Dungeon Version 2: Looking around

Let’s add more features! We'd like to see the map, so we can see where we are. With the "look around" function, we will see where we can and cannot move!

In this version, you will implement:

- Showing the map
- Looking around

What to do

In addition to the functions you already have:

1. Implement the `display_map` function that would display the map, including the current position of the player. The player’s position would be depicted using the **@** character.

This function should **not** change the grid.

At the start of the game, the **S** in the map would be replaced by **@**. For example, if we are using `cave_map.txt`, a call to this function would display the following:

```
**@--
*----
***--
--**F
```

If the player eventually moves to the third row and first column, a call to this function would display the following:

```
**S--
*----
@**--
--**F
```

If the player moves to the final location, a call to this function will display the following:

```
**S--
*----
```

```
***--
--**@
```

2. Write two small helper functions:

a) `get_grid_size()` would return a list of two integers: number of rows and number of columns in the map. For the example above, it would return `[4, 5]`.
b) `is_inside_grid()` will use the helper function `get_grid_size()` to check whether a given position is valid or not. For the example above:
`is_inside_grid(grid, [1, 1]) == True`
`is_inside_grid(grid, [6, 1]) == False`

3. Implement the `look_around` function that would return a list of allowed directions. This function will use the helper function `is_inside_grid` to return a list of allowed directions. For the example above, the player can move either to the **north** ("up") or to the **east** ("right"). So, the function would return `['north', 'east']` (in any order). Please note that the players **cannot** move diagonally in the grid.

Remember: the path is depicted by * **(stars)**. Start and Finish points are also valid destinations.

4. Update the `main` function:

a) At every step of the game, it should display valid directions.
b) In addition to the **escape** command, the **show map** command would display the map.

Use the following template. All functions defined in the template **must be present and implemented** in your code (you may **not** omit functions or change the function definitions). You **may** add extra functions if needed.

```
MAP_FILE = 'cave_map.txt'

def load_map(map_file: str) -> list[list[str]]:
    """
    Loads a map from a file as a grid (list of lists)
    """
    # Implemented in version 1

def find_start(grid: list[list[str]]) -> list[int, int]:
    """
    Finds the starting position of the player on the map.
    """
```

```
    # Implemented in version 1

def get_command() -> str:
    """
    Gets a command from the user.
    """
    # Implemented in version 1

def display_map(grid: list[list[str]], player_position: list[int, int]) -> None:
    """
    Displays the map.
    """
    # TODO: implement this function

def get_grid_size(grid: list[list[str]]) -> list[int, int]:
    """
    Returns the size of the grid.
    """
    # TODO: implement this function

def is_inside_grid(grid: list[list[str]], position: list[int, int]) -> bool:
    """
    Checks if a given position is valid (inside the grid).
    """
    grid_rows, grid_cols = get_grid_size(grid)
    # TODO: implement the rest of the function

def look_around(grid: list[list[str]], player_position: list[int, int]) -> list:
    """
    Returns the allowed directions.
    """
    allowed_objects = ('S', 'F', '*')
    row = player_position[0]
    col = player_position[1]
    directions = []
    if is_inside_grid(grid, [row - 1, col]) and grid[row - 1][col] in
allowed_objects:
        directions.append('north')
    # TODO: implement the rest of the function

def main():
    """
    Main entry point for the game.
    """
    # TODO: update the main() function
```

```
if __name__ == '__main__':
    main()
```

Hints

- Your `look_around()` function will likely have four separate if statements.

Program name

Save your program as `dungeon2.py`.

Demo

In this demo, `cave_map.txt` is used.

https://asciinema.org/a/a9g0UvnslKeo9c4I22x6Yw5QE

Testing

To make sure your program works correctly, you should test it.

Good news: we wrote the unit tests for you: **test_dungeon2.py**

To test your functions, simply run the unit tests:

```
$ python -m pytest test_dungeon2.py
```

All tests should pass.

Submitting

Submit `dungeon2.py` via eClass.