15-110: Principles of Computing, Spring 2018

Programming Assignment 11 (PA11)

Due: Tuesday, May 1 by 9PM

IMPORTANT ANNOUNCEMENT
You cannot drop this assignment even if it is your lowest PA score.
Failure to submit this assignment on time will result in a 0 which will be included in your PA total.
We only drop the lowest PA score from PA1-PA8.

Note: You are **responsible for protecting your solutions** to the following problems from being seen by other students both physically (e.g., by looking over your shoulder or verbal discussion) and electronically. In particular, since the lab machines use the Andrew File System (AFS) to share files worldwide, you need to be careful that you do not put files in a place that is publicly accessible.

If you are doing the assignment on the Gates-Hillman Cluster machines we use in the lab or on `unix.andrew.cmu.edu`, please remember to have your solutions inside a `private` folder (which is under your home directory). Our recommendation is that you create a `pa11` folder under `~/private/15110` for this assignment. That is, the new directory `pa11` is inside the directory named `15110`, which is inside the `private` directory.

Overview

For this assignment, you will create a short program that illustrates some very simple "intelligence" using the notion of a binary tree that you learned earlier this semester. In this assignment, you will use a new feature of Python that we haven't used before: you will read from files and write to files. You should save all Python files in a folder named `pa11`. Once you have every Python file, you should zip up the `pa11` folder and submit the zipped file on Autolab.

Assignment

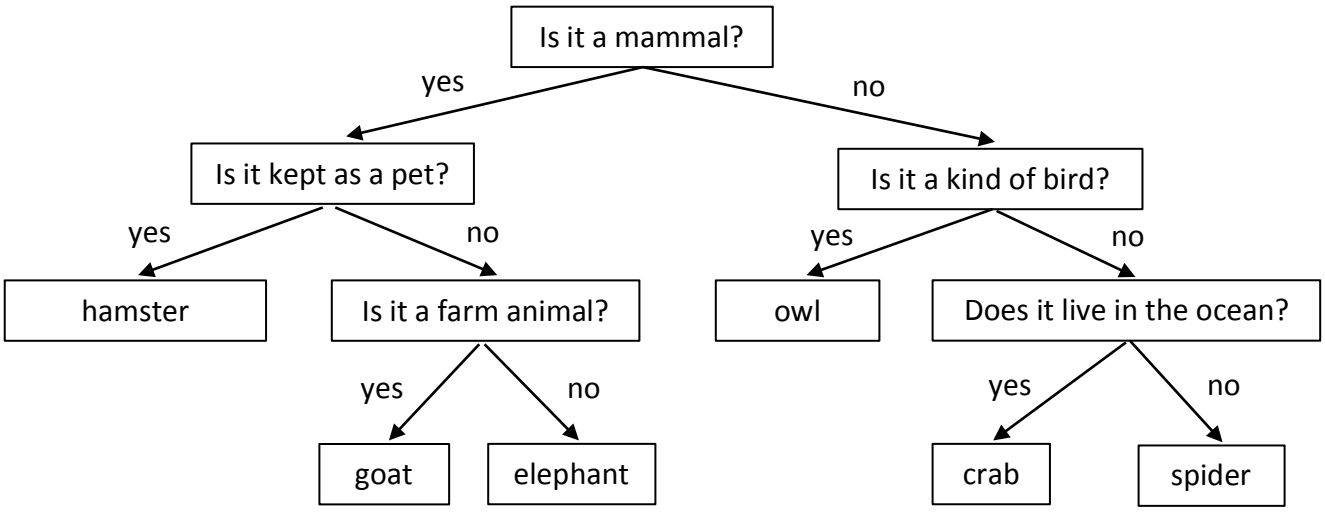
For this assignment, you will write a complete program that plays an animal guessing game between the computer and a human. The computer first reads a file that contains its current "knowledge" and builds a binary tree of this information. Then, the human thinks of an animal, and the computer tries to guess the animal by asking yes/no questions and moving down the tree.

Here is a sample text file and its corresponding binary tree:

```
Is it a mammal?
Is it kept as a pet?
Is it a kind of bird?
hamster
Is it a farm animal?
owl
Does it live in the ocean?


goat
elephant


crab
spider
```



Note that some lines in the text file are blank. These correspond to missing nodes in the binary tree. Each line that ends with a question mark is a question. Otherwise, it is assumed to be an animal.

Recall that a binary tree can be stored simply in Python as a list. The binary tree above would be stored as follows (not all `None` values are shown):

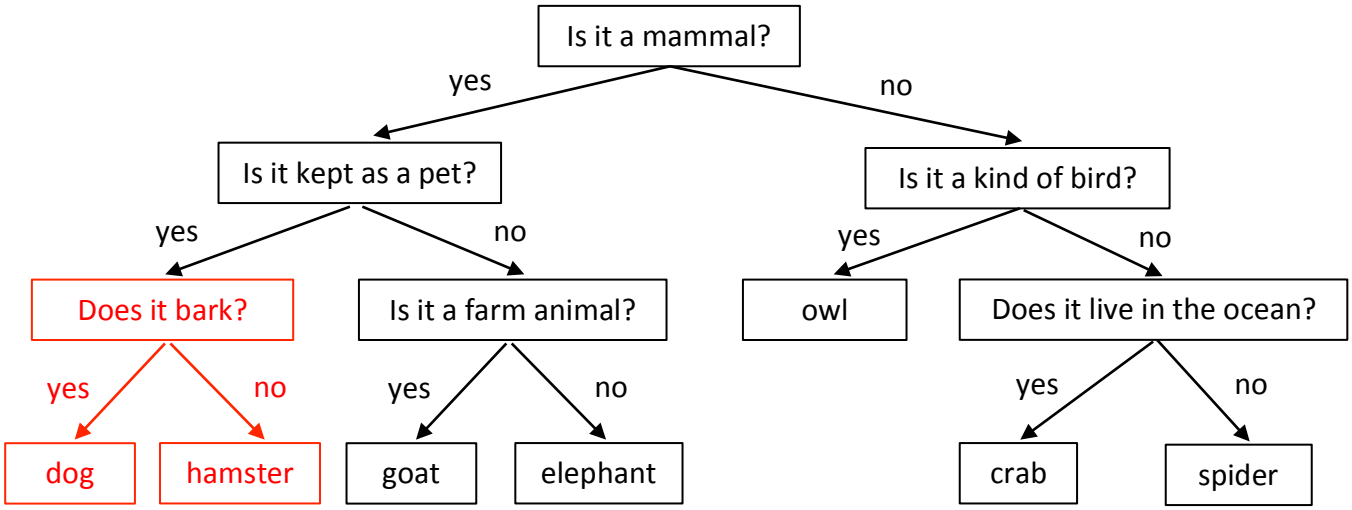
```
[15, 'Is it a mammal?', 'Is it kept as a pet?', 'Is it a kind of
bird?', 'hamster', 'Is it a farm animal?', 'owl', 'Does it live in
the ocean?', None, None, 'goat', 'elephant', None, None, 'crab',
'spider', None, None, None, None, None, None, None, None, None, None,
None, None, None, None, ..., None]
```

The value 15, stored at index 0, indicates the index of the last non-`None` entry (i.e. the last node).

In the binary tree, the computer stores questions in non-leaf nodes, and answers (i.e. animals) in leaf nodes. The computer starts by asking the question in the root node. If the answer from the human is "yes", then the computer goes down the left branch; otherwise if the answer is "no", it goes down the right branch. If the new node it arrives at contains a question, it asks that, again branching left or right. Eventually, it will get to a leaf. At this point, it will ask "Is it a(n) ____?" filling in the blank with the animal in the leaf node. If the human says "yes", then the computer wins and gloats. If the human says "no", then the computer asks for a new question to put in that node, and moves the correct answer and the wrong answer down one level below the new question. For example, suppose the human thinks of the animal "dog". The dialog will look something like this (human input is shown in *red italics*):

```
Let's play a guessing game. Answer yes or no only.
Is it a mammal?
yes or no? yes
Is it kept as a pet?
yes or no? yes
Is it a(n) hamster ?
yes or no? no
Ok, help me learn for next time.
What animal was it? dog
What question can I ask that will distinguish your animal from mine?
Does it bark?
Thank you.
```

Once this conversation is done, the computer's binary tree should be updated as follows:



You will build this program step by step until you have a complete working program. Your program will read in the data from the input file to build the initial tree of knowledge. It will then play the game and if the computer guesses wrong, it will update its tree based on the user input. Once the game is done, we will be able to save the tree into a new file for use next time so that the total knowledge is not lost.

Follow the instructions carefully. **BE SURE TO TYPE CAREFULLY SINCE OUR AUTOGRADER WILL DO EXACT MATCHING ON LETTERS AND SPACING.**

1. (1 pt) Write a function `build_empty_tree()` in the file `animalgame.py` that creates and returns a list representing the binary tree containing 64 entries, all set to `None`. This will allow us to store a complete binary tree with 6 levels = 63 nodes. (We will use index 0 for something else.) Test your function fully and submit to Autolab to check that you get maximum credit before moving on.

Sample usage:
```
python3 –i animalgame.py
>>> t = build_empty_tree()
>>> t
[None, None, None, ..., None]
```

Once you have this completed, copy your current solution into **`animalgame_v1.py`** in the `pa11` folder at the command line prompt after leaving the `python3` interpreter. Do not change this copy.

`cp animalgame.py animalgame_v1.py` ← **UPDATED**

2. (1 pt) Write a function `get_answer()` in the file `animalgame.py` that prompts the user for an answer of "`yes`" or "`no`" using the prompt "`yes or no? `" exactly as shown (with one space after the question mark). If the answer is not exactly correct, repeat the prompt until you get a valid answer. (HINT: Use a `while` loop.) Return the answer from the function (do not print it out).

Sample usage (user input shown in *red italics*):
```
python3 –i animalgame.py
>>> answer = get_answer()
yes or no? yes
>>> answer
'yes'
>>> answer = get_answer()
yes or no? no
>>> answer
'no'
>>> answer = get_answer()
yes or no? Yes
yes or no? NO!
yes or no? si
yes or no? yes
>>> answer
'yes'
```

Once you have this completed, copy your current solution into **`animalgame_v2.py`** in the `pa11` folder at the command line prompt after leaving the `python3` interpreter. Do not change this copy.

`cp animalgame.py animalgame_v2.py` ← **UPDATED**

3. (3 pts) Download the file `startingtree.txt` from the course website into your `pa11` folder. If you examine the file, it should look exactly like the file shown on page 2 of this handout.

Write a function `initialize_tree(filename, tree)` in the file `animalgame.py` that requires a filename of a file that stores the data for the tree and the list representing the `tree` which should be "empty" (i.e. 64 None values from step 1). The function should read the questions and answers from the data file and return the complete question/answer tree as the final result.

Algorithm: Open the file for reading. Set a *counter* equal to 1. Then for each string in the file, remove the newline character, and then see what remains (call this *clean_string*). If *clean_string* is empty, then set *node* to `None`. Otherwise set *node* to the string *clean_string*. After you set *node*, store *node* in *tree* at the index given by your *counter* and then update the *counter* by 1. Once your loop ends (i.e. the file is completely read), store the index of the last used node in position 0 of the tree and return the tree.

Sample usage (assume your input data file is named `startingtree.txt`):

```
python3 –i animalgame.py
>>> t = build_empty_tree()
>>> t = initialize_tree("startingtree.txt", t)
>>> t
[15, 'Is it a mammal?', 'Is it kept as a pet?', 'Is it a kind of
bird?', 'hamster', 'Is it a farm animal?', 'owl', 'Does it live in
the ocean?', None, None, 'goat', 'elephant', None, None, 'crab',
'spider', None, None, None, None, None, None, None, None, None, None,
None, None, None, None, None, ..., None]
```

Once you have this completed, copy your current solution into **`animalgame_v3.py`** in the `pa11` folder at the command line prompt after leaving the `python3` interpreter. Do not change this copy.

`cp animalgame.py animalgame_v3.py` ← **UPDATED**

4. (2 pts) Write the function `play_game(tree)` in the file `animalgame.py` that plays one round of the game as described earlier. The computer starts with the greeting, **exactly as shown**:

```
Let's play a guessing game. Answer yes or no only.
```

In this version, the computer will not update its tree. Instead it will just guess an animal and either win or lose. This version will help make sure you can traverse the tree correctly.

Algorithm:

1. Print the greeting.
2. Set *treeindex* equal to the index of the root of the tree. (Where is the root stored?)
3. Set *done* equal to `False`.

4. While *done* is not `True`, do the following:
    a. Set *node* equal to *tree[treeindex]*.
    b. If *node* is equal to `None`, then you somehow fell off the tree, so print an error message "NO NODE!" and return `None`. (And then fix your code!)
    c. Otherwise if *node* represents a question, then do the following:
        i. Print *node* (which should be the question).
        ii. Use the `get_answer` function to get the user's yes/no answer.
        iii. If the user's answer is yes, then set *treeindex* to the index of *node*'s left child.
        iv. Otherwise set *treeindex* equal to the index of *node*'s right child.
    d. Otherwise (the node should have an animal), set *done* to `True`.

Once step 4 ends, *treeindex* should be the index of a node containing the name of an animal.

5. Print "Is it a(n)" followed by the name of the animal. (You should save this animal in a variable since you will need it later in the next version.)
6. Use the `get_answer` function to get the user's yes/no answer.
7. If the user answers yes, then print "I am a very smart computer."
8. Otherwise, print "Better luck next time."
9. Return `None`.

Sample usage (user input shown in *red italics*):

```
python3 –i animalgame.py
>>> t = build_empty_tree()
>>> t = initialize_tree("startingtree.txt", t)
>>> play_game(t)
Let's play a guessing game. Answer yes or no only.
Is it a mammal?
yes or no? yes
Is it kept as a pet?
yes or no? yes
Is it a(n) hamster ?
yes or no? yes
I am a very smart computer.
>>> play_game(t)
Let's play a guessing game. Answer yes or no only.
Is it a mammal?
yes or no? no
Is it a kind of bird?
yes or no? yes
Is it a(n) owl ?
yes or no? no
Better luck next time.
```

Once you have this completed, copy your current solution into **`animalgame_v4.py`** in the `pa11` folder at the command line prompt after leaving the `python3` interpreter. Do not change this copy.

`cp animalgame.py animalgame_v4.py` ← **UPDATED**

5. (2 pts) Update your `play_game(tree)` function in `animalgame.py` so that it updates the tree if it incorrectly guesses the animal. To do this, you need to update steps 8 and 9 of the algorithm as follows:

8. Otherwise, do the following:
    a. Print "Ok, help me learn for next time."
    b. Ask the user to input the animal using the prompt "What animal was it?"
    c. Ask the user to input a question for the new animal using the prompt
    "What question can I ask that will distinguish your animal from mine?\n"
    (we will assume the input question ends with a question mark)
    d. Set the tree list at *treeindex* to the question input by the user in step 8c.
    e. Set the tree list at the left child of *treeindex* to the answer input by the user in step 8b.
    f. Set the tree list at the right child of *treeindex* to the string you saved in step 5.
    g. If the index of the right child is greater than the number of nodes stored in index 0 of the tree list, update the number of nodes to the index of the right child.
    g. Print "Thank you."
9. Return tree as the final step.

You may assume that the game will not go beyond 6 levels of nodes.

Sample usage (user input shown in *red italics*):

```
>>> t = build_empty_tree()
>>> t = initialize_tree("startingtree.txt", t)
>>> t = play_game(t)
Let's play a guessing game. Answer yes or no only.
Is it a mammal?
yes or no? yes
Is it kept as a pet?
yes or no? yes
Is it a(n) hamster ?
yes or no? no
Ok, help me learn for next time.
What animal was it? dog
What question can I ask that will distinguish your animal from mine?
Does it bark?
Thank you.
>>> t
[15, 'Is it a mammal?', 'Is it kept as a pet?', 'Is it a kind of
bird?', 'Does it bark?', 'Is it a farm animal?', 'owl', 'Does it live
in the ocean?', 'dog', 'hamster', 'goat', 'elephant', None, None,
'crab', 'spider', None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, ..., None]
>>>  t = play_game(t)
Let's play a guessing game. Answer yes or no only.
Is it a mammal?
yes or no? no
```

(sample usage continued)

```
Is it a kind of bird?
yes or no? no
Does it live in the ocean?
yes or no? no
Is it a(n) spider ?
yes or no? no
Ok, help me learn for next time.
What animal was it? turtle
What question can I ask that will distinguish your animal from mine?
Does it have a hard shell?
Thank you.
>>> t
[31, 'Is it a mammal?', 'Is it kept as a pet?', 'Is it a kind of
bird?', 'Does it bark?', 'Is it a farm animal?', 'owl', 'Does it live
in the ocean?', 'dog', 'hamster', 'goat', 'elephant', None, None,
'crab', 'Does it have a hard shell?', None, None, None, None, None,
None, None, None, None, None, None, None, None, None, 'turtle',
'spider', None, ..., None]
>>> t = play_game(t)
Let's play a guessing game. Answer yes or no only.
Is it a mammal?
yes or no? yes
Is it kept as a pet?
yes or no? yes
Does it bark?
yes or no? yes
Is it a(n) dog ?
yes or no? yes
I am a very smart computer.
```

Once you have this completed, copy your current solution into **`animalgame_v5.py`** in the `pa11` folder at the command line prompt after leaving the `python3` interpreter. Do not change this copy.

`cp animalgame.py animalgame_v5.py` ← **UPDATED**

6. (1 pt) Write the function `save_tree(filename, tree)` in the file `animalgame.py` that opens the given filename for writing, and then stores the contents of the tree in the file. Questions and answers should be stored as given and `None` should be stored as a blank line. Every line you store should end with a newline. Do not store the count of nodes at index 0 of tree in the file but you should use that to help you control the loop so that you only include the necessary lines. You should have no additional lines after the last line of information. You should be able to read this file in as the starting tree when you play the game again so it can continue to grow in its "knowledge" over time.

Sample usage (assuming you ran through the example for Problem 5):

```
>>> save_tree("endingtree.txt", t)
```

The file `endingtree.txt` should have 31 lines and look like this:

```
Is it a mammal?
Is it kept as a pet?
Is it a kind of bird?
Does it bark?
Is it a farm animal?
owl
Does it live in the ocean?
dog
hamster
goat
elephant


crab
Does it have a hard shell?









turtle
spider
```

The last line should have a newline so it may seem that there is another blank line after the last line, but your cursor should not be able to move beyond the first position of the next line when you examine this file in an editor.

Once you have this completed, copy your current solution into **`animalgame_v6.py`** in the `pa11` folder at the command line prompt after leaving the `python3` interpreter. Do not change this copy.

`cp animalgame.py animalgame_v6.py` ← **UPDATED**

Submission

At each step, you should zip up what you currently have into `pa11.zip` and submit this to Autolab for grading. You should check to see that you have maximum credit for each step you've completed before moving on to the next step, since each step depends on the previous steps. For later steps, you will get 0's since you won't have those files in your zip file, but that's ok as you work on each part.

At the end of this assignment, you should now have the `pa11` folder that contains the following files:

```
animalgame_v1.py
animalgame_v2.py
animalgame_v3.py
animalgame_v4.py
animalgame_v5.py
animalgame_v6.py
startingtree.txt
endingtree.txt
```

You may have additional files in this folder which will not be graded. Zip up the folder and submit the zipped file named as `pa11.zip` on Autolab your final time for your final score. Be sure to check your submissions to make sure you are handing in the correct files.