Sudoku Solver With Statistics

Release 1.0

Marcin Borowski

Nov 23, 2023

CONTENTS:

CHAPTER

ONE

FINAL_PROJECT

1.1 project module

project.**argparse_logic**() → tuple[Namespace, ArgumentParser]

Returns argparse Namespace instance containing all args and flags. Assign args after returning or change what is to be returned via this function.

Raises
argparse.ArgumentError – If -m is not an int in specified range

Returns
An object containing names of args provided via cli

Return type
argparse.Namespace

project.**main**()

Main wrapper function containing all logic and loops.

project.**show_stats**(*db_name: str*) → None

Main logic loop for CLS menu which gives the user the ability to brow solve stats, drop db contents into a csv file, delete entries, or even drop the wole db.

Parameters
db_name – Name of database file

project.**test_pipeline**(*args: Namespace*, *sudoku_data: list[dict[str, str]]*, *db_name: str*) → None

Test pipeline for sudoku algorithms. Elegant way for testing and writing to DB

Parameters

- **args** – Argparse Namespace
- **sudoku_data** – A dict containing sudoku puzzles and/or solutions

1.2 helpers package

1.2.1 Submodules

1.2.2 helpers.dbops module

helpers.dbops.**conn_to_db**(*db_name: str*) → tuple[Cursor, Connection]

Returns a cursor of DB object. DB name "stats.db"

Returns
DB cursor object

Return type
SQLite3 object

helpers.dbops.**get_data**(*db_name: str*, *db_content: str*, *target=''*) → None

Connects to DB and reads the data to be fetched from DB table. Then prints them to the screen.

Parameters

- **db_name** – Name of database file
- **db_content** – What is to be read from the db
- **target** – A str describing what should be modified when dropping entry or table

helpers.dbops.**setup_path**()

helpers.dbops.**termination**(*db_conn: object*) → None

Terminates the program via sys.exit()

Parameters
db_conn – SQLite3 connection object

helpers.dbops.**write_to_db**(*presentation_method: str*, *solve_method: str*, *test_date: object*, *start_time: str*, *duration_time: float*, *puzzles_read: int*, *solutions_found: int*, *avg_solve_time: float*, *db_name: str*) → None

Establishes a SQLite DB if there is no DB called "stats.db", or creates the table called "statistics" if there is none. Writes data to the DB.

Parameters

- **presentation_method** – Type of data presentation method. Available test or tofile
- **solve_method** – Algo name used to solve given dataset
- **test_date** – Date of execution
- **start_time** – Time at which the test has started
- **duration_time** – Operation duration time
- **puzzles_read** – How many puzzles there were in the dataset
- **solutions_found** – How many puzzles were solved
- **avg_solve_time** – Average time needed to solve one puzzle using given method
- **db_name** – A string representing db name

1.2.3 helpers.fileops module

helpers.fileops.**db_to_file**(*db_name: str*) → None

Saves DB contents into a csv file

Parameters
db_name – Name of the db file for connection purposes

helpers.fileops.**lines_positions**(*file: str*) → list[int]

Returns a list with 200 new line locations from a file. Uses tell() function If file has less lines stops at file end.

Parameters
file – filename or filepath as a str

Raises
IOError – If something's wrong with the file

Returns
A list composed of new line starting locations as int

Return type
list

helpers.fileops.**linesread**(*file: str*, *line_loc: int*) → str

Returns line read from a file to help determine which column is the puzzle, and which is the solution.

Parameters

- **file** – filename or filepath as a str
- **line_loc** – Integer number of byte position for a given line

Raises
IOError – If something's wrong with the file

Returns
A string containing puzzle and solution separated with comma

Return type
str

helpers.fileops.**open_file**(*filename: str*, *mode='r'*) → IO

Opens the file of given name or path.

Parameters

- **filename** – filename or filepath
- **mode** – File open mode. Here read mode.

Raises
IOError – If something's wrong with the file

Returns
A IO object

Return type
file object

helpers.fileops.**read_file**(*file: str*, *solutions: bool*) → list[str]

Returns list of dicts with sudoku puzzles or puzzles and solutions pairs provided by the user. Solutions are provided by the user. Output is going to be used to fed into the solver.

Parameters

- **file** – A string containing filename with extension or path to the file
- **solutions** – A boolean value depending on flag given by the user

Returns
A list of dicts containing sudoku data. Puzzles and solutions or puzzles only

Return type
list

helpers.fileops.**write_to_file**(*sudoku_data: dict[str, str]*, *grid: list*, *method_name*, *comparison=False*) → None

Writes a bunch of data to a file called "results.csv". Creates file if it doesn't exist, appends if it does.

Parameters

- **sudoku_data** – A dict contain ing puzzle and/or solution provided for comparison
- **grid** – A list of list representing generated sudoku solution
- **comparison** – Result of comparison if generated solution is equivalent to provided one
- **method_name** – A string representing solve method name

1.2.4 helpers.gridops module

helpers.gridops.**grid_to_str**(*grid: list[list[str]]*) → str

Creates string form of a grid ready for writing to a file, or comparison with provided dataset.

Parameters
grid – A list of list representing sudoku puzzle

Returns
A string concatenated from the grid

Return type
str

helpers.gridops.**make_grid**(*entry: dict[str, str]*) → list[list[str]]

Takes a dict as an input. Manipulates a string and creates a 9x9 grid out of it. The grid is a list of list made of 9 rows with 9 separated string chars inside.

Parameters
entry – A string containing sudoku puzzle to solve

Returns
A list of lists (2d grid) with strings inside

Return type
list

helpers.gridops.**print_grid**(*grid: list[list[str]]*) → object

returns grid using NumPy matrix method

Parameters
grid – List of list representing sudoku puzzle

Returns
A grid being a NumPy object

Return type
numpy matrix object

1.2.5 helpers.miscellaneous module

helpers.miscellaneous.**clear_screen**()

Clears terminal window.

helpers.miscellaneous.**compare**(*grid: list[list[str]]*, *sol_str: dict[str]*) → bool

Compares generated solution to the solution provided via file by the user

Parameters

- **grid** – A list of list representing generated sudoku solution
- **sol_str** – A string with provided solution to given puzzle for comparison

Returns
True or false

Return type
bool

1.2.6 helpers.printops module

helpers.printops.**confirmation**() → None

Confirmation menu for DB drop

helpers.printops.**print_date**() → str

Returns valid date string if input matches the pattern and adopted conventions for dates.

Returns
A date entered by the user after validation.

helpers.printops.**print_main_menu**() → None

Manin menu

helpers.printops.**print_other_actions**() → None

Other actions menu

helpers.printops.**print_presentation_method_menu**() → None

Presentation by method menu

helpers.printops.**print_solve_menu**() → None

Presentation by solve method menu

1.2.7 helpers.solvers module

helpers.solvers.**bact_r_solve**(*grid: list[list[str]]*, *free_fields: list[tuple]*, *depth=0*) → bool

A backtracking recursive algorithm for solving sudoku puzzle. Algorithm writes values to the list which was the argument in function call.

Parameters

- **grid** – A list of list representing sudoku puzzle
- **free_fields** – A list of tuples representing free cells of sudoku grid
- **depth** – A value indicating where for loop should start iteration over free cells

Returns
Boolean value indicating if solution was found or not

Return type
bool

helpers.solvers.**boost_bact_r_solve**(*grid: list[list[str]]*, *valid_vals: dict[tuple]*) → bool

A backtracking recursive algorithm for solving sudoku puzzle. Takes additional argument being a dict of possible values for given board field. Algorithm writes values to the list which was the argument in function call.

Parameters

- **grid** – A list of list representing sudoku puzzle
- **valid_vals** – A dict containing possible values for each free board field

Returns
Boolean value indicating if solution was found or not

Return type
bool

helpers.solvers.**cake_algo**(*grid: list[list[str]]*) → str

If you do this test calculation, You get a CAKE!

Parameters
grid – A list of list representing sudoku puzzle

Returns
A string containing life thoughts.

Return type
str

helpers.solvers.**dlxsudoku_module**(*data: dict[str]*) → list[list[str]]

Sudoku solver found on PyPl using induction, Dancing Links and brute force. https://pypi.org/project/dlxsudoku/

Parameters
data – Raw sudoku data containing puzzle and/or solution

Returns
Grid containing solution to given sudoku puzzle

Return type
list

helpers.solvers.**find_empty**(*grid: list[list[str]]*) → tuple

Checks if given board field is empty or not returning row, col tuple. Returns None otherwise.

Parameters
grid – A list of list representing sudoku puzzle

Returns
A tuple containing row and colum indices or None value

Return type
tuple or None value

helpers.solvers.**find_valid_vals**(*grid: list[list[str]]*, *position: tuple[int, int]*) → set

Finds possible values for given sudoku field (row col) tuple. Adds them to the set, and returns it

Parameters

- **grid** – A list of list representing sudoku puzzle

- **position** – A tuple with row and column indices of a grid

Returns
A set of possible values for given grid's field

Return type
set

helpers.solvers.**list_of_free_fields**(*grid: list[list[str]]*) → list[tuple[int, int]]

Make list of free fields to speed up recursion & backtrack algorithm.

Parameters
grid – A list of list representing sudoku puzzle

Returns
A list of tuples containing free cells addresses

Return type
list of tuples of integers

helpers.solvers.**ordered_valid_vals**(*valid_vals: dict[tuple, list[str]]*) → dict[tuple, list[str]]

Checks the frequency od found valid values for given grid field, and orders them by ascending order

Parameters
valid_vals – A dict of strings containing available values for given fields

Returns
A dict of strings containing available values for given fields in ascending order

Return type
dict of strings

helpers.solvers.**random_walk**(*grid: list[list[str]]*) → bool

Made in association with CS50 Duck debugger. It solves, but its random... Sometimes it gets stuck, sometimes it takes some time to solve, and sometimes its solves the problem quite fast (rarely)...

Parameters
grid – A list of list representing sudoku puzzle

Returns
A boolean value for indication if there is a solution or not

Return type
bool

helpers.solvers.**scan_for_valid_vals**(*grid: list[list[str]]*) → dict[tuple, list[str]]

Function scanning given sudoku grid, and filling a dictionary with tuple (row, col) as keys, and a list of possible values for given free field of sudoku grid. List is provided by "find_valid_vals" function.

Parameters
grid – A list of list representing sudoku puzzle

Returns
A dict with key[(row, col) tuple]: [list of possible values]

Return type
dict of strings

helpers.solvers.**validator**(*grid: list*, *digit: str*, *position: tuple*) → bool

Algorithm for validating if given value at given grid's position can be put in it, based of games rules, row, column and 3x3 square can't contain duplicates of any 1-9 value.

Parameters

- **grid** – A list of list representing sudoku puzzle
- **digit** – A value to put into given grid's field
- **position** – A tuple with row and column indices

Returns
True or False for given value

Return type
bool

1.2.8 helpers.validateops module

helpers.validateops.**determine_headers**(*file: str*, *lines_loc: list[int]*) → list[str]

Returns header list for csv.DictReader() based on the sample of data read from a file.

Parameters

- **file** – filename or filepath as a str
- **lines_loc** – List of integer number of byte position for a given line

Raises
IOError – If something's wrong with the file

Returns
A list of headers for csv.DictReader() determining where are the puzzles,

and where are the solutions :rtype: list

helpers.validateops.**validate_date**(*chosen_date: str*) → bool

Validates if a string is in valid date format, and a valid date.

Parameters
chosen_date – A string containing date for validation

Returns
Output either false or true

Return type
bool

helpers.validateops.**validate_file**(*filename: str*) → bool

Validates if file of given filename/filepath exists and returns T or F.

Parameters
filename (*str*) – Filename or path and file for validation

Returns
True or False

Return type
bool

helpers.validateops.**validate_rows**(*row: dict[str, str]*, *header: list[str]*, *solutions: bool*) → bool

Returns True if row is valid or False if not.

Parameters

- **row** – A dict containing line read from a file
- **header** – A list containing fieldnames for validation

- **solutions** – A boolean value depending on flag given by the user

Returns
True if validation is positive, False if it's negative

Return type
bool

1.2.9 Module contents

1.3 test_project module

test_project.**test_argparse_logic**(*monkeypatch*)

Tests if argparse_logic function is doing what is should. Uses monkeypatch fixture to provide sys.argv arguments

Parameters
monkeypatch – Pytest builtin fixture

Raises

- **AssertionError** – If positive test isn't valid
- **AttributeError** – If no errors are raised where they should be.

test_project.**test_bact_r_solve**()

Checks if tested function properly utilizes list_of_free_fields().

Function should return a list containing tuples with address cells of "0" in the grid. If the function was implemented correctly solver algorithm will find a solution to given grid.

Raises
AssertionError – If test isn't valid

test_project.**test_cake_algo**()

Tests if the output of Cake algorithm is as it should be.

Raises
AssertionError – If test isn't valid

test_project.**test_confirmation_negative**()

Test written for checking if print output of a function is valid or not.

Test is executed by the usage of contextlib and io modules. Contextlib uses redirect_stdout for print output than with help of StringIO an object is created, and the data extraction processed by getvalue().

Raises
AssertionError – If test ends with a success

test_project.**test_confirmation_positive**()

Test written for checking if print output of a function is valid or not.

Test is executed by the usage of contextlib and io modules. Contextlib uses redirect_stdout for print output than with help of StringIO an object is created, and the data extraction processed by getvalue().

Raises
AssertionError – If function's output isn't exact match.

test_project.**test_determine_headers**()

Checks what is the output for give data set. This is another step of the data provided in the input file.

Raises
AssertionError – If test isn't valid

test_project.**test_find_empty**()

Test if output is a tuple with row and col of the first free field.

Crucial functionality for sudoku validator.

Raises
AssertionError – If test isn't valid

test_project.**test_find_full**()

Test if output is a tuple with row and col of the first occupied field.

Crucial functionality for sudoku validator.

Raises
AssertionError – If test isn't valid

test_project.**test_grid_to_str**()

Test if joining grit to string gives the same result.

This is needed for further test, comparison and programs functionality.

Raises
AssertionError – If test isn't valid

test_project.**test_linesread**()

Checks if string returned is of 163 chars length and presence of 0

The length is fixed for every line, because of the tested file's construction. Every line is built of sudoku puzzle and solution separated by comma. If all elements are present it means that tested function returned proper line for further processing. Every line should be of 163 chars long, contain 0 only in the left part.

Raises
AssertionError – If test isn't valid

test_project.**test_make_grid**()

Checks if string conversion to a grid is valid

Grid is necessary for most of solving algorithms. So conversion in both ways is crucial.

Raises
AssertionError – If test isn't valid

test_project.**test_open_file**()

Checks if returned object is a file Tested function opens a file using built in method.

Raises
AssertionError – If test isn't valid

test_project.**test_print_date_neg**(*monkeypatch*)

Test written for checking if print output of a function is valid or not.

First step is overriding users input, which is required by tested function. Without it program waits infinitely. This is Achieved by using builtin pytest's fixture monkeypatch. It uses setattr pointing to readline for data input.

Test is executed by the usage of contextlib and io modules. Contextlib uses redirect_stdout for print output than with help of StringIO an object is created, and the data extraction processed by getvalue().

Parameters
monkeypatch – Builtin Pytest functionality

Raises
AssertionError – If function's output isn't an exception.

test_project.**test_print_date_pos**(*monkeypatch*)

Test written for checking if print output of a function is valid or not.

First step is overriding users input, which is required by tested function. Without it program waits infinitely. This is Achieved by using builtin pytest's fixture monkeypatch. It uses setattr pointing to readline for data input.

Test is executed by the usage of contextlib and io modules. Contextlib uses redirect_stdout for print output than with help of StringIO an object is created, and the data extraction processed by getvalue().

Parameters
monkeypatch – Builtin Pytest functionality

Raises
AssertionError – If function's output isn't exact match.

test_project.**test_print_main_menu_neg**()

Test written for checking if print output of a function is valid or not.

Test is executed by the usage of contextlib and io modules. Contextlib uses redirect_stdout for print output than with help of StringIO an object is created, and the data extraction processed by getvalue().

Raises
AssertionError – If test ends with a success

test_project.**test_print_main_menu_pos**()

Test written for checking if print output of a function is valid or not.

Test is executed by the usage of contextlib and io modules. Contextlib uses redirect_stdout for print output than with help of StringIO an object is created, and the data extraction processed by getvalue().

Raises
AssertionError – If function's output isn't exact match.

test_project.**test_print_other_actions_neg**()

Test written for checking if print output of a function is valid or not.

Test is executed by the usage of contextlib and io modules. Contextlib uses redirect_stdout for print output than with help of StringIO an object is created, and the data extraction processed by getvalue().

Raises
AssertionError – If test ends with a success

test_project.**test_print_other_actions_pos**()

Test written for checking if print output of a function is valid or not.

Test is executed by the usage of contextlib and io modules. Contextlib uses redirect_stdout for print output than with help of StringIO an object is created, and the data extraction processed by getvalue().

Raises
AssertionError – If function's output isn't exact match.

test_project.**test_print_presentation_method_menu_neg**()

Test written for checking if print output of a function is valid or not.

Test is executed by the usage of contextlib and io modules. Contextlib uses redirect_stdout for print output than with help of StringIO an object is created, and the data extraction processed by getvalue().

Raises
AssertionError – If test ends with a success

test_project.**test_print_presentation_method_menu_pos**()

Test written for checking if print output of a function is valid or not.

Test is executed by the usage of contextlib and io modules. Contextlib uses redirect_stdout for print output than with help of StringIO an object is created, and the data extraction processed by getvalue().

Raises
AssertionError – If function's output isn't exact match.

test_project.**test_print_solve_menu_neg**()

Test written for checking if print output of a function is valid or not.

Test is executed by the usage of contextlib and io modules. Contextlib uses redirect_stdout for print output than with help of StringIO an object is created, and the data extraction processed by getvalue().

Raises
AssertionError – If test ends with a success

test_project.**test_print_solve_menu_pos**()

Test written for checking if print output of a function is valid or not.

Test is executed by the usage of contextlib and io modules. Contextlib uses redirect_stdout for print output than with help of StringIO an object is created, and the data extraction processed by getvalue().

Raises
AssertionError – If function's output isn't exact match.

test_project.**test_read_file**()

Checks if returned object is of list type.

Tested function should read through the file and return a list containing every valid line, of read file. So if we have a list object, we should be ready to proceed.

Raises
AssertionError – If test isn't valid

test_project.**test_validate_file**()

Validates whether a file exists and if it is a file indeed. Tested function uses os.path.isfile() for the validation.

Raises
AssertionError – If test isn't valid

test_project.**test_validate_rows_noflag**()

Tests puzzle for data validation. Noflag for solutions.

Tested function should check if a read row is valid for further manipulation, and solving processes. It is done by checking if provided header of csv file is same as file contents. If i.g. there should be no comma in any line, and it should contain "0" values.

Raises
AssertionError – If test isn't valid

test_project.**test_validate_rows_puzzle**()

Test puzzle data validation.

Tested function should check if a read row is valid for further manipulation, and solving processes. It is done by checking if provided header of csv file is same as file contents. If i.g. puzzle, solution header is provided, "0" values should be only on the left part of each line.

Raises
AssertionError – If test isn't valid

test_project.**test_validate_rows_solution**()

Tests solutions data validation.

Tested function should check if a read row is valid for further manipulation, and solving processes. It is done by checking if provided header of csv file is same as file contents. If i.g. puzzle, solution header is provided, there should be no "0" values in the solution part of the line. part of each line.

Raises
AssertionError – If test isn't valid

test_project.**test_validator**()

Tests if validator is returning right boolean output

No solution if it is against the rules. So validator should assert if given digit, can fit into checked cell.

Raises
AssertionError – If test isn't valid

test_project.**test_write_to_file**()

Test write to file functionality.

When users desires to write sudoku puzzle solutions into the file, tested function should fulfill that desire. Test is simple, after using this function 'results.csv' file should be created. If so, when opened it should be an instance of parent class.

Raises
AssertionError – If test isn't valid

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

h

p

t