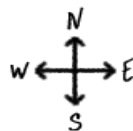
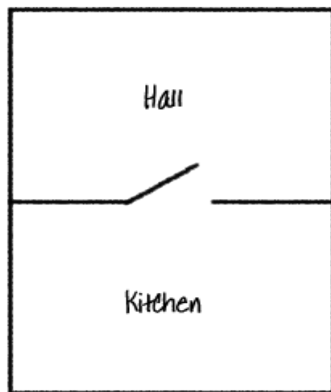


Adventure

"Colossal Cave Adventure" is a very old computer game – written in 1976. It is based on the idea of typing commands to navigate your way around a "world", collecting items and solving problems on the way. This class of game is called a Role Playing Game (RPG). This week, we'll use a very simple RPG and modify it. If we have time we'll also play Colossal Cave Adventure.

The aim of our RPG is to collect objects and escape from a house, making sure to avoid all the monsters!

Start up IDLE, then use `File->Open` and navigate to the `code_club/python` directory and open the file `RPG`. Now, make your own copy by using `File->Save as`. I called mine `RPG_neal`. If you get stuck, ask help. Now, press `F5` to run your program. Here's a map of the game so far:



It only has 2 rooms. You can type `go south` to move from the hall to the kitchen, and then `go north` to go back to the hall again!

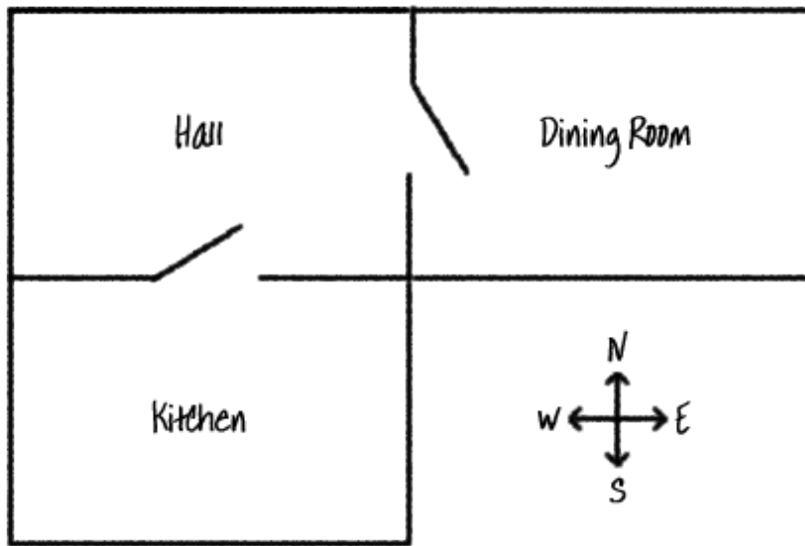
You can also see what happens when you type in a direction that you cannot go. For example, there is no room to the west of the hall, so if you type `go west`, you'll get a friendly error message.

If you edit the `RPG.py` file, you can see that the map is coded as a dictionary of rooms:

```
#a dictionary linking a room to other room positions
rooms = {
    1 : { "name" : "Hall" ,
          "south" : 2
        } ,
    2 : { "name" : "Kitchen" ,
          "north" : 1
        }
}
```

Actually, this is a dictionary that links a room number to another dictionary, containing all of the information about the room. For example, room 1 in the code above is the hall. The hall is linked to room 2 (the kitchen) to the south. Room 2 (the kitchen) also links to room 1 (the hall) to the north.

Let's add another room to your map (a dining room), to make it a bit more interesting!



This new dining room is linked to the hall (to the west). Let's add this new room into the code (the code you need to add is shown in red):

```
#a dictionary linking a room to other room positions
rooms = {
    1 : { "name" : "Hall" ,
          "south" : 2 ,
          "east" : 3
        } ,
    2 : { "name" : "Kitchen" ,
          "north" : 1
        } ,
    3 : { "name" : "Dining Room",
          "west" : 1
        }
}
```

To add this room to the game, notice that you need to add the new room (room number 3), and give it a name. You also need to link to room 1 (the hall) to the west of the new room. You need to add information to the dictionary for the hall, to allow you to enter the dining room to the east.

LOOK at how the braces {} match up in pairs, and CHECK that you have added the two commas (highlighted above in yellow).

Run the program and make sure that you can now move into and out of the new dining room. If you can't or your program doesn't run, CHECK the commas again! Ask for help if you get stuck.

Extra: Add two more rooms to the house, both opening off the hall. Sketch them on the map first to help you. Run the program with your changes and make sure it works as you expect. Ask for help if you get stuck.

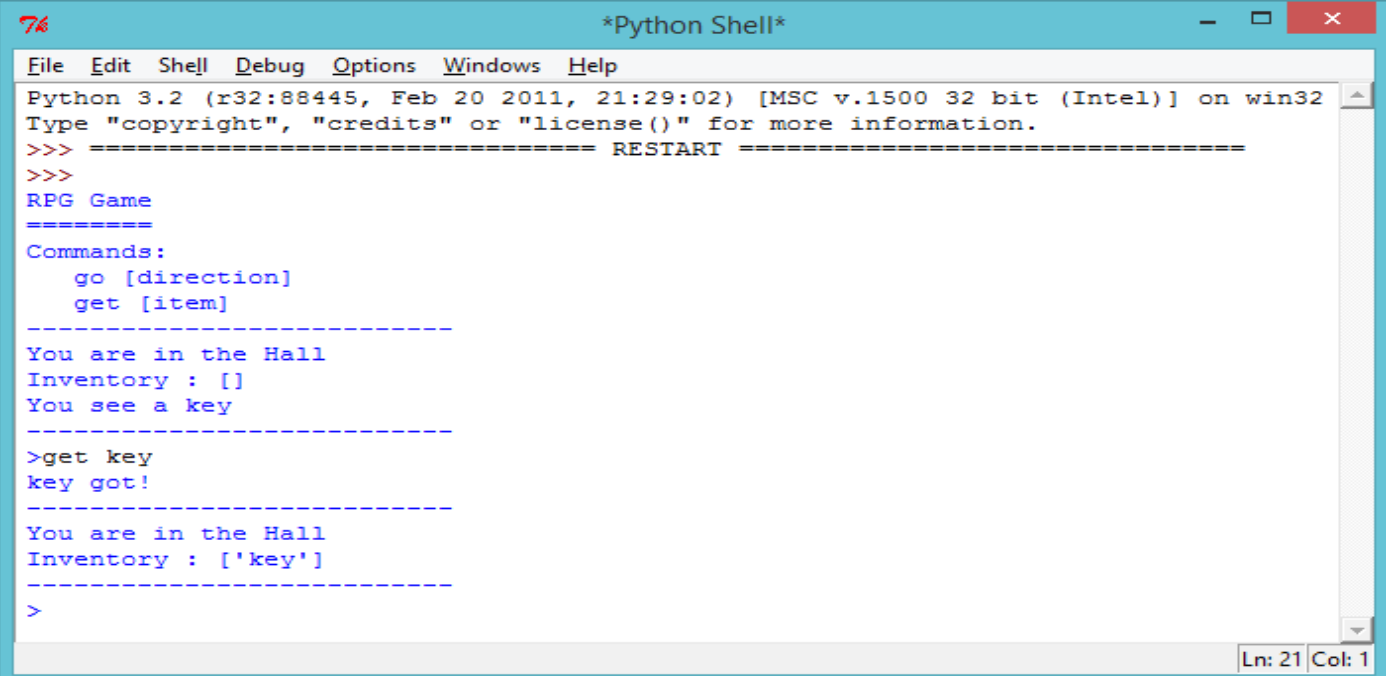
Now that you have lots of rooms, let's leave items in the rooms for the player to collect as they move through the maze.

Adding an item into a room is easy. You can just add it to the dictionary of a room. For example, let's put a key in the hall (the new code is shown in red):

```
#a dictionary linking a room to other room positions
rooms = {
    1 : { "name" : "Hall" ,
          "south" : 2 ,
          "east" : 3 ,
          "item" : "key"
        } ,
    2 : { "name" : "Kitchen" ,
          "north" : 1
        } ,
    3 : { "name" : "Dining Room",
          "west" : 1
        }
}
```

Remember to put a comma after the line above the new item, or your program won't run!

If you run your game after adding the code above, you can now see a key in the hall, and you can pick it up (by typing `get key`) and add it to your inventory.



```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
RPG Game
=====
Commands:
    go [direction]
    get [item]
-----
You are in the Hall
Inventory : []
You see a key
-----
>get key
key got!
-----
You are in the Hall
Inventory : ['key']
-----
>
```

Extra: Add an item to some of the rooms in your game. You can add anything that you think would be helpful in trying to escape the house! For example, a shield or a torch or a potion. Our program is quite simple, you can only use one word to name the item (For

example, "Potion" but not "Magic potion"). Run your modified program and check that it works as you expect.

This game is too easy! Let's add enemies to some rooms that the player must avoid.

Adding an enemy to a room is as easy as adding any other item. Let's add a hungry monster to the kitchen (the new code is in red):

```
#a dictionary linking a room to other room positions
rooms = {
    1 : { "name" : "Hall" ,
          "south" : 2 ,
          "east" : 3 ,
          "item" : "key"
        } ,
    2 : { "name" : "Kitchen" ,
          "north" : 1 ,
          "item" : "monster"
        } ,
    3 : { "name" : "Dining Room",
          "west" : 1
        }
}
```

Also, we want to make sure that the game ends if the player enters a room with a monster in. You can do this with the following code, which you should add to the end of the game:

```
#player loses if they enter a room with a monster
if "item" in rooms[currentRoom] and "monster" in
rooms[currentRoom]["item"]:
    print("A monster has got you... GAME OVER!")
    print("Press any key to quit")
    input()
    break
```

This code checks whether there is an item in the room, and if so, whether that item is a monster. Notice that this code is indented, putting it in line with the code above it. This means that the game will check for a monster every time the player moves into a new room.

Test out your code by going into the kitchen, which now contains a monster. If you get stuck, ask for help.

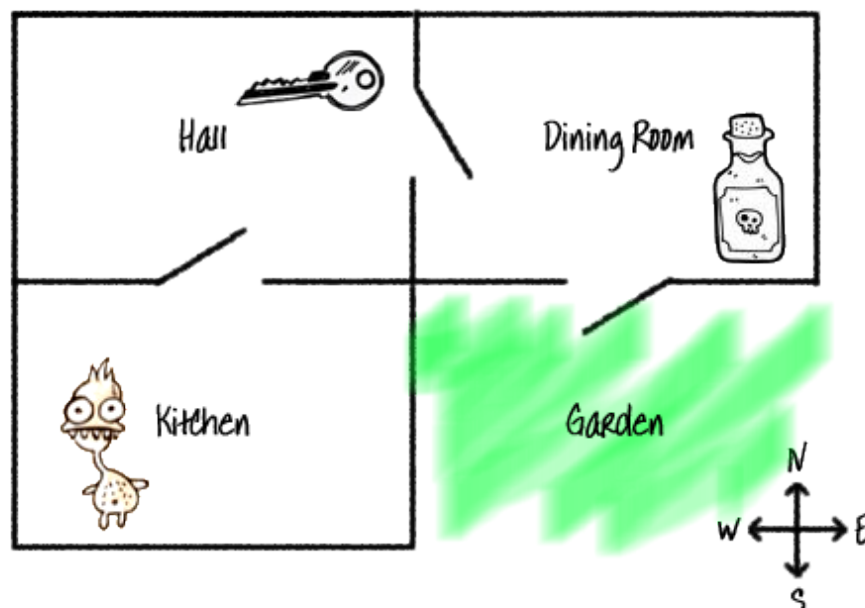
```
76 *Python Shell*
File Edit Shell Debug Options Windows Help
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
RPG Game
=====
Commands:
  go [direction]
  get [item]
-----
You are in the Hall
Inventory : []
You see a key
-----
>go east
A monster has got you... GAME OVER!
Press any key to quit
Ln: 18 Col: 0
```

Extra: Add more monsters to your game, to make it harder to escape the house! Remember that you need at least one route through the house with no monster, otherwise the game is impossible!

Finally, let's give the player a mission, which needs to be completed to win the game.

Here's our mission: *In this game, the player wins by getting to the garden and escaping the house. They also need to have the key with them, and the magic potion.*

Here's a map of the game (sketch in any extra rooms that you've added):



Notice that there's another fourth "room" (the garden) that links to the dining room (to the north).

Use the examples above to add a garden to your game. Remember to add doors, to link to other rooms in the house.

Use the examples above to add a potion to the dining room (or another room in your house).

To allow the player to win the game when they get to the garden with the key and the potion, add this code to the end of your game:

```
#player wins if they get to the garden with a key and a potion
if currentRoom == 4 and 'key' in inventory and 'potion' in
inventory:
    print("You escaped the house... YOU WIN!")
    print("Press any key to quit")
    input()
    break
```

Again, make sure this code is indented, and in line with the code above it. This code means that the message ...YOU WIN! is displayed if the player is in room 4 (the garden) and if the key and the potion are in the inventory. If you have more than 4 rooms, you may have to use a different room number for your garden in the code above.

Test your game to make sure the player can win!

As a final step, add some instructions to your game, so that the player knows what they have to do. Edit the `showInstructions()` function to include more information:

```
def showInstructions():
    #print a main menu, the commands and instructions
    print('''
RPG Game
=====
Instructions:
    Get to the garden
    ...
=====
Commands:
    go [direction]
    get [item]
''')
```

Change the instructions to tell the user what items they need to collect, and what they need to avoid!

Extra: Use what you've learnt to create your own game. Add lots of rooms, monsters to avoid and items to collect. Remember to modify the code so that the player wins when

they get to a certain room with some of the objects in their inventory. It may help you to sketch a map before you start coding! You could even add stairs to your map and have more than one level of rooms, by typing `go up` and `go down`. (**Hint:** you can use "up" and "down" in exactly the same way as the directions north/south/east/west)

Extra: Play the original Colossal Cave Adventure. From the python prompt:

```
>>> import sys
>>> sys.path.append("S:/Coding Club/python/adventure-1.2")
>>> import adventure
>>> adventure.play()
```

If you get totally stuck you can start again like this:

```
>>> adventure.play()
```

This was derived from a codeclub.org.uk project. codeclub.org.uk projects are for use inside the UK only. All Code Clubs **must** be registered. You can check registered clubs on the map at www.codeclub.org.uk. This coursework is developed in the open on GitHub (github.com/CodeClub), come and join us!

Original © 2014 Code Club. This work is licensed under a [BY-NC-SA 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).

Changes © 2014 Neal Crook. CC-BY-SA