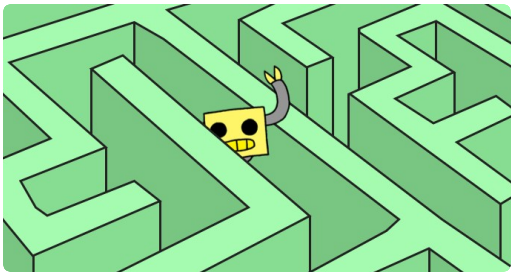
RPG

Design and code your own RPG maze game.

Python



Step 1 Introduction:

In this project, you'll design and code your own RPG maze game. The aim of the game will be to collect objects and escape from a house, making sure to avoid all the monsters!

```
RPG Game
========

Get to the Garden with a key and a potion
Avoid the monsters!

Commands:
  go [direction]
  get [item]

----------------------------
You are in the Hall
Inventory : []
You see a key
----------------------------
> go east
----------------------------
You are in the Dining Room
Inventory : []
You see a potion
----------------------------
>
```

Additional information for club leaders

If you need to print this project, please use the **Printer friendly version** (https://projects.raspberrypi.org/en/projects/rpg/print).

Club leader notes

Introduction:

This project teaches game design through the development of an RPG maze game. In this game, the player has to pick up objects within a house and get to a specific room, while avoiding monsters lurking in some of the rooms. This game will be achieved by manipulating dictionaries and lists.

Online Resources

This project uses Python 3. We recommend using **trinket** (https://trinket.io/) to write

Python online. This project contains the following Trinkets:

- **'RPG' starting point – jumpto.cc/rpg-go** (http://jumpto.cc/rpg-go)

There is also a trinket containing the finished project:

- **'RPG' Finished – trinket.io/python/d06adeb527** (https://trinket.io/python/d06adeb527)

Offline Resources

This project can be **completed offline** (https://www.codeclubprojects.org/en-GB/resources/python-working-offline/) if preferred. You can access the project resources by clicking the 'Project Materials' link for this project. This link contains a 'Project Resources' section, which includes resources that children will need to complete this project offline. Make sure that each child has access to a copy of these resources. This section includes the following files:

- rpg/rpg.py

You can also find the completed project project in the 'Volunteer Resources' section, which contains:

- rpg-finished/rpg.py

(All of the resources above are also downloadable as project and volunteer `.zip` files.)

Learning Objectives

- Game design;
- Editing:
  - Lists;
  - Dictionaries.
- Boolean expressions.

This project covers elements from the following strands of the **Raspberry Pi Digital Making Curriculum** (http://rpf.io/curriculum):

- **Combine programming constructs to solve a problem.** (https://www.raspberrypi.org/curriculum/programming/builder)

Challenges

- Adding new rooms;
- Adding items to collect;
- Adding enemies to avoid;
- Develop your own game.

Frequently Asked Questions

- Children may need reminding that elements of a dictionary/list are separated by a comma. For example, when adding a new room to the 'rooms' dictionary, a comma needs to be added between the new room being added and the previous room.
- When adding a new room, children may forget to add a link to an existing room to the newly created room. This will mean that children can leave a room, but not enter it!
- The code for checking whether the player has won or lost the game needs to be indented, to ensure that this check is performed upon entering each new room. If the code isn't indented, then it sits outside of the main game loop and is never run.

Project materials

Project resources

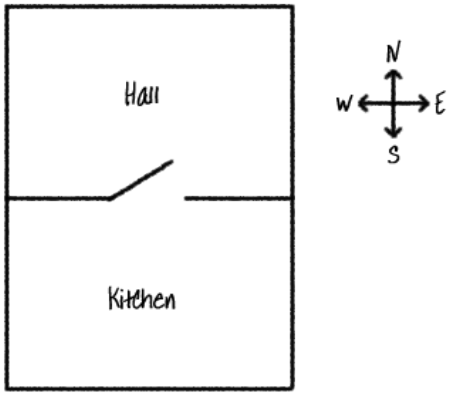
- **.zip file containing all project resources** (https://projects-static.raspberrypi.org/projects/rpg/0a3a5c2bc39a2ccbaba225a99ac39bf0bf0eb1b0/en/resources/r project-resources.zip)
- **Online Trinket containing all 'RPG' project resources** (http://jumpto.cc/rpg-go)
- **rpg/rpg.py** (https://projects-static.raspberrypi.org/projects/rpg/0a3a5c2bc39a2ccbaba225a99ac39bf0bf0eb1b0/en/resources/r rpg.py)

Club leader resources

- **.zip file containing all completed project resources** (https://projects-static.raspberrypi.org/projects/rpg/0a3a5c2bc39a2ccbaba225a99ac39bf0bf0eb1b0/en/resources/r volunteer-resources.zip)
- **Online completed Trinket project** (https://trinket.io/python/d06adeb527)
- **rpg-finished/rpg.py** (https://projects-static.raspberrypi.org/projects/rpg/0a3a5c2bc39a2ccbaba225a99ac39bf0bf0eb1b0/en/resources/r finished-rpg.py)

Step 2 Adding new rooms

- Some code for this game has been provided for you. Open this trinket: **jumpto.cc/rpg-go** (http://jumpto.cc/rpg-go).
- This is a very basic RPG game that only has 2 rooms. Here's a map of the game:



You can type `go south` to move from the hall to the kitchen, and then `go north` to go back to the hall again!

```
RPG Game
========
Commands:
  go [direction]
```

```
  get [item]

----------------------------
You are in the Hall
Inventory : []
----------------------------
> go south
----------------------------
You are in the Kitchen
Inventory : []
----------------------------
> go north
----------------------------
You are in the Hall
Inventory : []
----------------------------
>
```

- What happens when you type in a direction that you cannot go? Type `go west` in the hall and you'll get a friendly error message.

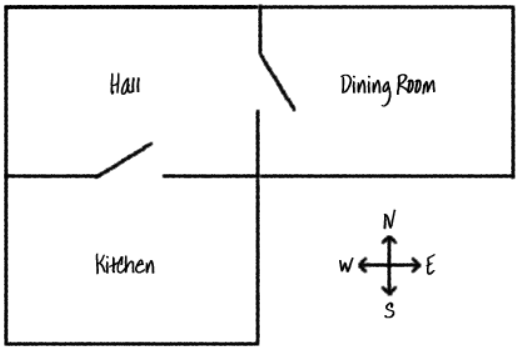
```
> go west
You can't go that way!
----------------------------
You are in the Hall
Inventory : []
----------------------------
>
```

- If you find the `rooms` variable, you can see that the map is coded as a dictionary of rooms:

```
#a dictionary linking a room to other rooms
rooms = {

            'Hall' : {
                    'south' : 'Kitchen'
                },

            'Kitchen' : {
                    'north' : 'Hall'
                }

        }
```

Each room is a dictionary and rooms are linked together using directions.

- Let's add a dining room to your map, to the east of the hall.



You need to add a 3rd room, called the `dining room`. You also need to link it to the hall to the west. You also need to add data to the hall, so that you can move to the dining room to the east.

```
#a dictionary linking a room to other rooms
rooms = {

            'Hall' : {
                    'south' : 'Kitchen',
                    'east'  : 'Dining Room'
                },

            'Kitchen' : {
                    'north' : 'Hall'
                },

            'Dining Room' : {
                    'west'  : 'Hall'
                }

        }
```

- Try out the game with your new dining room:

```
RPG Game
========
Commands:
  go [direction]
  get [item]

----------------------------
You are in the Hall
Inventory : []
----------------------------
> go east
----------------------------
You are in the Dining Room
Inventory : []
----------------------------
> go west
----------------------------
You are in the Hall
Inventory : []
----------------------------
>
```

If you can't move in and out of the dining room, just check that you added all of the code above (including the extra commas to the lines above).

Step 3 Challenge: Add new rooms

Can you add more rooms to your game? For example, you could create a living room to the south of the dining room. Remember to add a door to/from one of the other rooms!

Step 4 Adding items to collect

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 a room's dictionary. Let's put a key in the hall.

```
#a dictionary linking a room to other rooms
rooms = {

            'Hall' : {
                  'south' : 'Kitchen',
                  'east'  : 'Dining Room',
                  'item'  : 'key'
                },

            'Kitchen' : {
                  'north' : 'Hall'
                },

            'Dining Room' : {
                  'west'  : 'Hall'
                }

        }
```

  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 even pick it up (by typing `get key`) which adds it to your inventory!

```
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']
----------------------------
>
```

Step 5 Challenge: Add new items

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 magic potion.

Step 6 Adding enemies

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:

```
#a dictionary linking a room to other rooms
rooms = {

            'Hall' : {
                  'south' : 'Kitchen',
                  'east'  : 'Dining Room',
                  'item'  : 'key'
                },

            'Kitchen' : {
                  'north' : 'Hall',
                  'item'  : 'monster'
                },

            'Dining Room' : {
                  'west'  : 'Hall'
                }

        }
```

- You also 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:

```
    #otherwise, if the item isn't there to get
    else:
      #tell them they can't get it
      print('Can\'t get ' + move[1] + '!')

# 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!')
    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.

```
RPG Game
========
Commands:
  go [direction]
  get [item]

---------------------------
You are in the Hall
Inventory : []
You see a key
---------------------------
> go south
A monster has got you... GAME OVER!
```

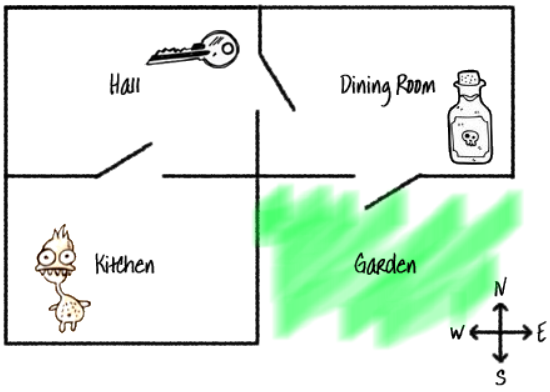
Step 7 Challenge: Adding more monsters

Add more monsters to your game, to make it harder to escape the house!

Step 8 Winning the game

Let's give your player a mission, which needs to completed to win the game.

- 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.



- First, you need to add a garden to the south of the dining room. Remember to add doors, to link to other rooms in the house.

```
#a dictionary linking a room to other rooms
rooms = {

            'Hall' : {
                  'south' : 'Kitchen',
                  'east'  : 'Dining Room',
                  'item'  : 'key'
                },

            'Kitchen' : {
                  'north' : 'Hall',
                  'item'  : 'monster'
                },

            'Dining Room' : {
                  'west'  : 'Hall',
                  'south' : 'Garden'
                },

            'Garden' : {
                  'north' : 'Dining Room'
                }

        }
```

- Add a potion to the dining room (or another room in your house).

```
            'Dining Room' : {
                  'west'  : 'Hall',
                  'south' : 'Garden',
                  'item'  : 'potion'
                },
```

- Add this code to allow the player to win the game when they get to the garden with the key and the potion:

```
# 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!')
  break

# player wins if they get to the garden with a key and a shield
if currentRoom == 'Garden' and 'key' in inventory and 'potion' in inventory:
  print('You escaped the house... YOU WIN!')
```

```
        break
```

Make sure this code is indented, in line with the code above it. This code means that the message `You escaped the house...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!

```
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']
---------------------------
> go east
---------------------------
You are in the Dining Room
Inventory : ['key']
You see a potion
---------------------------
> get potion
potion got!
---------------------------
You are in the Dining Room
Inventory : ['key', 'potion']
---------------------------
> go south
You escaped the house... YOU WIN!
```

- Finally, let's 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 and the commands
    print('''
RPG Game
========

Get to the Garden with a key and a potion
Avoid the monsters!

Commands:
  go [direction]
  get [item]
''')
```

You will need to add instructions to tell the user what items they need to collect, and what they need to avoid!

- Test your game and you should see your new instructions.

```
RPG Game
========

Get to the Garden with a key and a potion
```

```
Avoid the monsters!

Commands:
  go [direction]
  get [item]
```

Step 9 Challenge: Develop your own game

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`.