NOTICE:

NOTICE: the zip had images missing the last snake tail pixel. The files have been updated. RE-DOWNLOAD IT. The VM will be updated shortly.

You can assume that:

the snake initial posizion is always empty (never obstacle nor food).

the command input contains at least one command

the snake can move diagonally but cannot cross itself diagonally.

See the example in the HW text (but other cases could happen).

At start the snake is ١ pixel long (else we would give you initial lenght and pixels' positions), and is placed at the posizion passed as parameter.

The initial picture contains only background, food or obstacles pixels (never snake nor trail).

When the snake dies it NEVER replace the obstacle pixel or the diagonal pixel.

Corrections to the exercise text will be posted here, keep an eye on this page

NOTICE: to disable some safety checks and the timeout, change test_٠١.py by setting DEBUG=True

NOTICE: to keep safety checks but use a bigger timeout, change test_٠١.py by setting a bigger WARP factor

Instructions

To complete the homework:

Install all the required libraries (INCLUDING stopit)

Download the HW٦req.zip file and unzip it in a directory

The archive contains:

a problem to be solved, described at the beginning of file program٠١.eng.py;

your job is to complete the ex١ function inside program٠١.eng.py to solve the problem – adding other functions is allowed;

a file named program٠١.txt where you should describe your algorithm in English (keep it anonymous: no id/names or source code);

all other libraries and test files needed to run tests on your machine.

Enter the directory created by unzipping the archive and

rename program٠١.eng.py as program٠١.py and edit the file to solve the problem;

edit file program٠١.txt to describe your algorithm.

NOTICE: the program should not use input() or print(). All needed parameters are passed to the ex١ function by the test system. Your results are returned to the tests by using return.

BEWARE: global variables are forbidden.

BEWARE: it's forbidden to import other libraries apart from the ones already imported in the text.

NOTICE: your program should be applicable to any correct input (do not exploit data repetitions or particularities).

After the final deadline some secret tests will be applied and your program could fail tests not showing the same patterns.

DO NOT LEAVE YOUR TEST FUNCTIONS INSIDE THE PROGRAM, UNLESS THEY ARE AFTER THE LINE

if __name__ == '__main__':

Here you can enter your own test instructions

To test your program:

open an “Anaconda Prompt” window and enter the directory obtained by unzipping the file

run the following command

pytest test_٠١.py -v -rA

or (test printing also a list of the slowest runs)

pytest test_٠١.py -v -rA --durations ٠

or (test printing a list of the ٢٠ slowest functions executed)

pytest test_٠١.py -v -rA --profile

To stop tests at the first error, add the -x option

If you use Spyder ٤: (version ٥ is not supported by these plugins)

you can run tests from Spyder by first installing the spyder-unittest plugin (but you cannot add parameters to the test execution);

you can profile the functions from Spyder by first installing the spyder-line-profiler plugin.

NOTICE: to open and edit text files in the UTF^ format, DO NOT use Notepad (as it does not handle well the Unix linefeed character '\n'). Use Spyder or Notepad++ instead.