University of Zurich[UZH]

s.e.a.l. software evolution & architecture lab

Informatics I – HS18

Exercise 5

Submission Details

- **Submission Format:** ZIP file containing 2 .py files
- **Submission Deadline:** 12:00, Tuesday 23rd October
- The name of the ZIP file must have the following format:
  firstname_lastname_studentidentificationnumber_info1_exercise_5.zip,
  e.g. *hans_muster_12345678_info1_exercise_5.zip*.
- Your ZIP file should contain 2 files, `task_1.py` and `task_2.py` **Do not rename these files.**
- **Important:** Please follow the naming conventions very strictly, otherwise we will not be able to grade your submission.
- Please submit the assignment using OLAT. You may upload multiple solutions, but note that **only the last submission** will be evaluated.
- Your submissions will be tested on systems running **Python 3.7.X**. Make sure that your solutions work on this version.
- **Important:** Since you will mostly write functions now, the setup of the code files is a bit different. Please read closely in the comments where you need to write your code. The code you write in `if __name__ == '__main__':` is solely for testing and will not be evaluated at all.

1 Task: Battleship *Solitaire* (5 Points)

A rival company tried to destroy your masterpiece *Battleship* game! Fortunately, they did not manage to delete all of your code. You still have your trusty test suite, and the signature and documentation to each function, even of those where the implementation was deleted!

```
1  def initialize_board():
2      """
3      Initializes an empty board.
4      :return: a BOARD_SIZE * BOARD_SIZE list of lists (matrix). Each cell
       initialized with WATER.
5      """
6      pass
```

Listing 1: Example of a function where the implementation was deleted.

```
1  def print_board(board, show_ships=False):
2      """
3      Prints the game board to the command line.
4      :param board: game board to print
5      :param show_ships: whether to show where the ships are placed or not.
6      """
7      print('____________')
8      for row in board:
9          for value in row:
10             value = WATER if not show_ships and value == SHIP else value
11             print('|' + value, end="")
12
13         print('|\n-----------')
14     print('')
```

Listing 2: Example of a function which survived the purge.

Game requirements

This is a simplified one player version of the game *Battleship*[1]. In this version, the player is the only one firing at the enemy ships, placed at random at the beginning of the game. Each ship has size 1 and turn after turn the player take guesses at where they are. As soon as a ship is hit, it is sunk and it should be represented accordingly on the board. The game is over when the player managed to hit all of the enemies ship. The player cannot lose in this version of the game.

The expected outcome of the game once working can be seen in the following listing.

```
1  Good luck!
2  ____________
3  | | | | | |
4  -----------
5  | | | | | |
```

[1] https://en.wikipedia.org/wiki/Battleship_(game)

```
-----------
| | | | | |
-----------
| | | | | |
-----------
| | | | | |
-----------

Coordinates? 0,1
____________
| |-| | | |
-----------
| | | | | |
-----------
| | | | | |
-----------
| | | | | |
-----------
| | | | | |
-----------

Missed...
Turns taken 1
Hit ratio 0.0
Ships still floating 5
Coordinates? 2,1
____________
| |-| | | |
-----------
| | | | | |
-----------
| |X| | | |
-----------
| | | | | |
-----------
| | | | | |
-----------

Direct hit!
Turns taken 2
Hit ratio 0.5
Ships still floating 4
```

Listing 3: Excerpt of expected output.

Getting started

Start by reading through the code and the documentation. Each function without implementation should have enough details in the docstring to help you write the missing code. To check your

code you can (and should!) try running it but keep in mind that you also have the tests to check if everything is working as expected.

You can execute single tests from the suite, through PyCharm [2] or by running, for example, `python3 task_1_test.py Task1Test.test_initialize_board -v -b` from the command line.

The game loop

The game loop is responsible for handling the flow of the game: from getting input, to executing actions, to checking game over conditions. This function thankfully survived. Start implementing the missing functions following the logic flow from here.

```
def play():
    """
    Game logic.

    Initialize an empty board.
    Place enemy ships.
    Print initial board.

    While not game over
        1. Get user input
        2. fire
        3. print board, statistics
    """

    print("Good luck!")
    game_board = initialize_board()
    place_enemy_ships(game_board)
    print_board(game_board)

    while not is_game_over(game_board):
        coordinates = get_coordinates_from_user()
        has_hit = fire(game_board, coordinates)

        print_board(game_board)
        print("{}".format('Direct hit!' if has_hit else 'Missed...'))

        for stat in compute_statistics(game_board):
            print(stat[0], stat[1])

    print("Game over!")
```

Listing 4: Game loop.

Note: The code as is, crashes almost immediately. It is your job to fix it!

Task: Fill in the missing function implementations in order to make the battleship game work again!

[2] https://www.jetbrains.com/help/pycharm/testing-your-first-python-application.html#run-test

2 Task: Basic Data Science (5 Points)

In this task, you will perform some basic data science on a small dataset containing information about dogs in the city of Zurich. You are given a csv file with the raw data and have to implement some functions that help you answer statistical questions about the data.

Getting started

Start by familiarizing yourself with the dataset. Look at the csv and then read through the code and the documentation. The function that reads the csv is already implemented for you. The functions without implementation should have enough details in the docstring to help you write the missing code. To check if your code is working you can try running the code but keep in mind that you also have the tests to check if your code is working as expected. You can also create a toy dataset with the same format, but only a few lines, so that you can manually check whether your functions work correctly.

You can execute single tests from the suite, through PyCharm [3] or by running, for example, `python3 task_2_test.py Task2Test.test_preview_dognames -v -b` from the command line.

Note: We provided you with only a sample from the original dataset, and will use the whole set or another sample of it to test your implementation, as well as a more complete test suite.

Task: Fill in the missing function implementations of the following functions:

- `preview_dognames`
- `dognames_count`
- `top_n_dognames`
- `is_valid_row`
- `filter_dognames`

[3] https://www.jetbrains.com/help/pycharm/testing-your-first-python-application.html#run-test