**DEERWALK INSTITUTE OF TECHNOLOGY**



**LAB1:ANIMAL GUESSING GAME**

|  |  |  |  |
| --- | --- | --- | --- |
| **SUBMITTED BY:** |  |  | **SUBMITTED TO:** |
| **NAME: KUSAL BISTA** |  |  |  |
| **PROGRAM: B.SC.CSIT (SECOND SEM)** |  |  |  |
| **ROLL NO.: 512** |  |  |  |
| **SECTION: B** |  |  |  |
| **DATE: 2MARCH 2018** |  |  | **RAMESH RIMAL** |

**KATHMANDU, NEPAL**

**2018**

Introduction:

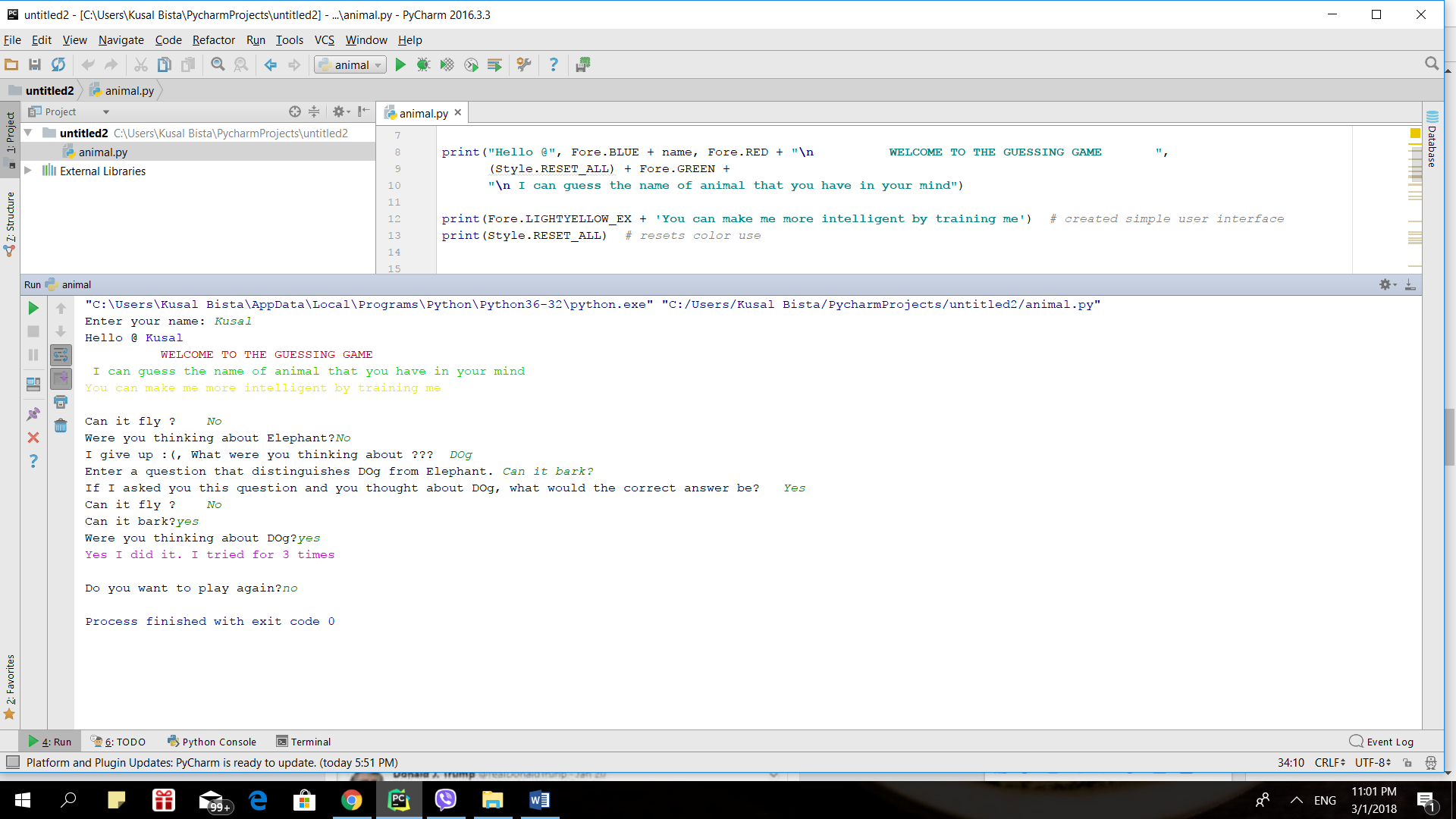
Animal Guessing Game is a very simple game in AI where player thinks of an animal and the computer tries to guess which animal the player is imagining by asking questions which can be answered 'yes' or 'no'. Before beginning the game, player should think of an animal. For example, suppose you are thinking of a dog. The computer will generate the question at the root node “Are you a mammal?" as the first question. The user has to press 'y' or 'n' (1 or 0) as the response. Based on the response the computer will generate the second question. The game continues until the leaf node is reached. At the leaf node the computer will generate the guess (dog, mouse, trout or sparrow in our example).

Theory:

The game can be built by the implementing a binary tree. It is a data structure where a node has at most two children, usually referred to as the “left" child and the “right" child. Each node can store some information as well. A k-ary tree is a tree in which each node has no more than k children. It is also sometimes known as a k-way tree, an N-ary tree, or an M-ary tree. A binary tree is the special case where k=2. Any k-ary tree can be represented as a binary tree where in each node the left pointer points to the first child and the right pointer points to the next sibling. Such a tree has been given the name left-child, right-sibling binary tree (LC-RS binary tree).

Program:

*/\*  
Animal Name Guessing Game  
Kusal Bista  
512  
\*/***import** sys  
  
**from** pip.\_vendor.colorama **import** Fore, Back, Style *# import to use color for text  
  
# print(Fore.RED + 'some red text')*name = str(input(**"Enter your name: "**))  
  
print(**"Hello @"**, Fore.BLUE + name, Fore.RED + **"\n WELCOME TO THE GUESSING GAME "**,  
 (Style.RESET\_ALL) + Fore.GREEN +  
 **"\n I can guess the name of animal that you have in your mind"**)  
  
print(Fore.LIGHTYELLOW\_EX + **'You can make me more intelligent by training me'**) *# created simple user interface*print(Style.RESET\_ALL) *# resets color use***def** Boolean(answer): *# answer can be either true or false* **if** answer[0] == **'y'**:  
 **return True  
 else**:  
 **return False  
  
  
def** makeQuestion(question, yes, no):  
 **return** [question, yes, no]  
  
  
**def** createQuestion(question, rightAns, wrongAns):  
 **return** [question, rightAns, wrongAns]  
  
  
**def** test(ans): *# validate the answer given by player should be either yes or no* **if** (ans == **"yes" or** ans == **"no"**):  
 **return True  
 else**:  
 **return False  
  
  
def** askQuestion(question): *# checks whether there is question in list left or not?* **if** (type(question).\_\_name\_\_ == **"list"**):  
 **return** str(input(question[0])).lower()  
 **else**:  
 **return** str(input(**"Were you thinking about %s?"** % question).lower())  
  
  
**def** playAgain(): *# whether user want to plain again or quit game* **return** Boolean(str(input(**"Do you want to play again?"**)))  
  
  
**def** nextQuestion(question, answer): *# returns questions from the list and prints acknowldgement* **global** count  
 count += 1  
 **if** (type(question).\_\_name\_\_ == **"list"**):  
 **if** answer:  
 **return** question[1]  
 **else**:  
 **return** question[2]  
 **else**:  
 **if** answer:  
 print(Fore.MAGENTA + **"Yes I did it. I tried for "** + str(count) + **" times"**)  
 print(Style.RESET\_ALL)  
 count = 0  
 **if** playAgain(): *# starts game again* count = 0  
 **return** firstQues  
 **else**:  
 sys.exit(0) *# exits game* **else**:  
 **return** makeNewQuestion(question)  
  
  
**def** replaceAnswer(tree, find, replace):  
 **if not** (type(tree).\_\_name\_\_ == **"list"**):  
 **if** tree == find:  
 **return** replace  
 **else**:  
 **return** tree  
 **else**:  
 **return** makeQuestion(tree[0], replaceAnswer(tree[1], find, replace), replaceAnswer(tree[2], find, replace))  
  
  
**def** makeNewQuestion(wrongAns):  
 **global** firstQues, count  
 correctAns = str(input(**"I give up :(, What were you thinking about ??? \t"**))  
 question = str(input(**"Enter a question that distinguishes %s from %s. "** % (correctAns, wrongAns)))  
 answer = Boolean(str(input(  
 **"If I asked you this question and you thought about %s, what would the correct answer be?\t"** % correctAns).lower()))  
 **if** answer:  
 newQuestion = [question, correctAns, wrongAns]  
 **else**:  
 newQuestion = [question, wrongAns, correctAns]  
 ques = replaceAnswer(firstQues, wrongAns, newQuestion) *# after training game starts from initial state* firstQues = ques  
 count = 0  
 **return** ques  
  
  
count = 0 *# initially count is zero*firstQues = createQuestion(**"Can it fly ?\t"**, **"Bat"**, **"Elephant"**)  
  
*# call createQuestion function where (question, rightAns, wrongAns) are the parameters*ques = firstQues  
  
*# print(ques)***while True**: *# if answer is True* ans = askQuestion(ques)  
 **while not** test(ans): *# if the user reply is not yes or no* print(**"Your answer must be either yes or no\n "**)  
 ans = askQuestion(ques)  
 ques = nextQuestion(ques, Boolean(ans))

OUTPUT:

LIMITATION

The above program has the following limitations:

1. The program can only read ‘YES’ or ‘NO’ value while traversing the tree. However, it can read other text when learning from the user.

2. The program cannot differentiate between duplicate data (both question and answer.)

3. The program does not store new information in secondary storage. Hence, it will not retain the information when terminated.