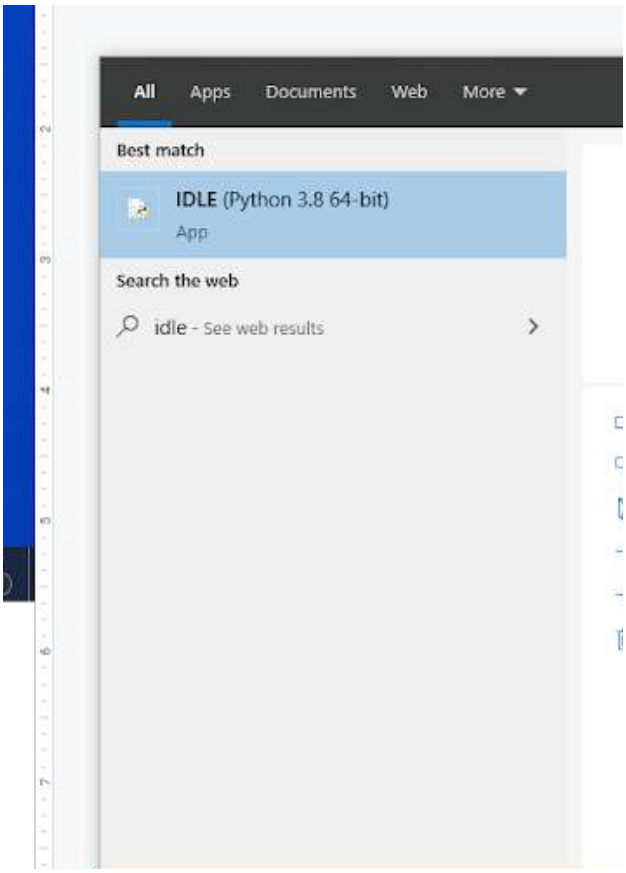
Part 9
Space Mission Directions

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

2. Locate the Space Mission Part 9 assignment.

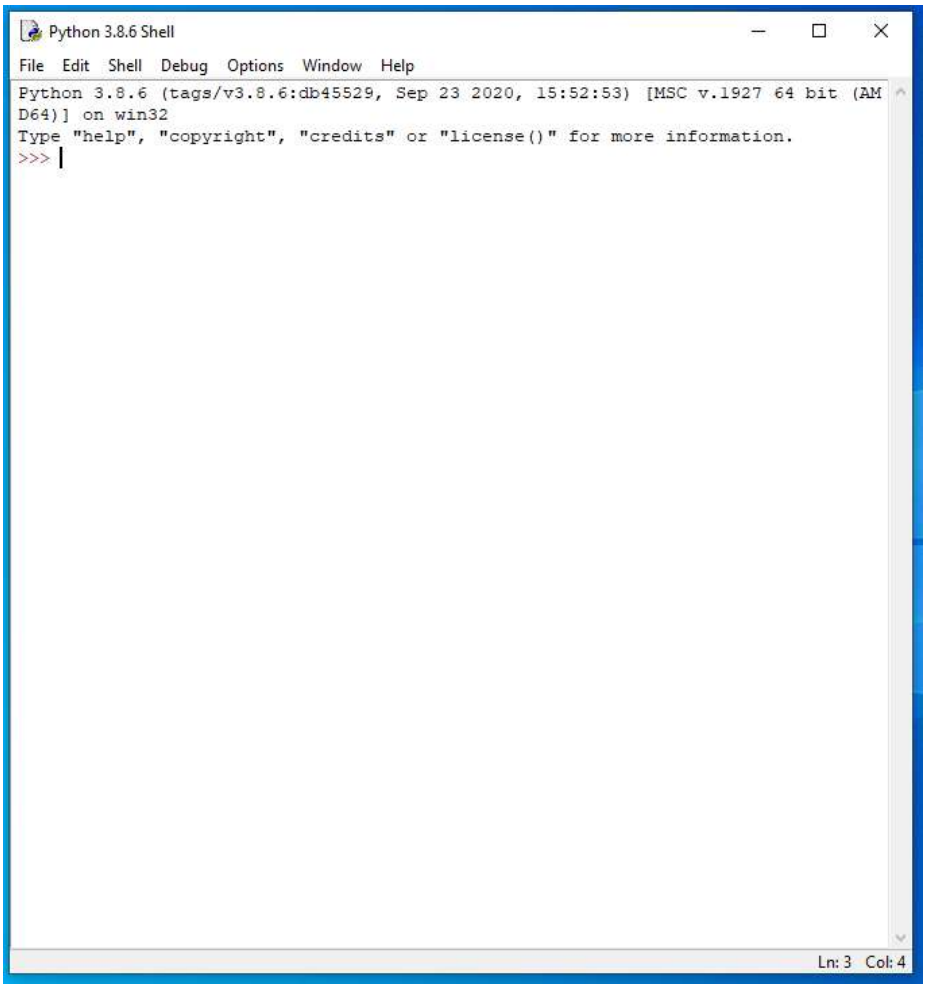
3. We are now ready to start adding code to our file. Using your Windows button menu, find and launch your IDLE program.



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.

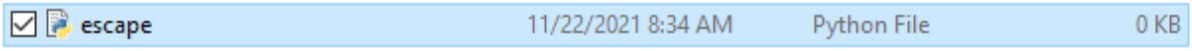
4. Your IDLE window should look something like this once it has launched.:



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 479.

```
468     for prop_number, prop_info in props.items():
469         prop_room = prop_info[0]
470         prop_y = prop_info[1]
471         prop_x = prop_info[2]
472         if (prop_room == current_room and
473             room_map[prop_y][prop_x] in [0, 39, 2]):
474                 room_map[prop_y][prop_x] = prop_number
475                 image_here = objects[prop_number][0]
476                 image_width = image_here.get_width()
477                 image_width_in_tiles = int(image_width / TILE_SIZE)
478                 for tile_number in range(1, image_width_in_tiles):
479                     room_map[prop_y][prop_x + tile_number] = 255
480
481
482 ################
483 ## GAME LOOP ##
484 ################
```

8. Press ENTER twice.

9. Type the code you see on Lines 481 – 483 of the screenshot below. Ensure your indentation and punctuation matches what is shown in the screenshot.

```
468     for prop_number, prop_info in props.items():
469         prop_room = prop_info[0]
470         prop_y = prop_info[1]
471         prop_x = prop_info[2]
472         if (prop_room == current_room and
473             room_map[prop_y][prop_x] in [0, 39, 2]):
474                 room_map[prop_y][prop_x] = prop_number
475                 image_here = objects[prop_number][0]
476                 image_width = image_here.get_width()
477                 image_width_in_tiles = int(image_width / TILE_SIZE)
478                 for tile_number in range(1, image_width_in_tiles):
479                     room_map[prop_y][prop_x + tile_number] = 255
480
481         hazard_map = [] # empty list
482         for y in range(room_height):
483             hazard_map.append( [0] * room_width )
484
485
486 ###############
487 ## GAME LOOP ##
488 ###############
```

Line 481 creates a new empty list called hazard_map.

Lines 482 – 483 fill the harzard_map list with rows of 0s that are as wide as the room width.

10. Ensure that your “GAME LOOP” comment runs on Lines 486 – 488 of your code, as shown in the screenshot below.

```
468     for prop_number, prop_info in props.items():
469         prop_room = prop_info[0]
470         prop_y = prop_info[1]
471         prop_x = prop_info[2]
472         if (prop_room == current_room and
473             room_map[prop_y][prop_x] in [0, 39, 2]):
474                 room_map[prop_y][prop_x] = prop_number
475                 image_here = objects[prop_number][0]
476                 image_width = image_here.get_width()
477                 image_width_in_tiles = int(image_width / TILE_SIZE)
478                 for tile_number in range(1, image_width_in_tiles):
479                     room_map[prop_y][prop_x + tile_number] = 255
480
481         hazard_map = [] # empty list
482         for y in range(room_height):
483             hazard_map.append( [0] * room_width )
484
485
486 ###############
487 ## GAME LOOP ##
488 ###############
```

11. Scroll and click at the end of Line 495.

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

12. Press ENTER.

13. Type the code you see on Line 496 of the screenshot below to call the hazard_start() method.

```
486 ###############
487 ## GAME LOOP ##
488 ###############
489
490 def start_room():
491     global airlock_door_frame
492     show_text("You are here: " + room_name, 0)
493     if current_room == 26: # Room with self-shutting airlock door
494         airlock_door_frame = 0
495         clock.schedule_interval(door_in_room_26, 0.05)
496     hazard_start()
497
498 def game_loop():
499     global player_x, player_y, current_room
500     global from_player_x, from_player_y
```

14. Scroll and click at the end of Line 550.

```
548 # check for exiting the room
549     if player_x == room_width: # through door on RIGHT
550         #clock.unschedule(hazard_move)
551         current_room += 1
552         generate_map()
553         player_x = 0 # enter at left
554         player_y = int(room_height / 2) # enter at door
555         player_frame = 0
556         start_room()
557         return
558 
559     if player_x == -1: # through door on LEFT
560         #clock.unschedule(hazard_move)
561         current_room -= 1
562         generate_map()
563         player_x = room_width - 1  # enter at right
564         player_y = int(room_height / 2) # enter at door
565         player_frame = 0
566         start_room()
567         return
568 
569     if player_y == room_height: # through door at BOTTOM
570         #clock.unschedule(hazard_move)
571         current_room += MAP_WIDTH
572         generate_map()
573         player_y = 0 # enter at top
574         player_x = int(room_width / 2) # enter at door
575         player_frame = 0
576         start_room()
577         return
578 
579    if player_y == -1: # through door at TOP
580         #clock.unschedule(hazard_move)
581         current_room -= MAP_WIDTH
582         generate_map()
583         player_y = room_height - 1 # enter at bottom
584         player_x = int(room_width / 2) # enter at door
585         player_frame = 0
586         start_room()
587         return
```

15. Uncomment the clock.unschedule(hazard_move) commands that you see on Lines 550, 560, 570, and 580 of your code by deleting the "#" symbol at the beginning of each line. The screenshot below shows the uncommented code that you should have after you have completed this step.

```
548 # check for exiting the room
549     if player_x == room_width: # through door on RIGHT
550         clock.unschedule(hazard_move)
551         current_room += 1
552         generate_map()
553         player_x = 0 # enter at left
554         player_y = int(room_height / 2) # enter at door
555         player_frame = 0
556         start_room()
557         return
558 
559     if player_x == -1: # through door on LEFT
560         clock.unschedule(hazard_move)
561         current_room -= 1
562         generate_map()
563         player_x = room_width - 1  # enter at right
564         player_y = int(room_height / 2) # enter at door
565         player_frame = 0
566         start_room()
567         return
568 
569     if player_y == room_height: # through door at BOTTOM
570         clock.unschedule(hazard_move)
571         current_room += MAP_WIDTH
572         generate_map()
573         player_y = 0 # enter at top
574         player_x = int(room_width / 2) # enter at door
575         player_frame = 0
576         start_room()
577         return
578 
579    if player_y == -1: # through door at TOP
580         clock.unschedule(hazard_move)
581         current_room -= MAP_WIDTH
582         generate_map()
583         player_y = room_height - 1 # enter at bottom
584         player_x = int(room_width / 2) # enter at door
585         player_frame = 0
586         start_room()
587         return
```

16. Scroll and click at the end of Line 707.

```
697                 if objects[item_here][1] is not None: # If object has a shadow
698                     shadow_image = objects[item_here][1]
699                     # if shadow might need horizontal tiling
700                     if shadow_image in [images.half_shadow,
701                                         images.full_shadow]:
702                         shadow_width = int(image.get_width() / TILE_SIZE)
703                         # Use shadow across width of object.
704                         for z in range(0, shadow_width):
705                             draw_shadow(shadow_image, y, x+z)
706                     else:
707                         draw_shadow(shadow_image, y, x)
708
709         if (player_y == y):
710                 draw_player()
711
712     screen.surface.set_clip(None)
```

17. Press ENTER twice.

18. Type the code you see on Lines 709 – 711 of the screenshot below. Ensure your indentation and punctuation match what is shown in the screenshot.

```
697                 if objects[item_here][1] is not None: # If object has a shadow
698                     shadow_image = objects[item_here][1]
699                     # if shadow might need horizontal tiling
700                     if shadow_image in [images.half_shadow,
701                                         images.full_shadow]:
702                         shadow_width = int(image.get_width() / TILE_SIZE)
703                         # Use shadow across width of object.
704                         for z in range(0, shadow_width):
705                             draw_shadow(shadow_image, y, x+z)
706                     else:
707                         draw_shadow(shadow_image, y, x)
708
709             hazard_here = hazard_map[y][x]
710             if hazard_here != 0: # If there's a hazard at this position
711                 draw_image(objects[hazard_here][0], y, x)
712
713         if (player_y == y):
714                 draw_player()
715
716     screen.surface.set_clip(None)
```

Line 709 creates sets the value of the hazard_here variable to the current value of the y and x coordinate positions on the hazard_map.

Line 710 will check to see if the value of the hazard_here variable is not equal to 0. Remember, when we created the hazard_map list, we set all of the values to 0. We will eventually change the values in the hazard_map list if there is, in fact, a hazard in that particular tile. This function will check to see if the value in the hazard_map list is NOT zero, meaning there is a hazard at that tile.

If this is true, Line 711 will draw the appropriate image that represents the hazard at the appropriate location.

19. Scroll and click at the end of Line 1184.

```
1183      objects[21][0] = frames[airlock_door_frame]
1184      objects[21][1] = shadow_frames[airlock_door_frame]
1185
1186
1187  ################
1188  ##    START    ##
1189  ################
1190
1191  generate_map()
1192  clock.schedule_interval(game_loop, 0.03)
1193  clock.schedule_interval(adjust_wall_transparency, 0.05)
1194  clock.schedule_unique(display_inventory, 1)
```

20. Press ENTER three times.

21. Type the code you see on Lines 1187 – 1203 of the screenshot below. Ensure your indentation, punctuation, and line spacing match what is shown in the screenshot.

```
1183      objects[21][0] = frames[airlock_door_frame]
1184      objects[21][1] = shadow_frames[airlock_door_frame]
1185
1186
1187  ################
1188  ##     AIR     ##
1189  ################
1190
1191  def draw_energy_air():
1192      box = Rect((20, 765), (350, 20))
1193      screen.draw.filled_rect(box, BLACK)
1194      screen.draw.text("AIR", (20, 766), color=BLUE)
1195      screen.draw.text("ENERGY", (180, 766), color=YELLOW)
1196
1197      if air > 0:
1198          box = Rect((50, 765), (air, 20))
1199          screen.draw.filled_rect(box, BLUE) # Draw new air bar.
1200
1201      if energy > 0:
1202          box = Rect((250, 765), (energy, 20))
1203          screen.draw.filled_rect(box, YELLOW) # Draw new energy bar.
1204
1205
1206  ################
1207  ##    START    ##
1208  ################
```

Lines 1187 – 1189 create a new section in your code called AIR.

Line 1191 creates a new method called draw_energy_air.

We begin this function by drawing a black box over the status area at the bottom of the screen to clear it (Lines 1192 – 1193). We then add the AIR label in blue and the ENERGY label in yellow

(Lines 1194 – 1195). This function will use the air and energy variables, which are already set to 100 in the VARIABLES part of the program at the beginning of the code.

If the player has some air left (if the variable air is more than 0), a box is created that uses the air variable for its width (Lines 1197 – 1198). The box is then filled with the color blue (Line 1199). This draws the AIR indicator bar, which starts off being 100 pixels wide and gets smaller as the AIR variable decreases.

We use similar instructions on Lines 1201 – 1203 to draw the energy bar, but the bar's start position is farther to the right.

22. Press ENTER twice.

23. Type the code you see on Lines 1205 – 1225 of the screenshot below. Ensure your indentation, punctuation, and line spacing match what is shown in the screenshot.

```
1187 ###############
1188 ##    AIR    ##
1189 ###############
1190
1191 def draw_energy_air():
1192     box = Rect((20, 765), (350, 20))
1193     screen.draw.filled_rect(box, BLACK)
1194     screen.draw.text("AIR", (20, 766), color=BLUE)
1195     screen.draw.text("ENERGY", (180, 766), color=YELLOW)
1196
1197     if air > 0:
1198         box = Rect((50, 765), (air, 20))
1199         screen.draw.filled_rect(box, BLUE) # Draw new air bar.
1200
1201     if energy > 0:
1202         box = Rect((250, 765), (energy, 20))
1203         screen.draw.filled_rect(box, YELLOW) # Draw new energy bar.
1204
1205 def end_the_game(reason):
1206     global game_over
1207     show_text(reason, 1)
1208     game_over = True
1209     sounds.say_mission_fail.play()
1210     sounds.gameover.play()
1211     screen.draw.text("GAME OVER", (120, 400), color = "white",
1212                      fontsize = 128, shadow = (1, 1), scolor = "black")
1213
1214 def air_countdown():
1215     global air, game_over
1216     if game_over:
1217         return # Don't sap air when they're already dead.
1218     air -= 1
1219     if air == 20:
1220         sounds.say_air_low.play()
1221     if air == 10:
1222         sounds.say_act_now.play()
1223     draw_energy_air()
1224     if air < 1:
1225         end_the_game("You're out of air!")
1226
1227
1228 ###############
1229 ##   START   ##
1230 ###############
```

Line 1205 creates a new function called end_the_game. This function will require the reason to be input whenever it is called.

Line 1206 converts the game_over variable to a global variable.

Line 1207 will display the reason for the character's death on the screen.

Line 1208 will change the value of the game_over variable to True.

Line 1209 will play the say_mission_fail sound and Line 1210 will play the gameover sound.

Lines 1211 – 1212 will draw text that says GAME OVER on the screen in white font with a black drop shadow (scolor means shadow color).

Line 1214 creates another function called air_countdown.

Line 1215 converts and air and game_over variables to global variables.

Line 1216 will check to see if the game_over variable is True. If so, we use a return statement on Line 1217 to exit out of the function and onto the next line of code below of the air_countdown method. We do not need to continue executing the function if the game is over.

Line 1218 will reduce the value of the air variable by 1.

Line 1219 will check to see if the value of the air variable is equal to 20. If this is true, the say_air_low sound will play (Line 1220).

Line 1221 will check to see if the value of the air variable is equal to 10. If this is due, the say_act_now sound will play (Line 1222).

Line 1223 will redraw the energy and air bars using the draw_energy_air function.

Finally, Line 1224 will check to see if the value of the air variable is less than 1. If this is true, Line 1225 will run the end_the_game function using the reason, “You’re out of air!”

24. Press ENTER twice.

25. Type the code you see on Lines 1227 – 1231 of the screenshot below. Ensure your indentation and punctuation match what is shown in the screenshot.

```
1214 def air_countdown():
1215     global air, game_over
1216     if game_over:
1217         return # Don't sap air when they're already dead.
1218     air -= 1
1219     if air == 20:
1220         sounds.say_air_low.play()
1221     if air == 10:
1222         sounds.say_act_now.play()
1223     draw_energy_air()
1224     if air < 1:
1225         end_the_game("You're out of air!")
1226
1227 def alarm():
1228     show_text("Air is running out, " + PLAYER_NAME
1229               + "! Get to safety, then radio for help!", 1)
1230     sounds.alarm.play(3)
1231     sounds.say_breach.play()
1232
1233
1234 ###############
1235 ##   START   ##
1236 ###############
```

Line 1227 creates a new function called alarm.

Lines 1228 – 1229 will display a new message that tells the player that they are running out of air. Lines 1230 and 1231 will play the alarm and say_breach sounds.

26. Press ENTER three times.

27. Type the code you see on Lines 1234 – 1248 of the screenshot below. Ensure your indentation, punctuation, and line spacing match what is shown in the screenshot.

```
1227 def alarm():
1228     show_text("Air is running out, " + PLAYER_NAME
1229               + "! Get to safety, then radio for help!", 1)
1230     sounds.alarm.play(3)
1231     sounds.say_breach.play()
1232
1233
1234 ###############
1235 ##  HAZARDS  ##
1236 ###############
1237
1238 hazard_data = {
1239     # room number: [[y, x, direction, bounce addition to direction]]
1240     28: [[1, 8, 2, 1], [7, 3, 4, 1]], 32: [[1, 5, 4, -1]],
1241     34: [[5, 1, 1, 1], [5, 5, 1, 2]], 35: [[4, 4, 1, 2], [2, 5, 2, 2]],
1242     36: [[2, 1, 2, 2]], 38: [[1, 4, 3, 2], [5, 8, 1, 2]],
1243     40: [[3, 1, 3, -1], [6, 5, 2, 2], [7, 5, 4, 2]],
1244     41: [[4, 5, 2, 2], [6, 3, 4, 2], [8, 1, 2, 2]],
1245     42: [[2, 1, 2, 2], [4, 3, 2, 2], [6, 5, 2, 2]],
1246     46: [[2, 1, 2, 2]],
1247     48: [[1, 8, 3, 2], [8, 8, 1, 2], [3, 9, 3, 2]]
1248     }
1249
1250
1251 ###############
1252 ##   START   ##
1253 ###############
```

Lines 1234 – 1236 create a new section in your code called HAZARDS.

Lines 1238 – 1248 create the hazard_data dictionary. This dictionary uses the room numbers as dictionary keys. For each room, there is a list that contains the data for all the hazards. The data for each hazard is in a list that contains the hazard's y position, x position, starting direction, and number to add to the direction of the hazard when it hits something.

As a note, direction 1 is up, 2 is right, 3 is down, and 4 is left.

28. Press ENTER twice.

29. Type the code you see on Lines 1250 – 1267 of the screenshot below. Ensure your indentation, punctuation, and line spacing match what is shown in the screenshot.

```
1234 ###############
1235 ##  HAZARDS  ##
1236 ###############
1237
1238 hazard_data = {
1239     # room number: [[y, x, direction, bounce addition to direction]]
1240     28: [[1, 8, 2, 1], [7, 3, 4, 1]], 32: [[1, 5, 4, -1]],
1241     34: [[5, 1, 1, 1], [5, 5, 1, 2]], 35: [[4, 4, 1, 2], [2, 5, 2, 2]],
1242     36: [[2, 1, 2, 2]], 38: [[1, 4, 3, 2], [5, 8, 1, 2]],
1243     40: [[3, 1, 3, -1], [6, 5, 2, 2], [7, 5, 4, 2]],
1244     41: [[4, 5, 2, 2], [6, 3, 4, 2], [8, 1, 2, 2]],
1245     42: [[2, 1, 2, 2], [4, 3, 2, 2], [6, 5, 2, 2]],
1246     46: [[2, 1, 2, 2]],
1247     48: [[1, 8, 3, 2], [8, 8, 1, 2], [3, 9, 3, 2]]
1248     }
1249
1250 def deplete_energy(penalty):
1251     global energy, game_over
1252     if game_over:
1253         return # Don't sap energy when they're already dead.
1254     energy = energy - penalty
1255     draw_energy_air()
1256     if energy < 1:
1257         end_the_game("You're out of energy!")
1258
1259 def hazard_start():
1260     global current_room_hazards_list, hazard_map
1261     if current_room in hazard_data.keys():
1262         current_room_hazards_list = hazard_data[current_room]
1263         for hazard in current_room_hazards_list:
1264             hazard_y = hazard[0]
1265             hazard_x = hazard[1]
1266             hazard_map[hazard_y][hazard_x] = 49 + (current_room % 3)
1267         clock.schedule_interval(hazard_move, 0.15)
1268
1269
1270 ###############
1271 ##   START   ##
1272 ###############
```

Line 1250 creates a new method called deplete_energy. This method will require the penalty to be input whenever it is called.

Line 1251 concerts the energy and game_over variables to global variables.

Line 1252 will check to see if the game_over variable is True. If so, Line 1253 will execute a return statement to exit out of the deplete_energy method and discontinue its execution.

Line 1254 will subtract the penalty value from the current value of the energy variable to calculate the new value of the energy variable. Then, the energy bar will be redrawn using the draw_energy_air method on Line 1255.

The Line 1256 will check to see if the value of the energy variable is less than 1. If so, Line 1257 will execute the end_the_game method using the reason, “You’re out of energy!”

Line 1259 creates another new function called hazard_start.

Line 1260 converts the current_room_harzards_list and the hazard_map to global variables.

Line 1261 will check to see if the current_room that the player is in is in the hazard_data.keys, meaning that the current room number is listed as a key in the hazard_data list and that room has a hazard in it.

If so, Line 1262 will input the hazard data for that room to the current_room_hazards_list to store all of the hazards in that room.

Lines 1263 – 1266 will use the hazard data from the current_room_hazards_list to valvulate the y and x location of each room hazard. The three hazard objects have the numbers 49, 50, and 51 in the objects dictionary. The program uses a simple calculation to work out which one goes into a particular room. As you’ve seen before, Python’s % operator gives you the remainder after doing a division. When you divide any number by 3, the remainder will be 0, 1, or 2. So the program divides the room number by 3 and adds the remainder to 49 to pick an object number (Line 1266).

The function finishes on Line 1267 by scheduling the hazard)move function to run every 0.15 seconds.

30. Press ENTER twice.

31. Type the code you see on Lines 1269 – 1274 of the screenshot below. Ensure your indentation, punctuation, and line spacing match what is shown in the screenshot.

```
1259 def hazard_start():
1260     global current_room_hazards_list, hazard_map
1261     if current_room in hazard_data.keys():
1262         current_room_hazards_list = hazard_data[current_room]
1263         for hazard in current_room_hazards_list:
1264             hazard_y = hazard[0]
1265             hazard_x = hazard[1]
1266             hazard_map[hazard_y][hazard_x] = 49 + (current_room % 3)
1267         clock.schedule_interval(hazard_move, 0.15)
1268
1269 def hazard_move():
1270     global current_room_hazards_list, hazard_data, hazard_map
1271     global old_player_x, old_player_y
1272
1273     if game_over:
1274         return
1275
1276
1277 ###############
1278 ##   START   ##
1279 ###############
```

Line 1269 creates a new function called hazard_move.

Lines 1270 and 1271 convert the current_room_hazards_list, hazard_data, hazard_map, old_player_x, and old_player_y to global variables.

Line 1273 will check to see if the game_over variable is True. If so, Line 1274 will execute a return statement to exit out of this function and stop its execution.

32. Press ENTER twice.

33. Type the code you see on Lines 1276 – 1294 of the screenshot below. Ensure your indentation, punctuation, and line spacing match what is shown in the screenshot.

```
1269 def hazard_move():
1270     global current_room_hazards_list, hazard_data, hazard_map
1271     global old_player_x, old_player_y
1272
1273     if game_over:
1274         return
1275
1276     for hazard in current_room_hazards_list:
1277         hazard_y = hazard[0]
1278         hazard_x = hazard[1]
1279         hazard_direction = hazard[2]
1280
1281         old_hazard_x = hazard_x
1282         old_hazard_y = hazard_y
1283         hazard_map[old_hazard_y][old_hazard_x] = 0
1284
1285         if hazard_direction == 1: # up
1286             hazard_y -= 1
1287         if hazard_direction == 2: # right
1288             hazard_x += 1
1289         if hazard_direction == 3: # down
1290             hazard_y += 1
1291         if hazard_direction == 4: # left
1292             hazard_x -= 1
1293
1294         hazard_should_bounce = False
1295
1296
1297 ################
1298 ##   START    ##
1299 ################
```

The hazard_move function uses an idea similar to the player movement. The hazard's position is stored in the old_hazard_x and old_hazard_y variables. The hazard is then moved using the if statements on Lines 1285 – 1294.

34. Press ENTER twice.

35. Type the code you see on Lines 1296 – 1301 of the screenshot below. Ensure your indentation and punctuation match what is shown in the screenshot.

```
1276      for hazard in current_room_hazards_list:
1277          hazard_y = hazard[0]
1278          hazard_x = hazard[1]
1279          hazard_direction = hazard[2]
1280
1281          old_hazard_x = hazard_x
1282          old_hazard_y = hazard_y
1283          hazard_map[old_hazard_y][old_hazard_x] = 0
1284
1285          if hazard_direction == 1: # up
1286              hazard_y -= 1
1287          if hazard_direction == 2: # right
1288              hazard_x += 1
1289          if hazard_direction == 3: # down
1290              hazard_y += 1
1291          if hazard_direction == 4: # left
1292              hazard_x -= 1
1293
1294          hazard_should_bounce = False
1295
1296           if (hazard_y == player_y and hazard_x == player_x) or \
1297             (hazard_y == from_player_y and hazard_x == from_player_x
1298              and player_frame > 0):
1299              sounds.ouch.play()
1300              deplete_energy(10)
1301              hazard_should_bounce = True
1302
1303
1304 ################
1305 ##    START    ##
1306 ################
```

Lines 1296 – 1298 will check to see if the hazard has hit the player. If so, Line 1299 will play the ouch sound. Line 1300 will run the deplete_energy function to deplete the player's energy by 10. Line 1301 will change the value of the hazard_should_bounce variable to True (more on this later).

36. Press ENTER twice.

37. Type the code you see on Lines 1303 – 1315 of the screenshot below. Ensure your indentation and punctuation match what is shown in the screenshot.

```
1294            hazard_should_bounce = False
1295
1296            if (hazard_y == player_y and hazard_x == player_x) or \
1297               (hazard_y == from_player_y and hazard_x == from_player_x
1298                and player_frame > 0):
1299                sounds.ouch.play()
1300                deplete_energy(10)
1301                hazard_should_bounce = True
1302
1303            # Stop hazard going out of the doors
1304            if hazard_x == room_width:
1305                hazard_should_bounce = True
1306                hazard_x = room_width - 1
1307            if hazard_x == -1:
1308                hazard_should_bounce = True
1309                hazard_x = 0
1310            if hazard_y == room_height:
1311                hazard_should_bounce = True
1312                hazard_y = room_height - 1
1313            if hazard_y == -1:
1314                hazard_should_bounce = True
1315                hazard_y = 0
1316
1317
1318 ###############
1319 ##   START   ##
1320 ###############
```

Line 1303 contains a comment.

Lines 1304 – 1315 contain statements that check to see if the hazard has gone out the room door by checking the hazards x and y locations. In each instance, the hazard_should_bounce variable is changed to True and the hazard is moved back into the edge of the room.

38. Press ENTER twice.

39. Type the code you see on Lines 1317 – 1334 of the screenshot below. Ensure your indentation, punctuation, and line spacing match what is shown in the screenshot.

```
1294            hazard_should_bounce = False
1295
1296            if (hazard_y == player_y and hazard_x == player_x) or \
1297               (hazard_y == from_player_y and hazard_x == from_player_x
1298                and player_frame > 0):
1299                sounds.ouch.play()
1300                deplete_energy(10)
1301                hazard_should_bounce = True
1302
1303            # Stop hazard going out of the doors
1304            if hazard_x == room_width:
1305                hazard_should_bounce = True
1306                hazard_x = room_width - 1
1307            if hazard_x == -1:
1308                hazard_should_bounce = True
1309                hazard_x = 0
1310            if hazard_y == room_height:
1311                hazard_should_bounce = True
1312                hazard_y = room_height - 1
1313            if hazard_y == -1:
1314                hazard_should_bounce = True
1315                hazard_y = 0
1316
1317            # Stop when hazard hits scenery or another hazard.
1318            if room_map[hazard_y][hazard_x] not in items_player_may_stand_on \
1319                    or hazard_map[hazard_y][hazard_x] != 0:
1320                hazard_should_bounce = True
1321
1322            if hazard_should_bounce:
1323                hazard_y = old_hazard_y # Move back to last valid position.
1324                hazard_x = old_hazard_x
1325                hazard_direction += hazard[3]
1326                if hazard_direction > 4:
1327                    hazard_direction -= 4
1328                if hazard_direction < 1:
1329                    hazard_direction += 4
1330                hazard[2] = hazard_direction
1331
1332            hazard_map[hazard_y][hazard_x] = 49 + (current_room % 3)
1333            hazard[0] = hazard_y
1334            hazard[1] = hazard_x
1335
1336
1337 ################
1338 ##   START    ##
1339 ################
```

Line 1317 contains a comment.

Lines 1318 – 1319 will check to see if the hazard has hit scenery or another hazard in the room. If this is true, Line 1320 will change the value of the hazard_should_bounce variable to True.

Line 1322 will check to see if the hazard_should_bounce variable is set to True. In the past few steps, we have set this variable to True if the hazard has hit anything (a player, the edge of the room, scenery, or another hazard).

If this is true, Lines 1323 – 1324 will reset the hazard's position back to its old position. Line 1325 will change the direction of the hazard by adding the last number it its list of data to the direction number. If adding this number increases the direction number to a number more than 4, the function subtracts 4 because 4 is the highest valid direction number (Lines 1326 – 1327). On the other hand, if adding the hazard_direction number decreases the direction number to less than 1, the function adds 4 (Lines 1328 – 1329). Finally, the new direction is saved in the hazard data list (Line 1330).

At the end of the function, the appropriate hazard is put into the hazard map at the appropriate y and x location (Lines 1332 – 1334).

40. Ensure your “START” comment runs on Lines 1337 – 1339, as shown in the screenshot below.

```
1327                hazard_direction -= 4
1328            if hazard_direction < 1:
1329                hazard_direction += 4
1330            hazard[2] = hazard_direction
1331
1332        hazard_map[hazard_y][hazard_x] = 49 + (current_room % 3)
1333        hazard[0] = hazard_y
1334        hazard[1] = hazard_x
1335
1336
1337 ###############
1338 ##   START   ##
1339 ###############
```

41. Scroll and click at the end of Line 1344.

```
1337 ###############
1338 ##   START   ##
1339 ###############
1340
1341 generate_map()
1342 clock.schedule_interval(game_loop, 0.03)
1343 clock.schedule_interval(adjust_wall_transparency, 0.05)
1344 clock.schedule_unique(display_inventory, 1)
```

42. Press ENTER.

43. Type the code you see on Lines 1345 - 1349 of the screenshot below.

```
1337 ###############
1338 ##   START   ##
1339 ###############
1340
1341 generate_map()
1342 clock.schedule_interval(game_loop, 0.03)
1343 clock.schedule_interval(adjust_wall_transparency, 0.05)
1344 clock.schedule_unique(display_inventory, 1)
1345 clock.schedule_unique(draw_energy_air, 0.5)
1346 clock.schedule_unique(alarm, 10)
1347 # A higher number below gives a longer time limit.
1348 clock.schedule_interval(air_countdown, 5)
1349 sounds.mission.play() # Intro music
```

Lines 1345 – 1346 will use the clock.schedule_unique function to run the draw_energy_air and alarm functions once, after a delay of .5 seconds and 10 seconds, respectively. These functions will run once, after a delay, when the program starts.

Line 1347 contains a comment.

Line 1348 will run the air_countdown function every 5 seconds.

Line 1349 will play the mission sound.

44. Go to File > Save. Your game is complete!

Final Code:

```
# Escape

import time, random, math

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

WIDTH = 800 #window size
HEIGHT = 800

#PLAYER variables
PLAYER_NAME = "Alice"
FRIEND1_NAME = "Jack"
FRIEND2_NAME = "Matthew"
current_room = 32 # start room = 31

top_left_x = 100
top_left_y = 160

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)

TILE_SIZE = 30

player_y, player_x = 2, 5
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_80,
    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 = (120, 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
        ["Noodle Mission Control", 9, 13, False, True], # room 28
        ["The viewing gallery", 9, 13, 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 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", 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 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 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 rock. Its coarse surface feels like a wheatstone", "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. Use 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, suit, toxic floor.
items_player_may_stand_on = items_player_may_carry + [9, 39, 2, 48]


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

# Scenery describes objects that cannot move between rooms.
# room number: [[object number, y position, x position]...]
scenery = {
    26: [[38,8,2]],
    27: [[33,5,5], [33,1,1], [33,1,9], [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], [10,1,1], [19,4,2], [17,4,4]],
    30: [[34,1,1], [35,1,9]],
    31: [[11,1,1], [19,1,5], [46,1,3]],
    32: [[48,2,2], [48,2,3], [48,2,4], [48,3,2], [48,3,3],
         [48,3,4], [48,4,2], [48,4,3], [48,4,4]],
    33: [[13,1,1], [13,1,9], [13,1,5], [13,1,10], [48,2,1],
         [48,2,7], [48,3,6], [48,3,8]],
    34: [[37,2,2], [32,6,7], [37,10,4], [28,5,3]],
    35: [[16,2,9], [16,2,2], [16,3,3], [16,3,8], [16,8,9], [16,8,2], [16,1,5],
         [16,1,3], [12,0,6], [12,9,5], [12,9,8],
         [15,5,6], [12,7,1], [12,7,11]],
    36: [[4,3,1], [9,1,7], [8,1,9], [8,1,9],
         [5,5,4], [6,5,7], [10,1,1], [12,1,2]],
    37: [[48,3,1], [48,3,2], [48,7,1], [48,5,2], [48,5,3],
         [48,7,2], [48,9,2], [48,9,3], [48,11,1], [48,11,2]],
    38: [[43,0,2], [6,2,2], [6,9,5], [6,4,7], [6,2,9], [48,1,10]],
    39: [[38,1,1], [7,3,4], [7,6,4], [5,3,6], [5,6,6],
         [6,3,9], [6,6,9], [45,1,11], [12,1,8], [12,1,4]],
    40: [[41,5,9], [41,5,7], [41,9,9], [41,9,7],
         [13,1,1], [19,1,8], [42,1,12]],
    41: [[4,3,1], [10,3,5], [4,5,1], [10,5,5], [4,7,1],
         [10,7,5], [12,1,1], [12,1,5]],
    42: [[46,4,3], [46,4,5], [25,1,1], [19,1,3],
         [25,1,5], [52,4,7], [14,1,8]],
    45: [[48,2,1], [48,2,2], [48,3,3], [48,3,4], [48,1,4], [48,1,11],
    46: [[20,1,1], [4,1,2], [5,1,7], [8,1,8], [8,1,9], [5,4,3], [7,3,21],
    47: [[9,1,1], [9,1,2], [10,1,3], [21,1,7], [5,8,4], [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],
         [17,8,1], [17,8,2], [17,8,3], [17,8,4],
         [17,8,5], [17,8,6], [17,8,7], [14,1,11]],
    49: [[14,2,2], [14,2,4], [7,5,1], [5,5,3], [45,3,3], [48,3,4]],
    50: [[45,4,8], [11,1,1], [13,1,8], [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, 20)]]

# 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][-2] # 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] = 255

```

```
    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

        hazard_map = [] # empty list
        for y in range(room_height):
            hazard_map.append( [0] * room_width )


###############
## 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)
    hazard_start()

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(0.05)
        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 += HALF_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), (800, top_left_y + (room_height - 1)*80))
    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], 5, 2)
        image_on_pad = room_map[5][2]
        if image_on_pad > 0:
            draw_image(objects[image_on_pad][0], 5, 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 - 2
                    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)

            hazard_here = hazard_map[y][x]
            if hazard_here != 0: # If there's a hazard at this position
                draw_image(objects[hazard_here][0], 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, 5, 6], 53: [45, 1, 5], 54: [0, 0, 0], 55: [0, 0, 0],
    56: [0, 0, 0], 57: [35, 4, 6], 58: [0, 0, 0], 59: [32, 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: [39, 5, 6], 67: [46, 1, 2],
    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: [28, 1, 11], 75: [0, 0, 0], 76: [41, 3, 5], 77: [0, 0, 0],
    78: [35, 5, 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) == 57, "Expected 57 prop items"
print("Prop checksum", checksum)
assert checksum == 61414, "Error in props data"


in_my_pockets = [56]
selected_item = 0 # the first item
item_carrying = in_my_pockets[selected_item]


RECIPES = {
    (62, 35, 63): (76, 26, 77), (76, 38, 54): (73, 74, 75),
    (59, 54, 60): (77, 55, 56), (56, 57, 58): (71, 65, 72),
    (68, 58, 69): (59, 66, 90), (67, 35, 66):
    }

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 INTERACTION ##
######################

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, (25 + (46 * item_counter), 90))

    box_left = (selected_marker * 46) - 9
    box = Rect((22 + box_left, 85), (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, 38]: # 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(0.6)
    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: "Air is running 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 than 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(0.6)
        # 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[58][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 13, outside."
        props[49][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(72)
            use_message = "You found the Foodle 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 == 68 or item_player_is_on == 68:
        energy = 100
        use_message = "You use the food to restore your energy"
        remove_object(68)
        draw_energy_air()

    if suit_stitched and air_fixed: # open airlock access
        if current_room == 31 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] == 31:
            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, 600))
    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 < 9:
        draw_image(images.rescue_ship, 8 - launch_frame, 6)
        draw_shadow(images.rescue_ship_shadow, 8 + launch_frame, 6)
        clock.schedule(game_completion_sequence, 0.25)
    else:
        screen.surface.set_clip(None)
        screen.draw.text("MISSION", (200, 350), color = "white",
                         fontsize = 128, shadow = (1, 1), scolor = "black")
        screen.draw.text("COMPLETE", (145, 480), 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][1] = 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 == 8 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 != 8 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 as 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]


##############
##    AIR   ##
##############

def draw_energy_air():
    box = Rect((20, 765), (350, 20))
    screen.draw.filled_rect(box, BLACK)
    screen.draw.text("AIR", (20, 766), color=BLUE)
    screen.draw.text("ENERGY", (180, 766), color=YELLOW)

    if air > 0:
        box = Rect((50, 765), (air, 20))
        screen.draw.filled_rect(box, BLUE) # Draw new air bar.

    if energy > 0:
        box = Rect((250, 765), (energy, 20))
        screen.draw.filled_rect(box, YELLOW) # Draw new energy bar.
```

```
def end_the_game(reason):
    global game_over
    show_text(reason, 1)
    game_over = True
    sounds.say_mission_fail.play()
    sounds.gameover.play()
    screen.draw.text("GAME OVER", (120, 400), color = "white",
                     fontsize = 128, shadow = (1, 1), scolor = "black")

def air_countdown():
    global air, game_over
    if game_over:
        return # Don't sap air when they're already dead.
    air -= 1
    if air == 20:
        sounds.say_air_low.play()
    if air == 10:
        sounds.say_act_now.play()
    draw_energy_air()
    if air < 1:
        end_the_game("You're out of air!")

def alarm():
    show_text("Air is running out, " + PLAYER_NAME
              + "! Get to safety, then radio for help!", 1)
    sounds.alarm.play(3)
    sounds.say_breach.play()


###############
##  HAZARDS  ##
###############

hazard_data = {
    # room number: [[y, x, direction, bounce addition to direction]]
    28: [[2, 8, 2, 1], [7, 3, 4, 1]], 32: [[1, 5, 4, -1]],
    34: [[5, 1, 1, 1], [5, 5, 1, 2]], 35: [[4, 4, 1, 2], [2, 5, 2, 2]],
    36: [[2, 1, 2, 2]], 38: [[1, 4, 3, 2], [5, 0, 1, 2]],
    40: [[3, 1, 3, -1], [6, 5, 2, 2], [7, 5, 4, 2]],
    41: [[4, 5, 2, 2], [6, 3, 4, 2], [8, 1, 2, 2]],
    42: [[2, 1, 2, 2], [4, 3, 2, 2], [6, 5, 2, 2]],
    46: [[2, 1, 2, 2]],
    48: [[1, 0, 2, 2], [5, 8, 1, 2], [8, 9, 3, 2]]
    }

def deplete_energy(penalty):
    global energy, game_over
    if game_over:
        return # Don't sap energy when they're already dead.
    energy = energy - penalty
    draw_energy_air()
    if energy < 1:
        end_the_game("You're out of energy!")

def hazard_start():
    global current_room_hazards_list, hazard_map
    if current_room in hazard_data.keys():
        current_room_hazards_list = hazard_data[current_room]
        for hazard in current_room_hazards_list:
            hazard_y = hazard[0]
            hazard_x = hazard[1]
            hazard_map[hazard_y][hazard_x] = 49 + (current_room % 3)
        clock.schedule_interval(hazard_move, 0.15)

def hazard_move():
    global current_room_hazards_list, hazard_data, hazard_map
    global old_player_x, old_player_y

    if game_over:
        return

    for hazard in current_room_hazards_list:
        hazard_y = hazard[0]
        hazard_x = hazard[1]
        hazard_direction = hazard[2]

        old_hazard_x = hazard_x
        old_hazard_y = hazard_y
        hazard_map[old_hazard_y][old_hazard_x] = 0

        if hazard_direction == 1: # up
            hazard_y -= 1
        if hazard_direction == 2: # right
            hazard_x += 1
        if hazard_direction == 3: # down
            hazard_y += 1
        if hazard_direction == 4: # left
            hazard_x -= 1

        hazard_should_bounce = False

        if (hazard_y == player_y and hazard_x == player_x) or \
           (hazard_y == from_player_y and hazard_x == from_player_x
            and player_frame > 0):
            sounds.ouch.play()
            deplete_energy(10)
            hazard_should_bounce = True

        # Stop hazard going out of the doors
        if hazard_x == room_width:
            hazard_should_bounce = True
            hazard_x = room_width - 1
        if hazard_x == -1:
            hazard_should_bounce = True
            hazard_x = 0
        if hazard_y == room_height:
            hazard_should_bounce = True
            hazard_y = room_height - 1
        if hazard_y == -1:
            hazard_should_bounce = True
            hazard_y = 0

        # Stop when hazard hits scenery or another hazard.
        if room_map[hazard_y][hazard_x] not in items_player_may_stand_on \
            or hazard_map[hazard_y][hazard_x] != 0:
            hazard_should_bounce = True

        if hazard_should_bounce:
            hazard_y = old_hazard_y # Move back to last valid position.
            hazard_x = old_hazard_x
            hazard_direction += hazard[3]
            if hazard_direction > 4:
                hazard_direction -= 4
            if hazard_direction < 1:
                hazard_direction += 4
            hazard[2] = hazard_direction

        hazard_map[hazard_y][hazard_x] = 49 + (current_room % 3)
        hazard[0] = hazard_y
        hazard[1] = hazard_x


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

generate_map()
clock.schedule_interval(game_loop, 0.03)
clock.schedule_interval(adjust_wall_transparency, 0.05)
clock.schedule_unique(display_inventory, 1)
clock.schedule_unique(draw_energy_air, 0.5)
clock.schedule_unique(alarm, 10)
# A higher number below gives a longer time limit.
clock.schedule_interval(air_countdown, 5)
sounds.mission.play() # Intro music
```