# ECS 32A: Programming Assignment #6

Instructor: Aaron Kaloti

Summer Session #1 2020

## Contents

## 1 Changelog

You should always refer to the latest version of this document.

- v.1: Initial version.
- v.2: Removed a small, possible contradiction in the directions for part #2. Clarified part #3 and added actual code examples.
- v.3: Updated weights for each of the parts. Added directions for part #4.
- v.4: Changed the way in the first example output part #4.3 that the types of `g.hero`, the instances in `g.objects`, and the instance in `g.buildings` are printed; I now use the `type()` function, so the unnecessary memory address is not shown.
- v.5: Added clarification on newline character being at end of file. This applies to multiple parts, so I put it at the beginning of the Problems section.
- v.6: Added note (in red) to part #3 about importing the `Entity` class. You can find the note before the examples.
- v.7: Clarified that buildings' doors cannot be entered in part #4.
- v.8: Added autograder details. Added to part #4.3 that you are guaranteed that there is at least one object.

## 2 Due Date

This assignment is due the night of Thursday, July 30. Gradescope will say 12:30 AM on Friday, July 31, due to the "grace period" (as described in the syllabus). *Do not rely on the grace period for extra time; this is risky.*

Some students tend to email me very close to the deadline. This is also a bad idea. There is no guarantee that I will check my email right before the deadline.

## 3 Reminder about Gradescope Active Submission

Once the deadline occurs, whatever submission is your **active submission** will be the one that dictates your final score. By default, your active submission will be your latest submission. However, you can change which submission is your active submission, which I talked about during the Thursday (6/25) lecture.

## 4 Manual Review

There will be no manual review similar to the previous two programming assignments in which you had to conform to the style guide. However, we will check that you made proper use of the classes given in part #4 and didn't just ignore all of the classes or combine them into one jumbo class.

## 5 Grading Breakdown

This assignment is worth 15% of your final grade and will be worth 150 points on Gradescope. Here is the breakdown of the 90 points by part:

- Part #1: 10
- Part #2: 15
- Part #3: 15
- Part #4: 110
  - Part #4.1: 25
  - Part #4.2: 10
  - Part #4.3: 35
  - Part #4.4: 40

## 6 Note about Cheating

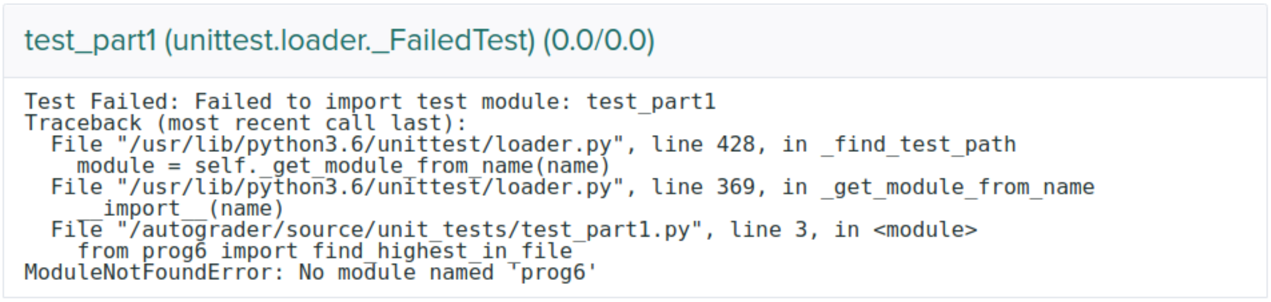
As stated in the Canvas announcement here, **any cheating on this assignment (as determined through the OSSJA) will result in an F in this entire course** (not just a zero on this assignment), even if you email me to confess the very second after you submit this assignment.

## 7 Autograder Details

In the end, the files that you should submit are `prog6.py` and `game.py`.

All of the details about the autograder that were given in the directions for the previous programming assignments are still relevant here, so I do not repeat them here.

As a reminder, if you see an error message like the one below, then that likely means that either you did not properly name the file or gave the function (for whichever test case it is) the wrong name. Oddly, for parts #1-3, if you get the below error message, you will *not* also be shown the failed test cases (although they will be marked as failures), but for part #4, if you get the below error message, you *will* be shown the failed test cases. (This isn't a big deal, but I wanted to point it out so as to avoid confusion.)



test_part1 (unittest.loader._FailedTest) (0.0/0.0)

```
Test Failed: Failed to import test module: test_part1
Traceback (most recent call last):
  File "/usr/lib/python3.6/unittest/loader.py", line 428, in _find_test_path
    module = self._get_module_from_name(name)
  File "/usr/lib/python3.6/unittest/loader.py", line 369, in _get_module_from_name
    __import__(name)
  File "/autograder/source/unit_tests/test_part1.py", line 3, in <module>
    from prog6 import find_highest_in_file
ModuleNotFoundError: No module named 'prog6'
```

If you have not completed part #4.4, then you may want to put a `return` statement at the beginning of the `run()` method in `class Game`, or else the test cases for part #4.4 will take forever (even though they will all fail), since the default `run()` (given to you in the starter code) will loop infinitely by default.

### 7.1 Test Cases' Inputs

See `hw6_visible_cases_inputs.pdf` on Canvas.

## 8 Problems

Update for v.5: Technically, all text files (i.e. files that contain human-readable contents) are supposed to end in a newline character. Unfortunately, not all editors automatically do this. However, the autograder (when it is applicable) will always use text files that end in a newline character. Note that this is not the same as saying that the file will end in a blank line. I am saying that you are guaranteed that every line in the file ends in a newline character, including the last line.

**You are not allowed to `import` any *built-in* modules.**

Of all the string, list, and dictionary methods, **only the below ones are permitted** (not all of them are necessary). Any method not listed below is prohibited. As mentioned in lecture, a method is a special kind of function that is called by specifying a value (usually a variable) followed by a dot/period and finally followed by the function call, e.g. `split()` is a method and can be called through something like `s.split()`, where `s` is a string.

- String methods: `format()`, `split()`.
- List methods: `append()`, `extend()`.
- Dictionary methods: `items()`, `values()`, `setdefault()`.

**None of what is said above applies to methods of files or user-defined classes.**

### 8.1 Part #1: Find Highest in File

File: `prog6.py`

Write a function called `find_highest_in_file()` that takes as its only argument the name of a file expected to contain one integer per line. Assuming the file's contents are valid (see below), the function should return the highest of all of the integers in the file.

The function should return `False` if at least one line in the file contains more than one integer or contains something that is not an integer.

*Hint*: For detecting invalid input, take advantage of the fact that `int()` throws an exception if it cannot convert its argument to an integer. This also works for checking if a line contains more than one integer; for example, the call `int("5 3")` throws an exception. You shouldn't have to do anything other than catching any exceptions that `int()` may throw.

#### 8.1.1 Examples

Suppose we have the following two files, called `values.txt` and `bad_values.txt`, respectively.

```
53
22
28
37
60
40
```

```
53
22
28 30
37
60
40
```

Here is the output of some calls of `find_highest_in_file()`:

```
>>> find_highest_in_file("values.txt")
60
>>> find_highest_in_file("bad_values.txt")
False
```

## 8.2 Part #2: Find Matches

File: `prog6.py`

Write a function called `find_matches` that takes four arguments. The first argument is the name of a text file. `find_matches()` should copy each line in this text file that matches a desired pattern to a file whose name is given by the second argument. (This second file's original contents should be erased.) The desired pattern is specified by the third and fourth parameters. The third parameter is a string, and the fourth parameter is a boolean value (i.e. `True` or `False`). A line matches the pattern if either of the following is true:

- The fourth parameter is `True`, and the line contains the string given in the third parameter.
- The fourth parameter is `False`, and the line *does not contain* the string given in the third parameter.

Put less rigidly, this function copies over any line from the input file to the output file that contains the third argument or does not contain the third argument, depending on if the fourth argument is `True` or `False`.

Any line that does not match the pattern should be ignored. The function must return the number of lines that were written to the output file. The lines that are written to the output file must be written in the order in which they are found in the input file.

*Hint*: We have primarily used the `in` operator to check if an integer is in a list, but don't forget that you can use the `in` operator to check if a string is a substring of another string.

### 8.2.1 Examples

Suppose we have the following file called `file1.txt`.

```
abcdef
abc
def
qwerty
wyxz
abcbcd
```

Next, we make some calls to `find_matches()`:

```
>>> find_matches("file1.txt", "output1.txt", "bc", True)
3
>>> find_matches("file1.txt", "output2.txt", "bc", False)
3
>>> find_matches("file1.txt", "output3.txt", "cbc", True)
1
>>> find_matches("file1.txt", "output4.txt", "cbc", False)
5
```

Below are the contents of the generated output files.

`output1.txt`:

```
abcdef
abc
abcbcd
```

`output2.txt`:

```
def
qwerty
wyxz
```

`output3.txt`:

```
abcbcd
```

output4.txt:

```
abcdef
abc
def
qwerty
wyxz
```

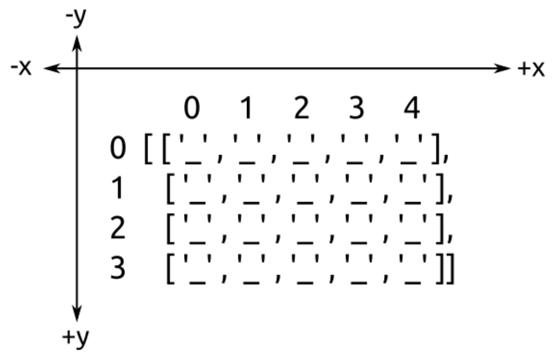
## 8.3 Part #3: Draw Entity

File: `prog6.py`

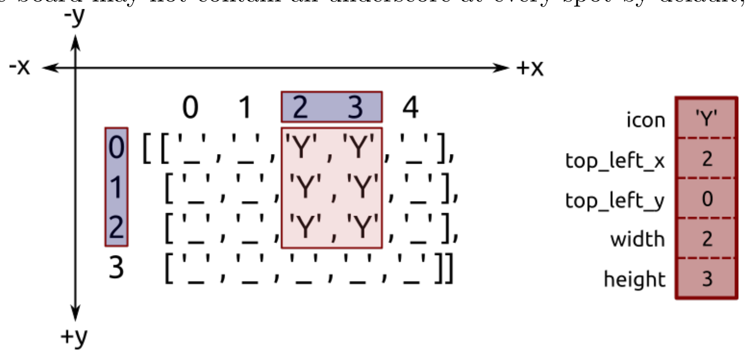
On Canvas, you will find a file called `entity.py` that contains the contents shown below. (Do not submit this file to the autograder; the autograder will ignore your version if you do. The autograder will use the intended version of `entity.py`; this effectively means you cannot modify `entity.py`.) The `icon` member is a single character, and the other four members – `top_left_x`, `top_left_y`, `width`, and `height` – are integers.

```python
class Entity:
    def __init__(self, icon, top_left_x, top_left_y, width, height):
        self.icon = icon
        self.top_left_x = top_left_x
        self.top_left_y = top_left_y
        self.width = width
        self.height = height
```

Still in `prog6.py`, implement a function called `draw_entity` that takes two arguments: an instance of `class Entity` and a 2D list representing a game board. The board represents a coordinate grid. See the below image. (Note that it is common in graphical applications or two-dimensional games to think of the coordinate grid with the positive values of the y-axis going downwards.)



`draw_entity()` should "draw" the given `Entity` instance's icon at the appropriate locations on the board, based on the fields of the `Entity` instance. In other words, the board should be modified to have the `Entity` instance's icon at the appropriate locations, based on the `Entity` instance's coordinates and dimensions. See the image below for a visual example. Note that the board may not contain an underscore at every spot by default, e.g. one could draw two entities onto the same board.



The function should return `True` if the drawing was succesful and `False` if the drawing failed due to an out-of-bounds indexing. (*Hint*: Use what was taught about exceptions to determine if `False` should be returned. You can and should modify the board (i.e. the 2D list, which is the second argument) directly; do not make a copy of the board, because the caller/autograder will not be able to check this copy[1].) If the function is supposed to return `False`, then the 2D list will not

[1]Technically speaking, I suppose you could make a shallow copy of the 2D list and modify that shallow copy (which would result in modifying original 2D list). It's creating a deep copy that you cannot do. In any event, there is no need to do this.

be checked; in other words, when the autograder expects the return value to be `False`, it will not check the board at all, and you do not need to undo any changes made to the board.

Below are some examples involving code. Except for the first board, I create the blank board using the workaround that I discussed towards the very end of the lecture slides on classes/references; I would recommend using the workaround. In case you wish to copy/paste any of the below without encountering the annoying errors about "unrecognized characters" that occur when one copy/pastes single quotation marks from a PDF generated from Latex, I have included all of the below output in a different PDF (generated from Markdown) on Canvas here. Update for v.6: You must `import` the `Entity` class wherever you use the `Entity` initializer, i.e. wherever you call `Entity(...)`. You can import it in your file, if you are trying the below lines in your file as opposed to on the interpreter. Make sure that `entity.py` is in the same folder as `prog6.py` when you try this on your own.

```
>>> from entity import Entity
>>> b = [['_', '_', '_', '_'],
...      ['_', '_', '_', '_'],
...      ['_', '_', '_', '_'],
...      ['_', '_', '_', '_'],
...      ['_', '_', '_', '_']]
>>> b
[['_', '_', '_', '_'], ['_', '_', '_', '_'], ['_', '_', '_', '_'], ['_', '_', '_', '_'], ['_', '_', '_', '
    _']]
>>> e = Entity('X', 1, 2, 2, 1)
>>> draw_entity(e, b)
True
>>> b
[['_', '_', '_', '_'], ['_', '_', '_', '_'], ['_', 'X', 'X', '_'], ['_', '_', '_', '_'], ['_', '_', '_', '
    _']]
>>> for row in b:
...     print(row)
...
['_', '_', '_', '_']
['_', '_', '_', '_']
['_', 'X', 'X', '_']
['_', '_', '_', '_']
['_', '_', '_', '_']
>>> e2 = Entity('Y', 0, 3, 3, 2)
>>> draw_entity(e2, b)
True
>>> for row in b:
...     print(row)
...
['_', '_', '_', '_']
['_', '_', '_', '_']
['_', 'X', 'X', '_']
['Y', 'Y', 'Y', '_']
['Y', 'Y', 'Y', '_']
>>> b = [['_'] * 6 for i in range(7)]
>>> e = Entity('Y', 2, 0, 2, 4)
>>> draw_entity(e, b)
True
>>> for row in b:
...     print(row)
...
['_', '_', 'Y', 'Y', '_', '_']
['_', '_', 'Y', 'Y', '_', '_']
['_', '_', 'Y', 'Y', '_', '_']
['_', '_', 'Y', 'Y', '_', '_']
['_', '_', '_', '_', '_', '_']
['_', '_', '_', '_', '_', '_']
['_', '_', '_', '_', '_', '_']
>>> e = Entity('Y', 2, 0, 2, 4)
>>> board = [['_'] * 3 for i in range(7)]
>>> draw_entity(e, board)
False
>>>
```

## 8.4 Part #4: Game

For this large part, you will start with the `game.py` file given on Canvas. Make sure to read the comments in this file. This file is what you will submit to the autograder for this part (with your many, many modifications, of course). In this game, you will play as a hero represented as a single character on a “board” (which is a 2D list). You can think of this board as being similar to the one in the previous part, except that the default characters are whitespaces instead of underscores.

The objective of this game is to collect all of the objects (each of which is represented by an instance of `class Object`). These objects are scattered across the board. The player, who (as stated above) is represented by a single character on a board, can move along the board by entering any of the movement commands (which are read from a file). However, the player cannot walk through the walls of buildings or walk out of the borders of the board.

I do not think it is possible to provide a sufficiently detailed, easily understood description of the game that you would only need to read once. The full nature of the game hopefully becomes clearer in the directions for part #4.4, where the game becomes playable. However, I will note that even without understanding the game too well, parts #4.1 and parts #4.2 – both of which only regard `class Board` – are doable.

Below are high-level descriptions of the four classes involved in this part.

- `class Game`: This class brings all details of the game together, from interaction with the user to collision detection to determining if the game has ended.
- `class Object`: This class contains information about an object that the user/hero can collect.
- `class Hero`: This class contains information about the hero that you play as.
- `class Building`: This class contains information about a building that serves as a structure on the board that the user cannot walk through.

For convenience, I provide the following function in `game.py`. It is sometimes used in the examples that follow.

```
def print_board(board):
    for row in board:
        for spot in row:
            print(spot, end='')
        print()
```

All coordinates are technically zero-based, and the y-axis goes downward as was the case for part #3 of this assignment. However, do note that the coordinates on the borders of the board (including the coordinate $(0, 0)$ at the top-left of the board) are invalid coordinates, since anything placed at those coordinates would collide with the borders; thus, you will never get a building whose top-left coordinate is $(0, 0)$ or $(1, 0)$ or $(0, 4)$, for example.

The directions for this part are split into sections. The first two sections focus on the `Building` class and ignore the `Game` class or the idea of even running the game. This exposes you to one of the benefits of splitting complex code into functions/classes in an effective manner: you can focus on implementing one piece at a time. For each part, you can and should implement as many helper functions as you would like; this is particularly recommended for part #4.4, or else the `run()` method may become exceptionally large.

### 8.4.1 Part #4.1: `Building` method: `draw_on_board()`

You are already provided some of the implementation of the `Building` class in `game.py`. Implement the `draw_on_board()` method. This method is very similar to the function you implemented in part #3 of this assignment in that the `draw_on_board()` method draws the building (represented by `self`) on the appropriate locations on the board (which is expected to be a 2D list of characters). Unlike in the function you implemented in part #3, a building is not a rectangle consisting of the same character. As you can see in the examples below, a buliding is represented like so:

```
------
|    |
| && |
--&&--
```

As you can see above, the top and bottom of the building are just dashes, and the sides of the building are vertical bars. The door is represented with ampersands. The building that your method draws must match the one above (while of course being in the correct position). Note that the `x` and `y` coordinates of the building refer to where it’s top-left point is located. The building’s dimensions are hard-coded into the class definition that you are given in `game.py`; these dimensions will never change.

Below are examples of how the method should behave.

```
>>> board = [[' '] * 12 for i in range(10)]
>>> print_board(board)  # prints 10 lines with 12 whitespaces each
                                                                   
                                                                   
                                                                   
                                                                   
                                                                   
                                                                   
                                                                   



>>> for row in board:
...     print(row)
...
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
>>> b1 = Building(2, 1)
>>> b1.draw_on_board(board)
>>> print_board(board)

  ------
  |    |
  | && |
  --&&--





>>> for row in board:
...     print(row)
...
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', '-', '-', '-', '-', '-', '-', ' ', ' ', ' ', ' ']
[' ', ' ', '|', ' ', ' ', ' ', ' ', '|', ' ', ' ', ' ', ' ']
[' ', ' ', '|', ' ', '&', '&', ' ', '|', ' ', ' ', ' ', ' ']
[' ', ' ', '-', '-', '&', '&', '-', '-', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
>>> b2 = Building(6, 5)
>>> b2.draw_on_board(board)
>>> print_board(board)

  ------
  |    |
  | && |
  --&&--
      ------
      |    |
      | && |
      --&&--

>>> for row in board:
...     print(row)
...
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', '-', '-', '-', '-', '-', '-', ' ', ' ', ' ', ' ']
[' ', ' ', '|', ' ', ' ', ' ', ' ', '|', ' ', ' ', ' ', ' ']
[' ', ' ', '|', ' ', '&', '&', ' ', '|', ' ', ' ', ' ', ' ']
[' ', ' ', '-', '-', '&', '&', '-', '-', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', '-', '-', '-', '-', '-', '-']
[' ', ' ', ' ', ' ', ' ', ' ', '|', ' ', ' ', ' ', ' ', '|']
[' ', ' ', ' ', ' ', ' ', ' ', '|', ' ', '&', '&', ' ', '|']
[' ', ' ', ' ', ' ', ' ', ' ', '-', '-', '&', '&', '-', '-']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
>>>
```

### 8.4.2 Part #4.2: `Building` method: `contains()`

Implement the `contains()` method of the `Building` class. This method takes as explicit arguments a pair of `x` and `y` coordinates, and it should return `True` if the given coordinates are in the space taken up by the building. Below are examples of how your program should behave. In the case of the building `b1`, any coordinates where `x` is between 2 and 7 (inclusive) and `y` is between 1 and 4 (inclusive) should result in `True` when passed to `b1.contains(...)`.

```
>>> b1 = Building(2, 1)
>>> b1.contains(2, 1)
True
>>> b1.contains(2, 2)
True
>>> b1.contains(2, 4)
True
>>> b1.contains(2, 5)
False
>>> b1.contains(7, 4)
True
>>> b1.contains(8, 4)
False
>>> b1.contains(15, 6)
False
```

### 8.4.3 Part #4.3: `Game` method: `read_input_file()`

This method loads the following information (in order) from the file whose name was explicitly passed as an argument:

- Dimensions of the board. Upon reading these, the `read_input_file()` method should create a 2D list (in which each slot is one whitespace character) to represent the board. Below is the line of code that I use for this; it is similar to what I talked about towards the end of the lecture on classes/references. *You may assume these dimensions will always be positive, even integers.*
  - `self.board = [[' '] * self.width for i in range(self.height)]`
  - You should also add the borders to the board. As you can see in one of the examples below,
- Hero's information: symbol and starting `x` and `y` coordinates.
- The key on the keyboard for moving the character up.
- The key for moving left.
- The key for moving down.
- The key for moving right.
- Any remaining lines are information on objects and buildings. (These can be in a mixed order, as they are in the below example.) Each line containing an object's information should result in the creation of an instance of `Object` (one of the provided classes in `game.py`) that is added to the `objects` member of the `Game` class. Each line containing a building's information should result in the creation of an instance of `Building` that is added to the `buildings` member of the `Game` class. There will always be at least one object.

Below is an example input file.

```
20 16
X 3 8
w
a
s
d
o $ 8 14
b 5 2
o $ 17 10
```

Below are a list of members of the `Game` class that could be checked by the autograder for this part (and thus that must be correct). I will set up the autograder so that getting just one of these members incorrect will not automatically cause you to fail all of the test cases.

- `hero`. This should be an instance of `class Hero`.
- `num_objects`. This should equal the number of objects created, i.e. the size of `objects`.
- `objects`: list containing all created instances of `class Object`.
- `buildings`: list containing all created instances of `class Building`.
- `board`. This must be a 2D list of the appropriate dimensions. The board must already have the hero, buildings, objects, and borders drawn onto it.

- `down_key`, `up_key`, `right_key`, `left_key`. Each of these should contain the corresponding character on the keyboard that must be entered in order to perform that movement (e.g. to move left).
  - *Hint*: Be mindful about the fact that when you read a line from a file, it contains a newline character at the end, even if the line looks like it only contains one character to us.

There are many reasons that the input file could be an invalid setup for the game, e.g. two objects could be on the same spot, a building stretches half past the edge of the board, etc. **Do not worry about such instances of invalid input. Assume that the buildings, objects, and borders do not collide with each other. Assume that no building, object, or the hero are given invalid (e.g. negative) coordinates.**

Below are examples of how your method should behave. As you can see in the provided code for the initializer of the `Game` class, the `read_input_file()` method should not be called explicitly, since the initializer already calls it.

`input1.txt`:

```
20 16
X 3 8
w
a
s
d
o $ 8 14
b 5 2
o $ 17 10
```

Using `input1.txt`:

```
>>> g = Game("input1.txt")
>>> type(g)
<class 'game.Game'>
>>> type(g.hero)
<class 'game.Hero'>
>>> print(g.hero.symbol)
X
>>> print(g.hero.x)
3
>>> print(g.hero.y)
8
>>> print(g.num_objects)
2
>>> for row in g.board:
...     print(row)
...
[' ', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', ' ']
['|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', ' ', '-', '-', '-', '-', '-', '-', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', ' ', '|', ' ', ' ', ' ', ' ', '|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', ' ', '|', ' ', '&', '&', ' ', '|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', ' ', '-', '-', '&', '&', '-', '-', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', 'X', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '$', ' ', '|']
['|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '$', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
[' ', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', ' ']
>>> print_board(g.board)
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|  X               |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
>>> print(len(g.objects))
2
>>> # Print the type of each object:
...
>>> print([type(obj) for obj in g.objects])
[<class 'game.Object'>, <class 'game.Object'>]
>>> print(len(g.buildings))
1
>>> type(g.buildings[0])
<class 'game.Building'>
>>> print(g.down_key)
s
>>> print(g.right_key)
d
```

`input2.txt`:

```
18 14
Y 1 2
u
l
d
r
o E 1 1
o E 3 1
b 4 3
b 10 7
```

Using `input2.txt`:

```
>>> g = Game("input2.txt")
>>> print_board(g.board)
 ----------------
|E E             |
|Y               |
|   ------       |
|   |    |       |
|   | && |       |
|   --&&--       |
|         ------ |
|         |    | |
|         | && | |
|         --&&-- |
|                |
|                |
 ----------------
>>> for row in g.board:
...     print(row)
...
[' ', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', ' ']
['|', 'E', ' ', 'E', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', 'Y', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', '-', '-', '-', '-', '-', '-', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', '|', ' ', ' ', ' ', ' ', '|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', '|', ' ', '&', '&', ' ', '|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', '-', '-', '&', '&', '-', '-', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '-', '-', '-', '-', '-', '-', ' ', '|']
['|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|', ' ', ' ', ' ', ' ', '|', ' ', '|']
['|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|', ' ', '&', '&', ' ', '|', ' ', '|']
['|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '-', '-', '&', '&', '-', '-', ' ', '|']
['|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
['|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|']
[' ', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', ' ']
>>> print(g.num_objects)
2
>>> print(g.down_key)
d
```

### 8.4.4 Part #4.4: `Game` method: `run()`

In this part, you will put together the pieces of this game by incorporating movement. Below is a list of steps.

- **User input**: If the user enters a movement command (i.e. any of down, up, left, or right by entering the corresponding keyboard key), then the function should try to *determine* where this would move the hero. The hero only moves one spot at a time, and there is no diagonal movement. For example, entering the key for moving right would suggest an attempt at moving the hero one spot right along the row that they currently inhabit. Note that the ideal setup of the game should be a constant interaction with the user in which the program shows the user the board, prompts them for input, reacts to their input, then shows them the board again, and so on.
- **Collision detection part #1: Illegal movements**: The reason that I said "determine" above is because it is entirely possible that the hero cannot move where the user wants them to. This can happen if the hero would collide with a building or a border of the board. Thus, you need to implement such logic, i.e. if the hero would move to a spot that puts them in collision with a building or a border, then they should not be allowed to move there, and they should remain in the location they were at. Note that there is no difference between the player colliding with a building's wall vs. a building's door. Buildings cannot be entered in this game. Entering a door should be prohibited just as entering a wall is.
- **Actually moving the hero**: If no collision would occur, change where the hero is at on the board.
- **Collision detection part #2: Collecting objects**: If the hero moves to a spot occupied by an object, then the user should "collect" this object. This will require you to update the appropriate members of the `Game` instance. Do remember that collecting objects is how the user wins the game; thus, collecting an object should bring the game one step closer to a state in which the `game_ended()` method that I provided returns `True`.

Below are examples of how your game should behave.

`input1.txt`:

```
20 16
X 3 8
w
a
s
d
o $ 8 14
b 5 2
o $ 17 10
```

```
>>> g = Game("input1.txt")
>>> g.run()
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|  X               |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|   X              |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
```

```
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|    X             |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|     X            |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|      X           |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|       X          |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|        X         |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|         X        |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|          X       |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|           X      |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|            X     |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|             X    |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|              X   |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|               X  |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|                X |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: s
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|                  |
|                X |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: s
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|                  |
|                  |
|                X |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: a
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|                  |
|                  |
|               X  |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: a
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|                  |
|                  |
|              X   |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: a
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|                 |
|                 |
|             X   |
|                 |
|                 |
|                 |
|       $         |
 -----------------
Enter: a
 -----------------
|                 |
|    ------       |
|    |    |       |
|    | && |       |
|    --&&--       |
|                 |
|                 |
|                 |
|                 |
|            X    |
|                 |
|                 |
|                 |
|       $         |
 -----------------
Enter: a
 -----------------
|                 |
|    ------       |
|    |    |       |
|    | && |       |
|    --&&--       |
|                 |
|                 |
|                 |
|                 |
|           X     |
|                 |
|                 |
|                 |
|       $         |
 -----------------
Enter: a
 -----------------
|                 |
|    ------       |
|    |    |       |
|    | && |       |
|    --&&--       |
|                 |
|                 |
|                 |
|                 |
|          X      |
|                 |
|                 |
|                 |
|       $         |
 -----------------
Enter: a
 -----------------
|                 |
|    ------       |
|    |    |       |
|    | && |       |
|    --&&--       |
|                 |
|                 |
|                 |
|                 |
|         X       |
|                 |
|                 |
|                 |
|       $         |
 -----------------
Enter: a
 -----------------
|                 |
|    ------       |
|    |    |       |
|    | && |       |
|    --&&--       |
|                 |
|                 |
|                 |
|                 |
|        X        |
|                 |
|                 |
|                 |
|       $         |
 -----------------
Enter: a
 -----------------
|                 |
|    ------       |
|    |    |       |
|    | && |       |
|    --&&--       |
|                 |
|                 |
|                 |
|                 |
|       X         |
|                 |
|                 |
|                 |
|       $         |
 -----------------
Enter: s
 -----------------
|                 |
|    ------       |
|    |    |       |
|    | && |       |
|    --&&--       |
|                 |
|                 |
|                 |
|                 |
|                 |
|       X         |
|                 |
|                 |
|       $         |
 -----------------
Enter: s
 -----------------
|                 |
|    ------       |
|    |    |       |
|    | && |       |
|    --&&--       |
|                 |
|                 |
|                 |
|                 |
|                 |
|                 |
|       X         |
|                 |
|       $         |
 -----------------
Enter: s
 -----------------
|                 |
|    ------       |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|                  |
|                  |
|                  |
|                  |
|                  |
|       X          |
|       $          |
 ------------------
Enter: s
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|                  |
|                  |
|                  |
|                  |
|                  |
|                  |
|       X          |
 ------------------
Congratulations: you’ve collected all of the items!
```

```
>>> g = Game("input1.txt")
>>> g.run()
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|  X               |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: a
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
| X                |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: a
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|X                 |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: a
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|X                 |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
| X                |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|  X               |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|   X              |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: d
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|                  |
|    X             |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: w
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|    X             |
|                  |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: w
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|    X             |
|                  |
|                  |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter:
Invalid command
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|    X             |
|                  |
|                  |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: w
```

```
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|    X             |
|                  |
|                  |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: s
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|    X             |
|                  |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
Enter: q
 ------------------
|                  |
|    ------        |
|    |    |        |
|    | && |        |
|    --&&--        |
|                  |
|    X             |
|                  |
|                  |
|                $ |
|                  |
|                  |
|                  |
|       $          |
 ------------------
You are a quitter!
```

*Hint*: This is more of a suggestion than a hint, because many of you may not wish to try this (which is partially why I hid it after all of the long examples). Obviously, one part of finishing the `run()` method involves checking which direction the user wishes to move, and this easily leads one to add one conditional statement per each movement. I did not want to have four conditional statements that contain similar code, so what I instead did was create the following dictionary and used it to avoid needing four conditional statements. (I won't provide more details than that here.)

```
movement_deltas = {self.down_key: (0,1),
                   self.up_key: (0,-1),
                   self.right_key: (1,0),
                   self.left_key: (-1,0)}
```