map2.py

```python
# A simple program to draw a map of rooms and connections using Tkinter
# This program uses a JSON file to load room data and visualize the connections between them.
# The JSON file should contain room names and their connections in a specific format.
# The program uses Tkinter to create a graphical window and draw rectangles for rooms and lines for connections.
# The program also includes a recursive function to place rooms and draw connections based on the loaded data.

import tkinter as tk
import json

# Load room data from the JSON file
with open("game_data.json") as f:
    data = json.load(f)

rooms = data["rooms"]

directions = ["North","South","East","West","Up","Down","Out"]

# Define direction-based position offsets
direction_offsets = {
    "North": (0, -1),   #up
    "South": (0, 1),    #down
    "East": (1, 0),     #right
    "West": (-1, 0),    #left
    "Up": (1, -1),      #up-right
    "Down": (-1, 1),    #down-left
    "Out": (-5, 0)
}



# Tkinter window setup
root = tk.Tk()
root.title("Room Map Viewer")

canvas = tk.Canvas(root, width=1200, height=800, bg="white")
canvas.pack()


x0, y0 = 300, 400  # Center starting coordinates

room_coords = {}  # Store coordinates of each room to draw connections
sizex = 40 # size of box to draw
sizey = 40 # size of box to draw

# Create a Visited set . A set is unchangeable!
visited = set()


```

```python
#draw the room as a rectangle shape
def draw_room(name, x, y):
    room_coords[name] = (x, y)
    canvas.create_rectangle(x - sizex, y - sizey, x + sizex, y + sizey, fill="lightblue")
    canvas.create_text(x, y-30, text=name, font=("Arial", 8), justify="center")

    ## print items in the room
    items = rooms[name].get("Item", [])
    if items:
        item_text = items
        canvas.create_text(x, y + 10, text=item_text, font=("Arial", 7), fill="black")

    #print enemies in the room
    enemies = rooms[name].get("Enemy", [])
    if enemies:
        enemy_text = enemies
        canvas.create_text(x, y + 20, text=enemy_text, font=("Arial", 7), fill="red")

def draw_connection(from_room, to_room, direction):

    # only draw a connection if we've drawn it
    if to_room not in room_coords:  #room_coords are only added when we draw it ( see draw_room() function above )
        return

    x1, y1 = room_coords[from_room]
    x2, y2 = room_coords[to_room]
    canvas.create_line(x1, y1, x2, y2, arrow=tk.LAST)

    # draw direction in middle of line
    mid_x, mid_y = (x1 + x2) // 2, (y1 + y2) // 2
    #canvas.create_text(mid_x, mid_y - 10, text=direction, font=("Arial", 6), fill="grey")

# RECURSIVE function to place room and connections
def place_room(name, x, y):

    #if you've been here before then go back
    if name in visited:
        return

    #add room to visited set and draw
    visited.add(name)
    draw_room(name, x, y)

    # loop through all exits and draw those
    for direction, target in rooms[name].items():

        # Only draw directions
        if direction not in directions:
            continue

        # find where to draw the room - if unknown then draw on top
        dx, dy = direction_offsets.get(direction, (0, 0))
        new_x = x + dx * 110
        new_y = y + dy * 110

        #if not already drawn - then draw it
        if target not in room_coords:
            place_room(target, new_x, new_y)

        #now connect together
        draw_connection(name, target, direction)

# Start drawing from the starting room
start_room = data.get("start")
place_room(start_room, x0, y0)

root.mainloop()

```