**Assignment 1 Explanation Document**

James Humphrey and Justin Roy

**Introduction**

This is an expert system that helps you choose a pet based on your lifestyle. It considers allergies to pets, yearly cost of a pet, time needed for upkeep and how long they can be left alone for. There are six possible results Dog, Cat, Fish, Lizard, Bird, and Hamster. It is written in python and uses two python libraries. There is an executable file so that you do not need to install the libraries for it to work. If for any reason the executable file does not work, here are the libraries to install for the code to work in an IDE or on command line. The executable file can be found in the A1\_JamesH\_JustinR\_Executable\_Directory under the name A1\_JamesH\_JustinR.exe. The code can be found in the A1\_JamesH\_JustinR.py file.

**PySimpleGUI**

PySimpleGUI is required for the user interface to function.

PySimpleGUI installation instructions:

https://pypi.org/project/PySimpleGUI/4.18.1/

**Pandas**

Pandas is needed to read in the knowledgebase and turn it from a cvs to a dataframe which is manipulated as the expert system narrows down what type of pet you should get.

pandas installation instructions:

https://pandas.pydata.org/docs/getting\_started/install.html

**Knowledge base**

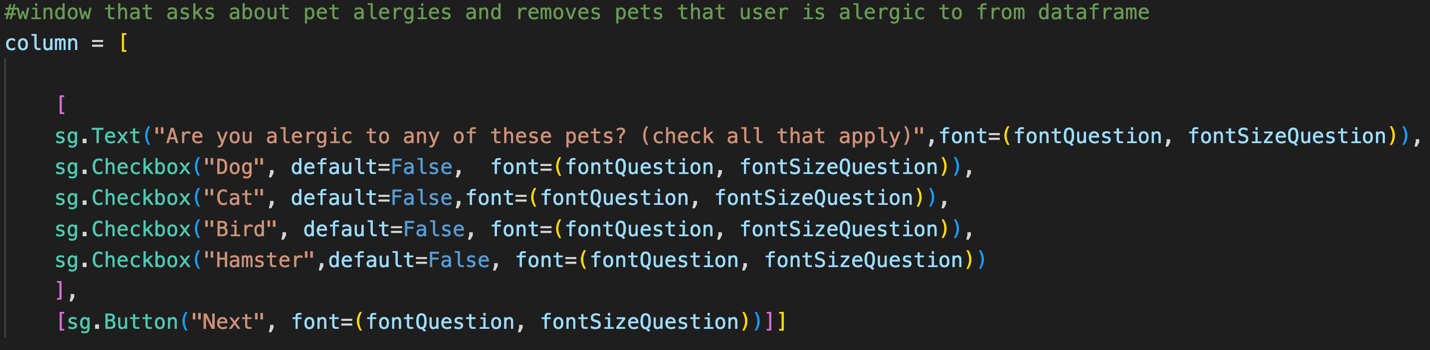
The knowledge base can be found at KnowledgeBase.csv it contains all the information on each pet the yearly cost of a pet in dollars, time needed for upkeep in minutes per day and how long they can be left alone for in hours at a time.

**Inference engine**

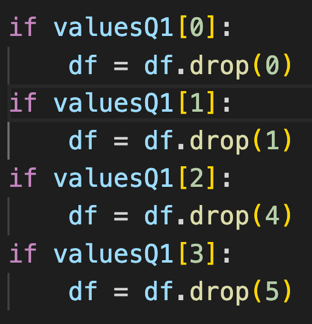
The inference engine is spread out throughout the code and makes an inference after each question is asked.

Here is one example:

Here a question is asked about which pets the user is allergic to.



Then the inference engine drops from the dataframe all the pets that have been selected.



**A "blackboard"/working area**

Every time a question is asked the results are stored in a values dictionary.

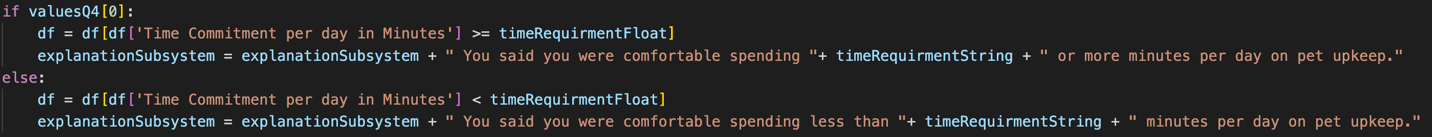
Text

Description automatically generated

Here all the values are stored in valuesQ1 and can be accessed by calling valuesQ1[i] where I is the number the question was asked. 0 is the first question, 1 is the next question and so on. In this expert system the values returned are boolean so if a radio button is clicked by a user that questions value will be true.

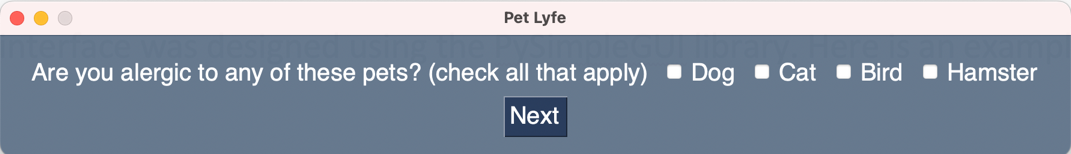
**Explanation subsystem**

The explanation subsystem starts with an empty string and then adds to it a string after each decision is made giving a final string recording all the decisions. Here is an example of the user deciding how much time commitment they are willing to give to the pet upkeep and the two possible explanations for each decision.

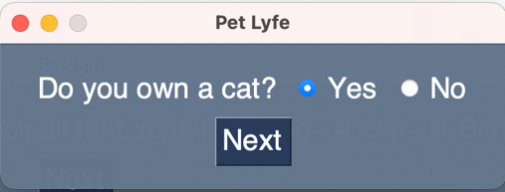


**User Interface**

The user interface was designed using the PySimpleGUI library. Here is an example of check boxes:

****

And here is an example using radio buttons:



**Forward Checking**

This expert system uses forward checking. Forward checking makes decisions as new information or user input becomes available instead of waiting for all the user input and then backtracking to form a solution. In this case forward checking was chosen because not all the questions need to be answered to produce a result. Here if you choose that you are allergic to 4 of the 6 possible pets then the answer to which you should chose can be done with a single question.