AE1205 Assignment 2: Animal guessing

In this assignment you will make a self-learning animal guessing program. An example of how the program will run, when finished (user input in red), can be seen below:

```
Think of an animal. I will try to guess it.
Press Enter to start.
Is it a mammal? (Yes/No)  No
Is it a fish? (Yes/No)  n
Is it an insect? (Yes/No)  yes
I think I know it!
Is it a bee? no
What a pity!
What was the correct animal? Ant
Give yes/no question to distinguish ant from bee : does it fly?
What is the answer for ant? No
Think of an animal. I will try to guess it.
Press Enter to start.
Is it a mammal? (Yes/No)  no
Is it a fish? (Yes/No)  y
I think I know it!
Is it a shark? n
What a pity!
What was the correct animal? Goldfish
Give yes/no question to distinguish goldfish from shark : can you keep one at home?
What is the answer for the goldfish? Yes
Ok, thanks.
Think of an animal. I will try to guess it.
Press Enter to start.
```

The program internally uses a table with questions, by going to a specific next question in the list depending on the answer being yes or no. This prevents it from asking irrelevant questions and thus it quickly zooms in on the animal.

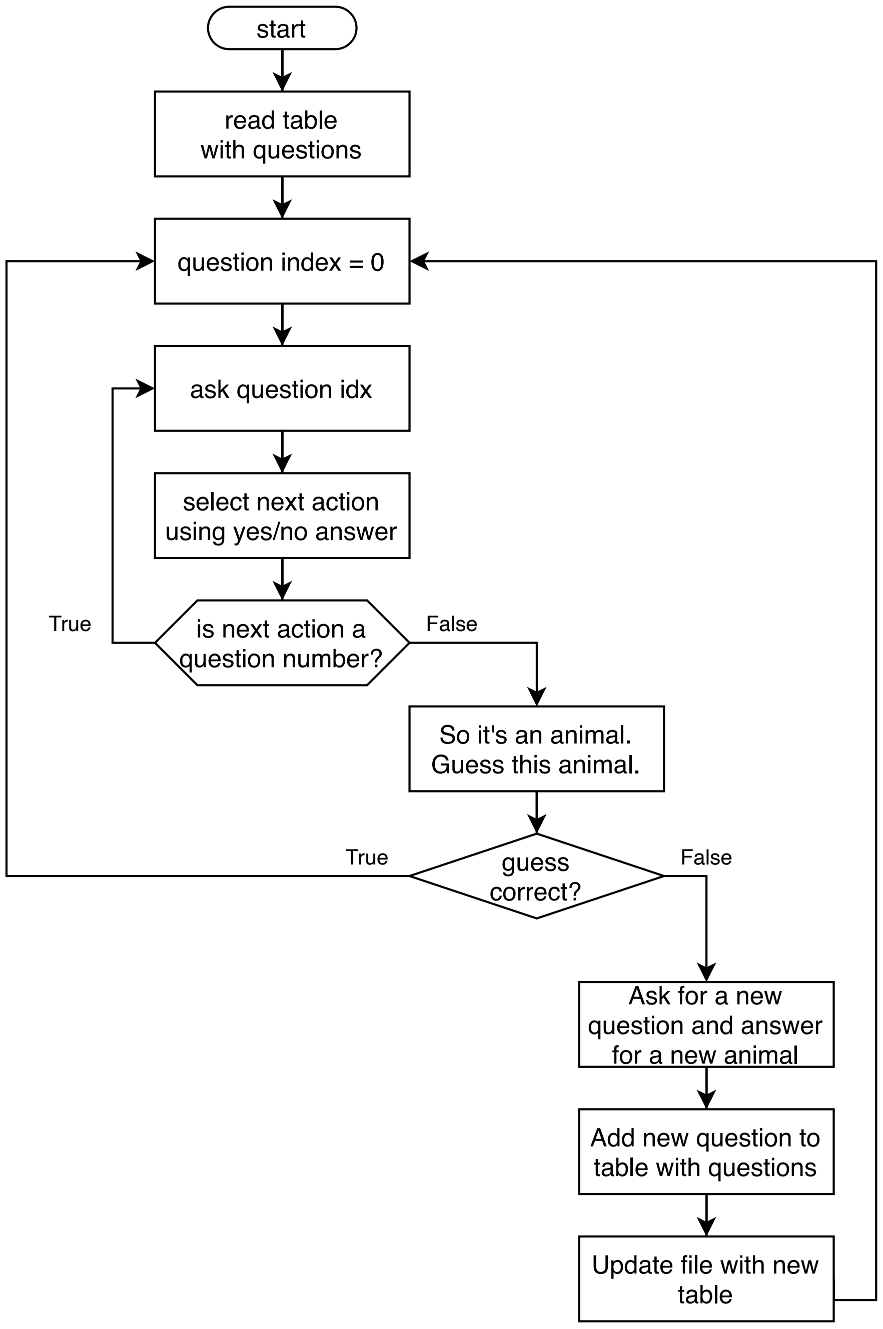
Every time a question is added to the table by the user, the program writes the table to the file `'animals.dat'` and thus learns more animals and questions to distinguish them every time. You will be surprised how quickly the program becomes quite good at this as it never makes the same mistake twice.

On the other side of this page, you can find an example program which reads and write text to a file, as well as interpreting and splitting some strings. Study this program.

Example of a different, admittedly rather useless, program which reads and writes text to a text file and does some string manipulation. This silly program shows the use of some methods which you might need for this assignment.

```
import os

# Let's split up the text and write it to a file:
story = [
    "ARTHUR: Go and tell your master that we have been charged by God with a
     ↪  sacred quest.",
    "ARTHUR: If he will give us food and shelter for the night,"+\
    "he can join us in our quest for the Holy Grail.",
    "FRENCH GUARD: Well, I'll ask him, but I don't think he'll be very keen.",
    "FRENCH GUARD:Uh, he's already got one, you see.",
    "ARTHUR: What?",
    "GALAHAD: He says they've already got one!"]

#-------------------WRITING A FILE----------------------------
with open("script.txt","w") as f:# Open this file for writing
    for line in story:
        # Split line using ":", parts is now a list
        parts = line.split(":")
        if len(parts)==2:
            role = parts[0]
            text = parts[1]
            f.write("Role: "+role+"\n") # write to file using write method
            print("Text:",text,file=f)  # You can also use print
            f.write("\n") # extra empty line

#-------------------READING A FILE----------------------------
# Checking whther it exists can be done using this function:
if os.path.exists("script.txt"):

    # Reading a file and print content on screen:
    with open("script.txt","r") as g:# Open file for reading
        nwords = 0
        for line in g.readlines(): #"for line in g:" also works

            # Remove trailing newline character and split using ":"
            # fields is now a list
            fields = line.strip("\n").split(":")
            keyword = fields[0].strip().lower()

            if keyword=="text":
                # Remove leading and trailing spaces
                # Replace apostrophe by a space to count words
                sentence = fields[1].strip().replace("'"," ")

                # Split it, using one or more spaces as a separator
                words = sentence.split()

                # Count the number of words and add it to th total
                nwords = nwords + len(words)

    print(nwords,"words in total.")
print("Ready.")
```

The final animal guessing program should use the logic as illustrated by the following flow chart:


start
read table with questions
question index = 0
ask question idx
select next action using yes/no answer
is next action a question number?
True
False
So it's an animal. Guess this animal.
guess correct?
True
False
Ask for a new question and answer for a new animal
Add new question to table with questions
Update file with new table

You will make this animal guessing program in three phases:

1. Read the data from the table
2. Making the loop asking the questions
3. Deal with a wrong guess by adding a question and updating the list and the file

Try to keep a working program to test the parts you add. First simple, then make it more complex in steps, which you test along the way.

Step 1: Read the table

Make a file folder called e.g. "animal guessing". Download the text file "start.dat", save it in the folder and make a copy of this file called "animals.dat".

If you open the file in a text editor like Notepad/TextEdit or IDLE, you will see that it contains a start of the question database. Copying this file to animals.dat resets the database to this minimal set of questions. This is the content of the file:

```
0-- Is it a mammal -- 1 -- 2
1-- Does it bark -- Dog -- Cat
2-- Is it a fish -- Shark -- 3
3-- Is it an insect -- Bee -- Spider
```

One row contains:

- a question number (not used, just for readability)
- a question
- the action in case of a "yes" answer
- the action in case of a "no" answer.

The fields are separated by two dashes "--" This first step is to read this file into a list, called `table`. Consider using string methods like `newtxt = txt.strip()` and `newtxt = txt.lower()` to remove spaces, newlines and make it all lower case. `txt.capitalize()` makes the first character into an upper case. Another very useful method is split:

```
>>> txt ="abc $ defg $efg"
>>> txt.split("$")
['abc ', ' defg ', 'efg']
```

As you can see, the `split` function splits a string into a list of strings. Using `lst = line.split("--")` will split a string called line with "--" and return a list. Now make a program to read this file and make sure to have a variable of the type list, called table, with the following content. Print it at the end of your program as a test.

```
>>> table
[['Is it a mammal', '1', '2'], ['Does it bark?', 'dog', 'cat'], , ['Is it a
↪ fish?', 'shark', '3'] , ['Is it an insect?', 'bee', 'spider']]
```

This list actually represents a table:

Row	Column 0:	Col 1: "Yes" action	Col 2: "No" action
0	Is it a mammal	1	2
1	Does it bark	Dog	Cat
2	Is it a fish	Shark	3
3	Is it an insect	Bee	Spider

Step 2: Using the table to ask questions

When running through this table, depending whether the answer of the user is “Yes” or “No”, the action is taken from column number1 (2nd column, with the yes-action) or the column 2 As you can see the string with the action can contain only digits, which can be tested in Python using the `txt.isdigit()` method on a string named txt or it is a text. In case of digits, it can be converted to an integer to get the next question with the `int()` function.

Try to run through the table with the ant or goldfish in mind. When you arrive at the final questions, it guesses wrong. For now we leave this and we only make it process the existing questions:

```
Think of an animal. I will try to guess it.
    Press Enter to start.
    Is it a mammal? (Yes/No)  No
    Is it a fish? (Yes/No)  n
    Is it an insect? (Yes/No)  yes
    I think I know it!
    Is it a bee? no
    What a pity!
    Think of an animal. I will try to guess it.
    Press Enter to start.
    Is it a mammal? (Yes/No)  no
    Is it a fish? (Yes/No)  y
    I think I know it!
    Is it a shark? n
    What a pity!
    Think of an animal. I will try to guess it.
    Press Enter to start.
```

Then as a next stage we will make the program learn, with the help of the user.

Step 3: Make the program learn

To do this we have to check whether the answer is correct or not, but if it is not correct, we need to ask the user for a question and add this to the bottom of the table. For this we first have to make sure we know from which column (1 or 2) and which row (the question index) our guessed animal came. We must also keep the guessed animal, as this will be one of the answers of the new question. So our guess, we should save like this, before we overwrite this cell of table:

```
guess = table[idx][icol]
```

Then when we need to add the new questions, instead of this guess it should now point to the new to be added question. If the table is currently 4 rows long, this will be row number 4 as we started counting with the zero.

```
table[idx][icol] = str(len(table))
```

Then to add a new question, we need some info from the user:

```
What was the correct animal? Ant
Give yes/no question to distinguish ant from bee: does it fly
What is the answer for ant? No
```

After also the goldfish in the example, the table will have expanded with two questions:

Row	Column 0:	Col 1: “Yes” action	Col 2: “No” action
0	Is it a mammal	1	2
1	Does it bark	Dog	Cat
2	Is it a fish	5	3
3	Is it an insect	4	Spider
4	Does it fly	Bee	Ant
5	Can you keep one at home	goldfish	Shark

Add the correct row to your table using the previously collected variable guess as well as the answer by the user to make the new list variable containing a row to be added to the table.

Run the program a few times, and every time you add a question, print the table to see whether your program works as intended. If it does, then immediately after adding a question, write a new file called “animals.dat”, to make sure your program knows the new questions the next time it runs.

Contents of animal.dat after the example run

```
0 -- Is it a mammal -- 1 -- 2
1 -- Does it bark? -- dog -- cat
2 -- Is it a fish? -- 5 -- 3
3 -- Is it an insect? -- 4 -- spider
4 -- Does it fly? -- bee -- ant
5 -- Can you keep one at home? -- goldfish -- shark
```

The function `os.path.exists("animals.dat")` can be used by importing the module called “`os`” (operating system). You can use this function to see whether the file animals.dat exists. If not, then you read “start.dat”, but you always write in “animals.dat”. In this case simply deleting the animals.dat file will reset the database with questions.