MyAdvent News

I wrote Python3 based code assuming that I would be the only person to ever use it, but I added several features just in case. It may be that these words will be found and someone sharing my interests will use my code to learn what I did. This code **does not work** with Python 2. It does not run within the Python IDLE debugger (F5). It only runs from a shell. ("python my_advent3.py") See the description under the "feel of text input" for why.

Features Unique to This Implementation

When I wrote an 8080 assembly version of this 350 point Adventure game in 2011, I had to keep an eye on the byte size goal (32 kilobytes) and the complete responsibility of low level coding. There were no fancy libraries for I/O or string manipulation. With an almost infinite set of Python libraries, feature creep is fun and it speeds up the deployment. The "news" command describes most of what I added as I wrote the code. The assembly version did have some of these features (f, b, load, save, room).

```
Adventure> news
This is a python 3.x based Colossal Cave Adventure.
It was tested on 32 bit Linux Raspberry PI OS
'forest door' runs the automated cave test

Special notes in this version:
 'load' restores the game state from a game 'save'
 'look' shows the current room long description without admonishment
 'map' shows the nearby room map moves.
 'map enable' displays a map with every move.
 'map disable' turns off automatic map display.
 'map all' shows moves for all rooms.
 'room' lists all possible moves not blocked by magic for the current room
 'save' preserves the current game state in a file named 'myadvent.sav'
 'jump' might be deadly in some rooms with pits.
Magazine magic can map the All Alike maze. Get one magazine, drop 12 monthly
editions.
TAB autocompletes (b)uilding, (d)ownstream, (f)orest, (g)rate, (i)nventory,
(r)eservoir, (v)ariables, (q)uit

'cheat' enables and disables commands for game design study.
Enter 'y' to see more about the 'cheat' commands.
```

Commands that are available in 'cheat' mode.
 'f' moves forward one room by internal room number.
 'b' moves back one room by internal room number
 'jump' moves to a specific room number. Jump is dangerous when not in debug mode.
 'words' lists all action and object words
 'variables' lists most internal game variables
 'showrooms' lists all unvisted rooms with room numbers.
Cheat mode adds these room displays:
- The internal map room number with every move.
- Pirate movements when in rooms 50-59.
- Dwarf movements and attack countdown when in rooms 40-45.

```
You are outside building.
Adventure>
```

The "feel" of text input

The original text based game had a special "feel" to the user interface. Some commands were only recognized after an "enter/return" keystroke. Some commands were laborious to type and a unix "autocomplete" capability was desirable. Some commands, in the spirit of the game "feel", needed to be a single keystroke, like the first and last prompts that respond to a single "y" or "n".

```
Welcome to Adventure! Would you like instructions?

You are standing at the end of a road before a small brick building. Around
you is a forest. A small stream flows out of the building and down a gully.
Adventure>

You are at east end of Hall of Mists.
ADVenture> quit
Are you sure you want to quit?
Your current score is 295 out of 350 after 277 moves.
Rooms= 119/125  Treasures= 96/120  Puzzles= 80/100  Endgame= 0/5
The next rank is 340 points.
Your score puts you in Master Adventurer class B.
```

In Python 3, there are two separate "getch" functions (Linux and Windows) that collect single keystrokes without the "enter/return" burden of "raw_input". "getch" does not work if the code is run from IDLE, which has its own keystroke capture mechanism for IDE controls. As a result, this game only runs from a command line, e.g. "python my_advent.py". The code would start the game and quickly find an exception, so it is coded to block execution in IDLE.

```
Welcome to Adventure! Would you like instructions?
Please run me on the command line, not in IDLE.
>>>
```

I implemented a Linux-like autocomplete for some of the commands. One types a single letter, followed by a "tab" key. The code rewrites the command prompt and command line, waiting for an enter/return keystroke. On a personal note, this helps with some of the commands that I eternally mistype…

```
TAB autocompletes (b)uilding, (d)ownstream, (f)orest, (g)rate, (i)nventory,
(r)eservoir, (v)ariables, (q)uit
```

Navigating the Cave

While I have found illustrated "Colossal Cave" maps on the Internet, I thought it would be useful to embed a map into the game. Below is an automated display of the nearby rooms, showing the current player location. The notation of compass directions and room numbers show the possible moves available. When "Outside Building" (room 0), the player can go "east" (with "e" or "east") and get to Inside Building (room 1) and the player can go "west" (with "w" or "west") and get to End of Road (room 2).

```
You are outside building.
--------------------------------------------------------------------------------
| 2 End of Road          * 0 Outside Building      * 1 Inside Building       |
| E0,W3,S6,N5,U3          * E1,W2                   * W0                     |
--------------------------*************************----------------------------
| 4 Forest near road      | 5 Open Forest          | 3 Hill in Road          |
| E6,S5                   | E4,N4                  | E2,W4,S5                |
--------------------------------------------------------------------------------
| 6 Valley                | 7 Slit in Streambed    | 8 Outside Grate         |
| E5,W5,S7,N2             | S8,N6                  | N7                      |
--------------------------------------------------------------------------------
Adventure>
```

There is a lot of logic running with every move in the game.

Dwarf Logic

The nasty little dwarf can kill the player, resetting the game. When the player moves, the dwarf appears in a random range of rooms 40 through 46. After 7 dwarf encounters of the player in the same room, the dwarf throws his axe at the player. There is a one in three chance that the axe hits the player and resets the game. If the dwarf misses, the axe is available to the player for a future dwarf encounter. There is a one in three chance that a player kills a dwarf in the room by throwing the axe. In the debug (a.k.a cheat) mode, one can watch the dwarf logic in play.   I did wander from the original game feel for player death. The original game asks the player for an optional re-incarnation, my logic skips that.

```
Adventure> e
You cannot get past the snake.
A huge green fierce snake bars the way!
dwarf now in room 40 attack downcounter= 1
A dwarf just walked around a corner, saw you, and threw a little axe at you!
Oh dear. The nasty little axe hit you!
 X
  X
   X
    X
     X
      X
      X
      X
      X
      X
    XXXXX
Oh dear, You have fallen into a pit.
You have broken everything including the scoreboard!
Poof!
You are standing at the end of a road before a small brick building. Around
you is a forest. A small stream flows out of the building and down a gully.
Adventure>
```

Bear Logic

The 800 pound bear befriends the player when the gold chain treasure is unlocked. The bear follows the player until it is told to go away.
If the player attempts to cross the rickety bridge, the 800 pound bear will follow, which destroys the bridge. There is a one in three chance that the rickety bridge collapse will kill the player and reset the game.

After the rickety bridge collapse, if the player survives, the rooms beyond the rickety bridge are no longer accessible in the game.

```
You are walking along a gently sloping north/south passage lined with oddly
shaped limestone formations.

Adventure> e
You are standing at the entrance to a large, barren room. A sign posted above
the entrance reads: "Caution! Bear in room!"

Adventure> e
The bear is locked to the wall with a golden chain!
You are in the barren room. The center of the room is completely empty except
for some dust. Marks in the dust lead away toward the far end of the room.
The only exit is the way you came in.
There is a ferocious cave bear eying you from the far end of the room!

Adventure> feed bear
The hungry bear eats your food.
There is a very large brown bear here.
You are inside a barren room.
There is a very large brown bear here.

Adventure> unlock chain
There is a contented-looking bear wandering about nearby.
You are inside a barren room.
There is a golden chain here!

Adventure> take chain
chain taken.
You are inside a barren room.

Adventure> w
You are being followed by a very large, tame bear.
You are at entrance of the barren room.

Adventure> w
You are being followed by a very large, tame bear.
You are in limestone passage.

Adventure> n
You are being followed by a very large, tame bear.
You are in a corridor. The path forks here. The left fork leads northeast.
```

```
A dull rumbling seems to get louder in that direction. The right fork leads
southeast down a gentle slope. The main corridor enters from the west.

Adventure> w
You are being followed by a very large, tame bear.
You are in a long, narrow E/W corridor stretching out of sight to the west.
A faint rumbling noise can be heard in the distance.

Adventure> w
You are being followed by a very large, tame bear.
A rickety wooden bridge extends across the chasm, vanishing into the mist.
A sign posted on the bridge reads, "STOP! Pay troll!"
You are on the far side of the chasm. A NE path leads away from the chasm
on this side.

Adventure> cross bridge
A burly troll stands by the bridge and insists you throw him a treasure before
you may cross.
You are on NE side of chasm.

Adventure> throw chain
The troll catches the chain and scurries away.
You are on NE side of chasm.

Adventure> cross bridge
Just as you reach the other side, the bridge buckles beneath the weight of the
bear,
which was still following you around.  You scrabble desperately for support, but as
the bridge collapses, you stumble back and fall into the chasm.
 X
  X
   X
    X
     X
      X
      X
      X
      X
      X
    XXXXX
Oh dear, You have fallen into a pit.
You have broken everything including the scoreboard!
Poof!
```

```
You are standing at the end of a road before a small brick building. Around
you is a forest. A small stream flows out of the building and down a gully.
Adventure>
```

Pirate Logic

This is a 350 point game, but one room point requires that a randomly placed Pirate rob the player.
The Pirate robs the player when randomly appearing in the same room for the third time.
The first two encounters are indicated with a message

```
There are faint rustling noises from the darkness behind you.
```

A Pirate robbery moves all treasures carried by the player to the Pirate Lair. The Pirate Lair is accessible from the All Alike Maze only after a robbery occurs. Without a robbery, the best a player can achieve is 124 out of 125 room visits in the Endgame. When robbed, the player can fully recover all of the treasures for a perfect score, provided some that there are efficient moves to the Pirate Lair to avoid game duration limits.

```
You are at NE end.
Adventure> showrooms
List of rooms not yet visited.
------------------------------
room 48 Pirate Lair
You are at NE end.

Adventure> score
Your current score is 349 out of 350 after 277 moves.
Rooms= 124/125  Treasures= 120/120  Puzzles= 100/100  Endgame= 5/5
You are at NE end.
current room= 123

Adventure>
```

Testing

I wanted to enjoy devising the game without burning out with too many test runs, playing it by hand. I invested some time in automated testing. I found a lot of bugs with this feature.   The test run goes to all but one room (Pirate Lair), runs all puzzles, collects all treasures and lands in the end game phase. The “forest” magic word is a simple function with 200 additional lines of Python code to implement the automated “forest door” testing. The automated test can only be run once for multiple reasons.

- At the end of a test run, all the treasures have been dropped in the Building.
- After 382 turns the lantern batteries die during the second run and the player falls in a pit
- Eventually the Dwarf lands the axe, killing the player.
- I had to disable the Pirate, he always managed to interfere.

```
You are standing at the end of a road before a small brick building. Around
you is a forest. A small stream flows out of the building and down a gully.
Adventure> forest door
############################################################
Start Cave Testing. Hit Enter/Return to start or stop.

(382 tests occur here)

Cave Test Complete.
############################################################
You are at NE end.
Adventure> score
Your current score is 349 out of 350 after 277 moves.
Rooms= 124/125  Treasures= 120/120  Puzzles= 100/100  Endgame= 5/5
You are at NE end.
Adventure> showrooms
List of rooms not yet visited.
------------------------------
room 48 Pirate Lair
You are at NE end.
Adventure>
```

While developing the tests, I wanted to see the logic turn by turn without hundreds of keystrokes. On the Mac, hundreds of tests run instantly, so I added a 2 second delay to permit passive observation. I also added a keyboard interrupt routine so that I did

not have to wait hundreds of seconds on a dead-end run or lose my program state with a control/c. Python execution on Windows 10 is far slower, making the 2 second delay unnecessary and the keyboard interrupt quite valuable.

```
    if True == kbhit():
        break
        time.sleep(2)
#
# detect a keystroke to interrupt forest door
# Returns True if a keypress is waiting to be read in stdin, False otherwise.
# code runs on Linux, Windows support ripped out https://stackoverflow.com/
questions/292095/polling-the-keyboard-detect-a-keypress-in-python
#
def kbhit():
    dr,dw,de = select.select([sys.stdin], [], [], 0)
    return dr != []
```

Game Duration

The original game design limited the number of player turns and the logic for this implementation interprets things in a similar way. After 300 turns, the lamp is turned off, assuring doom after 3 additional turns that send the player into a pit. If the coins treasure is given up, the player may continue, doomed to a less than perfect score.

```
#
# lamp life test
#
    if myvars[159] == 230:                  # 230 turns taken
        mysentence(mysent[436])              # lamp getting dim
    if myvars[159] == 265:                  # 265 turns taken
        mysentence(mysent[437])              # lamp is almost dead
    if myvars[159] > 300 and myvars[142] != 130:  # 300 turns, no coins used?
        mysentence(mysent[438])              # lamp is now dead
        myvars[158] = 4                     # lamp set to dark
```

Once the player has accumulated 12 of the treasures in the Building, the game logic forces the player toward the end game

even if the remaining treasures are not collected.

```
#
# cave closure test for 12 found treasures in Well House, magazine does not count
#
    count = 0
    treasures = [142,143,144,145,146,147,148,149,150,152,153,154,155,156,157]
    for trophy in treasures:
        if myvars[int(trophy)] == 1:  # this treasure found?
            count += 1
    if count >= 12:                   # 12 treasures in well house?
        myvars[171] += 1              # indicate cave is closing
    if myvars[171] > 9:               # every ten turns, announce closure
        myvars[171] = 1               # reset counter
        mysentence(mysent[469])       # cave is closing
```

Words and Sentences

There are many print statements in the code, but the item count only represents a small fraction of the displayed statements. In the earlier assembler version, I built a library of words and the sentences were named word lists. In assembly, this was a very efficient compression of the memory footprint with almost no processing overhead after the code was assembled. This collection of words and sentences were ported to Python with a few tool scripts. Examination of the code will not directly show most of the program run time sentences. It keeps some of the game intrigue and mystery in place despite access to the Python source code.