

Sudo Mario Bros

Assignment 1
CSSE1001/7030
Semester 2, 2019

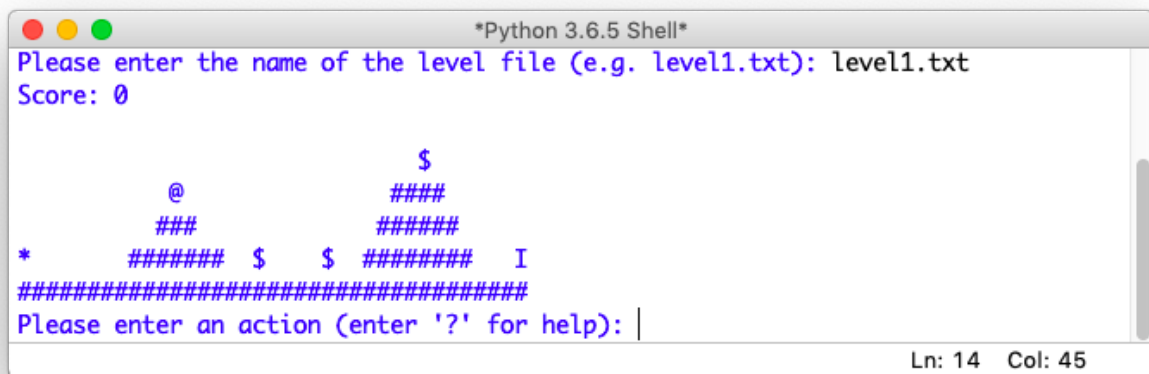
Version 1.0.0
10 marks

Due 23 Aug 19 20:30

Introduction

The goal of this assignment is to produce a simplistic 2D platformer game with Python.

This assignment will give you an opportunity to utilise the concepts that you have already learnt this semester. You will be asked to write several functions, outlined in this document, to successfully implement the expected game.



```
*Python 3.6.5 Shell*
Please enter the name of the level file (e.g. level1.txt): level1.txt
Score: 0

          $
        ####
      #####
* ##### $ $ ##### I
#####
Please enter an action (enter '?' for help): |
Ln: 14 Col: 45
```

Basic Running of the Game

Getting Started

The archive `a1_files.zip` contains all the necessary files to start this assignment. Some support code has been provided to assist with loading levels of the game. Additionally some map files have been provided (level1.txt, level2.txt, etc).

The main assignment file is `a1.py`, which contains the base you will use to implement your assignment.

The other provided file is `a1_support.py`, which contains some code to help you implement your assignment. You are not required to use this file to implement your assignment but it is strongly recommended.

Concepts

There are a couple of important concepts to help you understand how to complete the assignment.

levels: Levels are stored as text files which represent the playing field or map of the game. Levels are loaded from the text files using the `load_level` function in the support code. This will return a string representation of the level which is used by your program to play the game.

positions: Positions are represented as tuples which store two integers. The first integer in a position tuple is the **x** coordinate, starting at zero from the left side of the game window. The second integer in a position tuple is the **y** coordinate, starting at zero from the bottom side of the game window.

directions: Directions are represented by characters, and can only have one of the values "r", "l", "u" or "d" (representing right, left, up, and down respectively). For any function which deals with directions, you are not required to handle any inputs which do not have one of these values.

tiles: Tiles are characters from a level file. There are 7 different possible tiles.

Character	Description
	An air tile, represented by a space, a player can move onto these tiles without consequence.
@	A monster tile, a player needs to attack this before moving to this tile, otherwise the player will die.
\$	A coin tile, when a player moves to this tile their score will increase by one.
*	A player tile, this is how the player is represented in the game.
#	A wall tile, a player cannot pass through this tile, if they move onto a wall tile they will rise until there is an air tile available.
^	A checkpoint tile, only relevant for the bonus task, see the checkpoint section.
I	A goal tile, when a player moves onto this tile they win the game.

Implementation

To successfully complete this assignment you will need to implement the following functions exactly as they are described.

`get_position_in_direction(position, direction)`

Return the position that would result from moving from given position in the given direction.

If the direction "r" is given, increase the x coordinate by 1.

If the direction "l" is given, decrease the x coordinate by 1.

If the direction "u" is given, increase the y coordinate by 1.

If the direction "d" is given, decrease the y coordinate by 1.

`get_tile_at_position(level, position)`

Return the character representing the tile at the given position in a level string. Hint: refer to `position_to_index` in the support file.

`get_tile_in_direction(level, position, direction)`

Determine the new position which results from moving the given position in the given direction, and return the character representing the tile found at this new position.

`remove_from_level(level, position)`

Return a level string exactly the same as the one given, but with the given position replaced by an air tile.

`move(level, position, direction)`

Return the updated position that results from moving the character from the given position in the given direction.

If the tile at the updated position is a wall tile, adjust the position up until an air tile is found and return that as the position instead.

If the tile immediately below the next position is an air tile, adjust the position down until the tile below is not an air tile and

return that as the position instead.

It is important that you check whether the next position is a wall tile before checking if the tile below is an air tile.

`print_level(level, position)`

Print the level (i.e. string) with the tile of the given position replaced by the player tile.

`attack(level, position)`

Check if the position to the left of the player is a monster, if it is then print `Attacking the monster on your left!` and return the level with the monster tile removed.

Check if the position to the right of the player is a monster, if it is then print `Attacking the monster on your right!` and return the level with the monster tile removed.

If neither the left side nor the right side of the player contains a monster, then print `No monsters to attack!` and return the level unchanged.

`tile_status(level, position)`

If the tile at the position is the goal tile then print `Congratulations! You finished the level`

If the tile at the position is a monster tile then print `Hit a monster!`

If the tile at the position is either a coin tile or checkpoint tile, remove the tile from the level.

Finally, return a tuple containing the tile character and the level.

`main`

Handles the main interaction with the user.

When the game starts, it should ask the user for a file to load a level from as follows: `Please enter the name of the level file (e.g. level1.txt):`

Once the user enters the name of the level file, it should load that level from the file and print the current score as `Score: 0` followed by printing the loaded level with the player at the starting position of x=0, y=1.

The program should then repeatedly ask the user `Please enter an action (enter '?' for help):`

After each action is input, it should perform the action and print out the current score and the level. Actions are described in the table below.

Character	Description
?	Print out the string <code>HELP_TEXT</code> defined in <code>a1_support.py</code>
r	Move the player to the right, if the player hits a monster or goal, stop the game. If the player hits a coin, increase their score.
l	Move the player to the left, if the player hits a monster or goal, stop the game. If the player hits a coin, increase their score.
a	Attack a monster immediately left or right of the player.
n	Reset to the last checkpoint reached - Bonus Task Only
q	Stop the game (don't print the level or prompt the user again).

Bonus – Checkpoints

This task involves adding checkpoints to the game. A checkpoint is a tile in the game which triggers the saving of the current state (level map, player position and player score) when the player moves to that tile.

If the player is killed by an enemy then the level should be reset to the last state saved by a checkpoint.

Checkpoints should be implemented as apart of your `main` function.

The marks associated with this task are additional marks that will increase the total to above 10 marks. You will still be limited to a total of 10 marks for this assignment. The intention of this task is to allow you to demonstrate a good understanding of the

content that will allow for some marks to be lost in other areas while still receiving full marks.

Examples

Please enter the name of the level file (e.g. level1.txt): level1.txt

Score: 0

```

          $
        @   #####
      ###   #####
*   ##### $   $   ##### I
#####

```

Please enter an action (enter '?' for help): r

Score: 0

```

          $
        @   #####
      ###   #####
*   ##### $   $   ##### I
#####

```

Please enter an action (enter '?' for help): r

Score: 0

```

          $
        @   #####
      ###   #####
*   ##### $   $   ##### I
#####

```

Please enter an action (enter '?' for help): r

Score: 0

```

          $
        @   #####
      ###   #####
*   ##### $   $   ##### I
#####

```

Please enter an action (enter '?' for help): r

Score: 0

```

          $
        @   #####
      ###   #####
*   ##### $   $   ##### I
#####

```

Please enter an action (enter '?' for help): r

Score: 0

```

          $
        @   #####
      ###   #####
*   ##### $   $   ##### I
#####

```

Please enter an action (enter '?' for help): r

Score: 0

```

          $
        @   #####
      ###   #####
*   ##### $   $   ##### I
#####

```

Please enter an action (enter '?' for help): r

Score: 0

```

          $
        @      #####
      ###      #####
*##### $    $ ##### I
#####
Please enter an action (enter '?' for help): r
Score: 0

```

```

          $
        @      #####
      * ###      #####
    ##### $    $ ##### I
#####
Please enter an action (enter '?' for help): r
Score: 0

```

```

          $
        @      #####
      *###      #####
    ##### $    $ ##### I
#####
Please enter an action (enter '?' for help): r
Score: 0

```

```

          $
        *@      #####
      ###      #####
    ##### $    $ ##### I
#####
Please enter an action (enter '?' for help): a
Attacking the monster on your right!
Score: 0

```

```

          $
        *      #####
      ###      #####
    ##### $    $ ##### I
#####
Please enter an action (enter '?' for help): r
Score: 0

```

```

          $
        *      #####
      ###      #####
    ##### $    $ ##### I
#####
Please enter an action (enter '?' for help): r
Score: 0

```

```

          $
        *      #####
      ###      #####
    ##### $    $ ##### I
#####
Please enter an action (enter '?' for help): r
Score: 0

```

```

          $
        *      #####
      ***      #####
    ##### $    $ ##### I
#####
Please enter an action (enter '?' for help): r
Score: 0

```

```

          $
        ####
      ### *      #####
    ##### $    $ ##### I
#####
Please enter an action (enter '?' for help): r
Score: 0

```

```

          $
        ####
      ###      #####
    #####* $    $ ##### I
#####
Please enter an action (enter '?' for help): r
Score: 0

```

```

          $
        ####
      ###      #####
    ##### *$    $ ##### I
#####
Please enter an action (enter '?' for help): r
Score: 1

```

```

          $
        ####
      ###      #####
    ##### *    $ ##### I
#####
Please enter an action (enter '?' for help): l
Score: 1

```

```

          $
        ####
      ###      #####
    ##### *      $ ##### I
#####
Please enter an action (enter '?' for help): l
Score: 1

```

```

          $
        ####
      ###      #####
    #####*      $ ##### I
#####
Please enter an action (enter '?' for help):

```

Marking

The marks in this section add to a total of 20, which will be scaled down to the 10 marks available for this assignment. Marks will be awarded based on the formula below:

Total Mark = $\min((\text{functionality} + \text{bonus} + \text{style})/2, 10)$

Functionality Assessment

Your assignment will be marked automatically using the test suite provided. Marks will be awarded based on how many tests pass.

Syntax errors which prevent your assignment from running will be removed from your assignment where reasonable, deducting one mark per syntax error for up to 3 syntax errors. What is considered a reasonable fix to your assignment is left to the judgement of the tutor who marks your assignment and their judgement is final.

Function	Marks
<code>get_position_in_direction</code>	1
<code>get_tile_at_position</code>	1
<code>get_tile_in_direction</code>	1
<code>remove_from_level</code>	1
<code>move</code>	3
<code>print_level</code>	1
<code>attack</code>	2
<code>tile_status</code>	2
<code>main</code>	3
Total	15
Bonus Task	1

Code Style

The style of your assignment will be assessed by one of the tutors, and you will be marked on the broad categories listed below.

	Description	Marks
Readability	Code is readable. Appropriate and meaningful identifier names have been used. Simple and clear code structure. Repeated code has been avoided.	2
Simplicity	Code has been simplified where appropriate and is not overly convoluted.	1
Documentation	Documented clearly and concisely, without excessive or extraneous comments.	2
Total		5

Feedback Sessions

In week 6, you must attend the practical that you are signed onto on mySI-net where the tutor who assessed your assignment will provide you with feedback.

When you arrive at the feedback session please take a seat but don't sit in the row furthest from the door as this will be where tutors will be providing feedback and it will be need to be kept free for privacy.

Allow for being in attendance at the practical for the full 2 hours, when your name is called walk over to the tutor who called your name and they will provide you with feedback on your assignment. You can leave once you have discussed your assignment with a tutor.

Assignment Submission

Your assignment must be submitted via the assignment one submission link on Blackboard. You must submit a Python file, `a1.py`, containing your implementation of the assignment.

Late submission of the assignment will **not** be accepted. Do not wait until the last minute to submit your assignment, as the time to upload it may make it late. Multiple submissions are allowed, so ensure that you have submitted an almost complete version of the assignment *well* before the submission deadline. Your latest on-time, submission will be marked. Ensure that you submit the correct version of your assignment. An incorrect version that does not work **will** be marked as your final submission.

In the event of exceptional circumstances, you may submit a request for an extension. See the [course profile](#) for details of how to apply for an extension. Requests for extensions must be made **no later** than 48 hours prior to the submission deadline. The expectation is that with less than 48 hours before an assignment is due it should be substantially completed and submittable. Applications for extension, and any supporting documentation (e.g. medical certificate), must be submitted via [my.UQ](#). You must retain the original documentation for a *minimum period* of six months to provide as verification should you be requested to do so.

Change Log

Any changes to this document will be listed here.