Father Time and the New Year Maze

Overview

Father Time, who is in charge of all things time related (days of the week, hour glasses, national holidays, etc.), has gotten lost in his time maze looking for his hour glass.

Mother Nature has told Father Time countless times in the past "Why can't you just store all your time related things in a drawer or cupboard. It doesn't need to be a maze!" Father Time doesn't think it would be as fun. Well until he gets lost that is…

What you will make

For this activity you are going to design and create a simple maze for Father Time to walk through. You are going to use some advanced Python skills to create this Object Oriented Adventure.

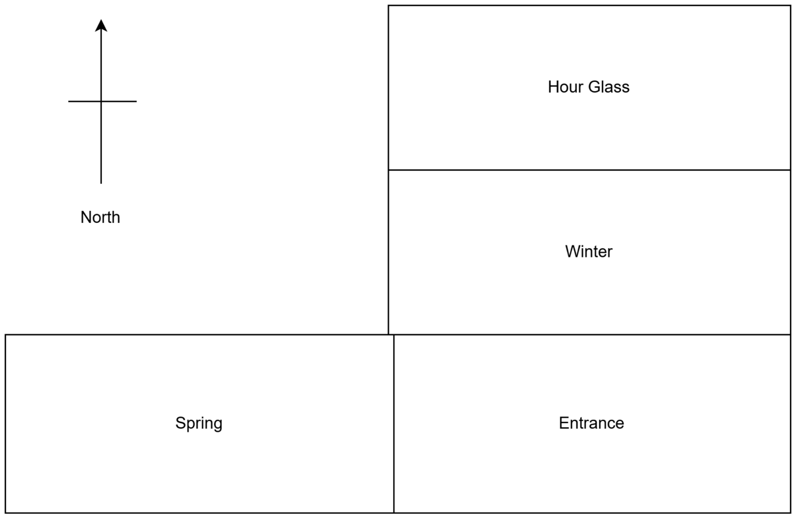
What you will need

You are going to need the following items:

- ☐ Pencil
- ☐ Paper
- ☐ The Starter Project
- ☐ A "can do" attitude

Activity 1 - Design your maze

For simplicity this tutorial is going to have a maze of 4 rooms. Yours should have about 5-7 rooms (more if you want).



In my example, The entrance is where Father Time will begin and the Hour Glass is where the goal is. I have also included a compass in the top left so you can see how we will move around:

- North
- South
- East
- West

Use your paper and pencil to draw and plan your own map. Remember to add names of your rooms and a description of what the room looks like

Activity 2 - Exploring the starter code

If you haven't already open the starter project

In there you will see two files:

- main.py - Where all the logic of our games are
- room.py - The blueprint of a room
- item.py - The blueprint for different items
- character.py - The blueprint for different characters

In main.py there is a lot of code that you can explore. We are going to do the majority of our coding here. **Look at the comments (lines starting with #) and try to explain to another creator or volunteer what this code does.**

Run the program to see if the code does what you thought. To run the code, click the play button at the top of the page.

In room.py we have code with the word class in. A class is like a blueprint that tells Python how to create an object. We use classes so that we can reuse the same code as many times as we like without having to write it again and again and again.

The **self** keyword is used a lot in our room class. This is because it helps our objects keep certain properties to it**self.** This stops other objects taking properties from other objects.

When the program asks, what will you do? If you type **exit** the program will end

Activity 3 - Creating Rooms

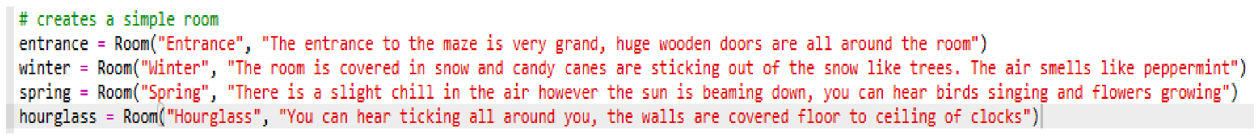
Let’s create a room. On line 8 you can create a room.

1. Create a variable name for your room (no spaces allowed)
2. Put in an equals sign (=)
3. Get python to use the room blueprint by using this code: Room()
4. Inside the brackets, put the name of your room within speechmarks
5. Place a comma
6. After the comma, put the description of your room within speechmarks

Your line of code should look something like this (The code is wrapped on multiple lines here, yours should be on one line):

winter = Room("Winter", "The room is covered in snow and candy canes are sticking out of the snow like trees. The air smells like peppermint")

Repeat the above steps for each of your rooms on your plan. Use my example below to help.



```
# creates a simple room
entrance = Room("Entrance", "The entrance to the maze is very grand, huge wooden doors are all around the room")
winter = Room("Winter", "The room is covered in snow and candy canes are sticking out of the snow like trees. The air smells like peppermint")
spring = Room("Spring", "There is a slight chill in the air however the sun is beaming down, you can hear birds singing and flowers growing")
hourglass = Room("Hourglass", "You can hear ticking all around you, the walls are covered floor to ceiling of clocks")
```

Activity 4 - Connecting your rooms

When you run your program at the moment, the entrance room will show up and it will say no rooms are connected. Let's change that!

First we need to change our Room class so that all our rooms get the ability to add new rooms.

1. Go to room.py
2. Find the line with the comment # The method that connects a room
3. Under that line add the following code

```
# The method that conencts a room
def connect_room(self, direction, room):
  self.connected_rooms[direction] = room
```

This code takes in a direction and a room, once it has them it will add the direction to a data structure called a dictionary. A dictionary uses a key-value pair. This means that we can assign different rooms to different directions.

4. Go back to main.py
5. Find the line with the comment # Connects Rooms
6. It is time for you to connect all your rooms. Use the following code to help you

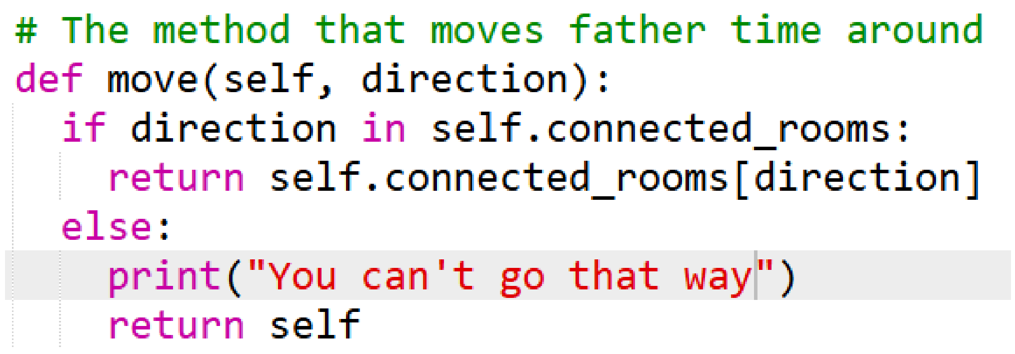
```
# Connects Rooms
entrance.connect_room("north", winter)
entrance.connect_room("west", spring)
spring.connect_room("east", entrance)
winter.connect_room("south", entrance)
winter.connect_room("north", hourglass)
hourglass.connect_room("south", winter)
```

Run your game and you should see that the entrance now has connected rooms.

Activity 5 - Moving around

Father Time is extremely happy that you have created the maze and connected all the rooms together, however he is growing more and more frustrated that he cannot move around the maze. Let's change that.

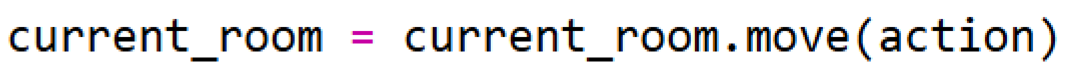
1. Go to room.py
2. Find the comment # The method that moves father time around
3. Add the following code



```python
# The method that moves father time around
def move(self, direction):
  if direction in self.connected_rooms:
    return self.connected_rooms[direction]
  else:
    print("You can't go that way")
    return self
```

The following code checks if the direction entered is within the connected rooms we defined in activity 4. If it is then it will return the new room. If it is not in the dictionary then it will return itself so that you end up in the same room.

4. See if you can find where the following code goes



```python
current_room = current_room.move(action)
```

5. **Run your game and see if you can move between your rooms**

Challenges

Now that you have the basics of:

- Making rooms
- Connecting the rooms
- Moving around the maze

Now it is time for you to add more things by yourself. Try the following challenges:

1. Add a win condition. E.g. if current_room.get_name() == “hourglass” then print(“You win!”) and playing = False
2. Use the item class to add items into your room. Tip: Create an attribute in the room class __init__ method called self.item
3. Try and make it so that Father Tiem can pick up an item
4. Use the character class to add a characters to your game, these could be enemies or friends
5. Add to the if statement in main.py to check if action is talk. If there is a character in the room then the character should talk

I keep getting stuck!

If you keep getting stuck look at my completed example for hints:
https://trinket.io/embed/python/582f342e3fef

Acknowledgements

This project is made with inspiration from the Raspberry Pi Projects:

- Escape the Maze
- Teach teens computing: Object-oriented Programming Python