
    Result = input("Are you happy with the result? If you're not we can try and provide you with a second option if available.")
    if Result == "yes" or Result.title() == "Yes":
        print("Please enjoy your holiday.")
        exit()

    elif Result == "no" or Result.title() == "No":

        print("We will offer the next option we have picked up on")
        CheckForce = len(ForceList)


        City = sorted(ForceList, key=str, reverse=True)
        FirstNumber = ForceList.count(City[0])
        at = FirstNumber
        Options = []
        for ae in range(0, len(ForceList)):
            if ForceList[ae] == City[ae]:
                Options.append(FirstCities[ae])


        print("The program has discovered", len(Options)-FirstNumber, "ideal destinations for your holiday so its second one through")
        print("These are the cities we have picked up on.")
        for af in range (len (Options)):
            print(Options[af])
        print("Please enjoy your holiday. We hope this mode could be helpful.")
        exit()


    else:
        print("We could not understand the given answer please say yes or no")
        Ask()
Ask()
```

Through testing I discovered that occasionally if your computer doesn't have a good enough cpu or you are running too many tasks in the background the program can often crash. I believe this may be the cause of the crash but it is difficult to tell as the crashes seem to be random and I haven't been able to see my cpu activity whilst the crash occurs.

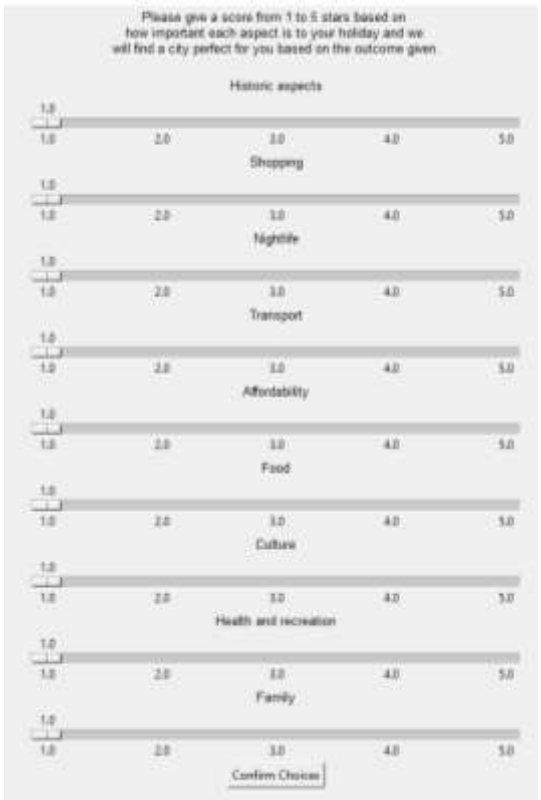


This is an example of the crash.

After this I tried to find the cause of the crash by running my computers task manager at the same time to see if I could see a spike in cpu usage

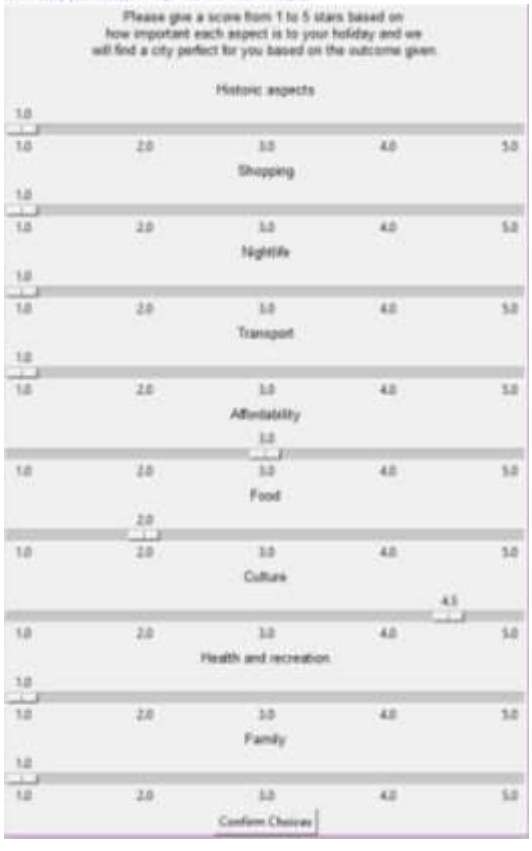


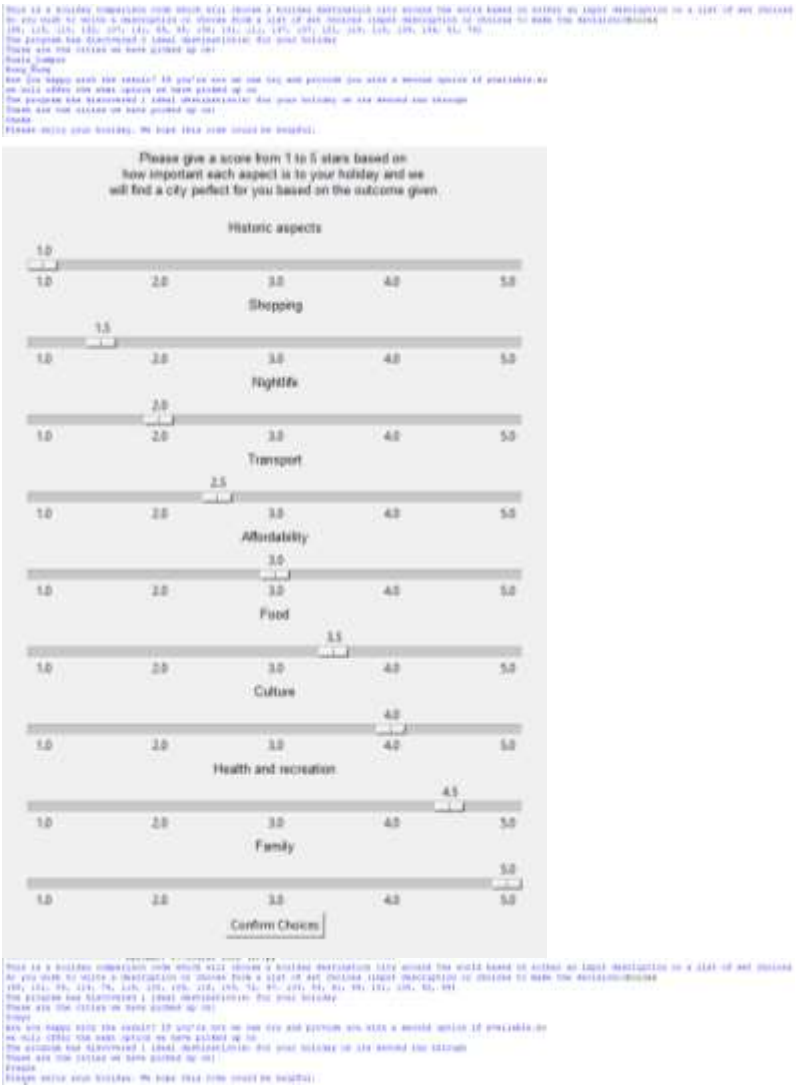
The results shown found that the cpu only spiked when opening the menu and memory usage was constantly at 33% I couldn't find the cause of the crashes due to there being so many reasons possible. This impacts overall usability however the functionality of the program is unchanged as once this crash occurs even if the window is closed a result is still given with this result being the expected value received every time.



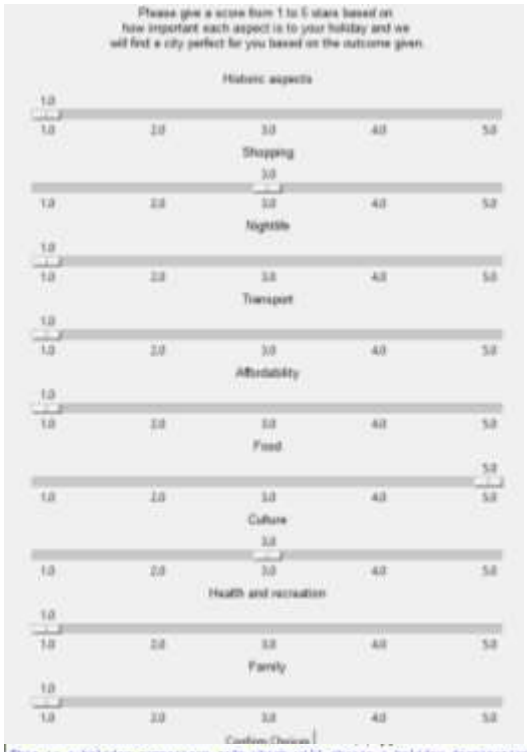
I used all of the same tests to see if it would work saying no for each one


Please give a score from 1 to 5 stars based on
how important each aspect is to your holiday and we
will find a city perfect for you based on the outcome given.
Historic aspects
5.0
1.0 2.0 3.0 4.0 5.0
Shopping
1.0
1.0 2.0 3.0 4.0 5.0
Nightlife
1.0
1.0 2.0 3.0 4.0 5.0
Transport
1.0
1.0 2.0 3.0 4.0 5.0
Affordability
1.0
1.0 2.0 3.0 4.0 5.0
Food
1.0
1.0 2.0 3.0 4.0 5.0
Culture
1.0
1.0 2.0 3.0 4.0 5.0
Health and recreation
1.0
1.0 2.0 3.0 4.0 5.0
Family
1.0
1.0 2.0 3.0 4.0 5.0
Confirm Choices


Please give a score from 1 to 5 stars based on
how important each aspect is to your holiday and we
will find a city perfect for you based on the outcome given.
Historic aspects
1.0
1.0 2.0 3.0 4.0 5.0
Shopping
1.0
1.0 2.0 3.0 4.0 5.0
Nightlife
1.0
1.0 2.0 3.0 4.0 5.0
Transport
1.0
1.0 2.0 3.0 4.0 5.0
Affordability
3.0
1.0 2.0 3.0 4.0 5.0
Food
2.0
1.0 2.0 3.0 4.0 5.0
Culture
4.5
1.0 2.0 3.0 4.0 5.0
Health and recreation
1.0
1.0 2.0 3.0 4.0 5.0
Family
1.0
1.0 2.0 3.0 4.0 5.0
Confirm Choices



From all of these test I could see that none of them had two second results so to make sure it was working for that as well I made another test.

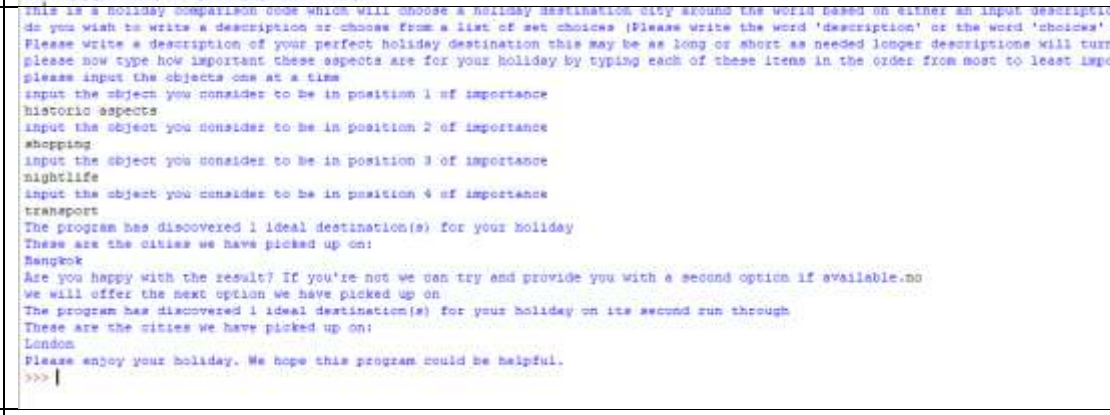
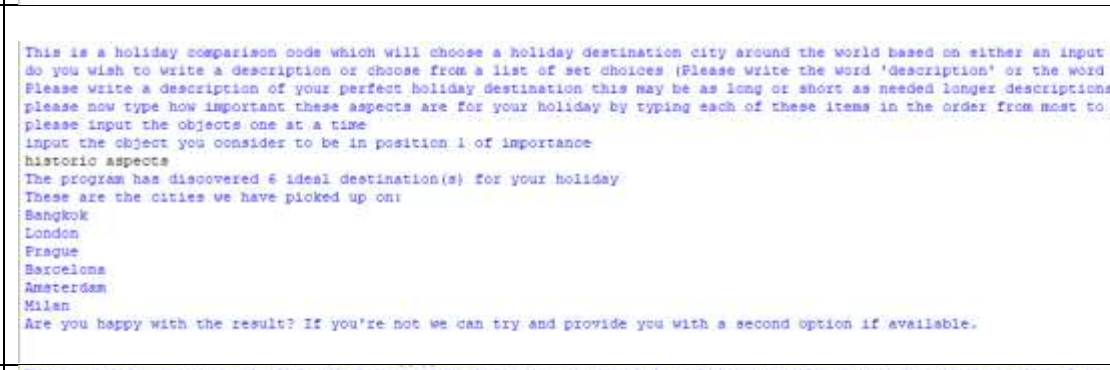
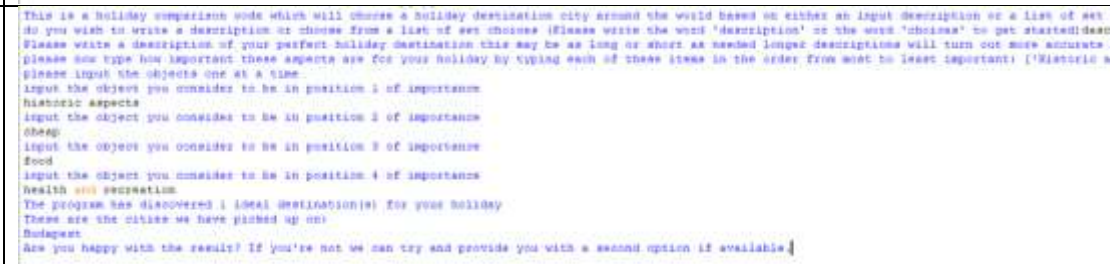
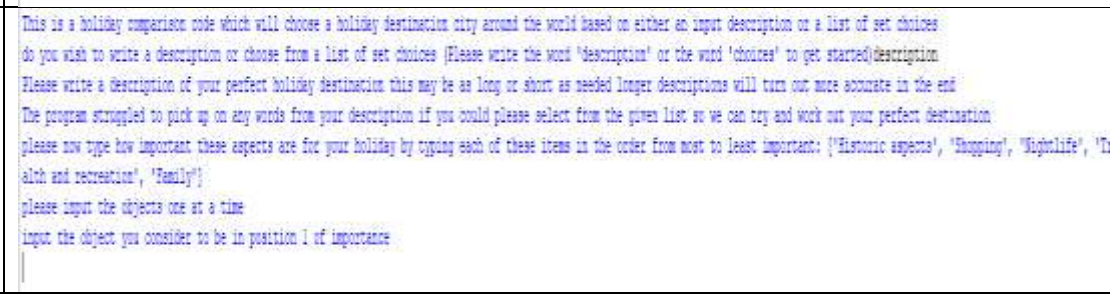



```
This is a holiday comparison code which will choose a holiday destination city around the world based on either an i
nput description or a list of set choices
do you wish to write a description or choose from a list of set choices (input description or choices to make the de
cision)choices
[120, 89, 131, 149, 139, 101, 126, 127, 158, 136, 129, 121, 151, 131, 130, 124, 155, 151, 100, 107]
The program has discovered 2 ideal destination(s) for your holiday
These are the cities we have picked up on!
Tokyo
Milan
Are you happy with the result? If you're not we can try and provide you with a second option if available.no
we will offer the next option we have picked up on
The program has discovered 2 ideal destination(s) for your holiday on its second run through
These are the cities we have picked up on:
Hong_Kong
Osaka
Please enjoy your holiday. We hope this code could be helpful.
```

This result proved that it works for showing two results.

With all of this testing the core code now works any changes made now are all for user experience. This will include adding any extra words and switching some of the wording of the program to suit a better user experience.

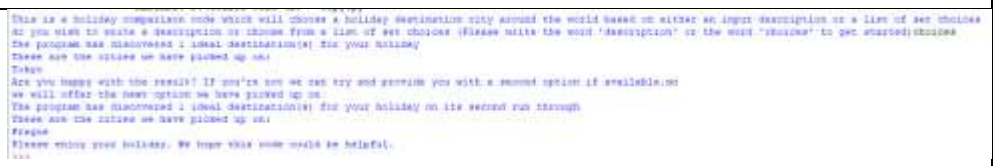
Test table results (Description)

Test Number	Input	Actual results
1	Historic being the only feature selected	
2	Historic, Shopping, Nightlife, Transport	
3	Historic being only feature selected with the user selecting no after options given	
4	Historic, Cheap , Food, Health	
5	No keywords given	

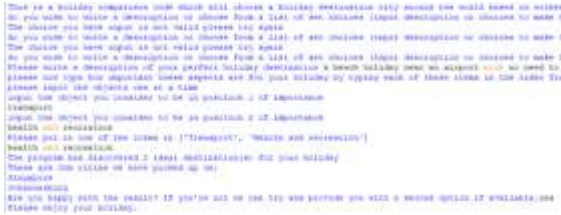
6	Wrong word put in when selecting	This is a holiday comparison code which will choose a holiday destination city around the world based on do you wish to write a description or choose from a list of set choices (Please write the word 'descript Please write a description of your perfect holiday destination this may be as long or short as needed lo please now type how important these aspects are for your holiday by typing each of these items in the or please input the objects one at a time input the object you consider to be in position 1 of importance shopping Please put in one of the items in ['Historic aspects']
7	Not putting in the words either choices or description	This is a holiday comparison code which will choose a holiday destination city around the do you wish to write a description or choose from a list of set choices (Please write the The choice you have input is not valid please try again do you wish to write a description or choose from a list of set choices (Please write the

Test table results (choices)

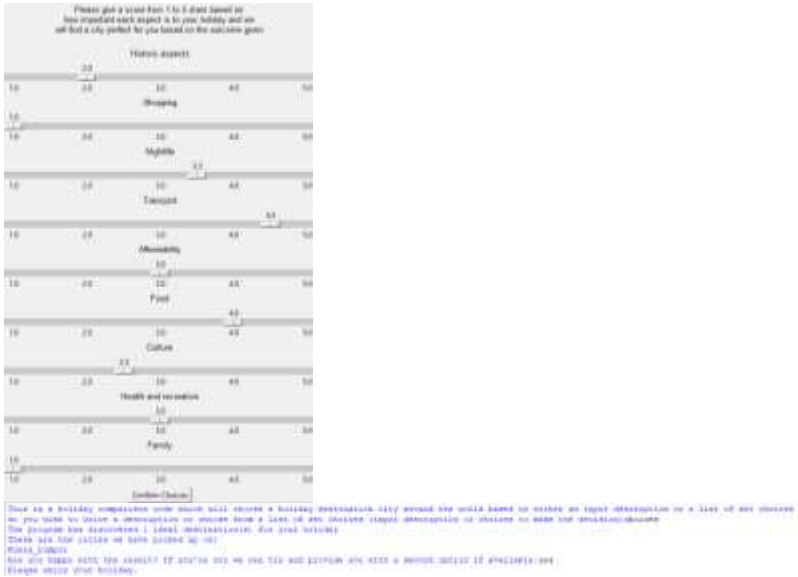
Test Number	Input	Actual output
1	Historic bar pulled to 5 stars	This is a holiday comparison code which will choose a holiday destination city around the world based on either do you wish to write a description or choose from a list of set choices (Please write the word 'description' or The program has discovered 1 ideal destination(s) for your holiday These are the cities we have picked up on: Bangkok Are you happy with the result? If you're not we can try and provide you with a second option if available.
2	Culture set to 5 shopping set to 4 Food and historic on 3 and nightlife set to 2	This is a holiday comparison code which will choose a holiday destination city around the world based on either an in do you wish to write a description or choose from a list of set choices (Please write the word 'description' or the w The program has discovered 2 ideal destination(s) for your holiday These are the cities we have picked up on: London Istanbul Are you happy with the result? If you're not we can try and provide you with a second option if available.
3	None of the scores changed	This is a holiday comparison code which will choose a holiday destination city around the world based on either an input description or a list of set choices do you wish to write a description or choose from a list of set choices (Please write the word 'description' or the word 'choices' to get started)choices The program has discovered 1 ideal destination(s) for your holiday These are the cities we have picked up on: Bangkok Are you happy with the result? If you're not we can try and provide you with a second option if available.
4	Each bar from smallest to biggest (histor	This is a holiday comparison code which will choose a holiday destination city around the world based on either an input description or a list of set choices do you wish to write a description or choose from a list of set choices (Please write the word 'description' or the word 'choices' to get started)choices The program has discovered 1 ideal destination(s) for your holiday These are the cities we have picked up on: Tokyo

	ic smallest health and recreation biggest)	
5	Saying no to the previous test	

After completing this iteration of the code I then proceeded to ask some members of the public and people that were available to test my code and give feedback. Their inputs where as follows:



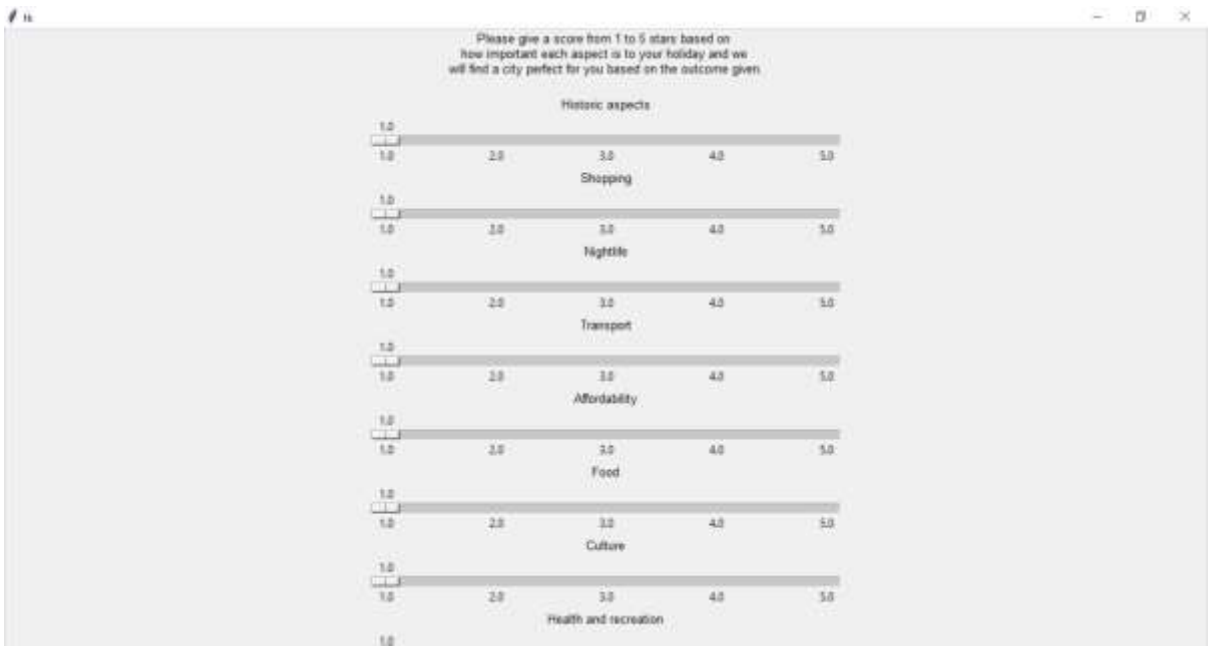
This first user gave the feedback that I should include different regions. This would mean if the user asked for a city in Europe the program would only select those cities found in Europe.



This user helped show some users will actually use the choices feature when available for less time typing.

From this last user what could be determined was that users do wish for the region selection to be a possibility.

All of my testing previously had been done on a tower pc so there was an issue that I hadn't found yet. On certain computers primarily laptops the screen size isn't big enough to fit all of the sliders and the button. The solution to solve this problem would be to create a scroll bar however a problem with this is you cannot scroll through packed widgets. Seen as all of my widgets are packed this is not solvable. The other solution could be to find the screen size of the computer being run on and change the details depending on that. This is an issue that due to time constraints I cannot solve so any testing involving the choices aspect of the program will have to be done on a tower pc. With the final product this may not be a problem as it would eventually be uploaded onto a website were scrolling would be possible but for the app on phones if that were to be made this would have to be changed.



As can be seen from the printed screenshot this is what the user can see on certain laptops and any information below this point is un-clickable. It is also possible that there may be a solution of some kind that I don't know about as I am not vastly experienced with using tkinter and many of my solutions for problems encountered are research based.
For example my biggest problem found so far has been finding a method of receiving slider values it

was after research that I found the .get() method however this was still not working for me. It was then that I discovered that buttons could run functions and I found my solution.

Test Table (Description)

Test number	Input	Expected output	Reasoning
1	Test with Europe and historic	London,Prague, Barcelona,Amsterdam, Moscow	This is all the places which have historic as the first attribute being in Europe.
2	Test with Africa and historic with no being said afterwards	Johannesburg should be the first city given with the program informing the user of the state of the program after.	I will be programing a response for if there are no cities left
3	The test for north America should be the same as the test for Africa with new York substituted.		
4	Test with Asia and historic	Bangkok	This is the only Asian city with historic as the first aspect.
5	Just Europe selected with no attributes selected	Should ask for an order and whichever item is put first all European cities with that attribute should be printed	Just selecting the region should cut down the number of available cities but should still ask for all attributes to be ordered.

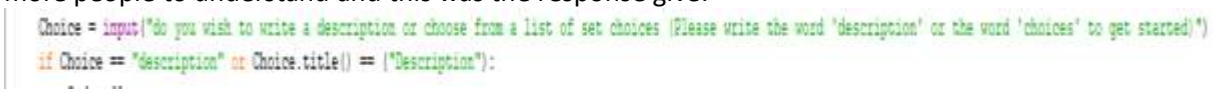
Test Table (Choices)

Test number	Input	Expected output	Reasoning
1	Europe selected without changing any attributes	London	This is because London is a European city which in the program before would be the second outcome after Bangkok which isn't a European city.
2	Africa selected with no being said after	Should print Johannesburg with users being informed afterwards	As with the first part I will be programming a response which will only be needed in this version due to a lack of cities.
3	Find an input which will print multiple outputs for the first	Should do exactly as intended	This will test if multiple region options can be printed

	option		
4	Find an input which will print multiple outputs for the second option	Should do exactly as intended	This will test if multiple region options can be printed with no being said to see if my program is now restricting anything
5	All regions selected	Should print as if no regions are selected	All items will be added so nothing will be different

Software Development and testing Iteration 2

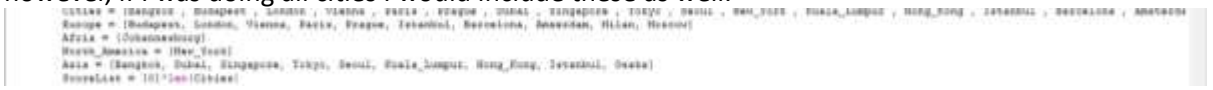
From the user feedback I received after the testing in iteration one it seemed that I could make some of the wording clearer to begin the code I asked the users how they would like it phrased as to allow more people to understand and this was the response give:



This led me to make another wording change with the description instructions and just clarify that the description can be any length:

This was the first thing I began to work on in the second iteration. I would first sort this out by creating a list named by the region. Within this list it would contain each city that is from that region. The regions will be the different continents: Europe, Africa, Asia, North America, South America, Oceania This excludes the continent Antarctica due to this not being a holiday destination.

When making the lists I found that I do not have any Oceanic or South American cities in my sample however, if I was doing all cities I would include these as well.



Next what I had to do was when picking up from the description I had to search for the region words those being (Europe, North America, Asia and Africa)
The testing for if this works will have to be done on the European and Asian continents as north America and Africa only have 1 city each from my sample list.

With these lists what I first tried to do was make cities equal to the variables of the different lists as follows:

This didn't do anything and would still print cities not from Europe.

The next thing I tried was to change the variable cities:

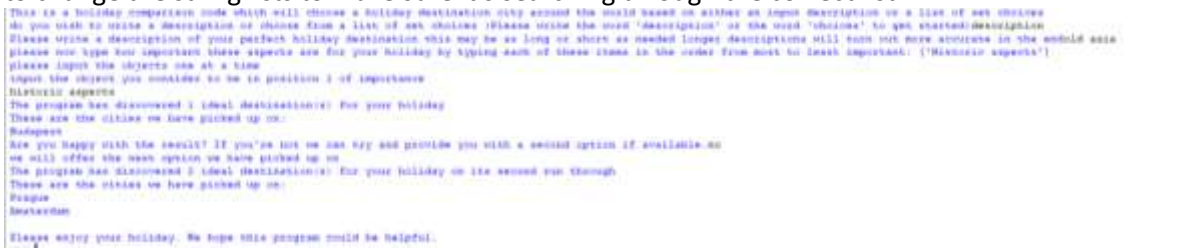
```
if ("Europe") or ("europe") in Description:
    Cities = [Budapest, London, Vienna, Paris, Prague, Istanbul, Barcelona, Amsterdam, Milan, Moscow]
if ("Africa") or ("africa") in Description:
    Cities = [Johannesburg]
if ("North America") or ("north america") or ("North america") or ("north America") in Description:
    Cities = [New_York]
if ("Asia") or ("asia") in Description:
    Cities = [Bangkok, Dubai, Singapore, Tokyo, Seoul, Kuala_Lumpur, Hong_Kong, Istanbul, Osaka]
```

When tested with Europe this gave all results that had a location in Europe:

To ensure this had actually worked, I tested the program with another region:

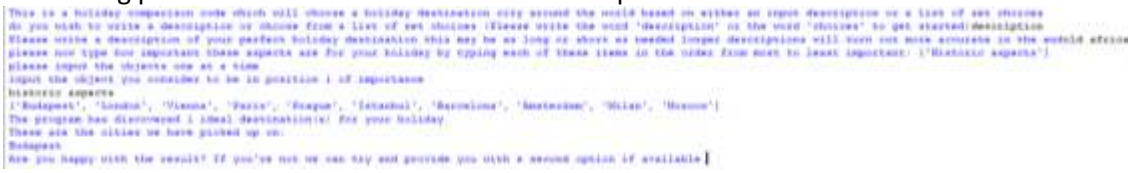
Seen as this gave the same result this told me that it had not worked. The fact it also changed the regular value of what would appear if just historic aspects was input would appear there is something else that needs to be changed.

After going through the code I quickly realised it was searching through the old cities list so I needed to change the string lists to make sure it's searching through the correct list.



The results clearly show cities not in Asian cities. This showed there was a problem with the code somewhere.

To find out where the issue was I would print what StrCities was after it should have been changed. For Europe the correct cities list was printed however, when I used African cities as an example the cities being printed were the exact same as the Europe list.



When testing without a region I found that the European list was still printed:

By checking if it was true the Europe was in the description list and all the other regions where as well I found that all of these results were returning true. This led me to change the method I was using to select the region.

The method I would be using would be using would be similar to ones I have used in the past with the searching algorithm where I could create a list of keywords and use that.

This is the list of key words I would be searching for. This is incomplete and isn't as long as intended however, I cannot thing of all the words that would be used to describe these regions.

This searches through each region keyword list for the length of the keyword list and checks if the words found are in the description. If they are found in the description it checks that the region is not already there and then appends the region name to a list called regions.

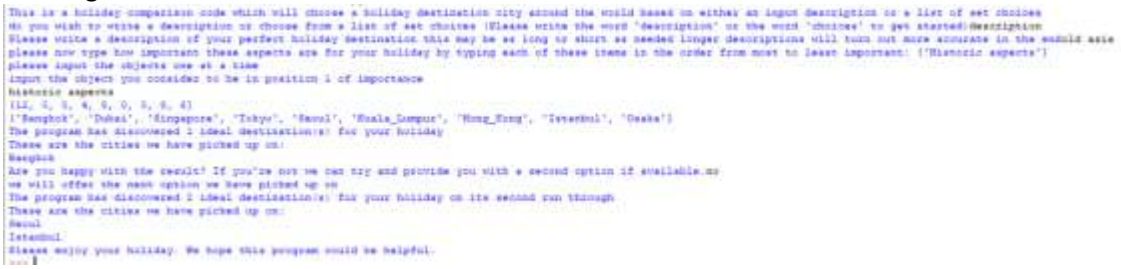
The code will then check if the region is in the regions list and if it is it will change the value of the list cities to equal only those cities within the specific region.

I then tested if this method had worked and the results were as follows:

This initial test was to see if the values for Europe were correct and being searched for.

I then tested for Africa and an error occurred as because the list is only of length 1 city there can be no second option available.

North America had the same problem. This prompted me to fix this by creating an if statement saying if the length of the cities found is 0 then we should apologise and say no secondary cities could be found. If more cities were included this problem wouldn't be encountered as there would be enough cities in the world but this will be the solution for the proof of concept I am currently building.



This final test was to see if it would search for multiple cities in the Asia list as well.

This shows that all the regions work to some extent. For the final testing of this section I will test with other attributes just to ensure it works with other values as well.

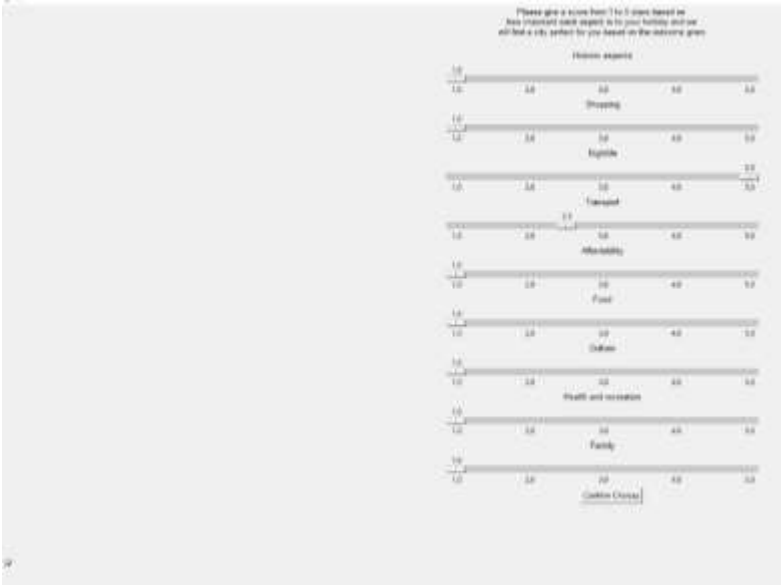
This is the code for testing if the second value is possible to find. This will only be applicable to the city of New York and Johannesburg when searched with region as these cities are only a list of size one.

The code functions as intended and exits before the error occurs.

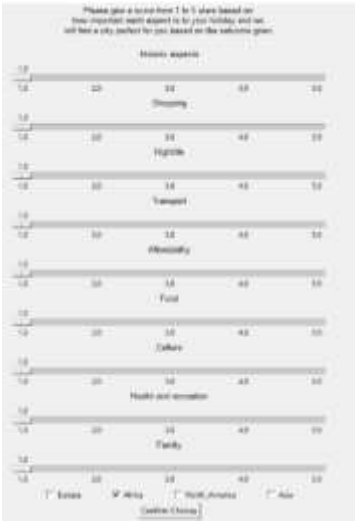
The next thing to do was I needed to add regions on to the choices section. I could do this by making the choices options be buttons contained at the side of the screen.

```
frame = Frame(width=800, height=600,)
frame.pack(side=TOP)

AAEurope = Checkbutton(frame, text = "Europe")
AAEurope.pack(side = LEFT, padx = 20)

ALAfrica = Checkbutton(frame, text = "Africa")
ALAfrica.pack(side = LEFT, padx = 20)

AANorth_America = Checkbutton(frame, text = "North_America")
AANorth_America.pack(side = LEFT, padx = 20)

AAAsia = Checkbutton(frame, text = "Asia")
AAAsia.pack(side = LEFT, padx = 20)
```

This is the code I used to create the buttons. Initially I did not have the frame line and all buttons were attached to master so when I padded the buttons to a particular side they appeared as shown below with them being very small and to the side.



After adding the frame the buttons now appear centred and on line with each other above the confirm choices button.



The pixel padding I chose to be 20. This was decided through trial and error as all the other looked either too small or too large whereas this look just right.

Currently these buttons don't actually conduct a task so pressing the confirm choices button will still serve the same purpose as before.
In order to make the check boxes do something what I need to do is implement them into the button I need to check the state they are in when the confirm choices button is pressed (that being ticked or

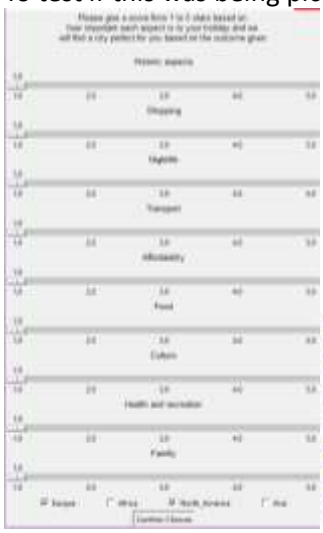
not) those that are ticked need to be the cities in the city list and those that aren't need to be discarded. This allows for implementation of the region selection into both parts of the holiday selection program.

```
ALEurope = Checkbutton(frame, text = "Europe", onvalue = 1, offvalue = 0, variable = EuropeVar)
ALEurope.pack(side = LEFT, padx = 20)

ALAfrica = Checkbutton(frame, text = "Africa", onvalue = 1, offvalue = 0, variable = AfricaVar)
ALAfrica.pack(side = LEFT, padx = 20)

ALNorth_America = Checkbutton(frame, text = "North America", onvalue = 1, offvalue = 0, variable = North_AmericaVar)
ALNorth_America.pack(side = LEFT, padx = 20)

ALAsia = Checkbutton(frame, text = "Asia", onvalue = 1, offvalue = 0, variable = AsiaVar)
ALAsia.pack(side = LEFT, padx = 20)
```

To give the check boxes a value I had to give them a value of 1 when on and 0 when off. To get the current state of the check boxes in tkinter what must be done is a variable must be set to the check box

```
print(ALEurope)
print(ALAfrica)
print(ALNorth_America)
print(ALAsia)
```

To test if this was being picked up on, I printed each one out.



```
This is a holiday comparison code which will choose a holiday destination city a
round the world based on either an input description or a list of set choices
do you wish to write a description or choose from a list of set choices (Please
write the word 'description' or the word 'choices' to get started)choices
1
0
1
0
The program has discovered 1 ideal destination(s) for your holiday
These are the cities we have picked up on:
Bangkok
Are you happy with the result? If you're not we can try and provide you with a s
econd option if available.
```

With the test on the left, the result on the right was what was printed. This was printed as intended so I knew that with that I could now manipulate the values and use them to change the number of available cities.

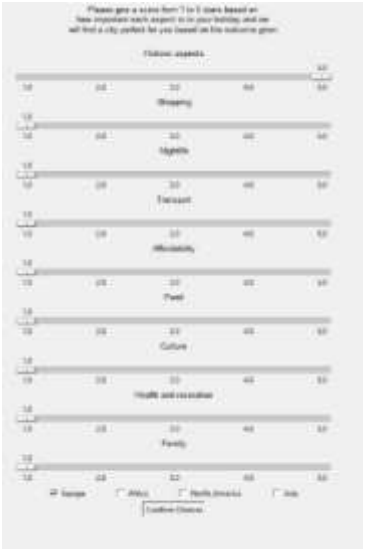
If the value is equal to one that means the check box is ticked. This means I can then change the value of cities of each region based on the indicated check box.

As with the descriptions part of the code what I had to change the values of the string cities too with the same changes to the cities list.

```
City = sorted(ScoreList,key=int, reverse=True)
FirstNumber = ScoreList.count(City[0])
ad = FirstNumber
Options = []
if FirstNumber >= len(ScoreList):
    print("We could not find any secondary cities. This program is a work in progress and does not contain all cities just yet.")
    exit()
for ae in range(0, len(ScoreList)):
```

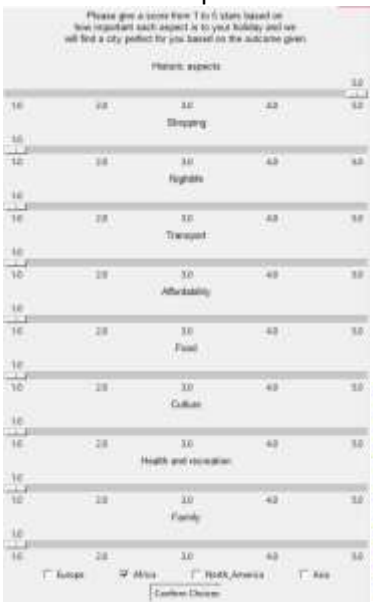
Because there is only one city in both the North America list and Africa list I need to make sure if the user is saying no to the provided list they are informed of the state of the program. In the real program this would not be included as more cities would be available from each region so it wouldn't be possible for the program to not find a second choice.

To test if the code had worked I conducted the same tests as I made with the descriptions part by testing each region individually with the historic option. It is not needed to test the program with the other search options as I already know it is searching correctly I just need to ensure it searches through the correct region list.



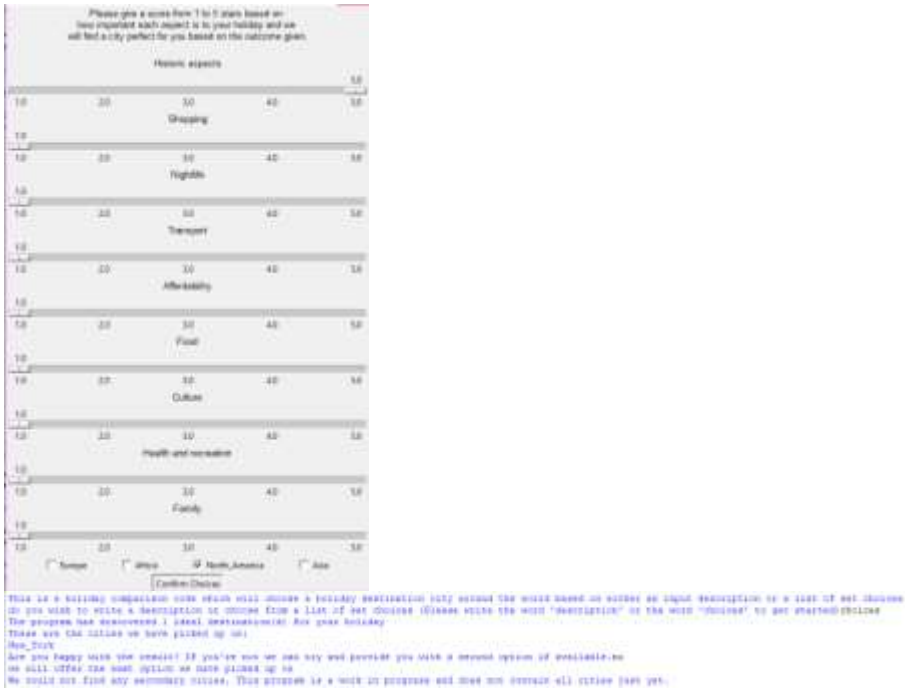
```
This is a holiday comparison code which will choose a holiday destination city a
round the world based on either an input description or a list of set choices
do you wish to write a description or choose from a list of set choices (Please
write the word 'description' or the word 'choices' to get started)choices
The program has discovered 1 ideal destination(s) for your holiday
These are the cities we have picked up on:
London
Are you happy with the result? If you're not we can try and provide you with a s
econd option if available,no
we will offer the next option we have picked up on
The program has discovered 1 ideal destination(s) for your holiday on its second
 run through
These are the cities we have picked up on:
Vienna
Please enjoy your holiday. We hope this code could be helpful.
```

The test for Europe worked as intended.

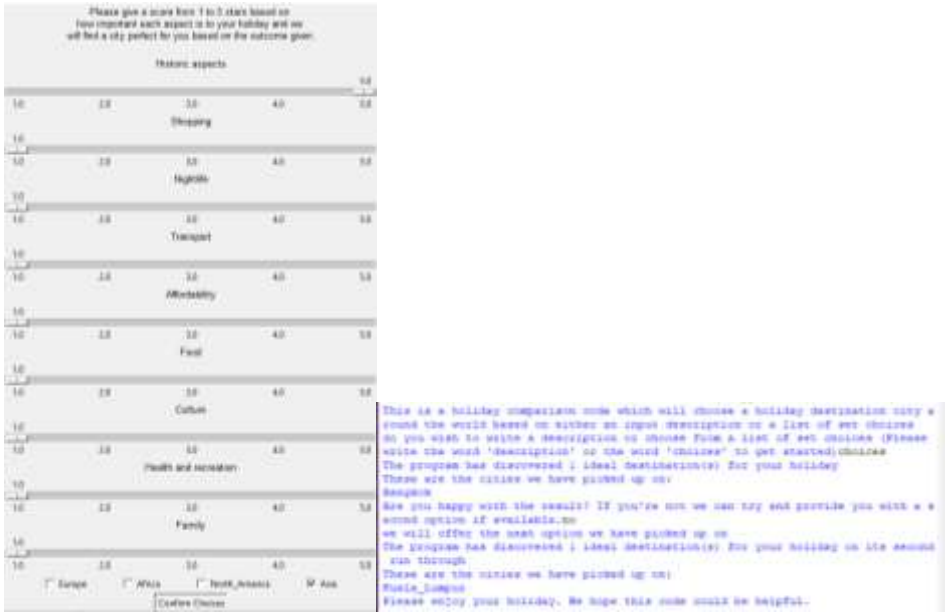


```
This is a holiday comparison code which will choose a holiday destination city a
round the world based on either an input description or a list of set choices
do you wish to write a description or choose from a list of set choices (Please
write the word 'description' or the word 'choices' to get started)choices
The program has discovered 1 ideal destination(s) for your holiday
These are the cities we have picked up on:
Johannesburg
Are you happy with the result? If you're not we can try and provide you with a s
econd option if available,no
we will offer the next option we have picked up on
We could not find any secondary cities. This program is a work in progress and d
oes not contain all cities just yet.
```

The test for Africa worked as intended



The test for North America worked as intended



The test for Asia worked as intended this showed that region selection was working.

After this I decided to test to see what would happen if multiple regions were selected for both the descriptions part and the menu.

When both regions of Europe and Asia were input a logical error occurred. This happened because my program is done via if and elif statements so only the first city list will be selected so the asia list never would. To sort this problem what I would have to do is append each city to the cities list based

on the region so that more can be added.

```
Cities= []
if "Europe" in Regions:
    City = [Budapest, London, Vienna, Paris, Prague, Istanbul, Barcelona, Amsterdam, Milan, Moscow]
    for AH in range(len(City)):
        Cities.append(City[AH])
if "Africa" in Regions:
    Cities.append(Johannesburg)
if "North America" in Regions:
    Cities.append(New_York)
if "Asia" in Regions:
    City = [Bangkok, Dubai, Singapore, Tokyo, Seoul, Kuala_Lumpur, Hong_Kong, Istanbul, Osaka]
    for AH in range(len(City)):
        Cities.append(City[AH])
if "Europe" and "Africa" and "North America" and "Asia" not in Regions:
    Cities = [Bangkok , Budapest , London , Vienna , Paris , Prague , Dubai , Singapore , Tokyo , Seoul , New_York , Kuala_Lumpur , Hong_Kong , Istanbul , Barcelona , Amst
```

To make it work as intended and append the statements as needed I created loops to append the cities individually based on the loop number.

```
StrCities = []
if "Europe" in Regions:
    StrCity = ["Budapest", "London", "Vienna", "Paris", "Prague", "Istanbul", "Barcelona", "Amsterdam", "Milan", "Moscow"]
    for AH in range(len(StrCity)):
        StrCities.append(StrCity[AH])
if "Africa" in Regions:
    StrCities.append ("Johannesburg")
if "North America" in Regions:
    StrCities.append("New_York")
if "Asia" in Regions:
    StrCity = ["Bangkok", "Dubai", "Singapore", "Tokyo", "Seoul", "Kuala_Lumpur", "Hong_Kong", "Istanbul", "Osaka"]
    for AH in range(len(StrCity)):
        StrCities.append(StrCity[AH])
if "Europe" and "Africa" and "North America" and "Asia" not in Regions:
    StrCities = ["Bangkok", "Budapest" , "London" , "Vienna" , "Paris", "Prague", "Dubai", "Singapore", "Tokyo","Seoul", "New_York" , "Kuala_Lumpur", "Hong_Kong","Istanbul
```

I had to make sure the string values were the same or they wouldn't be attached correctly.

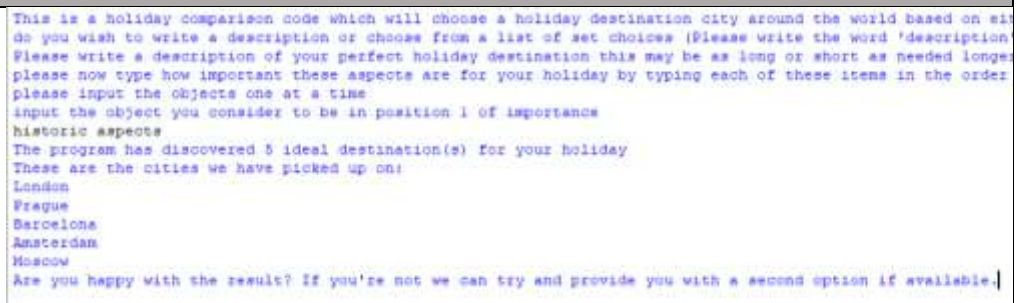
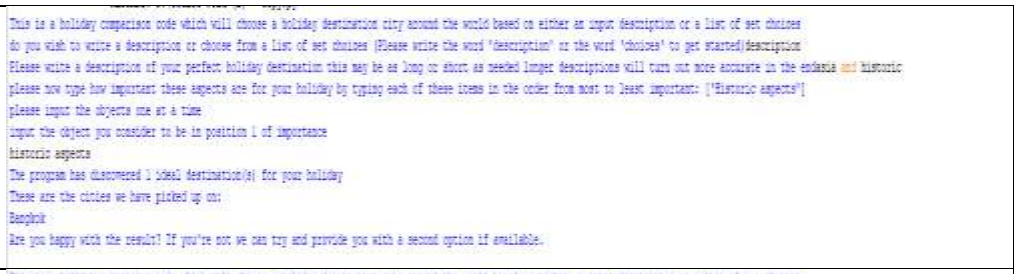
I then had to test if the values being used now are correct just in case I had reordered one of the lists. This shouldn't have happened as I appended by the index of the other list but I just have to make precautions.

The tests I made were according to the descriptions test table as this should be the final issue of this section of the code.

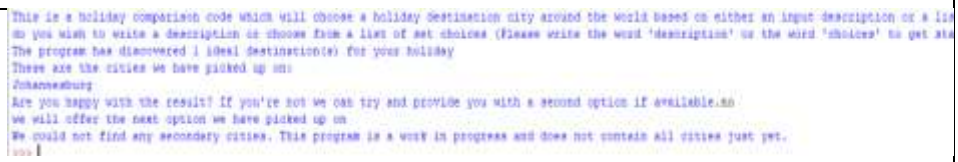
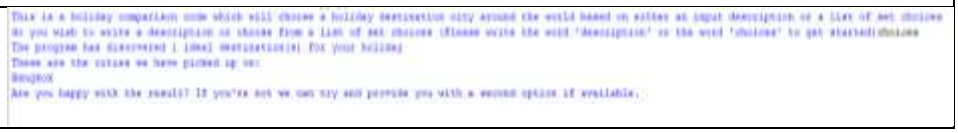
```
Cities = []
if AIEurope == 1:
    City = [Budapest, London, Vienna, Paris, Prague, Istanbul, Barcelona, Amsterdam, Milan, Moscow]
    for AH in range(len(City)):
        Cities.append(City[AH])

if AIAfrica == 1:
    Cities.append(Johannesburg)

if AINorth_America == 1:
    Cities.append(New_York)

if AIAsia == 1:
    City = [Bangkok, Dubai, Singapore, Tokyo, Seoul, Kuala_Lumpur, Hong_Kong, Istanbul, Osaka]
    for AH in range(len(City)):
        Cities.append(City[AH])
if AIEurope == 0 and AIAfrica == 0 and AINorth_America == 0 and AIAsia == 0:
    Cities = [Bangkok , Budapest , London , Vienna , Paris , Prague , Dubai , Singapore , Tokyo , Seoul , New_York , Kuala_Lumpur , Hong_Kong , Istanbul , Barcelona ,

StrCities = []
if AIEurope == 1:
    StrCities = ["Budapest", "London", "Vienna", "Paris", "Prague", "Istanbul", "Barcelona", "Amsterdam", "Milan", "Moscow"]

if AIAfrica == 1:
    StrCities.append("Johannesburg")

if AINorth_America == 1:
    StrCities.append("New_York")

if AIAsia == 1:
    StrCities = ["Bangkok", "Dubai", "Singapore", "Tokyo", "Seoul", "Kuala_Lumpur", "Hong_Kong", "Istanbul", "Osaka"]
if AIEurope == 0 and AIAfrica == 0 and AINorth_America == 0 and AIAsia == 0:
    StrCities = ["Bangkok", "Budapest" , "London" , "Vienna" , "Paris", "Prague", "Dubai", "Singapore", "Tokyo","Seoul", "New_York" , "Kuala_Lumpur", "Hong_Kong","Istan
```

This part will also be tested according to the test table too.

Test table Results (Description)

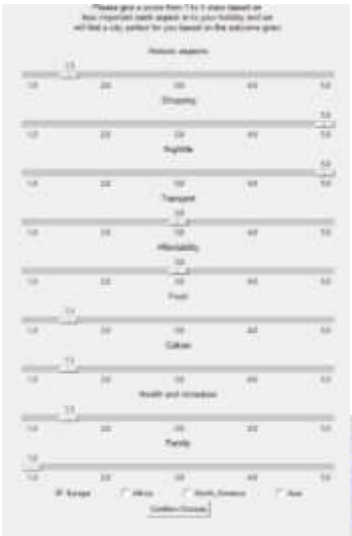
Test number	Input	Actual output
1	Test with Europe and historic	
2	Test with Africa and historic with no being said after wards	
3		
4	Test with Asia and historic	
5	Just Europe selected with no attributes selected	

Test table Results (Choices)

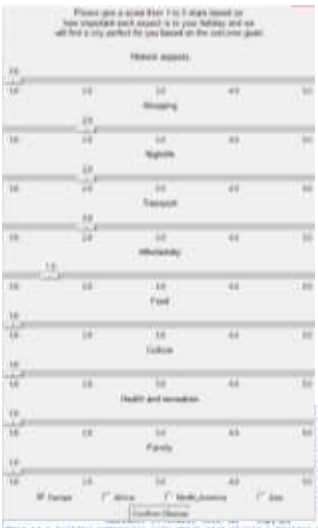
Test number	Input	Actual outcome
1	Europe selected without changing any attributes	
2	Africa selected with no being said after	
3	Find an input which will print multiple outputs for the first option	Its very difficult to find multiple printed cities as the program has far too many available variables to account for and there is no way for a human to calculate what the input should be
4	Find an input which will print multiple outputs for the second option	Its very difficult to find multiple printed cities as the program has far too many available variables to account for and there is no way for a human to calculate what the input should be
5	All regions selected	

After conducting these tests in the test table I asked some members of the public to help test my code. Each time I conduct user testing I am trying to find something I may not have thought of myself. As I am not a team of people like would be programmed with ordinarily, I cannot think of every possible writing style as many different age groups write differently. If this was further developed I would be using AI which would help account for all the different writing styles possible.

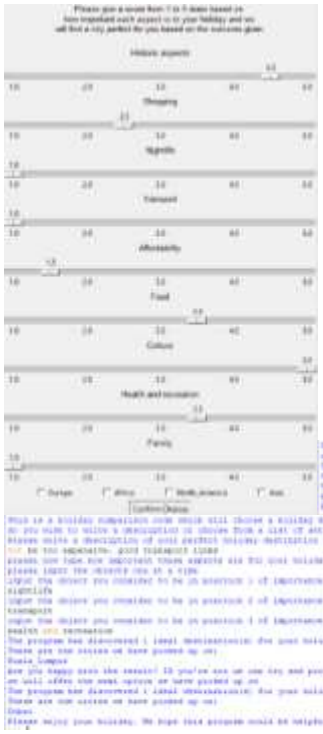
User 1



User 2



User 3



User 4

These test provided some words to be added to the lists to be picked up on as these all provide different writing styles that I couldn't have thought of myself. I got each user to test both the choices and the descriptions part of the code as the descriptions part would provide a writing style that I wouldn't have thought of and the choices menu tests it in ways I couldn't have thought of myself. The users asked for two different holidays in each case providing evidence as if I had 8 different people testing however having 8 different people would be more accurate but I do not have the resources available to sustain this. This is because each time I ask users to test this takes time from their day and many people in the target audience range have things that they need to be doing with their days so don't have the time to take out. Especially because I am a sixth form student many of the people attending my sixth form are too young for my target audience.

From this user feedback I could see that users wanted to be able to not include certain features in their holidays this would allow then to exclude certain features by putting a keyword in front. Putting a feature like that in would require some extra programing and a new prototype version which I will call iteration 3.

Software Development and testing Iteration 3

For this Iteration I will be changing the program to make it respond to negative words as per user feedback.
What this means is if someone were to say not Europe or no swimming or something similar the program will pick up on that and respond accordingly. This will be a challenge to integrate correctly as it will be difficult to determine what words not to include. This can be because there are many different ways to say you don't wish to do something. This would be easier to integrate with AI as you would then be able identify the different ways humans would say no with it only being inaccurate in the beginning.

My idea for how I was going to do this was to turn my input into a list of words and compare every two adjacent words to see if any of the words in my negative words list were there and discard the adjacent ones.
This will work as I am searching for the next attribute. I will only be searching for the next attribute within a given range after the negative word as after a certain point the negative word will no longer be referring to the attribute.

Test table (Description)

Test Number	Input	Desired output	Reasoning
1	Not word word word historic word	Not word word word word	The program should remove the next attribute after the word not within a 5 word range
2	Not not historic	Not not	Historic should be removed and no error should occur as I will account for list sizes
3	Not historic historic	Not historic	The program should only remove the first attribute after the negative word
4	Not shopping	not	This is to check that historic isn't the only aspect being removed
5	not	not	This is to check no error occurs
6	Not historic word word word word not historic word word word word	Not word word word word not word word word word	This checks that the program is iterating through and not just removing the first cycle of negative words used.
7	Not Europe	not	This checks the program is still

			working with region selection as well

Test table (Menu)

The choices section of the code will not need testing seen as it should be the same as in previous versions.

I thought to start by splitting the description into individual words in a list

```
Description = input("Please write a description of your perfect holiday destination this may be as long or short as needed longer descriptions will turn out more accurate
Des = Description.split()
```

I used the .split() method to do this as this would split the input by individual words. I also stored that in a variable so that I could keep and manipulate the list.

```
Not = ("no", "not in", "dont want", "not")
```

I created a list of keywords that the program should pick up on as the negative words. Of course because everyone has a different writing style as with the other keywords there is no way without using ai that I could account for every possible writing style. With my code I am trying to make it account for as much as possible however I also cannot think of every possible word that could be used. Once this section of the program is finished I will also be testing it with members of the public as to ensure anything I haven't thought of gets mentioned. This will help as it will offer a different perspective.

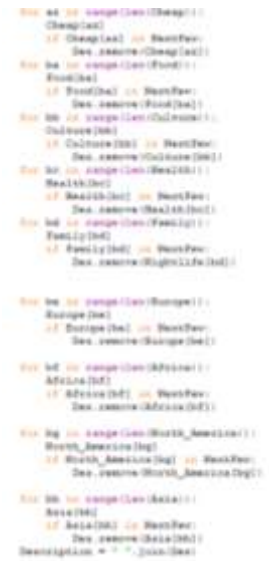
```
for ax in range(len(Des)):
    for ax in range(len(Not)):
        if Not[ax] in Des:
            num = Des.index(Not[ax])
            if len(Des)-num >= 6:
                NextFew = [Des[num+1],Des[num+2],Des[num+3],Des[num+4],Des[num+5]]
            elif len(Des)-num >= 5:
                NextFew = [Des[num+1],Des[num+2],Des[num+3],Des[num+4]]
            elif len(Des)-num >= 4:
                NextFew = [Des[num+1],Des[num+2],Des[num+3]]
            elif len(Des)-num >= 3:
                NextFew = [Des[num+1],Des[num+2]]
            elif len(Des)-num >= 2:
                NextFew = [Des[num+1]]
            else:
                break
            for ax in range(len(Historic)):
                Historic[ax]
                if Historic[ax] in NextFew:
                    Des.remove(Historic[ax])
            for ax in range(len(Shopping)):
                Shopping[ax]
                if Shopping[ax] in NextFew:
                    Des.remove(Shopping[ax])
            for ax in range(len(Nightlife)):
                Nightlife[ax]
                if Nightlife[ax] in NextFew:
                    Des.remove(Nightlife[ax])
            for ay in range(len(Transport)):
                Transport[ay]
                if Transport[ay] in NextFew:
                    Des.remove(Transport[ay])
            for ax in range(len(Cheap)):
                Cheap[ax]
                if Cheap[ax] in NextFew:
                    Des.remove(Cheap[ax])
            for ba in range(len(Food)):
                Food[ba]
                if Food[ba] in NextFew:
                    Des.remove(Food[ba])
            for bb in range(len(Culture)):
                Culture[bb]
                if Culture[bb] in NextFew:
```

Then to account for any list length errors as the Description word list or the list of keyword could both be longer and I cannot tell which will be. I iterate through both the length of the description word lit and the length of the keyword list. This means that the same keyword will be picked up multiple times so I will need to test to see if the method I come up with to solve this only removes the next keyword attribute and not all of them within that given range.

The first if statement checks whether or not any words in the negative words list are actually in the description word list.

If they are the program sets a range based on how many remaining words there are after the negative word. This is because the negative word could be near the end so the list would give an error. I set the amount of words to be 5 because I thought that would be an appropriate amount to pick up on the word. The loops check for 6 words after the length of description list minus the iteration number of the negative word as lists start on index number 0.

Then for each individual attribute I iterated through for the amount of keywords there are in each list and removed the attribute if it was there.

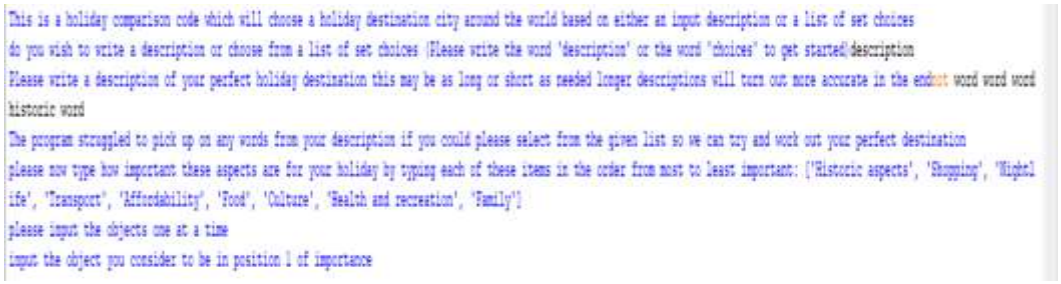
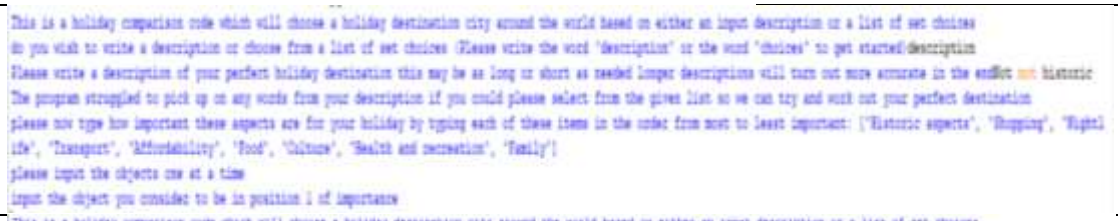
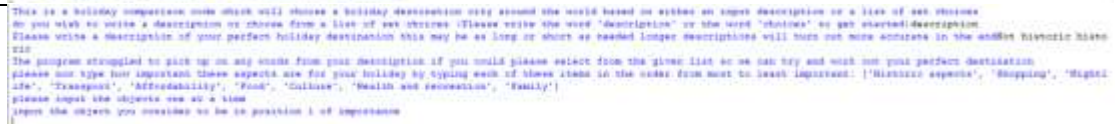
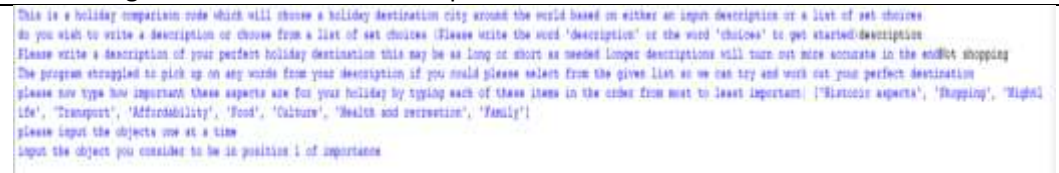
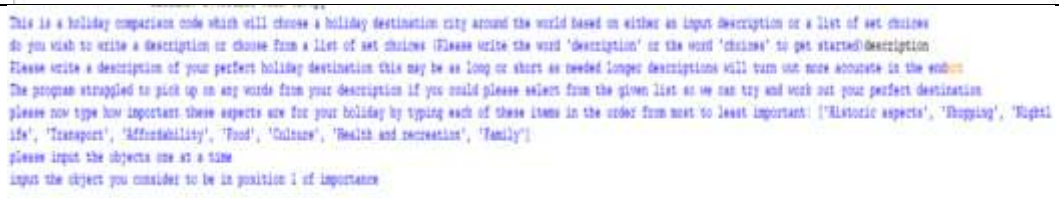


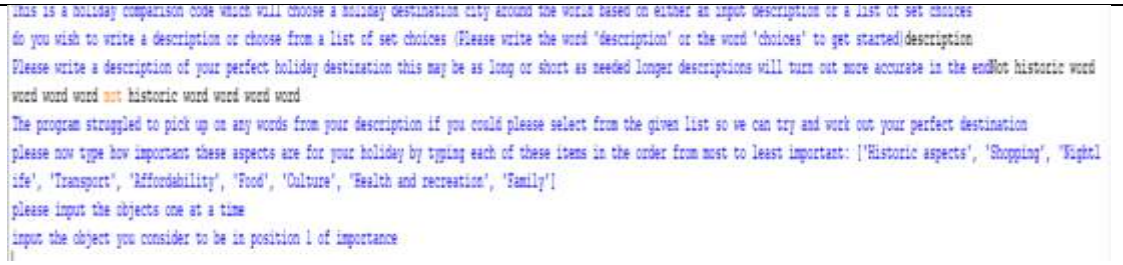
Then once all of the attribute words that I don't want included in the description are removed I then re-join the description back to a string form and not a list as my program is written with it searching through a string for the attributes and not a list for them instead.

The first test I conducted was to see if the program would work with not old and not Europe I choose to test these two aspects at the same time here as it would tell me if the program is working faster than testing them individually.

Seen as the program worked as intended I could see that the code I wrote was doing what it was intended to.
This requires further testing but to find a problem I decided to use the test table until an issue was found.

Test table

Test Number	Input	Actual Outcome
1	Not word word word historic word	 The issue I discovered with this test was I had missed the capital version of the word not as when tested with the lower case version the program functioned differently.  This was functioned as intended. To fix this issue I added the upper case version of not to the list `Not = ("no", "not in", "dont want", "not", "Not")`
2	Not not historic	
3	Not historic historic	 This didn't work as intended as this has shown that the program is removing all aspects after the word not. This will not affect any of the other tests in the test table so I will finish conducting the tests and then fix the problems after the table is finished.
4	Not shopping	
5	not	

6	Not historic word word word word word not historic word word word word	
7	Not Europe	

The only issue found in the program found in the test table was that after a negative word all attributes are being removed.

The first thing I did to try and resolve the issue was add a break to the end of the if statement to try and stop it removing more than one historic

```
for av in range(len(Historic)):
    Historic[av]
    if Historic[av] in NextFew:
        Des.remove(Historic[av])
        break
```

This did not work and yielded the same results as in the test table

The next thing I tried was to set a variable equal to zero and remove the item only if the value of the variable is zero and add one after the item has been removed.

When testing to see if this method had worked I found an issue.

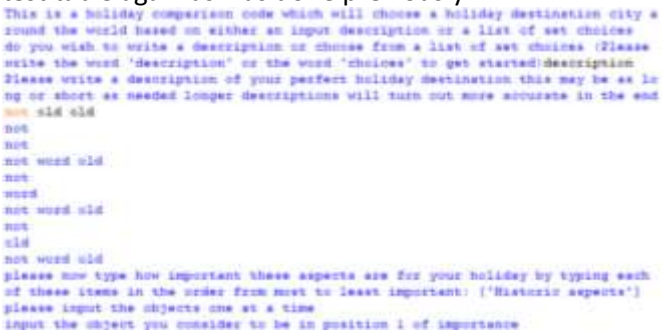
This was only removing the first attribute and then because the value of the variable would be greater than zero after that point it does not continue to remove anymore. This made me think about the problem some more and I realised I couldn't just create multiple variables what I would have to do is find a way to stop the loop another way.

The issue being had here is the program is checking all of the next 5 words after the negative word. This is an issue due as the attribute after the first attribute removed isn't one we want removed at all.
This means we have to find a way to stop the loop after the first attribute is found. An issue with this is that there can be multiple negative words in the input meaning after the first removal we need to find a way to ignore the next words until a negative word is found again.

The solution I came up with to solve this issue was to set a variable value to zero if the current item of the description word list is equal to the current word of the not word list and then only remove the next attribute if the value is equal to zero.

The outcome shows that it is performing the task I wanted it to however an error is printed as I have removed a value from the descriptions list so the program is trying to look for that value.
To solve this problem what I thought to do was to add a new word to the descriptions list in the place that had been removed. The word I add would have to be one which doesn't affect any of the search algorithms as it has to have the same effect as the attribute not existing on the program it would also have to be in a position of +1 to the current loop number as we remove the words after the word not and so to add that position back we have to add one.

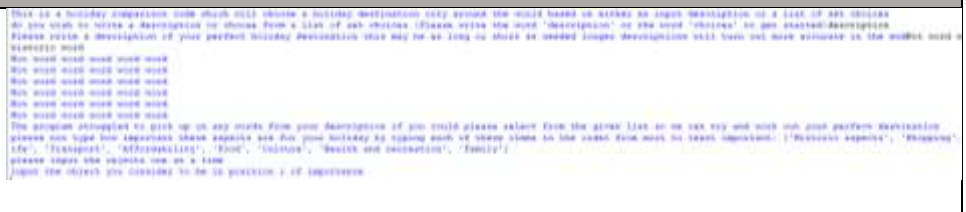
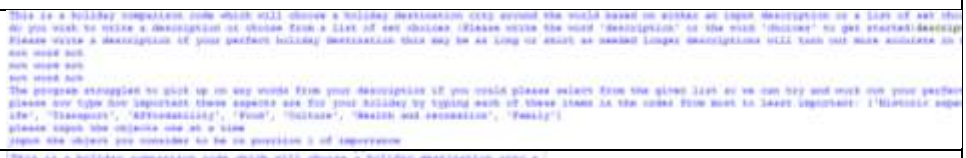
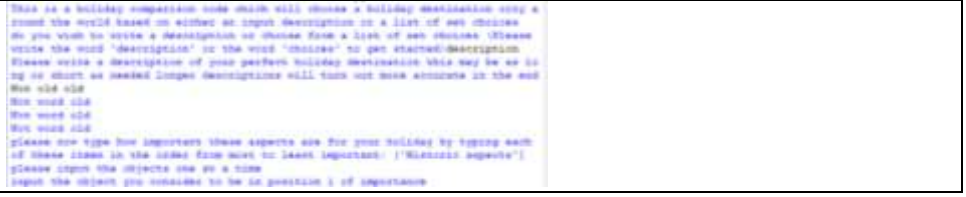
For testing purposes I have included the printed loop value and have only made this work for historic values. If these tests are successful I will then add the rest of the values in and input values from the test table again as was done previously.

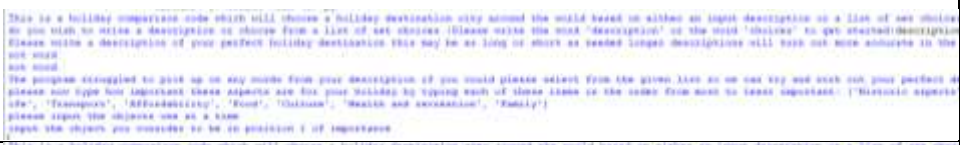
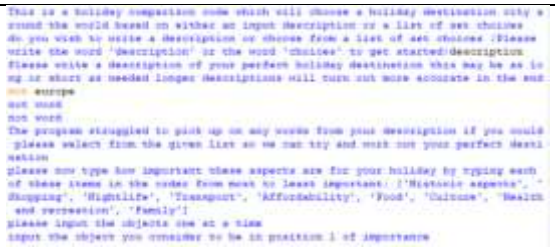


This method seems to have worked. I will now conduct the same test table tests as was conducted before all of the tests will have to be conducted again as changes have been made to all parts of this part of the program.

To show the outcomes I will keep the description printed however in the final version this line will be taken out. Anywhere where the word was expected to be removed is not expected to be replaced with the word “word” as this is the method I used to keep the index within the correct range.

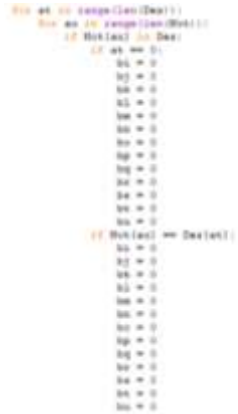
Test table results

Test Number	Input	Actual Outcome
1	Not word word word historic word	
2	Not not historic	
3	Not historic historic	

4	Not shopping	
5	not	
6	Not historic word word word word not historic word word word word	
7	Not Europe	

The tests all come to the expected results this led me to believe the testing was complete and I could now do further testing with members of the public and get feedback from each person.

When I gave the program to the user for testing an error came about. I didn't know why the error occurred but I was able to quickly find the solution by initially setting the values of each of the variables to 0 before the if statement is around.



These variables were labelled using letters of the alphabet as they are just in place in order to ensure the loop occurs once and doesn't remove all attributes.
This shouldn't have changed anything in the test table

After fixing the issue I returned the program to the user to complete the testing and this result was given.

Usability testing of final version

User 1

User 2

User 3

Places of interest should pick up the item historic however, this is hard to incorporate as each of the words that make up that phrase can be used in other ways so it's likely a phrase like that would have to use AI to help pick up on that.

User 4

These results showed that people would use the features I added to my program in iteration 2 and this current iteration.

What was found was there were a few unexpected results and things that need to be added to each of the aspect lists but nothing worth giving another iteration.

After receiving this positive feedback I decided to add comments to each line of my program. This was to ensure that anyone who wished to further develop this program in the future would understand the purpose and function of each part of the program. This would also allow me to work on the program in the future if needed to help me remember what each lines intention was.

Something that was brought to my attention by one of the users is if an input is put in wrong it gives the options of all the aspects mentioned again and not just the ones that are available. This could mean that the user can input the same item twice I left this in the program as intended as if the user was willing to not put in another aspect this may be because the other aspect wasn't wanted. I decided to make this being a possibility clearer in the program so that users are aware that if they do

not like the aspects given they don't have to take the aspects. This could be for a reason such as certain words contain parts of other words within so are picked up unnecessarily this works for things such as nightlife. Nightlife contains the word life so will get picked up on by health and recreation even though this isn't the intended feature. This happens with nightlife being picked up on so in this case the user can discard the health and recreation feature and just use the features that they intended on using from the beginning.

To make this clearer what was said is specified below:

```
print("please input the objects one at a time (If there is a feature picked up on that you dont want included in the search please input the last feature twice!")
```

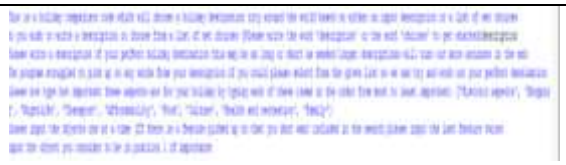
Resources used

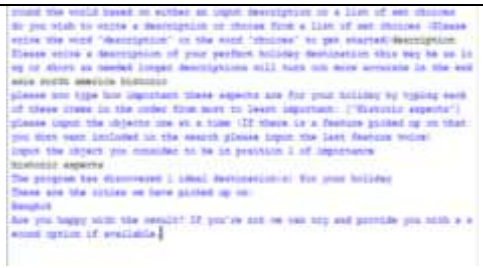
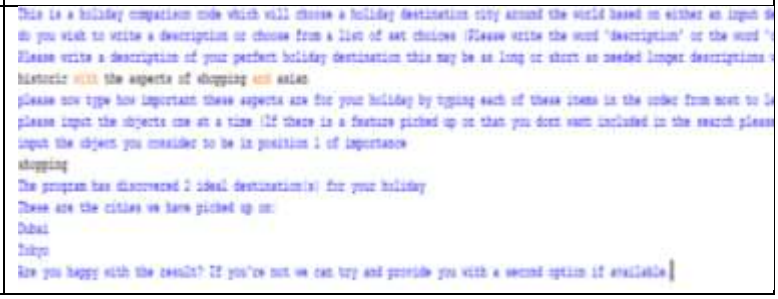
Throughout programming if and when I struggled to solve a particular problem I would often try and find similar solutions to certain issues on student forums. Usually these forums wouldn't have the exact solutions I was looking for but often I could use similar ideas to allow myself to adapt these solutions to my own problems. This is because my problems tended to be encountered due to other areas in the programming interfering with the current section I was working on and thus my problem would be unique but the idea of how the problem was solved on student forums would help lead me in the right direction.

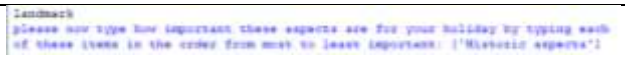
Testing for Robustness

With this testing for robustness there may be inaccuracies as it is difficult to predict what my algorithm will say the city is. This is due to the amount of variables to incorporate in finding the solution. If it wasn't difficult to predict the solution there wouldn't be any point in having this program in the first place as computers are able to make calculations much faster than any computer can.

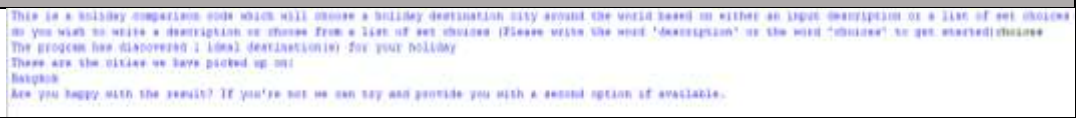
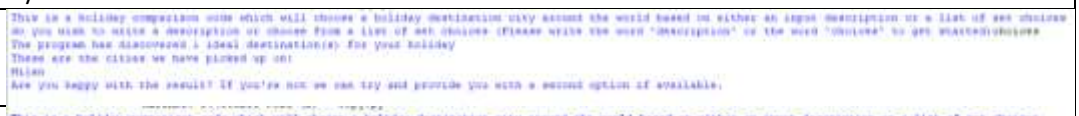
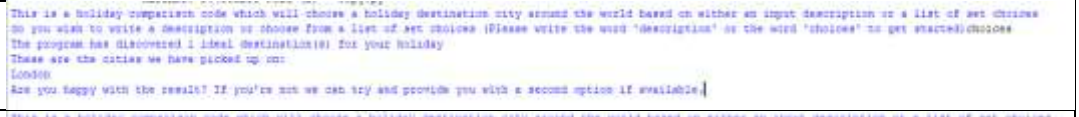
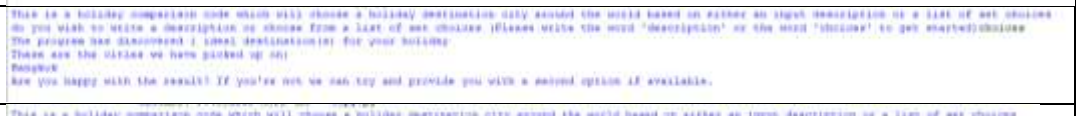
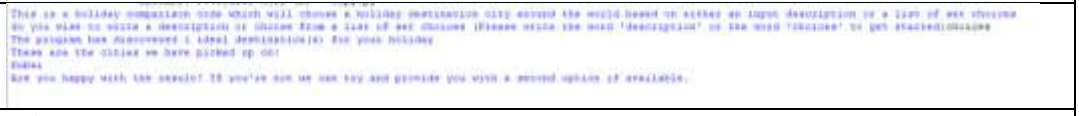
Test number	Test Description	Test passed	Output description	Screenshot (Description)
1	Test that Bangkok appears when key aspects of Bangkok are put in (Historic, shopping, nightlife, transport) For	yes	The output are each city with the aspects of historic as their first choice	Bangkok London Prague Barcelona Amsterdam Milan

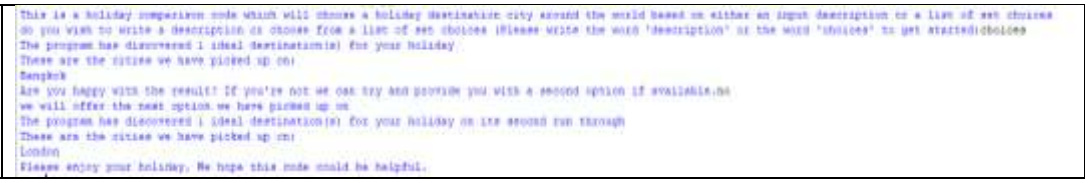
	the choices section just confirm choices immediately as the aspects are in order so the score should be highest for Bangkok			
2	Test that if no description is put in all attributes are given	yes	The program offered every attribute to be ordered and thus there is no issue	
3	Test with food set to 5 shopping set to 4 culture set to 3 and nightlife set to 2.5 this should give the city of Milan and if this test	yes	N/A	N/A

	works it should apply to all of them			
4	Test for Europe with historic as first value to make it easiest to identify the correct city. Expected city of London	yes	These cities are all European cities that contain historic as their first aspect	europe historic please now type ho of these items in please input the c you dont want incl input the object y historic aspects The program has di These are the citi London Prague Barcelona Amsterdam Moscow Are you happy with
5	Test for Asia and North America historic as first value with expected city of Bangkok (this tests for multiple regions)	yes	The answer given clearly shows the expected outcome as Bangkok is the only Asian or North American city with historic as its first aspect	
6	Input a holiday in Asia but not in Europe and not historic	yes	The output cities are all in Asian with the outcom	

	with a quality of shopping so I can more easily predict the outcome. should give city of Dubai		e of shopping as their first aspect.	
7	Type landmark to ensure the program is looking through the entire list of keywords.	yes	As historic was picked up on this proves the program is searching thought the entire keyword list	
8	Input a long description with the word not (this has caused an error in the user testing previously but should be fixed)	yes	As the error which previously occurred didn't appear and this is a long description with multiple negative words contained this	

			proved that the solution is fixed for the know occurrences of the bug.	
9	Say historic aspects and say no should give cities with historic as the second aspects	yes	The end resulting cities are all cities with historic as the second aspect which proves this still functions correctly.	

Test number	Screenshot of choices where appropriate
1	
2	N/A
3	
4	
5	
6	
7	N/A
8	N/A

9	

This list of tests should be appropriate to show each of the different solutions and show the program works as many of the different aspects are programmed in the same way. As well as this I have also already done testing previously throughout the documentation of the program. This testing for robustness was done to check that all the features I added worked together as intended as previously I had tried to just test the impact of the new section being added.

Seen as no errors occurred during this testing for robustness this shows that the program is strong and doesn't contain any errors that I can find. It is possible that with a larger number of people testing they may be able to find an interaction that I have not yet thought about but because of a lack of resources and access to such a large number of people I am unable to find anything as such.

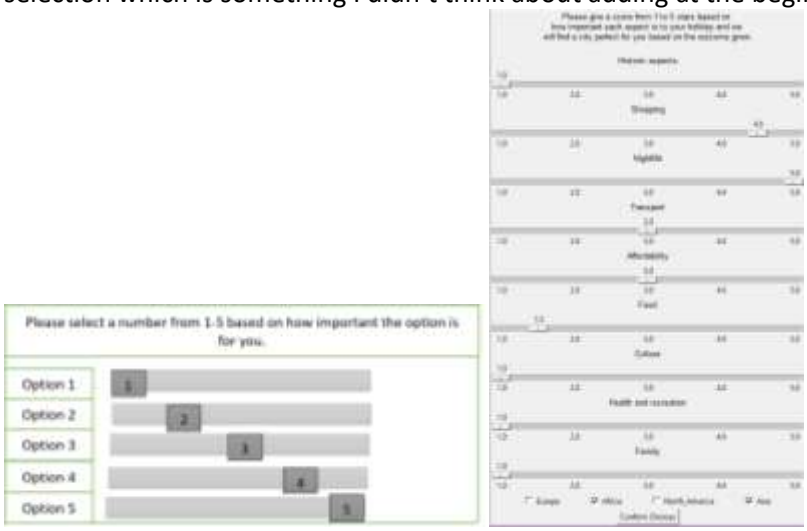
<u>Evidence of prototyping</u>

The program I created had 2 previous versions each which was tested by users between my age range I decided when planning.
The first prototype just contained a basic searching algorithm to find keywords in a list and compare them with keywords in a second list. It then created a score with these comparisons and the most alike city was given to the user. This had some problems however, some of these problems did carry over to the final version due to difficulty of implementing without using real AI modules. Other issues were fixed within the second and third iterations such as the issue of users selecting a specific region. Some users when testing said phrases such as "I want a holiday in Europe" previously my program would not be able to pick up on the fact Europe was mentioned and thus it gave the user holidays in other regions of the world. That told me that this was the next thing that needed to be implemented into my program.
This brought on to the second prototype. In this version region selection was added. What could have been done further would be to add things such as distances from lakes or beaches however I did not add this as I am only one person and that would require a lot of research on each of the cities which would ordinarily be done by a team of people. As I am only one person I did not add this however this is something I would add in the future. When users were given this prototype to test some of these users did use phrases such as these and were confused why some of the options given were not by the sea or lake as asked for. In these cases I explained the state of the program and these users provided feedback on other working aspects of the program.
Another thing was the number of cities used in the program. As I am only one person and creating aspects for each city is something it would take individuals a long time to do for every city this would not be possible for me. This meant that I only worked with 20 city lists that I created myself. Seen as I haven't been to many of these cities myself it was difficult to create these initial lists as this is very much based on user opinion and my judgments were mostly based of other people's experiences some of these lists may be inaccurate. This could be the reason for some of the users saying no after receiving the city back from the information they input. In the future this program would be adapted with AI and thus many of these city lists would be edited to find some sort of average based off the average user experience of visiting a place. I feel as though this would be a more fair way to decide what the key aspects of a city actually are as it's less biased. This method also requires much more information however and thus would not totally be sustainable for me to do as my access to information isn't as vast as trivagos database is or another larger companies. These review websites would each have a database with the different reviews in and this could serve as information to

contribute to the aspects of each city.
The last prototype that I added enabled for the user to disregard an aspect based on a list of keyword with different negative words that could be input. This was the hardest part to decide how this worked due to how the English language could be utilised in many different ways. There are far too many different ways to say that you do or don't want something in your holiday so my challenge was to decide what to remove. This would again be another thing that would be improved and become fairer with AI due to computers being able to process large amounts of test much faster than any other person could. This would allow for the computer to examine many different writing styles to see what would be most appropriate to remove and keep from the users answer.
The method I ended up option for was to say that if a negative word was used I should search thought the next 5 words to see if the attribute can be found in there and if nothing was found there the program could then continue to function as normal. This was the method I opted for as I thought that It was most probable that a user would say something with not and then an attribute but it was also possible that the user could say something along the lines of "I do not want to go to Europe" a phrase such as the one just use would use all 5 extra words with the attribute being the last one. It is also possible to use more than 5 words but the more words I use to search with like that the more chance there would be of my program picking out an attribute that was never intending on being removed. This the value of 5 words was a compromise and could never be accurate without the use of AI. An issue with using AI however is it could make the program more CPU and GPU intensive. This could be a problem as the menu section tends to already crash when run and AI would just increase the chance of this happening as more resources would be needed to run.

Success of solution

I would consider my project to be successful as all of the attributes in the created test tables were created and the goals I set out to achieve at the beginning of the project were all achieved. If I was to change something I made It may be the menu design as I may have tried to use the grid method instead If I was creating this project again in the future however the menu design I created is very similar to the menu design I wrote up in planning but the created menu just has the feature of region selection which Is something I didn't think about adding at the beginning of the project.



Above you can see the menu design side by side with the created menu. I feel all the changes I made to the menu were necessary and turned out to be improvements on the menu design as a whole. I ended up adding more attributes because I found that when creating ideal cities lists many of the cities were too different to be described by only 5 features together.

Earlier when creating my success criteria this is what I said:

Success Criteria

To ensure the program is a success it needs to be able to distinguish between similar cities and pick up words from the user description given. It needs to feel user friendly and accessible enough so even the least tech savvy people can understand what needs to be done. To ensure that happens I will keep in close connection with users so any wording changes or design changes can be made immediately.

For the program to be an improvement on Trivagos interface they already have it needs to add something that doesn't already exist. The city finder doesn't already exist and should help customers find where they want to go much easier.

For this project I will not be using AI this is because of the lack of available data to me so for this project to be successful the program needs to do all the already mentioned criteria whilst proving that if AI was used it would be possible for a more accurate and sustainable program to be created. Whilst my program should be successful in its own right. This will be because im also only searching through 20 cities.

The first part of my success criteria was achieved because my program does search though my list of 20 cities for the city with the highest score. Multiple cities are given at times when the scores of all these cities are the same however with more aspects added and more cities added this would happen less often due to the program being able to distinguish between each city much easier.
The program I have in place is also an improvement on Trivagos interface as this program doesn't exist yet and from my user feedback I was successful in making finding a holiday easier.
This positive feedback also supports the idea that my program is successful at what it aims to achieve even without the use of AI however this sample size is smaller than I would like it to be as for this to be as successful as possible it would have to work internationally meaning I would have to get samples from other counties as well.

Although the program did meet the success criteria successfully there was a number of things that could have been changed or added that would have created a more complete or consistent program with a more user friendly system. Many small changes can however entirely change the program due to the amount of different solutions this program has. My opinion of what makes a perfect holiday comparison program may be different to another person's but that doesn't change the success of either approach.
My program could also be seen as not 100% successful as when users were testing my program they may have had a city in mind that was not selected. This could have been for a number of different reasons such as missing attributes or missing selection features or the ability for the program to interpret what the user was saying. These are features I would consider adding in further development as they would improve the accuracy substantially.

Description of final product

What I have created is a proof of concept of something which doesn't exist in the holiday marketing industry yet. It is a program which searches though either a description or a menu to find a perfect matching holiday for you or give you a list of ideas of where you could go because this doesn't use AI accuracy is limited but I believe that with the 20 cities I have selected my program is reasonably accurate for the time and resources I had to work with. An idea such as this one could really be sold to a company like trivago however would need to be more accurate for when it Is actually published on their domain. I have made the integration for someone to take over the project very easy as I have provided comments on each line describing what it does and I believe the foundation of this program to be very strong for creating a program which would search though every city in the world. I choose to do cities rather than countries due to the accuracy. Deciding which country the user wishes to go to would be much harder than deciding which city due to a country having many different aspects spread around it and not always having one feature. Some countries are also very split in how the experience of visiting there would be like due to different levels of poverty but both choices between city and country would be complex due to the resources needed to decide.

Maintainability of my solution

Due to different aspects of cities changing and world politics many of the features I have given to each of the cities could change very rapidly causing a fault in the algorithm however, the only solution for keeping this maintained and up to date would be to have someone changing the program based off current events which would end up being a taxing job. The alternative to this is using AI by making the program keep up to date with current event and making it aware of how the world is changing it could then change the program immediately. Preferably without the user ever receiving a bad holiday destination. This is solvable computationally due to how computers can compare and link information much faster than a human can. A potential problem with this however would be that a computer may decide that a news story is a negative reason to visit a city and could then potentially remove the ideal city the user is after from the search. This would be solved by keeping the strength of the news editing AI very weak in the deciding factor to the aspects of a city.

The AI solution would also work off user reviews so would be maintained as long as people are using the program as the more people using the program the more data the AI would have to work with and thus the better it can become.

Seen as this will also be on the Trivago website another thing to maintain would be each of the servers that Trivagos search engine run on. This shouldn't be too much of an issue as this framework already exist as they are an established company.

In reality these ideal aspects for each of the cities would be constantly changing based off current events and other political events. It would be difficult to weight how much these events should effect how the program selects the city but the most important factor is the users health so the program cannot send the user to a country that is currently dangerous for the user to be in. This is another opinion based aspect as one person may be more of a thrill seeker and enjoy the different aspects of a more dangerous and wild holiday but another may not. This is something that would have to be decided on how to keep the user safe. This is something that a team of programmers would have to decide and the main rules would mostly have to be kept by AI due to the emotional aspect humans have in making choices and decisions.

With how many different inputs there are its very possible that there is an error I have not been able to find that could cause problems in the future this would mean that implementing a report system may be important so users can report the solutions received or the errors received and a programmer can fix the issues as and when they are found.

Further development of the solution

There are many ways in which this solution could be further developed to create a most sound program which works more accurately. Ultimately however, everyone's experience is different and so one users experience cannot be equated to another users experience. As well as this another user may rate their holiday differently to another. This makes the most sustainable approach to the solution being to work off the average user experience and reviews. To do this would require AI as the amount of data that would need to be handled would be too much for any team of people to deal with. Once the program is running more information will be received every minute and that means more information the team would have to work with. This is impossible and would require a machine to sort though and a machines processing power is much greater and the access to information is also much faster. Further development without AI would include adding a large list of

cities, adding more regions, adding more aspects, adding better regions selection this includes items such as lakes and beaches and if the program is to be uploaded to the trivago page the program should use the users IP address to locate the user and work out how far from each country the user is and possibly flight prices to make the expenses part of the program more accurate. This would be as it would be possible to use the algorithms that trivago already has in place in order to find the cheapest or "best deals" for the user. These things were not included by me in another iteration as they would have taken too much time and too many resources to handle to be sustainable for the scope of the project. This would also largely require the frame work trivago already has in order to find these best deals.
In future a map could also be included which would allow users to select areas from the map for more usability features.
Ultimately there are many features that could be added to improve how the program functions and how it could then be seen by the user.
Adding more aspects would help the program as it would help make each different city totally unique. There being so many different aspects to a city makes it hard to compare how historic a city is compared to another city as they may be historic in different ways and it's just not feasible for one user to compare in this way. This is the method I had to use to compare the 20 city examples that I used and I found that from my experience of comparing the 20 cities there were already many arguments I could make about were and how I was placing each factor. This means that my example of my ideal cities aren't the exact lists that everyone else would put as there is no right answer and it's all opinion based. There being so many different opinion based answers in this solution it makes there being no right answer and the best that can be done is an average of some kind. This also means there are so many different things that can be done for further development with each programmer finding a different outcome. This would mean to find the exact solution trivago would want it would be needed to work very closely with a design or marketing team in trivago.
There is also a lot that can be done once you can determine how far the user is from each different country as this is mostly linked to the framework that already exists on the trivago website.
For a better front end of the program what could be done is open a menu with two buttons on one with description and one with choices on and describe what each button does with each different button opening a menu one would be the already existing choices menu and the other would be a test box for the user to input their description. This might make the program a little bit easier to use and slightly improve the overall user experience. With this front end a back button could be added to allow the user to return to the other option (choices or description depending on what was previously selected) This will allow the user to manoeuvre around the menu without having to fully restart the program to achieve this.
What could also be added is the option to select countries that you do not wish to go to. This could be a useful feature as if people have travelled to one particular location many times and they do not wish to go there anymore the program could discard this location. This would only affect a minute amount of users but it is best to try and help as many users as possible and the more methods to understand a description the better the functionality should be as long as the user isn't overwhelmed with features.

Usability features

I created a gui menu in order to make the menu section neater and more easy for the different users to use. As well as this I have thorough instructions at the beginning of the program which I had users help word when they didn't understand something. With each different iteration I asked users what could be improved afterwards and from that I changed how the program was understood and added any suggested features throughout the programming process I tried to keep as close as possible to the end users as these would be the people who would be using the program at the end so if they don't like it then the project can be seen as a failure. Thankfully after each test with users the users

seemed to be happy with the functionality of the program and the different features offered especially with the gui menu.
The gui menu never had any questions on how it functioned as it seemed to be something very similar to what people have experienced before and so no questions were asked and the first design I created ended up staying till the end.

One issue found with usability was the GUI menu used frequently crashes when used. I tried to find the solution to why this was by looking at task manager whilst running the program but this doesn't also tell you how much cpu usage python is capped at so it can be difficult to identify the issue. The other issue with identifying the cause is that the crashes weren't always consistent and occurred more frequently with more lines of code. Which does indicate it's a hardware issue rather than a fault in the program itself. This issue only effects usability as the overall functionality isn't harmed as when the window is closed an answer is already given which fits with the expected results. This has been appropriately tested.

Main problems of my solution

Everyone's experience when they go on holiday is different the experience is different weather you are going alone with friends or with family and its different for each individual person. Most users will know a general idea of what they want out of a holiday and this is what I was trying to find. The problem with this is that everyone's language is different and one person's use of a certain phrase may be interpreted differently by anyone else. As there are no definitive rules of language and it can mean different things for different people it can be very difficult for a computer to create rules to follow to then decide what someone wants because ultimately the user is the only person who can decide that and any program that I make will only be an estimate on this. I believe my program has the ability to make good estimates based on the resources I started with and the user feedback received. For this reason I believe I was successful in what I tried to achieve however, I know that my program cannot work for everyone and the most successful version will work for a majority of people. This version may also not work perfectly for each individual but it would achieve its job as best as it can. With the further development solutions I have proposed, with the main development feature being AI I believe this would create the most accurate version. The more data that's available the more assumptions that can be made therefore the better the program can function.