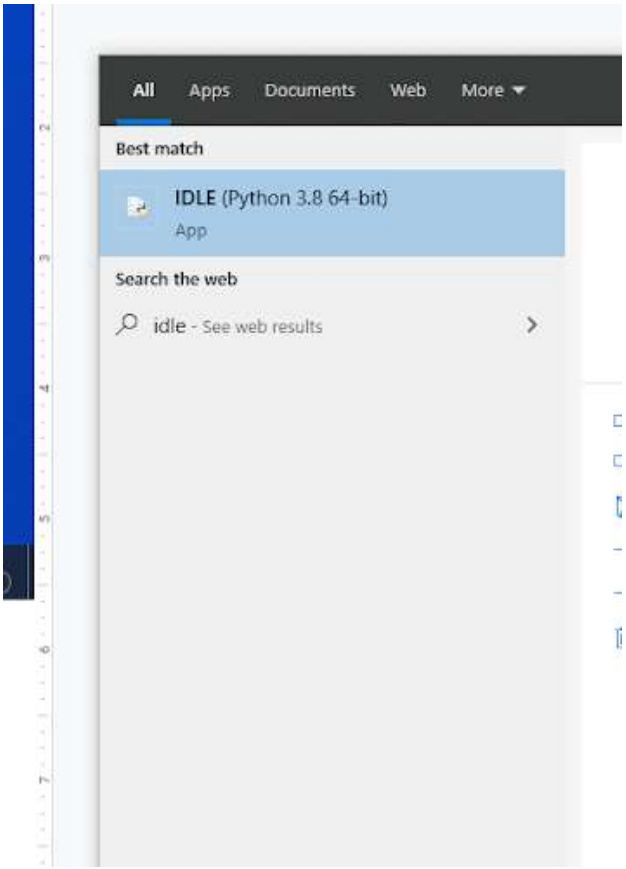
Part 2
Space Mission Directions

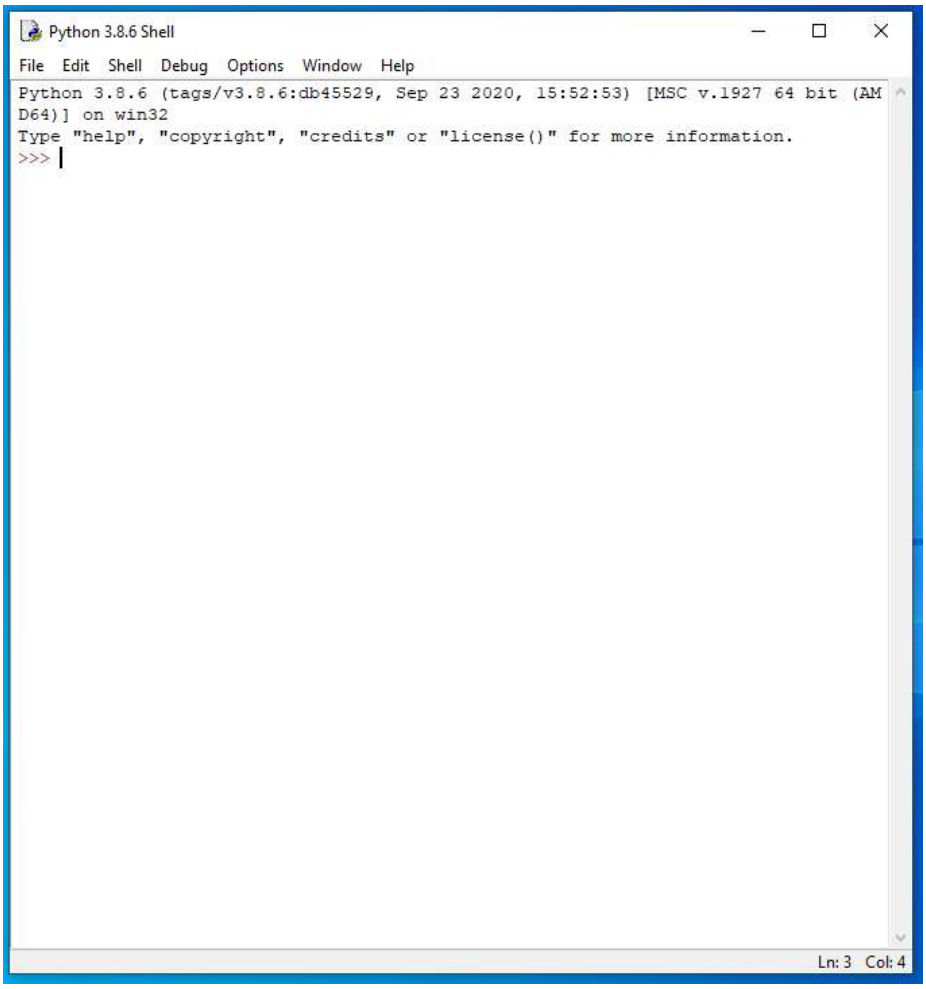
1. Navigate out to the Google Classroom for this class.
2. Locate the Space Mission Part 2 assignment.
3. Click on the, “Chapter 5 Object List Code to Copy/Paste” file that is attached to the assignment.
4. You will want this file later on in the exercise. Keep it handy.
5. We are now ready to start adding code to our file. Using your Windows button menu, find and launch your IDLE program.



IDLE is the integrated development environment associated with Python. It is made up of a code editor where you type your code along with other helpful tools that allow you to write, save, and test run programs.

IDLE is designed to recognize Python code, compile Python code, and provide basic debugging tips to programmers if there are problems with their code.

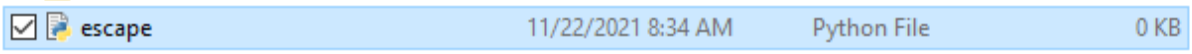
6. Your IDLE window should look something like this once it has launched.:



On Startup, IDLE will display the Python Shell, which can be used to give commands to the computer's operating system. Since we are viewing the shell through IDLE and not the actual command prompt window, the commands that we type into the Shell will not communicate directly with our operating system. However, you can type similar commands in the Python Shell directly from the Python program (not through IDLE) and, if you have permission to access the operating system's commands, you can communicate with the computer's operating system that way.

In IDLE, the shell is mainly used as a launching screen for other activities that we will do, like writing code for our game or debugging a file.

7. Go to File > Open and then browse in the Starting Files folder I gave you to find the escape python file that we have been working on.



8. Your escape.py file will open up.

9. Click at the end of Line 21.

```
12 #PLAYER variables
13 PLAYER_NAME = "Alice"
14 FRIEND1_NAME = "Jack"
15 FRIEND2_NAME = "Matthew"
16 current_room = 31 # start room = 31
17
18 top_left_x = 100
19 top_left_y = 150
20
21 DEMO_OBJECTS = [images.floor, images.pillar, images.soil]
```

10. Press ENTER twice.

11. Type the code you see on Lines 23 – 25 of the screenshot below.

```
18 top_left_x = 100
19 top_left_y = 150
20
21 DEMO_OBJECTS = [images.floor, images.pillar, images.soil]
22
23 LANDER_SECTOR = random.randint(1, 24)
24 LANDER_X = random.randint(2, 11)
25 LANDER_Y = random.randint(2, 11)
26
27
28 ###############
29 ##    MAP    ##
30 ###############
```

Line 23 creates the LANDER_SECTOR variable and sets its value to be equal to a random integer between 1 and 24. This bit of code will run once at the start of the game, so the lander location is different each time you play but doesn't change during the game.

Lines 24 and 25 create two more variables, LANDER_X and LANDER_Y, and sets their values to be random integers between 2 and 11. These variables will track the lander's x and y position within the LANDER_SECTOR.

12. Scroll and click at the end of Line 72.

```
68          ["The robot workshop", 9, 11, True, False] # room 50
69          ]
70
71  #simple sanity check on map above to check data entry
72  assert len(GAME_MAP)-1 == MAP_SIZE, "Map size and GAME_MAP don't match"
```

13. Press ENTER three times.

14. On Lines 75 – 77, create the OBJECTS section comment shown in the screenshot below.

```
71  #simple sanity check on map above to check data entry
72  assert len(GAME_MAP)-1 == MAP_SIZE, "Map size and GAME_MAP don't match"
73
74
75  ###############
76  ##  OBJECTS  ##
77  ###############
```

15. Press ENTER twice.

16. Navigate to the Google Doc file with the objects list the I gave to you on Google Classroom. Copy and paste this entire list onto Lines 79 – 224 of your code.

```
          "An MP3 player, with all the latest tunes", "an MP3 player"],
    71: [images.lander, None, "The Poodle, a small space exploration craft.
Its black box has a radio sealed inside.", "the Poodle lander"],
    72: [images.radio, None, "A radio communications system, from the \
Poodle", "a communications radio"],
    73: [images.gps_module, None, "A GPS Module", "a GPS module"],
    74: [images.positioning_system, None, "Part of a positioning system. \
Needs a GPS module.", "a positioning interface"],
    75: [images.positioning_system, None,
          "A working positioning system", "a positioning computer"],
    76: [images.scissors, None, "Scissors. They're too blunt to cut \
anything. Can you sharpen them?", "blunt scissors"],
    77: [images.scissors, None,
          "Razor-sharp scissors. Careful!", "sharpened scissors"],
    78: [images.credit, None,
          "A small coin for the station's vending systems",
          "a station credit"],
    79: [images.access_card, None,
          "This access card belongs to " + PLAYER_NAME, "an access card"],
    80: [images.access_card, None,
          "This access card belongs to " + FRIEND1_NAME, "an access card"],
    81: [images.access_card, None,
          "This access card belongs to " + FRIEND2_NAME, "an access card"]
    }
```

The code in the Google Doc that you copied and pasted contains the "objects" dictionary that lists 81 different objects that the player can see or interact with in the game. These objects are numbered and contain the object's image name(s) and description. Each object begins or ends with a square bracket and the objects in the dictionary are separated by a comma. The dictionary begins and ends with a curly bracket.

We will use the numbers at the beginning of each object list item as the "key" for this dictionary. The "key" is what we will use to search the dictionary for different items. Each list item's "key" must be unique to only that item in the dictionary. However, they "key" does not have to be a number. It can be words or letter combinations too. In our case, it seems like the simplest solution to assign each dictionary item its own unique number "key" and use these numbers to reference that particular entry in the dictionary later.

17. Click at the end of Line 224.

18. Press ENTER twice. Backspace all the way to the left margin if necessary.

19. Type the code you see on Lines 226 – 228 of the screenshot below.

```
222     81: [images.access_card, None,
223          "This access card belongs to " + FRIEND2_NAME, "an access card"]
224     }
225
226 items_player_may_carry = list(range(53, 82))
227 # Numbers below are for floor, pressure pad, soil, toxic floor.
228 items_player_may_stand_on = items_player_may_carry + [0, 39, 2, 48]
```

Line 226 creates another list named items_player_may_carry. This list stores the numbers of the objects from the objects dictionary that the player can pick up. The player can pick up all of the objects between dictionary items numbered 53 – 82.

Line 227 contains a comment.

Line 228 creates another list named items_player_may_stand_on. This list contains all items from the items_player_may_carry list plus the floor (number 0 in the dictionary), the pressure pad (number 39), the soil (number 2), and the toxic floor (number 48).

20. Scroll and click at the end of Line 308.

```
301 ################
302 ## EXPLORER  ##
303 ################
304
305 def draw():
306     global room_height, room_width, room_map
307     generate_map()
308     screen.clear()
```

21. Press ENTER.

22. Type the code you see on Lines 309 – 314 of the screenshot below.

```
301 ###############
302 ## EXPLORER  ##
303 ###############
304
305 def draw():
306     global room_height, room_width, room_map
307     generate_map()
308     screen.clear()
309     room_map[2][4] = 7
310     room_map[2][6] = 6
311     room_map[1][1] = 8
312     room_map[1][2] = 9
313     room_map[1][8] = 12
314     room_map[1][9] = 9
315
316     for y in range(room_height):
317         for x in range(room_width):
```

Lines 309 – 314 contain code that will place various objects into the room before the room image is generated. For example, Line 309 will place the object 7 from the objects dictionary (a chair) three rows from the top (index position 2) and five positions from the left. Lines 310 – 314 will place five more objects from the objects dictionary in the room.

23. Scroll and click at the end of Line 318.

```
314     room_map[1][9] = 9
315
316     for y in range(room_height):
317         for x in range(room_width):
318             image_to_draw = DEMO_OBJECTS[room_map[y][x]]
319             screen.blit(image_to_draw,
320                 (top_left_x + (x*30),
321                  top_left_y + (y*30) - image_to_draw.get_height()))
```

24. Backspace to erase all of the text to the right of the equal sign.

```
314     room_map[1][9] = 9
315
316     for y in range(room_height):
317         for x in range(room_width):
318             image_to_draw =
319             screen.blit(image_to_draw,
320                 (top_left_x + (x*30),
321                  top_left_y + (y*30) - image_to_draw.get_height()))
```

25. Type the text shown on Line 318 in the screenshot below to the right of the equal sign.

```
313     room_map[1][8] = 12
314     room_map[1][9] = 9
315
316     for y in range(room_height):
317         for x in range(room_width):
318             image_to_draw = objects[room_map[y][x]][0]
319             screen.blit(image_to_draw,
320                 (top_left_x + (x*30),
321                  top_left_y + (y*30) - image_to_draw.get_height()))
322
323 def movement():
```

To see the objects, we have to tell the game to use the new dictionary instead of the images in the DEMO_OBJECTS variable list.

Line 318 will not change the value of the image_to_draw variable to be the objects from the room_map list at the y and x position. The [0] at the end of this line will reference the first item in the objects list for that room, which is the room's image.

26. Go to File > Save Now to save your code.

Final Code:

```
# Escape

import time, random, math

###############
## VARIABLES ##
###############

WIDTH = 800 #window size
HEIGHT = 600

#PLAYER variables
PLAYER_NAME = "Alice"
FRIEND1_NAME = "Jack"
FRIEND2_NAME = "Matthew"
current_room = 31 # start room = 31

top_left_x = 100
top_left_y = 150

DEMO_OBJECTS = [images.floor, images.pillar, images.soil]

LANDER_SECTOR = random.randint(1, 24)
LANDER_X = random.randint(2, 11)
LANDER_Y = random.randint(2, 11)


###############
##    MAP    ##
###############

MAP_WIDTH = 5
MAP_HEIGHT = 10
MAP_SIZE = MAP_WIDTH * MAP_HEIGHT

GAME_MAP = [ ["Room 0 - where unused objects are kept", 0, 0, False, False] ]

outdoor_rooms = range(1, 26)
for planetsectors in range(1, 26): #rooms 1 to 25 are generated here
    GAME_MAP.append( ["The dusty planet surface", 18, 13, True, True] )

GAME_MAP  += [
        #["Room name", height, width, Top exit?, Right exit?]
        ["The airlock", 13, 6, True, False], # room 26
        ["The engineering lab", 13, 13, False, False], # room 27
        ["Foodle Mission Control", 9, 13, False, True], # room 28
        ["The viewing gallery", 9, 15, False, False], # room 29
        ["The crew's bathroom", 8, 8, False, False], # room 30
        ["The airlock entry bay", 7, 13, True, True], # room 31
        ["Left elbow room", 8, 7, True, False], # room 32
        ["Right elbow room", 7, 13, True, True], # room 33
        ["The science lab", 13, 13, False, True], # room 34
        ["The greenhouse", 13, 13, True, False], # room 35
        [PLAYER_NAME + "'s sleeping quarters", 9, 11, False, False], # room 36
        ["West corridor", 15, 5, True, True], # room 37
        ["The briefing room", 7, 13, False, True], # room 38
        ["The crew's community room", 11, 13, True, False], # room 39
        ["Main Mission Control", 14, 14, False, False], # room 40
        ["The sick bay", 12, 7, True, False], # room 41
        ["West corridor", 9, 7, True, False], # room 42
        ["Utilities control room", 9, 9, False, True], # room 43
        ["Systems engineering bay", 9, 11, False, False], # room 44
        ["Security portal to Mission Control", 7, 7, True, False], # room 45
        [FRIEND1_NAME + "'s sleeping quarters", 9, 11, True, True], # room 46
        [FRIEND2_NAME + "'s sleeping quarters", 9, 11, True, True], # room 47
        ["The pipeworks", 13, 11, True, False], # room 48
        ["The chief scientist's office", 5, 7, True, True], # room 49
        ["The robot workshop", 9, 11, True, False] # room 50
        ]

#simple sanity check on map above to check data entry
assert len(GAME_MAP)-1 == MAP_SIZE, "Map size and GAME_MAP don't match"


###############
##  OBJECTS  ##
###############

objects = {
    0: [images.floor, None, "The floor is shiny and clean"],
    1: [images.pillar, images.full_shadow, "The wall is smooth and solid"],
    2: [images.soil, None, "It's like a desert. Or should that be dessert?"],
    3: [images.pillar_low, images.half_shadow, "The wall is smooth and cold"],
    4: [images.bed, images.half_shadow, "A tidy and comfortable bed"],
    5: [images.table, images.half_shadow, "It's made from strong plastic."],
    6: [images.chair_left, None, "A chair with a soft cushion"],
    7: [images.chair_right, None, "A chair with a soft cushion"],
    8: [images.bookcase_tall, images.full_shadow,
        "Bookshelves, stacked with reference books"],
    9: [images.bookcase_small, images.half_shadow,
        "Bookshelves, stacked with reference books"],
    10: [images.cabinet, images.half_shadow,
         "A small locker, for storing personal items"],
    11: [images.desk_computer, images.half_shadow,
         "A computer. Use it to run life support diagnostics"],
    12: [images.plant, images.plant_shadow, "A spaceberry plant, grown here"],
    13: [images.electrical1, images.half_shadow,
         "Electrical systems used for powering the space station"],
    14: [images.electrical2, images.half_shadow,
         "Electrical systems used for powering the space station"],
    15: [images.cactus, images.cactus_shadow, "Ouch! Careful on the cactus!"],
    16: [images.shrub, images.shrub_shadow,
         "A spare lettuce. A bit limp, but amazing it's growing here!"],
    17: [images.pipes1, images.pipes1_shadow, "Water purification pipes"],
    18: [images.pipes2, images.pipes2_shadow,
         "Pipes for the life support systems"],
    19: [images.pipes3, images.pipes2_shadow,
         "Pipes for the life support systems"],
    20: [images.door, images.door_shadow, "Safety door. Opens automatically \
for astronauts in functioning spacesuits."],
    21: [images.door, images.door_shadow, "The airlock door. \
For safety reasons, it requires two person operation."],
    22: [images.door, images.door_shadow, "A locked door. It needs " \
         + PLAYER_NAME + "'s access card"],
    23: [images.door, images.door_shadow, "A locked door. It needs " \
         + FRIEND1_NAME + "'s access card"],
    24: [images.door, images.door_shadow, "A locked door. It needs " \
         + FRIEND2_NAME + "'s access card"],
    25: [images.door, images.door_shadow,
         "A locked door. It is opened from Main Mission Control"],
```

```
    26: [images.door, images.door_shadow,
         "A locked door in the engineering bay."],
    27: [images.map, images.full_shadow,
         "The screen says the crash site was Doctor: " \
         + str(LANDER_SECTOR) + " // X: " + str(LANDER_X) + " \
         + // Y: " + str(LANDER_Y)],
    28: [images.rock_large, images.rock_large_shadow,
         "A rock. Its coarse surface feels like a whetstone", "the rock"],
    29: [images.rock_small, images.rock_small_shadow,
         "A small but heavy piece of Martian rock"],
    30: [images.crater, None, "A crater in the planet surface"],
    31: [images.fence, None,
         "A fine gauze fence. It helps protect the station from dust storms"],
    32: [images.contraption, images.contraption_shadow,
         "One of the scientific experiments. It gently vibrates"],
    33: [images.robot_arm, images.robot_arm_shadow,
         "A robot arm, used for heavy lifting"],
    34: [images.toilet, images.half_shadow, "A sparkling clean toilet"],
    35: [images.sink, None, "A sink with running water", "the tap"],
    36: [images.globe, images.globe_shadow,
         "A giant globe of the planet. It gently glows from inside"],
    37: [images.science_lab_table, None,
         "A table of experiments, analysing the planet soil and dust"],
    38: [images.vending_machine, images.full_shadow,
         "A vending machine. It requires a credit.", "the vending machine"],
    39: [images.floor_pad, None,
         "A pressure sensor to make sure nobody goes out alone."],
    40: [images.rescue_ship, images.rescue_ship_shadow, "A rescue ship!"],
    41: [images.mission_control_desk, images.mission_control_desk_shadow, \
         "Mission Control stations."],
    42: [images.button, images.button_shadow,
         "The button for opening the time-locked door in engineering."],
    43: [images.whiteboard, images.full_shadow,
         "The whiteboard is used in brainstorms and planning meetings."],
    44: [images.window, images.full_shadow,
         "The window provides a view out onto the planet surface."],
    45: [images.robot, images.robot_shadow, "A cleaning robot, turned off."],
    46: [images.robot2, images.robot2_shadow,
         "A planet surface exploration robot, awaiting set-up."],
    47: [images.rocket, images.rocket_shadow, "A one-person craft in repair"],
    48: [images.toxic_floor, None, "Steam floor - do not walk on!"],
    49: [images.drone, None, "A delivery drone"],
    50: [images.mercury_ball, None, "An mercury ball - dangerous!"],
    51: [images.energy_ball2, None, "An energy ball - dangerous!"],
    52: [images.computer, images.computer_shadow,
         "A computer workstation for managing space station systems."],
    53: [images.clipboard, None,
         "A clipboard. Someone has doodled on it.", "the clipboard"],
    54: [images.bubble_gun, None,
         "A piece of sticky bubble gum. Strawberry flavour.", "bubble gun"],
    55: [images.yoyo, None, "A toy made of fins, strong string and plastic. \
Used for zerograv experiments.", PLAYER_NAME + "'s yoyo"],
    56: [images.thread, None,
         "A piece of fine, strong string", "a piece of string"],
    57: [images.needle, None,
         "A sharp needle from a cactus plant", "a cactus needle"],
    58: [images.threaded_needle, None,
         "A cactus needle, spearing a length of string", "needle and string"],
    59: [images.canister, None,
         "The air canister has a leak.", "a leaky air canister"],
    60: [images.canister, None,
         "It looks like the seal will hold!", "a sealed air canister"],
    61: [images.mirror, None,
         "The mirror throws a circle of light on the walls.", "a mirror"],
    62: [images.bin_empty, None,
         "A rarely used bin, made of light plastic", "a bin"],
    63: [images.bin_full, None,
         "A heavy bin full of water", "a bin full of water"],
    64: [images.rag, None,
         "An oily rag. Pick it up by one corner if you must!", "an oily rag"],
    65: [images.hammer, None,
         "A hammer. Maybe good for cracking things open...", "a hammer"],
    66: [images.spoon, None, "A large serving spoon", "a spoon"],
    67: [images.food_pouch, None,
         "A dehydrated food pouch. It needs water.", "a dry food pack"],
    68: [images.food, None,
         "A food pouch. See it to get 1300 energy.", "ready-to-eat food"],
    69: [images.book, None, "The book has the words 'Don't Panic' in the \
cover in large, friendly letters", "a book"],
    70: [images.mp3_player, None,
         "An MP3 player, with all the latest tunes", "an MP3 player"],
    71: [images.lander, None, "The Poodle, a small space exploration craft. \
Its black box has a radio sealed inside.", "the Poodle lander"],
    72: [images.radio, None, "A radio communications system, from the \
Poodle", "a communications radio"],
    73: [images.gps_module, None, "A GPS Module", "a GPS module"],
    74: [images.positioning_system, None, "Part of a positioning system. \
Needs a GPS module.", "a positioning interface"],
    75: [images.positioning_system, None,
         "A working positioning system", "a positioning computer"],
    76: [images.scissors, None, "Scissors. They're too blunt to cut \
anything. Can you sharpen them?", "blunt scissors"],
    77: [images.scissors, None,
         "Razor-sharp scissors. Careful!", "sharpened scissors"],
    78: [images.credit, None,
         "A small coin for the station's vending systems",
         "a station credit"],
    79: [images.access_card, None,
         "This access card belongs to " + PLAYER_NAME, "an access card"],
    80: [images.access_card, None,
         "This access card belongs to " + FRIEND1_NAME, "an access card"],
    81: [images.access_card, None,
         "This access card belongs to " + FRIEND2_NAME, "an access card"]
    }

items_player_may_carry = list(range(53, 82))
# Numbers below are for floor, pressure pad, exit, toxic floor.
items_players_may_stand_on = items_player_may_carry + [0, 39, 2, 48]


###############
## MAKE MAP  ##
###############

def get_floor_type():
    if current_room in outdoor_rooms:
        return 2 # soil
    else:
        return 0 # tiled floor
```

```
def generate_map():
# This function makes the map for the current room,
# using room data, scenery data and prop data.
    global room_map, room_width, room_height, room_name, hazard_map
    global top_left_x, top_left_y, wall_transparency_frame
    room_data = GAME_MAP[current_room]
    room_name = room_data[0]
    room_height = room_data[1]
    room_width = room_data[2]

    floor_type = get_floor_type()
    if current_room in range(1, 21):
        bottom_edge = 2 #wall
        side_edge = 2 #wall
    if current_room in range(21, 26):
        bottom_edge = 1 #wall
        side_edge = 2 #wall
    if current_room > 25:
        bottom_edge = 1 #wall
        side_edge = 1 #wall

    # Create top line of room map.
    room_map=[[side_edge] * room_width]
    # Add middle lines of room map (wall, floor to fill width, wall).
    for y in range(room_height - 2):
        room_map.append([side_edge]
                        + [floor_type]*(room_width - 2) + [side_edge])
    # Add bottom line of room map.
    room_map.append([bottom_edge] * room_width)

    # Add doorways.
    middle_row = int(room_height / 2)
    middle_column = int(room_width / 2)

    if room_data[4]: # If exit at right of this room
        room_map[middle_row][room_width - 1] = floor_type
        room_map[middle_row+1][room_width - 1] = floor_type
        room_map[middle_row-1][room_width - 1] = floor_type

    if current_room % MAP_WIDTH != 1: # If room is not on left of map
        room_to_left = GAME_MAP[current_room - 1]
        # If room on the left has a right exit, add left exit in this room
        if room_to_left[4]:
            room_map[middle_row][0] = floor_type
            room_map[middle_row + 1][0] = floor_type
            room_map[middle_row - 1][0] = floor_type

    if room_data[3]: # If exit at top of this room
        room_map[0][middle_column] = floor_type
        room_map[0][middle_column + 1] = floor_type
        room_map[0][middle_column - 1] = floor_type

    if current_room <= MAP_SIZE - MAP_WIDTH: # If room is not on bottom row
        room_below = GAME_MAP[current_room+MAP_WIDTH]
        # If room below has a top exit, add exit at bottom of this one
        if room_below[3]:
            room_map[room_height-1][middle_column] = floor_type
            room_map[room_height-1][middle_column + 1] = floor_type
            room_map[room_height-1][middle_column - 1] = floor_type

###############
## EXPLORER  ##
###############

def draw():
    global room_height, room_width, room_map
    generate_map()
    screen.clear()
    room_map[2][9] = 7
    room_map[2][6] = 6
    room_map[1][1] = 8
    room_map[1][2] = 5
    room_map[1][8] = 12
    room_map[1][9] = 9

    for y in range(room_height):
        for x in range(room_width):
            image_to_draw = objects[room_map[y][x]][0]
            screen.blit(image_to_draw,
                        (top_left_x + (x*30),
                         top_left_y + (y*30) - image_to_draw.get_height()))

def movement():
    global current_room
    old_room = current_room

    if keyboard.left:
        current_room -= 1
    if keyboard.right:
        current_room += 1
    if keyboard.up:
        current_room -= MAP_WIDTH
    if keyboard.down:
        current_room += MAP_WIDTH

    if current_room > 50:
        current_room = 50
    if current_room < 1:
        current_room = 1

    if current_room != old_room:
        print("Entering room: " + str(current_room))

clock.schedule_interval(movement, 0.05)
```