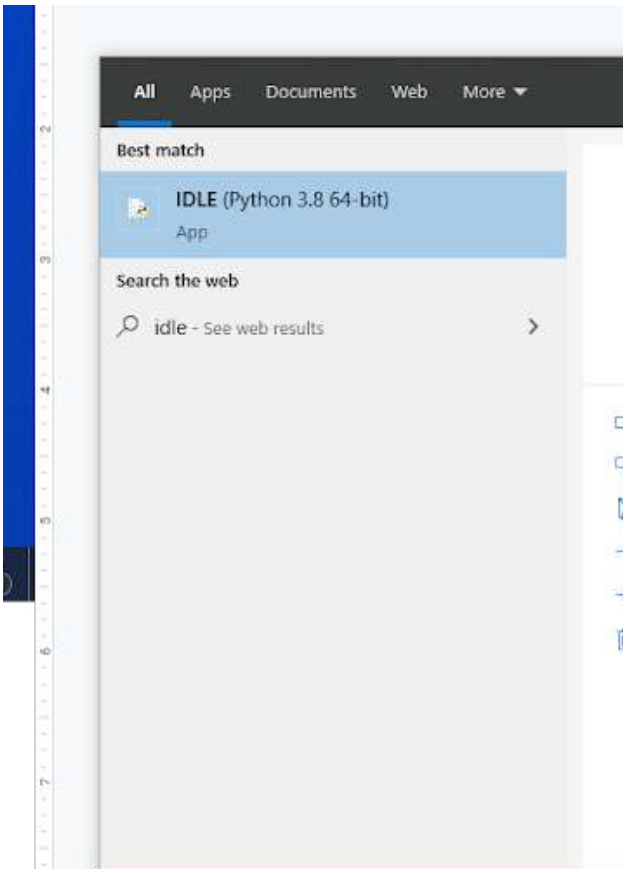
Part 8
Space Mission Directions

1. Navigate out to the Google Classroom for this class.

2. Locate the Space Mission Part 8 assignment.

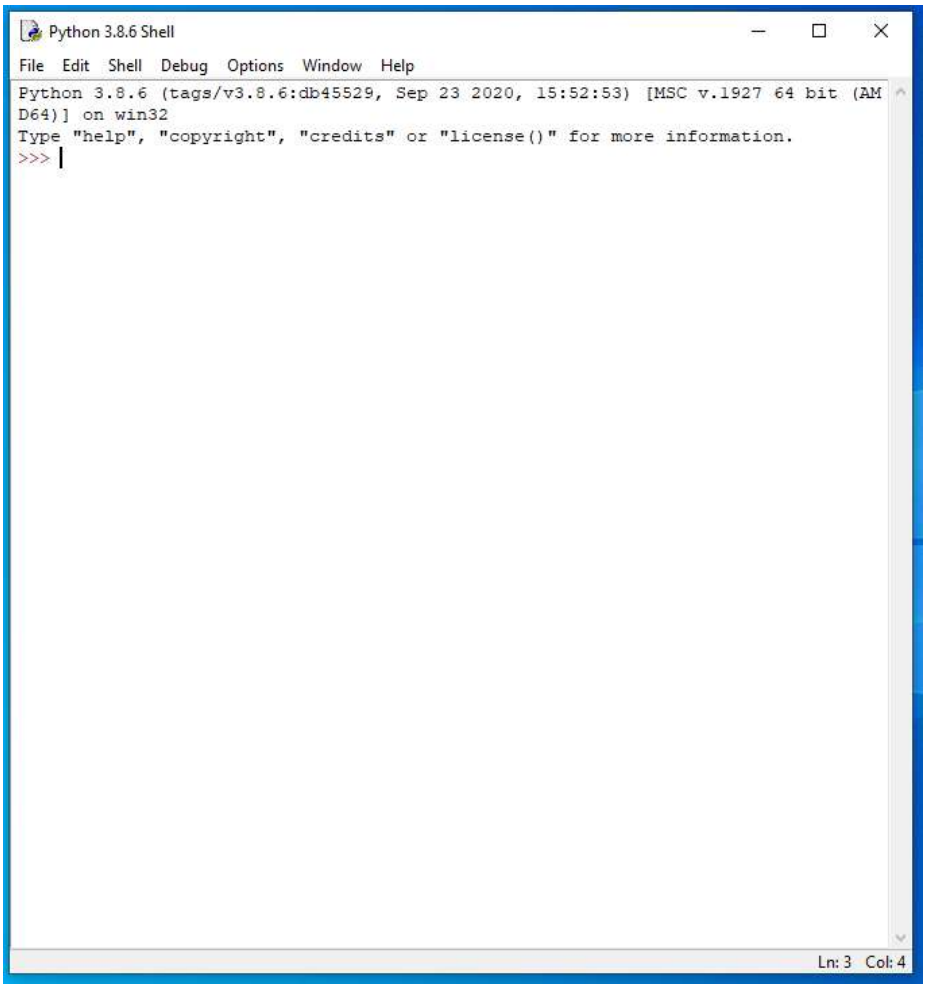
3. We are now ready to start adding code to our file. Using your Windows button menu, find and launch your IDLE program.



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.

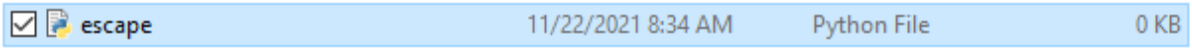
4. Your IDLE window should look something like this once it has launched.:



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.

5. 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.



6. Your escape.py file will open up.

7. Scroll and click at the end of Line 487.

```
482 ################
483 ## GAME LOOP ##
484 ################
485
486 def start_room():
487     show_text("You are here: " + room_name, 0)
488
489 def game_loop():
```

8. Modify the start_room method by adding the code you see on Lines 487, 489, 490, and 491 of the screenshot below.

```
482 ################
483 ## GAME LOOP ##
484 ################
485
486 def start_room():
487     global airlock_door_frame
488     show_text("You are here: " + room_name, 0)
489     if current_room == 26: # Room with self-shutting airlock door
490         airlock_door_frame = 0
491         clock.schedule_interval(door_in_room_26, 0.05)
492
493 def game_loop():
494     global player_x, player_y, current_room
495     global from_player_x, from_player_y
496     global player_image, player_image_shadow
497     global selected_item, item_carrying, energy
498     global player_offset_x, player_offset_y
499     global player_frame, player_direction
```

Line 487 converts the airlock_door_frame variable into a global variable so the start_room function can modify its value.

Line 488 is code that already existed in your game from a previous chapter.

Line 489 begins an “if” function that checks to see if the player is standing in room 26, which is the room that has a self-shutting airlock door. If the player is in room 26, Lines 490 and 491 will run.

Line 490 sets the value of the airlock_door_frame variable to 0. Line 491 sets the door_in_room_26 method to run every .05 seconds. We haven't created this method or initialized the airlock_door_frame variable yet.

9. Scroll and click at the end of Line 1038.

```
1022      for recipe in RECIPES:
1023          ingredient1 = recipe[0]
1024          ingredient2 = recipe[1]
1025          combination = recipe[2]
1026          if (item_carrying == ingredient1
1027              and item_player_is_on == ingredient2) \
1028              or (item_carrying == ingredient2
1029                  and item_player_is_on == ingredient1):
1030              use_message = "You combine " + objects[ingredient1][3] \
1031                            + " and " + objects[ingredient2][3] \
1032                            + " to make " + objects[combination][3]
1033              if item_player_is_on in props.keys():
1034                  props[item_player_is_on][0] = 0
1035                  room_map[player_y][player_x] = get_floor_type()
1036              in_my_pockets.remove(item_carrying)
1037              add_object(combination)
1038              sounds.combine.play()
1039
1040      show_text(use_message, 0)
1041      time.sleep(0.5)
```

10. Press ENTER twice.

11. Type the code you see on Lines 1040 – 1048 of the screenshot below. Ensure your indentation and punctuation match what is shown in the screenshot.

```
1033            if item_player_is_on in props.keys():
1034                props[item_player_is_on][0] = 0
1035                room_map[player_y][player_x] = get_floor_type()
1036            in_my_pockets.remove(item_carrying)
1037            add_object(combination)
1038            sounds.combine.play()
1039
1040        # {key object number: door object number}
1041    ACCESS_DICTIONARY = { 79:22, 80:23, 81:24 }
1042    if item_carrying in ACCESS_DICTIONARY:
1043        door_number = ACCESS_DICTIONARY[item_carrying]
1044        if props[door_number][0] == current_room:
1045            use_message = "You unlock the door!"
1046            sounds.say_doors_open.play()
1047            sounds.doors.play()
1048            open_door(door_number)
1049
1050    show_text(use_message, 0)
1051    time.sleep(0.5)
1052
1053def game_completion_sequence():
```

Line 1040 contains a comment.

This section of code enables players to use keys to open the doors. We create a new dictionary called ACCESS_DICTIONARY that uses the access card number as the dictionary key and the door number as the data. So object 79 (an access card) is used to open door 22, for example (Line 1041).

When the player presses U, the door opens (Line 1048) if they have selected one of the items in the dictionary for unlocking doors (Line 1042 - 1043) and if they are standing in the same room as the door it unlocks (Line 1044). We also play a sound effect of a computer voice saying “doors open” (Line 1046), play the doors sound (Line 1047), and change the value of the use_message variable (Line 1045).

12. Scroll down and click at the end of Line 1076.

```
1064      launch_frame += 1
1065      if launch_frame < 9:
1066          draw_image(images.rescue_ship, 8 - launch_frame, 6)
1067          draw_shadow(images.rescue_ship_shadow, 8 + launch_frame, 6)
1068          clock.schedule(game_completion_sequence, 0.25)
1069      else:
1070          screen.surface.set_clip(None)
1071          screen.draw.text("MISSION", (200, 380), color = "white",
1072                           fontsize = 128, shadow = (1, 1), scolor = "black")
1073          screen.draw.text("COMPLETE", (145, 480), color = "white",
1074                           fontsize = 128, shadow = (1, 1), scolor = "black")
1075          sounds.completion.play()
1076          sounds.say_mission_complete.play()
1077
1078
1079  ###############
1080  ##   START   ##
1081  ###############
```

13. Press ENTER three times.

14. Type the code you see on Lines 1079 – 1094 of the screenshot below. Ensure your line spacing, indentation, and punctuation match what is shown in the screenshot.

```
1075            sounds.completion.play()
1076            sounds.say_mission_complete.play()
1077
1078
1079    ###############
1080    ##    DOORS    ##
1081    ###############
1082
1083    def open_door(opening_door_number):
1084        global door_frames, door_shadow_frames
1085        global door_frame_number, door_object_number
1086        door_frames = [images.door1, images.door2, images.door3,
1087                       images.door4, images.floor]
1088        # (Final frame restores shadow ready for when door reappears).
1089        door_shadow_frames = [images.door1_shadow, images.door2_shadow,
1090                              images.door3_shadow, images.door4_shadow,
1091                              images.door_shadow]
1092        door_frame_number = 0
1093        door_object_number = opening_door_number
1094        do_door_animation()
1095
1096
1097    ###############
1098    ##    START    ##
1099    ###############
```

Lines 1079 – 1081 create a new section in the code called DOORS.

Line 1083 establishes a new function called open_door. This function will require the opening_door_number to be input whenever it is called.

Lines 1084 and 1085 establish the door_frames, door_shadow_frames, door_frame_number, and door_object_number as global variables.

The door animation to open the door consists of five frames, numbered 0 to 4. We store images for the animation in a list called door_frames (Lines 1086 – 1087) and store the frame number in the variable called door_frame_number (Line 1092).

Line 1088 contains a comment.

Lines 1089 – 1091 contain door frame animation images for the door's shadow, stored in a list called door_shadow_frames.

In the variable door_object_number (Line 1093), we store the object number of the door that will be opening or closing. After the variables and list have been set up, the function do_door_animation (Line 1094) is started to carry out the animation. We will create that function later.

15. Press ENTER twice.

16. Type the code that you see on Lines 1096 – 1112 of the screenshot below. Ensure your indentation and punctuation match what is shown in the screenshot.

```
1079 ###############
1080 ##   DOORS   ##
1081 ###############
1082
1083 def open_door(opening_door_number):
1084     global door_frames, door_shadow_frames
1085     global door_frame_number, door_object_number
1086     door_frames = [images.door1, images.door2, images.door3,
1087                    images.door4, images.floor]
1088     # (Final frame restores shadow ready for when door reappears).
1089     door_shadow_frames = [images.door1_shadow, images.door2_shadow,
1090                           images.door3_shadow, images.door4_shadow,
1091                           images.door_shadow]
1092     door_frame_number = 0
1093     door_object_number = opening_door_number
1094     do_door_animation()
1095
1096 def close_door(closing_door_number):
1097     global door_frames, door_shadow_frames
1098     global door_frame_number, door_object_number, player_y
1099     door_frames = [images.door4, images.door3, images.door2,
1100                    images.door1, images.door]
1101     door_shadow_frames = [images.door4_shadow, images.door3_shadow,
1102                           images.door2_shadow, images.door1_shadow,
1103                           images.door_shadow]
1104     door_frame_number = 0
1105     door_object_number = closing_door_number
1106     # If player is in same row as a door, they must be in open doorway
1107     if player_y == props[door_object_number][1]:
1108         if player_y == 0: # if in the top doorway
1109             player_y = 1 # move them down
1110         else:
1111             player_y = room_height - 2 # move them up
1112     do_door_animation()
1113
1114
1115 ###############
1116 ##   START   ##
1117 ###############
```

Line 1096 creates a new function called close_door. This function will require the closing_door_number to be input whenever it is called

As we did in the open_door function, Lines 1097 – 1098 convert the door_frames, door_shadow_frames, door_frame_number, door_object_number, and player_y variables to global variables.

Lines 1099 – 1103 create two different lists, one for the door animation and one for the door shadow animation, to store the images for the animation. Notice in these lists that the images are reversed from the open_door function. That is because we want to animate the door closing, not opening, so we will need to reverse the order of the images.

Line 1104 establishes sets the door_frame_number variable to 0 and Line 1105 sets the door_object_number variable to be the same as the closing_door_number.

Line 1106 contains a comment.

Lines 1107 – 1111 run through a series of checks to ensure the door doesn't close on top of a player. If the door is closing and the player is in the way, the program will change the player's position to move them out of the doorway.

Line 1112 will run the do_door_animation function, similar to what we did in the open_door function we wrote previously.

17. Press ENTER twice.

18. Type the code you see on Lines 1114 – 1126 of the screenshot below. Ensure your indentation and punctuation match what is shown in the screenshot.

```
1104      door_frame_number = 0
1105      door_object_number = closing_door_number
1106      # If player is in same row as a door, they must be in open doorway
1107      if player_y == props[door_object_number][1]:
1108          if player_y == 0: # if in the top doorway
1109              player_y = 1 # move them down
1110          else:
1111              player_y = room_height - 2 # move them up
1112      do_door_animation()
1113
1114  def do_door_animation():
1115      global door_frames, door_frame_number, door_object_number, objects
1116      objects[door_object_number][0] = door_frames[door_frame_number]
1117      objects[door_object_number][1] = door_shadow_frames[door_frame_number]
1118      door_frame_number += 1
1119      if door_frame_number == 5:
1120          if door_frames[-1] == images.floor:
1121              props[door_object_number][0] = 0 # remove door from props list
1122          # Regenerate room map from the props
1123          # to put the door in the room if required.
1124          generate_map()
1125      else:
1126          clock.schedule(do_door_animation, 0.15)
1127
1128
1129  ###############
1130  ##   START   ##
1131  ###############
```

Line 1114 creates a new function called do_door_animation.

Line 1115 converts the door_frames, door_frame_number, door_object_number, and objects variables to global variables.

The objects dictionary contains, among other things, the images to use for a particular object. This new function starts by changing the door's image and shadow image in that dictionary to the current animation frame (Lines 1116 - 1117). When the room is redrawn, it will now use that animation frame.

The function then increases the animation frame number by 1 (Line 1118) so the next animation frame can be shown next time this function runs.

If the frame is now 5 (Line 1119), it means we've reached the end of the animation. In that case, we check whether the door has opened (rather than closed) by seeing whether the final frame was a floor tile, showing no door (Line 1120). Remember, an index number of -1 will give you the last item in a list.

If the door was opened, the door will be removed from view (Line 1121) and moved to room 0.

Lines 1122 and 1123 contain comments.

Line 1124 runs the generate_map function to redraw the map.

Line 1125 contains an “else” function that will run if the door animation frame is not equal to 5. Line 1126 will run the do_door_animation again after .15 seconds. Essentially, we continue to run the door animation, increasing the animation frame images by 1 each time the function runs. When it gets to the final frame, if the door has opened, it will remove the door from view.

19. Press ENTER twice.

20. Type the code you see on Lines 1128 – 1138 of the screenshot below. Ensure your indentation and punctuation match what is shown in the screenshot.

```
1114 def do_door_animation():
1115     global door_frames, door_frame_number, door_object_number, objects
1116     objects[door_object_number][0] = door_frames[door_frame_number]
1117     objects[door_object_number][1] = door_shadow_frames[door_frame_number]
1118     door_frame_number += 1
1119     if door_frame_number == 5:
1120         if door_frames[-1] == images.floor:
1121             props[door_object_number][0] = 0 # remove door from props list
1122         # Regenerate room map from the props
1123         # to put the door in the room if required.
1124         generate_map()
1125     else:
1126         clock.schedule(do_door_animation, 0.15)
1127
1128 def shut_engineering_door():
1129     global current_room, door_room_number, props
1130     props[25][0] = 32 # Door from room 32 to the engineering bay.
1131     props[26][0] = 27 # Door inside engineering bay.
1132     generate_map() # Add door to room_map for if in affected room.
1133     if current_room == 27:
1134         close_door(26)
1135     if current_room == 32:
1136         close_door(25)
1137     show_text("The computer tells you the doors are closed.", 1)
1138     sounds.say_doors_closed.play()
1139
1140
1141 ###############
1142 ##   START   ##
1143 ###############
```

Line 1128 creates a new function called shut_engineering_door.

Line 1129 establishes the current_room, door_room_number, and props variables as global variables.

The shut_engineering_door function has two door props to work with, objects 25 and 26, because the player can see this door from either side depending on which room they're in. The first thing we do is update the props dictionary so these doors appear in the rooms (Lines 1130 – 1131).

Line 1132 will run the generate_map function to redraw the room map with the doors in place. If the player is in a room with one of these doors, this function updates the room map for the current room. In other cases, the generate_map function still runs but nothing changes.

Line 1133 will check to see if the player is in room 27. If so, Line 1134 will run the close_door function for object 26.

Line 1135 will check to see if the player is in room 32. If so, Line 1136 will fun the close_door function for object 25.

Line 1137 runs the show_text function to display a message on the screen. Line 1138 will play the say_doors_closed sound for the player.

21. Press ENTER twice.

22. Type the code you see on Lines 1140 – 1152 of the screenshot below. Ensure your indentation, punctuation, and line spacing match what is shown in the screenshot.

```
1128 def shut_engineering_door():
1129     global current_room, door_room_number, props
1130     props[25][0] = 32 # Door from room 32 to the engineering bay.
1131     props[26][0] = 27 # Door inside engineering bay.
1132     generate_map() # Add door to room_map for if in affected room.
1133     if current_room == 27:
1134         close_door(26)
1135     if current_room == 32:
1136         close_door(25)
1137     show_text("The computer tells you the doors are closed.", 1)
1138     sounds.say_doors_closed.play()
1139
1140 def door_in_room_26():
1141     global airlock_door_frame, room_map
1142     frames = [images.door, images.door1, images.door2,
1143               images.door3,images.door4, images.floor
1144               ]
1145
1146     shadow_frames = [images.door_shadow, images.door1_shadow,
1147                      images.door2_shadow, images.door3_shadow,
1148                      images.door4_shadow, None]
1149
1150     if current_room != 26:
1151         clock.unschedule(door_in_room_26)
1152         return
1153
1154
1155 ###############
1156 ##   START   ##
1157 ###############
```

Line 1140 creates a new function called door_in_room_26.

Line 1141 establishes the airlock_door_frame and room_map variables as global variables.

We store the animation frames for the door in the list frames, including the first frame that shows the door shut and the final frame that shows an empty floor tile instead of the door (Lines 1142 – 1144). We also create a separate list for the animation frames for the doors shadow on Lines 1146 – 1148. We leave the final frame out of the shadow animation since there will be no shadow when the door is completely open.

Line 1150 will check to see if the player is still in room 26. If the player has left the room, the clock.unschedule function will stop the door_in_room_26 function from running regularly and exit the function using a return statement so that the door animation stops.

23. Press ENTER twice.

24. Type the code you see on Lines 1154 – 1172 of the screenshot below. Ensure your indentation, punctuation, and line spacing match what is shown in the screenshot.

```
1140 def door_in_room_26():
1141     global airlock_door_frame, room_map
1142     frames = [images.door, images.door1, images.door2,
1143               images.door3,images.door4, images.floor
1144               ]
1145
1146     shadow_frames = [images.door_shadow, images.door1_shadow,
1147                      images.door2_shadow, images.door3_shadow,
1148                      images.door4_shadow, None]
1149
1150     if current_room != 26:
1151         clock.unschedule(door_in_room_26)
1152         return
1153
1154     # prop 21 is the door in Room 26.
1155     if ((player_y == 8 and player_x == 2) or props[63] == [26, 8, 2]) \
1156             and props[21][0] == 26:
1157         airlock_door_frame += 1
1158         if airlock_door_frame == 5:
1159             props[21][0] = 0 # Remove door from map when fully open.
1160             room_map[0][1] = 0
1161             room_map[0][2] = 0
1162             room_map[0][3] = 0
1163
1164     if ((player_y != 8 or player_x != 2) and props[63] != [26, 8, 2]) \
1165             and airlock_door_frame > 0:
1166         if airlock_door_frame == 5:
1167             # Add door to props and map so animation is shown.
1168             props[21][0] = 26
1169             room_map[0][1] = 21
1170             room_map[0][2] = 255
1171             room_map[0][3] = 255
1172         airlock_door_frame -= 1
1173
1174
1175 ###############
1176 ##   START   ##
1177 ###############
```

Line 1154 contains a comment.

Lines 1155 - 1156 will check to see if the player is standing on the pressure pad in room 26 and that the door is currently in the room. If this is true, Line 1157 will increase the animation door frame for the airlock door by 1.

Line 1158 will check to see if the airlock_door_frame value is equal to 5, which is the last image in the animation, meaning the door is fully open. If this is true, Line 1159 will remove the door from view (by moving the prop to room 0), and the room_map will also be updated.

Lines 1164 – 1172 will perform a similar set of commands in reverse to close the door. Lines 1164 – 1165 will check to see if the player is not standing on the pressure pad and that the airlock_door_frame value is larger than 0 (meaning that the door is at least partially open).

Line 1166 will check to see if the airlock_door_frame value is equal to 5, meaning that the door is currently fully open.

Line 1167 contains a comment.

If the door is fully open, the props and the room_map dictionaries will be updated to put the door back into the room so the player can see it. Remember, the number 255 is used to indicate an object that takes up more than one tile.

Line 1172 will subtract 1 from the value of the airlock_door_frame, moving backwards through the animation to begin closing the door.

25. Press ENTER twice.

26. Type the code you see on Lines 1174 – 1175 of the screenshot below. Ensure your indentation matches what is shown in the screenshot.

```
1164      if ((player_y != 8 or player_x != 2) and props[63] != [26, 8, 2]) \
1165              and airlock_door_frame > 0:
1166          if airlock_door_frame == 5:
1167              # Add door to props and map so animation is shown.
1168              props[21][0] = 26
1169              room_map[0][1] = 21
1170              room_map[0][2] = 255
1171              room_map[0][3] = 255
1172          airlock_door_frame -= 1
1173
1174      objects[21][0] = frames[airlock_door_frame]
1175      objects[21][1] = shadow_frames[airlock_door_frame]
1176
1177
1178  ###############
1179  ##    START    ##
1180  ###############
```

Lines 1174 and 1175 will change the image file for the door and door shadow in the objects dictionary to match the current animation frame.

27. Go to File > Save to save your code.

Final Code:

```
# Escape

import time, random, math

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

WIDTH = 800 #window size
HEIGHT = 800

#PLAYER variables
PLAYER_NAME = "Blaine"
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, 12)
LANDER_Y = random.randint(2, 12)

TILE_SIZE = 30

player_y, player_x = 2, 6
GAME_OVER = False

PLAYER = {
    "left": [images.spacesuit_left, images.spacesuit_left_1,
             images.spacesuit_left_2, images.spacesuit_left_3,
             images.spacesuit_left_4
             ],
    "right": [images.spacesuit_right, images.spacesuit_right_1,
              images.spacesuit_right_2, images.spacesuit_right_3,
              images.spacesuit_right_4
              ],
    "up": [images.spacesuit_back, images.spacesuit_back_1,
           images.spacesuit_back_2, images.spacesuit_back_3,
           images.spacesuit_back_4
           ],
    "down": [images.spacesuit_front, images.spacesuit_front_1,
             images.spacesuit_front_2, images.spacesuit_front_3,
             images.spacesuit_front_4
             ]
    }

player_direction = "down"
player_frame = 0
player_image = PLAYER[player_direction][player_frame]
player_offset_x, player_offset_y = 0, 0

PLAYER_SHADOW = {
    "left": [images.spacesuit_left_shadow, images.spacesuit_left_1_shadow,
             images.spacesuit_left_2_shadow, images.spacesuit_left_3_shadow,
             images.spacesuit_left_3_shadow
             ],
    "right": [images.spacesuit_right_shadow, images.spacesuit_right_1_shadow,
              images.spacesuit_right_2_shadow,
              images.spacesuit_right_3_shadow, images.spacesuit_right_3_shadow
              ],
    "up": [images.spacesuit_back_shadow, images.spacesuit_back_1_shadow,
           images.spacesuit_back_2_shadow, images.spacesuit_back_3_shadow,
           images.spacesuit_back_3_shadow
           ],
    "down": [images.spacesuit_front_shadow, images.spacesuit_front_1_shadow,
             images.spacesuit_front_2_shadow, images.spacesuit_front_3_shadow,
             images.spacesuit_front_3_shadow
             ]
    }

player_image_shadow = PLAYER_SHADOW["down"][0]

PILLARS = [
    images.pillar, images.pillar_95, images.pillar_90,
    images.pillar_60, images.pillar_50
    ]

wall_transparency_frame = 0

BLACK = (0, 0, 0)
BLUE = (0, 155, 255)
YELLOW = (255, 255, 0)
WHITE = (255, 255, 255)
GREEN = (0, 255, 0)
RED = (128, 0, 0)

air, energy = 100, 100
suit_stitched, air_fixed = False, False
launch_frame = 0


###############
##    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", 13, 13, True, True] )

GAME_MAP  += [
        #["Room name", height, width, Top exit?, Right exit?]
        ["The airlock", 13, 5, True, False], # room 26
        ["The engineering lab", 13, 13, False, False], # room 27
        ["Poodle Mission Control", 5, 13, False, True], # room 28
        ["The viewing gallery", 5, 15, False, False], # room 29
        ["The crew's bathroom", 5, 5, False, False], # room 30
        ["The airlock entry bay", 7, 11, True, True], # room 31
        ["Left elbow room", 9, 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 stor 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", 9, 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 cold"],
    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 cozy 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 space 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.pipes3_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 Sector: " \
         + str(LANDER_SECTOR) + " // X: " + str(LANDER_X) + " \
         " // Y: " + str(LANDER_Y)],
    28: [images.rock_large, images.rock_large_shadow,
         "A look. 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 taps"],
    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, analyzing 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, "Toxic floor - do not walk on!"],
    49: [images.drone, None, "A delivery drone"],
    50: [images.energy_ball, None, "An energy 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_gum, None,
         "A piece of sticky bubble gum. Spaceberry flavour.", "bubble gum"],
    55: [images.yoyo, None, "A toy made of fine, strong string and plastic. \
Used for antigrav 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.rags, 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. Has it to get 100% energy.", "ready-to-eat food"],
    69: [images.book, None, "The book has the words 'Don't Panic' on 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, soil, toxic floor,
items_player_may_stand_on = items_player_may_carry + [0, 39, 2, 48]


##############
##  SCENERY  ##
##############

# Scenery describes objects that cannot move between rooms.
# room number: [[object number, y position, x position]...]
```

```
scenery = {
    26: [[39,8,2]],
    27: [[33,5,5], [33,1,1], [33,1,8], [47,5,2],
         [47,3,10], [47,9,8], [42,1,6]],
    28: [[27,0,3], [41,4,3], [41,4,7]],
    29: [[7,2,6], [6,2,8], [12,1,13], [44,0,1],
         [36,4,10], [20,1,1], [19,4,2], [17,4,6]],
    30: [[34,1,1], [35,1,3]],
    31: [[11,1,1], [19,1,8], [46,1,3]],
    32: [[48,2,2], [45,2,3], [45,2,4], [45,3,2], [45,3,3],
         [45,3,4], [45,4,2], [45,4,3], [45,4,4]],
    33: [[13,1,1], [13,1,3], [13,1,8], [19,1,10], [45,2,1],
         [45,2,7], [45,3,6], [45,3,3]],
    34: [[37,2,2], [32,6,7], [37,10,4], [25,5,3]],
    35: [[14,2,9], [16,2,2], [16,3,3], [16,3,8], [16,8,9], [16,8,2], [16,1,8],
         [16,1,3], [12,8,6], [12,8,4], [12,9,8],
         [15,4,6], [12,7,1], [12,7,11]],
    36: [[4,9,1], [9,1,7], [9,1,8], [8,1,9],
         [5,5,4], [6,5,7], [10,1,1], [12,1,2]],
    37: [[45,3,1], [45,3,2], [45,7,1], [45,5,2], [45,5,3],
         [45,7,2], [45,9,2], [45,9,3], [45,11,1], [45,11,2]],
    38: [[43,0,2], [6,3,2], [6,3,5], [6,4,7], [6,2,9], [55,1,10]],
    39: [[38,1,1], [7,9,5], [7,6,4], [8,3,6], [5,6,6],
         [6,3,9], [6,6,9], [45,1,11], [12,1,8], [12,1,4]],
    40: [[41,5,3], [41,5,7], [41,9,3], [41,9,7],
         [13,1,1], [13,1,3], [42,1,12]],
    41: [[4,9,1], [10,9,5], [9,5,1], [10,6,5], [6,7,1],
         [10,7,5], [12,1,1], [12,1,5]],
    42: [[46,4,3], [46,4,5], [15,1,1], [19,1,3],
         [19,1,5], [52,4,7], [24,1,5]],
    43: [[45,2,1], [45,2,2], [45,5,7], [45,3,4], [45,1,4], [45,1,1]],
    44: [[10,1,1], [4,1,2], [8,1,7], [9,1,8], [8,1,9], [6,5,8], [7,3,3]],
    47: [[9,1,1], [9,1,2], [10,1,9], [12,1,7], [9,4,4], [6,4,7], [4,1,8]],
    48: [[17,4,1], [17,4,2], [17,4,3], [17,4,4], [17,4,5], [17,4,6], [17,4,7],
         [17,5,1], [17,5,2], [17,5,3], [17,5,4],
         [17,5,5], [17,5,6], [17,6,7], [14,1,1]],
    49: [[14,2,2], [14,2,4], [7,5,1], [5,5,3], [48,5,3], [48,3,4]],
    50: [[45,4,8], [11,1,1], [19,1,8], [33,2,1], [46,4,6]]
    }

checksum = 0
check_counter = 0
for key, room_scenery_list in scenery.items():
    for scenery_item_list in room_scenery_list:
        checksum += (scenery_item_list[0] * key
                     + scenery_item_list[1] * (key + 1)
                     + scenery_item_list[2] * (key + 2))
        check_counter += 1
print(check_counter, "scenery items")
assert check_counter == 161, "Expected 161 scenery items"
assert checksum == 200095, "Error in scenery data"
print("Scenery checksum: " + str(checksum))

for room in range(1, 26):# Add random scenery in planet locations.
    if room != 13: # Skip room 13.
        scenery_item = random.choice([16, 28, 29, 30])
        scenery[room] = [[scenery_item, random.randint(2, 10),
                          random.randint(2, 10)]]


# Use loops to add fences to the planet surface rooms.
for room_coordinate in range(0, 13):
    for room_number in [1, 2, 3, 4, 5]: # Add top fence
        scenery[room_number] += [[31, 0, room_coordinate]]
    for room_number in [1, 6, 11, 16, 21]: # Add left fence
        scenery[room_number] += [[31, room_coordinate, 0]]
    for room_number in [5, 10, 15, 20, 25]: # Add right fence
        scenery[room_number] += [[31, room_coordinate, 12]]

del scenery[21][-1] # Delete last fence panel in Room 21
del scenery[25][-1] # Delete last fence panel in Room 25


################
## 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 #soil
        side_edge = 2 #soil
    if current_room in range(21, 26):
        bottom_edge = 1 #wall
        side_edge = 2 #soil
    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

    if current_room in scenery:
        for this_scenery in scenery[current_room]:
            scenery_number = this_scenery[0]
            scenery_y = this_scenery[1]
            scenery_x = this_scenery[2]
            room_map[scenery_y][scenery_x] = scenery_number

            image_here = objects[scenery_number][0]
            image_width = image_here.get_width()
            image_width_in_tiles = int(image_width / TILE_SIZE)

            for tile_number in range(1, image_width_in_tiles):
                room_map[scenery_y][scenery_x + tile_number] = 256

    center_y = int(HEIGHT / 2) # Center of game window
    center_x = int(WIDTH / 2)
    room_pixel_width = room_width * TILE_SIZE # Size of room in pixels
    room_pixel_height = room_height * TILE_SIZE
    top_left_x = center_x - 0.5 * room_pixel_width
    top_left_y = (center_y - 0.5 * room_pixel_height) + 110

```

```
    for prop_number, prop_info in props.items():
        prop_room = prop_info[0]
        prop_y = prop_info[1]
        prop_x = prop_info[2]
        if (prop_room == current_room and
            room_map[prop_y][prop_x] in (0, 39, 2)):
                room_map[prop_y][prop_x] = prop_number
                image_here = objects[prop_number][0]
                image_width = image_here.get_width()
                image_width_in_tiles = int(image_width / TILE_SIZE)
                for tile_number in range(1, image_width_in_tiles):
                    room_map[prop_y][prop_x + tile_number] = 255

################
## GAME LOOP ##
################
def start_room():
    global airlock_door_frame
    show_text("You are here: " + room_name, 0)
    if current_room == 26: # Room with self-shutting airlock door
        airlock_door_frame = 0
        clock.schedule_interval(door_in_room_26, 0.05)

def game_loop():
    global player_x, player_y, current_room
    global from_player_x, from_player_y
    global player_image, player_image_shadow
    global selected_item, item_carrying, energy
    global player_offset_x, player_offset_y
    global player_frame, player_direction

    if game_over:
        return

    if player_frame > 0:
        player_frame += 1
        time.sleep(0.06)
        if player_frame == 5:
            player_frame = 0
            player_offset_x = 0
            player_offset_y = 0

#save player's current position
    old_player_x = player_x
    old_player_y = player_y

# MOVE IF key is pressed
    if player_frame == 0:
        if keyboard.right:
            from_player_x = player_x
            from_player_y = player_y
            player_x += 1
            player_direction = "right"
            player_frame = 1
        elif keyboard.left: #elif stops player making diagonal movements
            from_player_x = player_x
            from_player_y = player_y
            player_x -= 1
            player_direction = "left"
            player_frame = 1
        elif keyboard.up:
            from_player_x = player_x
            from_player_y = player_y
            player_y -= 1
            player_direction = "up"
            player_frame = 1
        elif keyboard.down:
            from_player_x = player_x
            from_player_y = player_y
            player_y += 1
            player_direction = "down"
            player_frame = 1

# check for exiting the room
    if player_x == room_width: # through door on RIGHT
        #clock.unschedule(hazard_move)
        current_room += 1
        generate_map()
        player_x = 0 # enter at left
        player_y = int(room_height / 2) # enter at door
        player_frame = 0
        start_room()
        return

    if player_x == -1: # through door on LEFT
        #clock.unschedule(hazard_move)
        current_room -= 1
        generate_map()
        player_x = room_width - 1  # enter at right
        player_y = int(room_height / 2) # enter at door
        player_frame = 0
        start_room()
        return

    if player_y == room_height: # through door at BOTTOM
        #clock.unschedule(hazard_move)
        current_room += MAP_WIDTH
        generate_map()
        player_y = 0 # enter at top
        player_x = int(room_width / 2) # enter at door
        player_frame = 0
        start_room()
        return

    if player_y == -1: # through door at TOP
        #clock.unschedule(hazard_move)
        current_room -= MAP_WIDTH
        generate_map()
        player_y = room_height - 1 # enter at bottom
        player_x = int(room_width / 2) # enter at door
        player_frame = 0
        start_room()
        return
```

```
    if keyboard.g:
        pick_up_object()

    if keyboard.tab and len(in_my_pockets) > 0:
        selected_item += 1
        if selected_item > len(in_my_pockets) - 1:
            selected_item = 0
        item_carrying = in_my_pockets[selected_item]
        display_inventory()

    if keyboard.d and item_carrying:
        drop_object(old_player_y, old_player_x)

    if keyboard.space:
        examine_object()

    if keyboard.u:
        use_object()

    # If the player is standing somewhere they shouldn't, move them back.
    if room_map[player_y][player_x] not in items_player_may_stand_on: #\
    #       or hazard_map[player_y][player_x] != 0:
        player_x = old_player_x
        player_y = old_player_y
        player_frame = 0

    if player_direction == "right" and player_frame > 0:
        player_offset_x = -1 + (0.25 * player_frame)
    if player_direction == "left" and player_frame > 0:
        player_offset_x = 1 - (0.25 * player_frame)
    if player_direction == "up" and player_frame > 0:
        player_offset_y = 1 - (0.25 * player_frame)
    if player_direction == "down" and player_frame > 0:
        player_offset_y = -1 + (0.25 * player_frame)


###############
##  DISPLAY  ##
###############

def draw_image(image, y, x):
    screen.blit(
        image,
        (top_left_x + (x * TILE_SIZE),
         top_left_y + (y * TILE_SIZE) - image.get_height())
        )

def draw_shadow(image, y, x):
    screen.blit(
        image,
        (top_left_x + (x * TILE_SIZE),
         top_left_y + (y * TILE_SIZE))
        )

def draw_player():
    player_image = PLAYER[player_direction][player_frame]
    draw_image(player_image, player_y + player_offset_y,
               player_x + player_offset_x)
    player_image_shadow = PLAYER_SHADOW[player_direction][player_frame]
    draw_shadow(player_image_shadow, player_y + player_offset_y,
                player_x + player_offset_x)

def draw():
    if game_over:
        return

    # Clear the game arena area.
    box = Rect((0, 150), (800, 600))
    screen.draw.filled_rect(box, RED)
    box = Rect ((0, 0), (500, top_left_y + (room_height - 1)*30))
    screen.surface.set_clip(box)
    floor_type = get_floor_type()

    for y in range(room_height): # Lay down floor tiles, then items on floor.
        for x in range(room_width):
            draw_image(objects[floor_type][0], y, x)
            # Next line enables shadows to fall on top of objects on floor
            if room_map[y][x] in items_player_may_stand_on:
                draw_image(objects[room_map[y][x]][0], y, x)

    # Pressure pad in room 26 is added here, so props can go on top of it.
    if current_room == 26:
        draw_image(objects[39][0], 8, 2)
        image_on_pad = room_map[8][2]
        if image_on_pad > 0:
            draw_image(objects[image_on_pad][0], 8, 2)

    for y in range(room_height):
        for x in range(room_width):
            item_here = room_map[y][x]
            # Player cannot walk on 255: it marks spaces used by wide objects.
            if item_here not in items_player_may_stand_on + [255]:
                image = objects[item_here][0]

                if (current_room in outdoor_rooms
                    and y == room_height - 1
                    and room_map[y][x] == 1) or \
                    (current_room not in outdoor_rooms
                    and y == room_height - 1
                    and room_map[y][x] == 1
                    and x > 0
                    and x < room_width - 1):
                    # Add transparent wall image in the front row.
                    image = PILLARS[wall_transparency_frame]

                draw_image(image, y, x)

                if objects[item_here][1] is not None: # If object has a shadow
                    shadow_image = objects[item_here][1]
                    # If shadow might need horizontal tiling
                    if shadow_image in [images.half_shadow,
                                        images.full_shadow]:
                        shadow_width = int(image.get_width() / TILE_SIZE)
                        # Use shadow across width of object.
                        for x in range(0, shadow_width):
                            draw_shadow(shadow_image, y, x+x)
                    else:
                        draw_shadow(shadow_image, y, x)

        if (player_y == y):
                draw_player()

    screen.surface.set_clip(None)

def adjust_wall_transparency():
    global wall_transparency_frame

    if (player_y == room_height - 2
        and room_map[room_height - 1][player_x] == 1
        and wall_transparency_frame < 4):
        wall_transparency_frame += 1 # Fade wall out.

    if ((player_y < room_height - 2
            or room_map[room_height - 1][player_x] != 1)
            and wall_transparency_frame > 0):
        wall_transparency_frame -= 1 # Fade wall in.

def show_text(text_to_show, line_number):
    if game_over:
        return
    text_lines = [15, 50]
    box = Rect((0, text_lines[line_number]), (800, 35))
    screen.draw.filled_rect(box, BLACK)
    screen.draw.text(text_to_show,
                     (20, text_lines[line_number]), color=GREEN)


###############
##  PROPS  ##
###############

# Props are objects that may move between rooms, appear or disappear.
# All props must be set up here. Props not yet in the game go into room 0.
# object number : [room, y, x]
```

```
props = {
    20: [31, 0, 4], 21: [26, 0, 1], 22: [41, 0, 2], 23: [39, 0, 5],
    24: [45, 0, 2],
    25: [32, 0, 2], 26: [27, 12, 5], # two sides of same door
    40: [0, 8, 6], 53: [45, 1, 5], 54: [0, 0, 0], 55: [0, 0, 0],
    56: [0, 0, 0], 57: [33, 4, 6], 58: [0, 0, 0], 59: [31, 1, 7],
    60: [0, 0, 0], 61: [36, 1, 1], 62: [36, 1, 6], 63: [0, 0, 0],
    64: [27, 8, 3], 65: [50, 1, 7], 66: [35, 8, 6], 67: [46, 1, 1],
    68: [0, 0, 0], 69: [30, 3, 3], 70: [47, 1, 3],
    71: [0, LANDER_Y, LANDER_X], 72: [0, 0, 0], 73: [27, 4, 6],
    74: [25, 1, 11], 75: [0, 0, 0], 76: [41, 3, 5], 77: [0, 0, 0],
    78: [35, 9, 11], 79: [26, 3, 2], 80: [41, 7, 5], 81: [29, 1, 1]
    }

checksum = 0
for key, prop in props.items():
    if key != 71: # 71 is skipped because it's different each game.
        checksum += (prop[0] * key
                     + prop[1] * (key + 1)
                     + prop[2] * (key + 2))
print(len(props), "props")
assert len(props) == 37, "Expected 37 prop items"
print("Prop checksum:", checksum)
assert checksum == 61414, "Error in props data"


in_my_pockets = [55]
selected_item = 0 # the first item
item_carrying = in_my_pockets[selected_item]


RECIPES = [
    [62, 35, 63], [76, 28, 77], [78, 38, 64], [73, 74, 75],
    [59, 54, 60], [77, 55, 56], [56, 57, 58], [71, 65, 72],
    [88, 58, 89], [89, 60, 90], [67, 35, 68]
    ]

checksum = 0
check_counter = 1
for recipe in RECIPES:
    checksum += (recipe[0] * check_counter
                 + recipe[1] * (check_counter + 1)
                 + recipe[2] * (check_counter + 2))
    check_counter += 3
print(len(RECIPES), "recipes")
assert len(RECIPES) == 11, "Expected 11 recipes"
assert checksum == 37296, "Error in recipes data"
print("Recipe checksum:", checksum)


######################
## PROP INTERACTIONS ##
######################

def find_object_start_x():
    checker_x = player_x
    while room_map[player_y][checker_x] == 255:
        checker_x -= 1
    return checker_x

def get_item_under_player():
    item_x = find_object_start_x()
    item_player_is_on = room_map[player_y][item_x]
    return item_player_is_on

def pick_up_object():
    global room_map
    # Get object number at player's location.
    item_player_is_on = get_item_under_player()
    if item_player_is_on in items_player_may_carry:
        # Clear the floor space.
        room_map[player_y][player_x] = get_floor_type()
        add_object(item_player_is_on)
        show_text("Now carrying " + objects[item_player_is_on][3], 0)
        sounds.pickup.play()
        time.sleep(0.5)
    else:
        show_text("You can't carry that!", 0)

def add_object(item): # Adds item to inventory.
    global selected_item, item_carrying
    in_my_pockets.append(item)
    item_carrying = item
    # Minus one because indexes start at 0.
    selected_item = len(in_my_pockets) - 1
    display_inventory()
    props[item][0] = 0 # Carried objects go into room 0 (off the map).

def display_inventory():
    box = Rect((0, 45), (800, 105))
    screen.draw.filled_rect(box, BLACK)

    if len(in_my_pockets) == 0:
        return

    start_display = (selected_item // 16) * 16
    list_to_show = in_my_pockets[start_display : start_display + 16]
    selected_marker = selected_item % 16

    for item_counter in range(len(list_to_show)):
        item_number = list_to_show[item_counter]
        image = objects[item_number][0]
        screen.blit(image, (26 + (46 * item_counter), 80))

    box_left = (selected_marker * 46) - 3
    box = Rect((22 + box_left, 55), (40, 40))
    screen.draw.rect(box, WHITE)
    item_highlighted = in_my_pockets[selected_item]
    description = objects[item_highlighted][2]
    screen.draw.text(description, (20, 130), color="white")

def drop_object(old_y, old_x):
    global room_map, props
    if room_map[old_y][old_x] in [0, 2, 39]: # places you can drop things
        props[item_carrying][0] = current_room
        props[item_carrying][1] = old_y
        props[item_carrying][2] = old_x
        room_map[old_y][old_x] = item_carrying
        show_text("You have dropped " + objects[item_carrying][3], 0)
        sounds.drop.play()
        remove_object(item_carrying)
        time.sleep(0.5)
    else: # This only happens if there is already a prop here
        show_text("You can't drop that there.", 0)
        time.sleep(0.5)

def remove_object(item): # Takes item out of inventory
    global selected_item, in_my_pockets, item_carrying
    in_my_pockets.remove(item)
    selected_item = selected_item - 1
    if selected_item < 0:
        selected_item = 0
    if len(in_my_pockets) == 0: # If they're not carrying anything
        item_carrying = False # Set item_carrying to False
    else: # Otherwise set it to the new selected item
        item_carrying = in_my_pockets[selected_item]
    display_inventory()

def examine_object():
    item_player_is_on = get_item_under_player()
    left_tile_of_item = find_object_start_x()
    if item_player_is_on in [0, 2]: # don't describe the floor
        return
    description = "You see: " + objects[item_player_is_on][2]
    for prop_number, details in props.items():
        # props = object number: [room number, y, x]
        if details[0] == current_room: # if prop is in the room
            # If prop is hidden (= at player's location but not on map)
            if (details[1] == player_y
                and details[2] == left_tile_of_item
                and room_map[details[1]][details[2]] != prop_number):
                add_object(prop_number)
                description = "You found " + objects[prop_number][3]
                sounds.combine.play()
    show_text(description, 0)
    time.sleep(0.5)


################
## USE OBJECTS ##
################
```

```
def use_object():
    global room_map, props, item_carrying, air, selected_item, energy
    global in_my_pockets, suit_stitched, air_fixed, game_over

    use_message = "You fiddle around with it but don't get anywhere."
    standard_responses = {
        4: "Are is rubbing out! You can't take this lying down!",
        6: "This is no time to sit around!",
        7: "This is no time to sit around!",
        32: "It shakes and rumbles, but nothing else happens.",
        34: "Ah! That's better. Now wash your hands.",
        35: "You wash your hands and shake the water off.",
        37: "The test tubes smoke slightly as you shake them.",
        54: "You chew the gum. It's sticky like glue.",
        55: "The yoyo bounces up and down, slightly slower then on Earth",
        56: "It's a bit too fiddly. Can you thread it on something?",
        59: "You need to fix the leak before you can use the canister",
        61: "You try signalling with the mirror, but nobody can see you.",
        62: "Don't throw resources away. Things might come in handy...",
        67: "To enjoy yummy space food, just add water!",
        75: "You are at Sector: " + str(current_room) + " // X: " \
            + str(player_x) + " // Y: " + str(player_y)
        }

    # Get object number at player's location.
    item_player_is_on = get_item_under_player()
    for this_item in [item_player_is_on, item_carrying]:
        if this_item in standard_responses:
            use_message = standard_responses[this_item]

    if item_carrying == 70 or item_player_is_on == 70:
        use_message = "Banging tunes!"
        sounds.steelmusic.play(2)

    elif item_player_is_on == 11:
        use_message = "AIR: " + str(air) + \
                      "% / ENERGY " + str(energy) + "% / "
        if not suit_stitched:
            use_message += "'ALERT' SUIT FABRIC TORN / "
        if not air_fixed:
            use_message += "'ALERT' SUIT AIR BOTTLE HISSING"
        if suit_stitched and air_fixed:
            use_message += " SUIT OK"
        show_text(use_message, 0)
        sounds.say_status_report.play()
        time.sleep(0.5)
        # If "on" the computer, player intention is clearly status update.
        # Return to stop another object use accidentally overriding this.
        return

    elif item_carrying == 60 or item_player_is_on == 60:
        use_message = "You fix " + objects[60][3] + " to the suit"
        air_fixed = True
        air = 90
        air_countdown()
        remove_object(60)

    elif (item_carrying == 58 or item_player_is_on == 58) \
        and not suit_stitched:
        use_message = "You use " + objects[56][3] + \
                      " to repair the suit fabric"
        suit_stitched = True
        remove_object(58)

    elif item_carrying == 72 or item_player_is_on == 72:
        use_message = "You radio for help. A rescue ship is coming. \
Rendezvous Sector 12, outside."
        props[40][0] = 13

    elif (item_carrying == 66 or item_player_is_on == 66) \
            and current_room in outdoor_rooms:
        use_message = "You dig..."
        if [current_room == LANDER_SECTOR
            and player_x == LANDER_X
            and player_y == LANDER_Y):
            add_object(71)
            use_message = "You found the Poodle lander!"

    elif item_player_is_on == 40:
        clock.unschedule(air_countdown)
        show_text("Congratulations, "+ PLAYER_NAME +"!", 0)
        show_text("Mission success! You have made it to safety.", 1)
        game_over = True
        sounds.take_off.play()
        game_completion_sequence()

    elif item_player_is_on == 16:
        energy += 1
        if energy > 100:
            energy = 100
        use_message = "You munch the lettuce and get a little energy back"
        draw_energy_air()

    elif item_player_is_on == 42:
        if current_room == 27:
            open_door(26)
        props[25][0] = 0 # Door from RM32 to engineering bay
        props[26][0] = 0 # Door inside engineering bay
        clock.schedule_unique(shut_engineering_door, 60)
        use_message = "You press the button"
        show_text("Door to engineering bay is open for 60 seconds", 1)
        sounds.say_doors_open.play()
        sounds.doors.play()

    elif item_carrying == 69 or item_player_is_on == 69:
        energy = 100
        use_message = "You use the food to restore your energy"
        remove_object(69)
        draw_energy_air()

    if suit_stitched and air_fixed: # open airlock access
        if current_room == 32 and props[20][0] == 31:
            open_door(20) # which includes removing the door
            sounds.say_airlock_open.play()
            show_text("The computer tells you the airlock is now open.", 1)
        elif props[20][0] == 32:
            props[20][0] = 0 # remove door from map
            sounds.say_airlock_open.play()
            show_text("The computer tells you the airlock is now open.", 1)

     for recipe in RECIPES:
        ingredient1 = recipe[0]
        ingredient2 = recipe[1]
        combination = recipe[2]
        if (item_carrying == ingredient1
            and item_player_is_on == ingredient2) \
            or (item_carrying == ingredient2
                and item_player_is_on == ingredient1):
            use_message = "You combine " + objects[ingredient1][3] \
                          + " and " + objects[ingredient2][3] \
                          + " to make " + objects[combination][3]
            if item_player_is_on in props.keys():
                props[item_player_is_on][0] = 0
                room_map[player_y][player_x] = get_floor_type()
            in_my_pockets.remove(item_carrying)
            add_object(combination)
            sounds.combine.play()

     # (key object number: door object number)
    ACCESS_DICTIONARY = { 79:22, 80:23, 81:24 }
    if item_carrying in ACCESS_DICTIONARY:
        door_number = ACCESS_DICTIONARY[item_carrying]
        if props[door_number][0] == current_room:
            use_message = "You unlock the door!"
            sounds.say_doors_open.play()
            sounds.doors.play()
            open_door(door_number)

    show_text(use_message, 0)
    time.sleep(0.5)

def game_completion_sequence():
    global launch_frame #(initial value is 0, set up in VARIABLES section)
    box = Rect((0, 150), (800, 480))
    screen.draw.filled_rect(box, (128, 0, 0))
    box = Rect ((0, top_left_y - 30), (800, 390))
    screen.surface.set_clip(box)

    for y in range(0, 13):
        for x in range(0, 13):
            draw_image(images.soil, y, x)
```

```
    launch_frame += 1
    if launch_frame < 5:
        draw_image(images.rescue_ship, 0 - launch_frame, 6)
        draw_shadow(images.rescue_ship_shadow, 0 + launch_frame, 6)
        clock.schedule(game_completion_sequence, 0.25)
    else:
        screen.surface.set_clip(None)
        screen.draw.text("MISSION", (200, 380), color = "white",
                         fontsize = 128, shadow = (1, 1), scolor = "black")
        screen.draw.text("COMPLETE", (145, 450), color = "white",
                         fontsize = 128, shadow = (1, 1), scolor = "black")
        sounds.completion.play()
        sounds.say_mission_complete.play()


################
##   DOORS    ##
################

def open_door(opening_door_number):
    global door_frames, door_shadow_frames
    global door_frame_number, door_object_number
    door_frames = [images.door1, images.door2, images.door3,
                   images.door4, images.floor]
    # (Final frame restores shadow ready for when door reappears).
    door_shadow_frames = [images.door1_shadow, images.door2_shadow,
                          images.door3_shadow, images.door4_shadow,
                          images.door_shadow]
    door_frame_number = 0
    door_object_number = opening_door_number
    do_door_animation()

def close_door(closing_door_number):
    global door_frames, door_shadow_frames
    global door_frame_number, door_object_number, player_y
    door_frames = [images.door4, images.door3, images.door2,
                   images.door1, images.door]
    door_shadow_frames = [images.door4_shadow, images.door3_shadow,
                          images.door2_shadow, images.door1_shadow,
                          images.door_shadow]
    door_frame_number = 0
    door_object_number = closing_door_number
    # If player is in same row as a door, they must be in open doorway
    if player_y == props[door_object_number][1]:
        if player_y == 0: # if in the top doorway
            player_y = 1 # move them down
        else:
            player_y = room_height - 2 # move them up
    do_door_animation()

def do_door_animation():
    global door_frames, door_frame_number, door_object_number, objects
    objects[door_object_number][0] = door_frames[door_frame_number]
    objects[door_object_number][2] = door_shadow_frames[door_frame_number]
    door_frame_number += 1
    if door_frame_number == 5:
        if door_frames[-1] == images.floor:
            props[door_object_number][0] = 0 # remove door from props list
        # Regenerate room map from the props
        # to put the door in the room if required.
        generate_map()
    else:
        clock.schedule(do_door_animation, 0.15)

def shut_engineering_door():
    global current_room, door_room_number, props
    props[25][0] = 32 # Door from room 32 to the engineering bay.
    props[26][0] = 27 # Door inside engineering bay.
    generate_map() # Add door to room_map for if in affected room.
    if current_room == 27:
        close_door(26)
    if current_room == 32:
        close_door(25)
    show_text("The computer tells you the doors are closed.", 1)
    sounds.say_doors_closed.play()

def door_in_room_26():
    global airlock_door_frame, room_map
    frames = [images.door, images.door1, images.door2,
              images.door3,images.door4, images.floor
              ]

    shadow_frames = [images.door_shadow, images.door1_shadow,
                     images.door2_shadow, images.door3_shadow,
                     images.door4_shadow, None]

    if current_room != 26:
        clock.unschedule(door_in_room_26)
        return

    # prop 21 is the door in Room 26.
    if ((player_y == 6 and player_x == 2) or props[63] == [26, 8, 2]) \
            and props[21][0] == 26:
        airlock_door_frame += 1
        if airlock_door_frame == 5:
            props[21][0] = 0 # Remove door from map when fully open.
            room_map[0][1] = 0
            room_map[0][2] = 0
            room_map[0][3] = 0

    if ((player_y != 6 or player_x != 2) and props[63] != [26, 8, 2]) \
            and airlock_door_frame > 0:
        if airlock_door_frame == 5:
            # Add door to props and map so animation is shown.
            props[21][0] = 26
            room_map[0][1] = 21
            room_map[0][2] = 255
            room_map[0][3] = 255
        airlock_door_frame -= 1

    objects[21][0] = frames[airlock_door_frame]
    objects[21][1] = shadow_frames[airlock_door_frame]


################
##   START    ##
################

generate_map()
clock.schedule_interval(game_loop, 0.03)
clock.schedule_interval(adjust_wall_transparency, 0.05)
clock.schedule_unique(display_inventory, 1)
```