Homework 5

***House***

**Skills you will learn**

* Using lists
* Practicing functions, loops, and decisions

|  |
| --- |
| House: A Treasure Finding "Game" |

**Description**

**Introduction**

You will build a game that requires the player to find all of the treasures in a house.  The player will use the n,e,s,w keys to move around.  The player should not be able to go through walls.  Rooms in the house may have locked doors.  In order to do through a locked door, the player must have previously picked up a key.  To win the game, the player must navigate around the house, going through locked doors as necessary, and pick up all the treasures.  
  
Initially, your program will read the house from a file.  For example, a file with the contents shown below describes a house.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*          \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*         0                         6      \*

\*\*\*\*\*\*\*\*\*\*\*   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*     \*

          \*   \*                         \*    \*

    \*\*\*\*\*\*\*\*5\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*            \*  t \*

    \*                  1   \*           \*\*\*\*\*\*

    \*       t            \*

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To create your own house file, simply click on "new file" in repl.it, name the file, and copy and paste the above text into it. In the file, the stars (\*) represent impassable walls. Numbers 0 through 4 represent a key, and the number 5 through 9 represent a door.  Key 0 opens door 5, key 1 opens door 6, and so on. Each “t” represents a treasure.

Below is a function that you will include in your program that loads a file into your Python program and represents it as a list of lists.  After calling the function below, you will be able to look at any cell in the house to determine what is here.  For example, if you use the following code:

house = build\_house();

Then you will be able to inspect house[0][0] and see that it is a '\*', which means that it is a piece of a wall.  Likewise, at house[2][12] is a key (key 0) that will open the locked door at house[5][12] (door 5).

**def** build\_house():

print("Please enter the house file: ")

house\_file = input()

housefp = open(house\_file, "r")

myhouse = []

line = housefp.readline()

**while** line:

myhouse.append(list(line))

line = housefp.readline()

**return** myhouse

**Part 1 Navigation**

You will write a main driver that contains a loop that asks the player for a direction, and then attempts to move the player in that direction.  The algorithm for the main is below.

|  |
| --- |
| main()   Build the house (see build\_house() function above)   # the player’s start location:   Set startrow and startcol to be indices of an empty space in the house   Set num\_treasures to the number of t’s in the house   Set tcount to 0   # how many treasures found so far   Loop while tcount is less that num\_treasures     Print the house     Print the directions that the user can go (n,e,s,w), anywhere not a wall     Read input from user into variable called command     # next, trow and tcol will represent where the player wants to go     # make sure you call check\_north, check\_south, ... as appropriate     if command is “n” set trow to startrow-1 and tcol to startcol     however if command is “e” set trow to startrow and tcol to startcol+1     however if command is “s” set trow to startrow+1 and tcol to startcol     however if command is “w” set trow to startrow and tcol to startcol-1     however if command is “q” then break out of the loop     if the character at index trow and tcol of the house is a \* then         tell the user that they cannot go that way and continue the loop     fi     # \*\*\* right now we are not going to handle doors or keys     # The IF statement below should be coded into get\_treasure()     #    except for the print (do not print anything in get\_treasure())     if calling get\_treasure() passing it trow and tcol returns True then         # the treasure is picked up:         # get\_treasure should replace the “t” at index trow and tcol with a         # space if a treasure was found there         add one to tcount         tell the user that they have found a treasure, and how many they have     fi     set startrow to trow and startcol to tcol   Pool   if tcount equals num\_treasures then       congratulate the user   fi niam |
| **Figure 1: Pseudocode for the house game - navigation.** |

Implement the above algorithms and test it.  Also, implement the following functions to help you implement the above algorithm (make sure that you use them in your code!).  
  
print\_house(house, row, col) - display the house and an "@" showing where the player is. This function is given below.  
check\_north(house, row, col) - returns true if the character at house[row][col] is not a wall  
Also implement functions check\_south, check\_east, and check\_west  
get\_treasure(house, row, col) - returns true is house[row][col] is a “t”.  Also replaces the “t” with a “ “ to indicate treasure was picked up.

The code for print\_house() is below:

**import** **copy**

**def** print\_house(h,sr,sc):

th = copy.deepcopy(h)

th[sr][sc] = "@"

**for** i **in** th:

print(''.join(str(x) **for** x **in** i), end='')

**Part 2 - Handling Door and Keys**  
  
Note that in the previous algorithm, you did not handle unlocking doors.  Therefore, in this part of the of the assignment, you will implement this missing feature.  
  
First, you need a data structure to keep track of what doors you can unlock (i.e. what keys you have found).  So, at the top of your main() function, set a variable called unlocked to a list containing five False values.   
  
Next, you will add the Python code that handles picking up keys and unlocking doors. A good spot to handle going through a door is to add code under the comment in the algorithm above that starts with three stars, i.e.  
# \*\*\* right now we are not going to handle doors or keys  
  
So, below that comment, add Python code that implements the following pseudocode.

|  |
| --- |
| if calling function is\_door() and passing it house, trow, and tcol is True      # Note: doors are 5-9, keys are 0-4.  Key 0 unlocks door 5, so if we are      #       trying to go through door 5, then the value of unlocked at index 0      #       should be True if we have already found the key that unlocks the door             if calling can\_unlock and passing it house, unlocked, trow, and tcol returns False.  # player can’t go through the door         set trow to startrow and tcol = startcol   # Keeps player from moving         tell player “Sorry, the door is locked and you do not have the key"      otherwise tell the player ”You unlocked the door.") else if calling get\_key() and passing it house, unlocked, trow, and tcol returs True         # found a key!         # get\_key() should set unlocked at index house[trow][tcol] to True         #   and replace character at index trow and tcol of house to a space         tell player “You found a key!” else # do the code for treasure in the above algorithm |
| **Figure 2: Pseudocode for the house game - doors and keys** |

When adding code to your Python program to implement the above modifications to the algorithm, implement the following functions and use them in your implementation of the pseudocode.  
is\_door(house,row,col) - returns true is the character at house[row][col] is a door (from 5 to 9), returns false otherwise.  
get\_key(house,unlocked,row,col) - Sets the value at the appropriate index of unlocked to True for the number found at house[row][col], if it is a number between 0 and 4 inclusive.  In other words, it sets unlocked[house[row][col]] to True if house[row][col] is a number between 0 and 4.  It also replaces house[row][col] with a space to represent that the key has been picked up.  
can\_unlock(house,unlocked,row,col) - returns true if unlocked for the corresponding door at house[row][col] is true.  Remember to subtract 5 from the door number to index into the unlocked list at the appropriate place.

**Turn In**

Below is a table describing what your maximum grade is depending on how much of the assignment you compete.   In the table below "did not use all the functions" means the you may have implemented the code, but you did not put it in a callable function. Turn in your Python script via Moodle.

|  |  |
| --- | --- |
| **Accomplishment** | **Max Grade** |
| Your program does not run, or runs with syntax errors | 0 |
| Did part 1, but did not use all the functions described in part 1 | 79 |
| Did part 1, used all the functions in part 1, but did not implement part 2. | 89 |
| Did part 2, but did not use all the functions (some missing from part 1, part 2, or both) | 89 |
| Did part 2 and used all the functions in part 1 and part 2 | 100 |