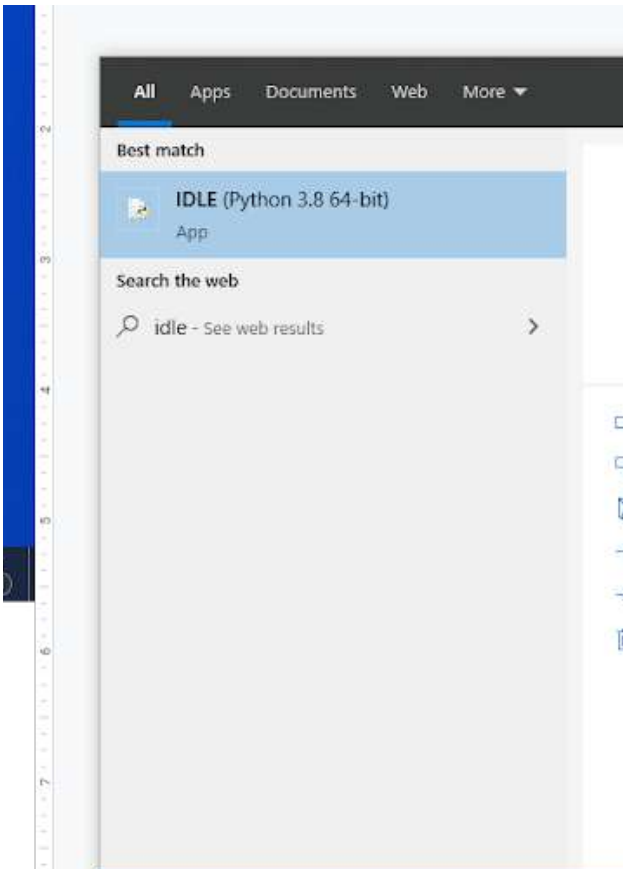
Part 7
Space Mission Directions

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

2. Locate the Space Mission Part 7 assignment.

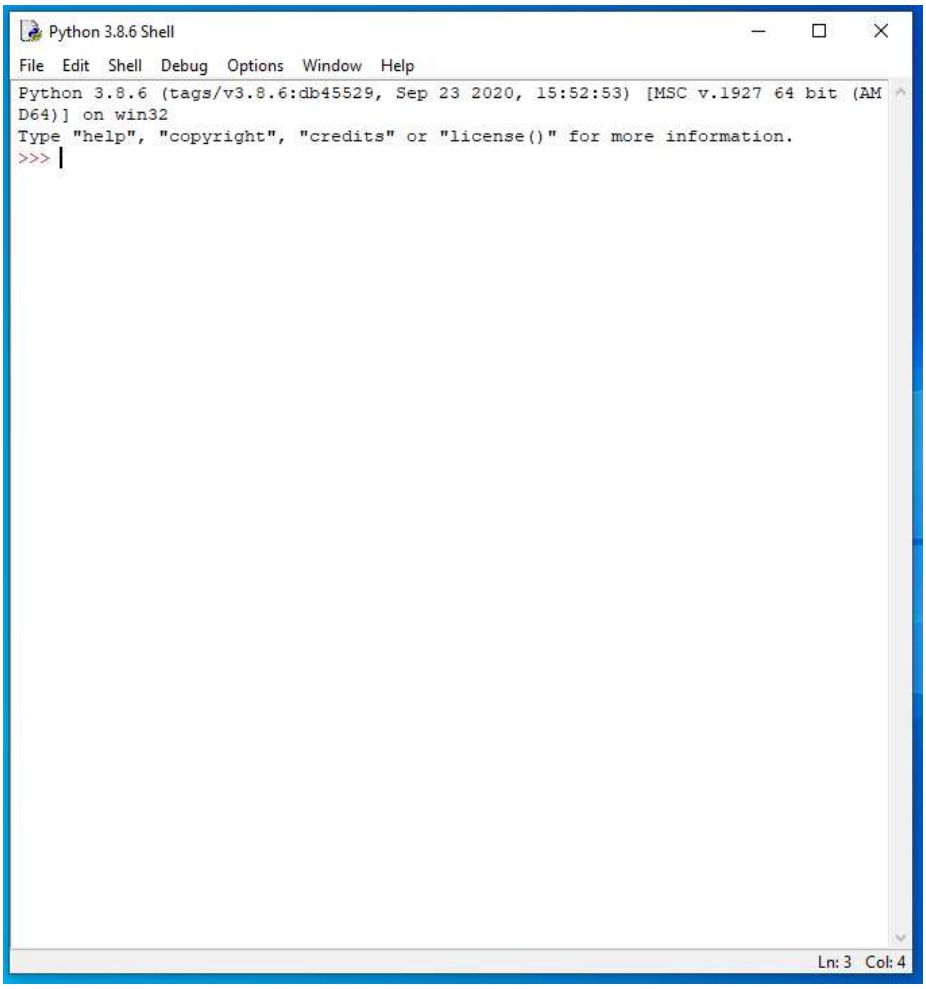
3. We are now ready to start adding code to our file. Using your Windows button menu, find and launch your IDLE program.



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.

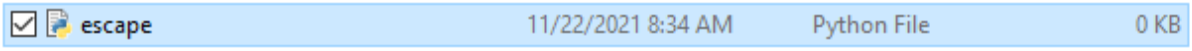
4. Your IDLE window should look something like this once it has launched.:



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 89.

```
82 wall_transparency_frame = 0
83
84 BLACK = (0, 0, 0)
85 BLUE = (0, 155, 255)
86 YELLOW = (255, 255, 0)
87 WHITE = (255, 255, 255)
88 GREEN = (0, 255, 0)
89 RED = (128, 0, 0)
90
91
92 ###############
93 ##    MAP    ##
94 ###############
```

8. Press ENTER twice.

9. Type the code you see on Lines 91 – 93 of the screenshot below.

```
84 BLACK = (0, 0, 0)
85 BLUE = (0, 155, 255)
86 YELLOW = (255, 255, 0)
87 WHITE = (255, 255, 255)
88 GREEN = (0, 255, 0)
89 RED = (128, 0, 0)
90
91 air, energy = 100, 100
92 suit_stitched, air_fixed = False, False
93 launch_frame = 0
94
95
96 ###############
97 ##    MAP    ##
98 ###############
```

Line 91 creates two new variables called air and energy. Both variables are set to 100.

Line 92 creates two more new variables called suit_stitched and air_fixed. Both variables are set to False.

Line 93 creates a fifth new variable called launch_frame. Its initial value is set to 0.

10. Ensure the “MAP” comment runs from Lines 96 – 98 of your code.

11. Scroll down and click at the end of Line 594.

```
590     if keyboard.d and item_carrying:
591         drop_object(old_player_y, old_player_x)
592
593     if keyboard.space:
594         examine_object()
595
596
597      # If the player is standing somewhere they shouldn't, move them back.
598     if room_map[player_y][player_x] not in items_player_may_stand_on: #\
599     #           or hazard_map[player_y][player_x] != 0:
600         player_x = old_player_x
```

12. Press ENTER twice.

13. Type the code you see on Lines 596 – 597 of the screenshot below. Ensure your indentation matches what is shown in the screenshot.

```
593     if keyboard.space:
594         examine_object()
595
596     if keyboard.u:
597         use_object()
598
599
600      # If the player is standing somewhere they shouldn't, move them back.
601     if room_map[player_y][player_x] not in items_player_may_stand_on: #\
602     #           or hazard_map[player_y][player_x] != 0:
603         player_x = old_player_x
604         player_y = old_player_y
605         player_frame = 0
```

Line 596 checks to see if the u key on the keyboard has been pressed. If it has, Line 597 will run the use_object function, which we haven’t written yet.

14. Delete the blank line on Line 598.

```
590     if keyboard.d and item_carrying:
591         drop_object(old_player_y, old_player_x)
592
593     if keyboard.space:
594         examine_object()
595
596     if keyboard.u:
597         use_object()
598
599      # If the player is standing somewhere they shouldn't, move them back.
600     if room_map[player_y][player_x] not in items_player_may_stand_on: #\
601     #           or hazard_map[player_y][player_x] != 0:
602         player_x = old_player_x
603         player_y = old_player_y
604         player_frame = 0
```

15. Scroll and click at the end of Line 764.

```
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]


########################
## PROP INTERACTIONS ##
########################
```

16. Press ENTER twice.

17. Type the code you see on Lines 767 – 771 of the screenshot below. Ensure your indentation and punctuation match what is shown in the screenshot below.

```
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, 54], [73, 74, 75],
    [59, 54, 60], [77, 55, 56], [56, 57, 58], [71, 65, 72],
    [88, 58, 89], [89, 60, 90], [67, 35, 68]
    ]

########################
## PROP INTERACTIONS ##
########################
```

In this game's programming, combinations are called recipes. A single recipe contains three object numbers in a list. The first two are the items that are combined, and the third one is the object number they make when they're combined.

When you combine objects, the new object goes into your inventory. The objects you combined are removed from the game if they're props. Sometimes one will be a piece of scenery and so will remain in the game.

Line 767 begins a list of recipes that lists the two items that can be combined to create the third item.

18. Press ENTER twice.

19. Type the code you see on Lines 773 – 783 of the screenshot below. Ensure your indentation and punctuation match what is shown in the screenshot.

```
RECIPES = [
    [62, 35, 63], [76, 28, 77], [78, 38, 54], [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 ##
########################
```

Just as we have done before, Lines 773 – 783 contain a checksum to ensure that all code has been entered accurately. If not, the game will not run.

20. Ensure that the “PROP INTERACTIONS” comment runs on Lines 786 – 788 of your code.

21. Scroll and click at the end of Line 891.

```
874 def examine_object():
875     item_player_is_on = get_item_under_player()
876     left_tile_of_item = find_object_start_x()
877     if item_player_is_on in [0, 2]: # don't describe the floor
878         return
879     description = "You see: " + objects[item_player_is_on][2]
880     for prop_number, details in props.items():
881         # props = object number: [room number, y, x]
882         if details[0] == current_room: # if prop is in the room
883             # If prop is hidden (= at player's location but not on map)
884             if (details[1] == player_y
885                 and details[2] == left_tile_of_item
886                 and room_map[details[1]][details[2]] != prop_number):
887                 add_object(prop_number)
888                 description = "You found " + objects[prop_number][3]
889                 sounds.combine.play()
890     show_text(description, 0)
891     time.sleep(0.5)
892
893
894 ################
895 ##   START    ##
896 ################
897
898 generate_map()
899 clock.schedule_interval(game_loop, 0.03)
900 clock.schedule_interval(adjust_wall_transparency, 0.05)
901 clock.schedule_unique(display_inventory, 1)
902
```

22. Press ENTER three times.

23. Type the code you see on Lines 894 – 920 of the screenshot below. Ensure your indentation, punctuation, and line spacing match what is shown in the screenshot.

```
890     show_text(description, 0)
891     time.sleep(0.5)
892
893
894 ##################
895 ## USE OBJECTS ##
896 ##################
897
898 def use_object():
899     global room_map, props, item_carrying, air, selected_item, energy
900     global in_my_pockets, suit_stitched, air_fixed, game_over
901
902     use_message = "You fiddle around with it but don't get anywhere."
903     standard_responses = {
904         4: "Air is running out! You can't take this lying down!",
905         6: "This is no time to sit around!",
906         7: "This is no time to sit around!",
907         32: "It shakes and rumbles, but nothing else happens.",
908         34: "Ah! That's better. Now wash your hands.",
909         35: "You wash your hands and shake the water off.",
910         37: "The test tubes smoke slightly as you shake them.",
911         54: "You chew the gum. It's sticky like glue.",
912         55: "The yoyo bounces up and down, slightly slower than on Earth",
913         56: "It's a bit too fiddly. Can you thread it on something?",
914         59: "You need to fix the leak before you can use the canister",
915         61: "You try signalling with the mirror, but nobody can see you.",
916         62: "Don't throw resources away. Things might come in handy...",
917         67: "To enjoy yummy space food, just add water!",
918         75: "You are at Sector: " + str(current_room) + " // X: " \
919             + str(player_x) + " // Y: " + str(player_y)
920         }
921
922
923 ###############
924 ##   START   ##
925 ###############
```

Lines 894 – 896 create the USE OBJECTS section of the code.

Line 898 creates a new function called use_object.

Lines 899 – 900 convert the room_map, item_carrying, air, selected_i9tem, energy, in_my_pockets, suit_stitched, air_fixed, and game_over variables to global variables so that the use_object function can modify them.

Line 902 creates the use_message variable and inputs a default message for the object.

Line 903 creates a standard_responses dictionary with a variety of messages that can be displayed for the player. Some of the objects have no real function in the game but will reward the player with a message when they try to use them. These messages could include clues as well as add to the game story. The dictionary standard_responses contains messages to show players when they use certain objects, identified by their object number. For example, if they want to use the bed, which is object 4, they see a message that says, “You can’t take this lying down!”

24. Press ENTER twice.

25. Type the code you see on Lines 922 – 930 of the screenshot below. Ensure your indentation, punctuation, and line spacing match what is being shown in the screenshot.

```
898 def use_object():
899     global room_map, props, item_carrying, air, selected_item, energy
900     global in_my_pockets, suit_stitched, air_fixed, game_over
901
902     use_message = "You fiddle around with it but don't get anywhere."
903     standard_responses = {
904         4: "Air is running out! You can't take this lying down!",
905         6: "This is no time to sit around!",
906         7: "This is no time to sit around!",
907         32: "It shakes and rumbles, but nothing else happens.",
908         34: "Ah! That's better. Now wash your hands.",
909         35: "You wash your hands and shake the water off.",
910         37: "The test tubes smoke slightly as you shake them.",
911         54: "You chew the gum. It's sticky like glue.",
912         55: "The yoyo bounces up and down, slightly slower than on Earth",
913         56: "It's a bit too fiddly. Can you thread it on something?",
914         59: "You need to fix the leak before you can use the canister",
915         61: "You try signalling with the mirror, but nobody can see you.",
916         62: "Don't throw resources away. Things might come in handy...",
917         67: "To enjoy yummy space food, just add water!",
918         75: "You are at Sector: " + str(current_room) + " // X: " \
919             + str(player_x) + " // Y: " + str(player_y)
920         }
921
922     # Get object number at player's location.
923     item_player_is_on = get_item_under_player()
924     for this_item in [item_player_is_on, item_carrying]:
925         if this_item in standard_responses:
926             use_message = standard_responses[this_item]
927
928     if item_carrying == 70 or item_player_is_on == 70:
929         use_message = "Banging tunes!"
930         sounds.steelmusic.play(2)
931
932
933 ###############
934 ##   START   ##
935 ###############
```

Line 922 contains a comment.

Line 923 begins the process of figuring out which message the display to the player. The variable item_the_player_is_on stores the object number at the player's position in the room (Line 923). Players can use objects they are carrying or standing on. On Line 924, we set up a loop that goes through a list the contains two items: the item number the player is standing on and the item number the player is carrying. If either of them is a key for the standard_responses dictionary (Line 925), the use_message is updated to the object's message from that dictionary (Line 926). The program prioritizes items you're carrying over items you're standing on if they both have standard messages.

Line 928 will check to see if the player is carrying item 70 or standing on item number 70, which is an MP3 player. If this is true, it will display the message “Banging tunes!” and play the steelmusic sound.

26. Press ENTER twice.

27. Type the code you see on Lines 932 – 953 of the screenshot below. Ensure your indentation, punctuation, and line spacing match what is shown in the screenshot.

```
922     # Get object number at player's location.
923     item_player_is_on = get_item_under_player()
924     for this_item in [item_player_is_on, item_carrying]:
925         if this_item in standard_responses:
926             use_message = standard_responses[this_item]
927
928     if item_carrying == 70 or item_player_is_on == 70:
929         use_message = "Banging tunes!"
930         sounds.steelmusic.play(2)
931
932     elif item_player_is_on == 11:
933         use_message = "AIR: " + str(air) + \
934                       "% / ENERGY " + str(energy) + "% / "
935         if not suit_stitched:
936             use_message += "*ALERT* SUIT FABRIC TORN / "
937         if not air_fixed:
938             use_message += "*ALERT* SUIT AIR BOTTLE MISSING"
939         if suit_stitched and air_fixed:
940             use_message += " SUIT OK"
941         show_text(use_message, 0)
942         sounds.say_status_report.play()
943         time.sleep(0.5)
944         # If "on" the computer, player intention is clearly status update.
945         # Return to stop another object use accidentally overriding this.
946         return
947
948     elif item_carrying == 60 or item_player_is_on == 60:
949         use_message = "You fix " + objects[60][3] + " to the suit"
950         air_fixed = True
951         air = 90
952         air_countdown()
953         remove_object(60)
954
955
956 ###############
957 ##   START   ##
958 ###############
```

Line 932 will check to see if the player is standing on item number 11, which is a computer. If this is true, a message will be generated (Line 933 – 934) and displayed (Line 941) that combines information from the air and energy variables and adding an alert if the suit or air bottle is faulty (Lines 935 – 940). There’s also a computer speech sound effect that says, “status report!” At the end of the program, there will be a brief pause (Line 943).

The return statement on Line 946 prevents the player from accidentally using another object when they intended to use the computer. If we didn't include this return instruction, the player might end up using another prop that they're carrying instead of the computer.

Line 948 will check to see if the player is carrying or is on item number 60, which is a sealed air canister.

Line 949 updates the use_message variable telling the player that they have fixed the sealed air canister to the suit. Line 950 updates the air_fixed variable to True. Line 951 changes the air variable to 90. Line 952 calls the air_countdown function, which we haven't written yet. Line 953 removes item 60 from the player's inventory using the remove_object function.

28. Press ENTER twice.

29. Type the code you see on Lines 955 – 965 of the screenshot below. Ensure your indentation, line spacing, and punctuation match what is shown in the screenshot.

```
944         # If "on" the computer, player intention is clearly status update.
945         # Return to stop another object use accidentally overriding this.
946         return
947
948     elif item_carrying == 60 or item_player_is_on == 60:
949         use_message = "You fix " + objects[60][3] + " to the suit"
950         air_fixed = True
951         air = 90
952         air_countdown()
953         remove_object(60)
954
955     elif (item_carrying == 58 or item_player_is_on == 58) \
956        and not suit_stitched:
957         use_message = "You use " + objects[56][3] + \
958                       " to repair the suit fabric"
959         suit_stitched = True
960         remove_object(58)
961
962     elif item_carrying == 72 or item_player_is_on == 72:
963         use_message = "You radio for help. A rescue ship is coming. \
964 Rendezvous Sector 13, outside."
965         props[40][0] = 13
966
967
968 ###############
969 ##   START   ##
970 ###############
```

Lines 955 – 956 will check to see if the player is standing on or carrying item 58, which is a needle and that the suit is NOT stitched. If this is true, Lines 957 – 958 will update the use_message variable to tell the player that they use the needle to fix the suit fabric. Line 959 will update the suit_stitched variable to True and Line 960 will remove item 58 from the player's inventor using the remove_object function.

Line 962 will check to see if the player is carrying or is standing on a radio, which is item 72.

Lines 963 – 964 will update the use_message variable to a message indicating the player uses the radio to call for help.

Line 965 will update the room number of prop 40 to room 13, which is an empty area of the ouside area of the space station.

30. Press ENTER twice.

31. Type the code you see on Lines 967 – 982 of the screenshot below. Ensure your indentation, line spacing, and punctuation match what is shown in the screenshot.

```
955     elif (item_carrying == 58 or item_player_is_on == 58) \
956        and not suit_stitched:
957         use_message = "You use " + objects[56][3] + \
958                       " to repair the suit fabric"
959         suit_stitched = True
960         remove_object(58)
961
962     elif item_carrying == 72 or item_player_is_on == 72:
963         use_message = "You radio for help. A rescue ship is coming. \
964 Rendezvous Sector 13, outside."
965         props[40][0] = 13
966
967     elif (item_carrying == 66 or item_player_is_on == 66) \
968             and current_room in outdoor_rooms:
969         use_message = "You dig..."
970         if (current_room == LANDER_SECTOR
971             and player_x == LANDER_X
972             and player_y == LANDER_Y):
973             add_object(71)
974             use_message = "You found the Poodle lander!"
975
976     elif item_player_is_on == 40:
977         clock.unschedule(air_countdown)
978         show_text("Congratulations, "+ PLAYER_NAME +"!", 0)
979         show_text("Mission success! You have made it to safety.", 1)
980         game_over = True
981         sounds.take_off.play()
982         game_completion_sequence()
983
984
985 ###############
986 ##   START   ##
987 ###############
```

Lines 967 - 968 will check to see if the player is on or carrying item 66, which is a large spoon, and that the player is in an outside room.

Line 969 will update the use_message variable.

Lines 970 - 972 will check to see if the player's position is equal to the location of the Poodle lander. If this is true, the Poodle lander will be added to the player's inventory using the add_object method (Line 973) and the use_message variable will be updated to tell the player that they have found the poodle lander (Line 974).

Line 976 will check to see if the player is on object 40, which is a rescue ship. If this is true, the air_countdown will stop (Line 977).

Lines 978 and 979 will display messages telling the player that they have made it to safety.

Line 980 will change the game_over variable to True.

Line 981 will play the take_off sound.

Line 982 will call the game_completion_sequence method, which we haven't written yet.

32. Press ENTER twice.

33. Type the code you see on Lines 984 – 1000 of the screenshot below. Ensure your indentation, line spacing, and punctuation match what is shown in the screenshot.

```
976     elif item_player_is_on == 40:
977         clock.unschedule(air_countdown)
978         show_text("Congratulations, "+ PLAYER_NAME +"!", 0)
979         show_text("Mission success! You have made it to safety.", 1)
980         game_over = True
981         sounds.take_off.play()
982         game_completion_sequence()
983 
984     elif item_player_is_on == 16:
985         energy += 1
986         if energy > 100:
987             energy = 100
988         use_message = "You munch the lettuce and get a little energy back"
989         draw_energy_air()
990 
991     elif item_player_is_on == 42:
992         if current_room == 27:
993             open_door(26)
994         props[25][0] = 0 # Door from RM32 to engineering bay
995         props[26][0] = 0 # Door inside engineering bay
996         clock.schedule_unique(shut_engineering_door, 60)
997         use_message = "You press the button"
998         show_text("Door to engineering bay is open for 60 seconds", 1)
999         sounds.say_doors_open.play()
1000        sounds.doors.play()
1001
1002
1003 ###############
1004 ##   START   ##
1005 ###############
```

Line 984 will check to see if the player is standing on a shrub (item 16). If this is true, the value of the energy variable will be increased by 1 (Line 985).

Line 986 checks to see if the energy variable is larger than 100. If this is true, it will be reset back to its max value of 100.

Line 988 will update the use_message variable. Line 989 will run the draw_energy_air function, which we haven't written yet.

Line 991 will check to see if the player is standing on a button to open a door (object 42). If this is true, and the current room number is 27 (Line 992), the open_door function for item 26 will run. We have not written this function yet.

Lines 994 and 995 will update the room number for props 25 and 26 to 0, removing the door from view.

Line 996 will run the shut_engineering_door function once 60 seconds have elapsed.

Line 997 will update the use_message variable.

Line 998 will display a message on the screen for the user so they know that the door is open for 60 seconds.

Lines 999 – 1000 will play the say_doors_open and doors sounds.

34. Press ENTER twice.

35. Type the code you see on Lines 1002 – 1016 of the screenshot below. Ensure your indentation, punctuation, and line spacing match what is shown in the screenshot.

```
991     elif item_player_is_on == 42:
992         if current_room == 27:
993             open_door(26)
994         props[25][0] = 0 # Door from RM32 to engineering bay
995         props[26][0] = 0 # Door inside engineering bay
996         clock.schedule_unique(shut_engineering_door, 60)
997         use_message = "You press the button"
998         show_text("Door to engineering bay is open for 60 seconds", 1)
999         sounds.say_doors_open.play()
1000        sounds.doors.play()
1001
1002    elif item_carrying == 68 or item_player_is_on == 68:
1003        energy = 100
1004        use_message = "You use the food to restore your energy"
1005        remove_object(68)
1006        draw_energy_air()
1007
1008    if suit_stitched and air_fixed: # open airlock access
1009        if current_room == 31 and props[20][0] == 31:
1010            open_door(20) # which includes removing the door
1011            sounds.say_airlock_open.play()
1012            show_text("The computer tells you the airlock is now open.", 1)
1013        elif props[20][0] == 31:
1014            props[20][0] = 0 # remove door from map
1015            sounds.say_airlock_open.play()
1016            show_text("The computer tells you the airlock is now open.", 1)
1017
1018
1019 ###############
1020 ##   START   ##
1021 ###############
```

Line 1002 will check to see if the player is carrying or standing on item 68, which is a food pouch. If this is true, the value of the energy variable will be set to 100 (Line 1003). Line 1004 will update the use_message variable, and Line 1005 will remove the object from the player's inventory using the remove_object method. Line 1006 will execute the draw_energy_air method, which we haven't written yet.

Line 1008 will check to see if the suit_stitched and air_fixed variables are both set to True.

If this is true, Line 1009 will check to see if the value of the player's current room is 31 and that prop 20 is in room 31. If this is true, Line 1010 will execute the open_door function on prop 20 and Line 1011 will play the say_airlock_open sound. Finally, Line 1012 will display text on the screen to tell the player that that the airlock is open.

Line 1013 will run to see if prop 20 is in room 31. This only checks the location of the prop, not the location of the player, as Line 1009 did. If this is true, Line 1014 will remove the prop 20 from view by moving it to room 0. Line 1015 will play the say_airlock_open sound and Line 1016 will display text on the screen to tell the player that the airlock is open.

36. Press ENTER twice.

37. Type the code you see on Lines 1018 – 1037 of the screenshot below. Ensure your indentation, line spacing, and punctuation match what is shown in the screenshot.

```
1008      if suit_stitched and air_fixed: # open airlock access
1009          if current_room == 31 and props[20][0] == 31:
1010              open_door(20) # which includes removing the door
1011              sounds.say_airlock_open.play()
1012              show_text("The computer tells you the airlock is now open.", 1)
1013          elif props[20][0] == 31:
1014              props[20][0] = 0 # remove door from map
1015              sounds.say_airlock_open.play()
1016              show_text("The computer tells you the airlock is now open.", 1)
1017
1018       for recipe in RECIPES:
1019          ingredient1 = recipe[0]
1020          ingredient2 = recipe[1]
1021          combination = recipe[2]
1022          if (item_carrying == ingredient1
1023              and item_player_is_on == ingredient2) \
1024              or (item_carrying == ingredient2
1025                  and item_player_is_on == ingredient1):
1026              use_message = "You combine " + objects[ingredient1][3] \
1027                            + " and " + objects[ingredient2][3] \
1028                            + " to make " + objects[combination][3]
1029              if item_player_is_on in props.keys():
1030                  props[item_player_is_on][0] = 0
1031                  room_map[player_y][player_x] = get_floor_type()
1032              in_my_pockets.remove(item_carrying)
1033              add_object(combination)
1034              sounds.combine.play()
1035
1036      show_text(use_message, 0)
1037      time.sleep(0.5)
1038
1039
1040  ###############
1041  ##   START   ##
1042  ###############
```

In the code above, we use a loop to go through all the items in the RECIPES list, and a new recipe goes into the recipe list each time. We put the ingredients and combination object numbers into variables to make the function easier to understand. Lines 1019 – 1021 assign the item ingredients in the RECIPES list to different variables.

The program checks whether the player is carrying the first ingredient and standing on the second one, or the other way around (Lines 1022 – 1025). If so, the use_message variable is updated to tell them what they combined and what they made (Lines 1026 – 1028).

When the combined object is made, it usually replaces the ingredient objects. If one of the objects is scenery instead of a prop, though, it remains in the game. So, the program checks whether the item the player is on is a prop, and if so, its room number is set to 0, removing it from the game. If it's a prop, it's also deleted from the room map for the current room (Lines 1029 – 1031).

The object that was being carried is removed from the player's inventory (Line 1032), and the newly created object is added to it (Line 1033).

Finally, Line 1034 plays the combine sound.

38. Press ENTER twice.

39. Type the code you see on Lines 1039 – 1048 of the screenshot below. Ensure your indentation, punctuation, and line spacing match what is shown in the screenshot.

```
1018      for recipe in RECIPES:
1019          ingredient1 = recipe[0]
1020          ingredient2 = recipe[1]
1021          combination = recipe[2]
1022          if (item_carrying == ingredient1
1023              and item_player_is_on == ingredient2) \
1024              or (item_carrying == ingredient2
1025                  and item_player_is_on == ingredient1):
1026              use_message = "You combine " + objects[ingredient1][3] \
1027                            + " and " + objects[ingredient2][3] \
1028                            + " to make " + objects[combination][3]
1029              if item_player_is_on in props.keys():
1030                  props[item_player_is_on][0] = 0
1031                  room_map[player_y][player_x] = get_floor_type()
1032              in_my_pockets.remove(item_carrying)
1033              add_object(combination)
1034              sounds.combine.play()
1035
1036      show_text(use_message, 0)
1037      time.sleep(0.5)
1038
1039  def game_completion_sequence():
1040      global launch_frame #(initial value is 0, set up in VARIABLES section)
1041      box = Rect((0, 150), (800, 600))
1042      screen.draw.filled_rect(box, (128, 0, 0))
1043      box = Rect ((0, top_left_y - 30), (800, 390))
1044      screen.surface.set_clip(box)
1045
1046      for y in range(0, 13):
1047          for x in range(0, 13):
1048              draw_image(images.soil, y, x)
1049
1050
1051  ###############
1052  ##    START    ##
1053  ###############
```

Line 1039 creates a new function called game_completion_sequence. We saw this function called earlier when the player has won the game.

Line 1040 establishes the launch_frame variable as a global variable.

Line 1041 creates a new rect object at the location of 0, 150. The object is 800 pixels wide and 600 pixels tall.

Line 1042 draws the filled rectangle on the screen using the RGB color values of 128, 0, 0.

Line 1043 creates another rec object. Line 1044 turns this rect object into a clipping area.

Line 1046 begins a loop to draw the soil image at the appropriate y and x coordinates.

40. Press ENTER twice.

41. Type the code you see on Lines 1050 – 1062 of the screenshot below. Ensure your indentation and punctuation match what is shown in the screenshot.

```
1039 def game_completion_sequence():
1040     global launch_frame #(initial value is 0, set up in VARIABLES section)
1041     box = Rect((0, 150), (800, 600))
1042     screen.draw.filled_rect(box, (128, 0, 0))
1043     box = Rect ((0, top_left_y - 30), (800, 390))
1044     screen.surface.set_clip(box)
1045
1046     for y in range(0, 13):
1047         for x in range(0, 13):
1048             draw_image(images.soil, y, x)
1049
1050     launch_frame += 1
1051     if launch_frame < 9:
1052         draw_image(images.rescue_ship, 8 - launch_frame, 6)
1053         draw_shadow(images.rescue_ship_shadow, 8 + launch_frame, 6)
1054         clock.schedule(game_completion_sequence, 0.25)
1055     else:
1056         screen.surface.set_clip(None)
1057         screen.draw.text("MISSION", (200, 380), color = "white",
1058                          fontsize = 128, shadow = (1, 1), scolor = "black")
1059         screen.draw.text("COMPLETE", (145, 480), color = "white",
1060                          fontsize = 128, shadow = (1, 1), scolor = "black")
1061         sounds.completion.play()
1062         sounds.say_mission_complete.play()
1063
1064
1065 ###############
1066 ##   START   ##
1067 ###############
```

Line 1050 will increase the value of the launch_frame variable by 1.

Line 1051 will check to see if the launch_frame variable is less than 9. If this is true, the appropriate image and shadow image will be drawn and the game_completion_sequence will run every .25 seconds (Lines 1053 - 1054).

Otherwise, Line 1056 will clear the clipping area previously set. Lines 1057 – 1060 will draw text on the screen to tell the player that their mission is complete. Lines 1061 and 1062 will play the completion and say_mission_complete sounds.

42. Ensure that the “START” comment runs on Lines 1065 – 1067 of your code.

43. 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 = "Alice"
FRIEND1_NAME = "Jack"
FRIEND2_NAME = "Matthew"
current_room = 31 # start room = 31

top_left_x = 100
top_left_y = 150

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

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

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_2_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 = (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", 9, 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 sick bay", 11, 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 postal 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.electrical1, 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 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_AIM, images.robot_AIM_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 barely 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, 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], [33,1,9], [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], [35,1,3]],
    31: [[11,1,1], [19,1,8], [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], [13,1,3], [13,1,5], [13,1,10], [48,2,1],
         [48,2,7], [48,3,6], [45,3,11],
    34: [[37,2,2], [32,6,7], [37,10,4], [23,5,3]],
    35: [[16,2,5], [16,2,2], [16,3,3], [16,3,5], [16,0,9], [16,8,2], [16,1,8],
         [16,1,3], [12,5,6], [12,9,4], [12,9,8],
         [15,4,6], [12,7,1], [12,7,11]],
    36: [[4,3,1], [9,1,7], [5,1,8], [6,1,9],
         [5,5,4], [6,5,7], [10,1,1], [12,1,2]],
    37: [[48,3,1], [48,3,2], [48,7,1], [48,3,2], [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], [6,3,6], [6,4,7], [6,2,5], [45,1,10]],
    39: [[38,1,1], [7,8,4], [7,6,4], [5,3,6], [5,6,6],
         [6,3,9], [6,6,9], [45,1,11], [12,1,5], [12,1,6]],
    40: [[41,5,3], [41,5,7], [41,9,3], [41,9,7],
         [13,1,1], [13,1,3], [42,1,12]],
    41: [[4,3,1], [10,9,5], [5,5,1], [10,6,5], [4,7,1],
         [10,7,5], [12,1,1], [12,1,5]],
    44: [[46,4,3], [46,4,5], [18,1,1], [19,1,3],
         [19,1,5], [32,4,7], [14,1,5]],
    45: [[48,2,1], [48,2,2], [48,3,5], [48,3,4], [48,1,4], [48,1,1]],
    46: [[10,1,1], [4,1,2], [8,1,7], [8,1,8], [8,1,9], [5,4,8], [7,9,3]],
    47: [[9,1,1], [9,1,2], [10,1,3], [12,1,7], [5,4,4], [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],
         [17,8,5], [17,8,6], [17,8,7], [14,1,12]],
    49: [[14,2,2], [14,2,4], [7,5,1], [5,5,3], [48,3,3], [48,3,4]],
    50: [[48,4,8], [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 == 141, "Expected 141 scenery items"
assert checksum == 200055, "Error in scenery data"
print("Scenery checksum: " + str(checksum))

for room in range(1, 26):# Add random scenery in planet locations.
    if room != 19: # Skip room 19.
        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


###############
## GAME 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, 99, 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():
    show_text("You are here: " + room_name, 0)

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 == 3:
            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.q:
        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)*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], 5, 2)
        image_on_pad = room_map[6][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)
                        # Draw shadow across width of object.
                        for z in range(0, shadow_width):
                            draw_shadow(shadow_image, y, x+z)
                    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: [35, 4, 6], 58: [0, 0, 0], 59: [31, 1, 7],
    60: [0, 0, 0], 62: [36, 2, 1], 62: [36, 2, 6], 63: [0, 0, 0],
    64: [27, 8, 3], 65: [50, 1, 7], 66: [39, 5, 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: [28, 1, 13], 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, 36, 63], [76, 28, 77], [70, 38, 64], [73, 74, 75],
    [59, 54, 60], [77, 59, 66], [56, 57, 58], [71, 69, 72],
    [59, 58, 59], [59, 60, 80], [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 == 87296, "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, (25 + (46 * item_counter), 90))

    box_left = (selected_marker * 46) - 3
    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, 99]: # 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: "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 = "Hanging 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 MISSING"
        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 == 56 or item_player_is_on == 56) \
        and not suit_stitched:
        use_message = "You use " + objects[56][3] + \
                      " to repair the suit fabric"
        suit_stitched = True
        remove_object(56)
```

```
    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[40][0] = 19

    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 Boodle 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()

    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, 590))
    screen.surface.set_clip(box)

    for y in range(0, 13):
        for x in range(0, 19):
            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, 990), color = "white",
                         fontsize = 128, shadow = (1, 1), scolor = "black")
        screen.draw.text("COMPLETE", (245, 480), color = "white",
                         fontsize = 128, shadow = (1, 1), scolor = "black")
        sounds.completion.play()
        sounds.say_mission_complete.play()


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

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