SimCity Land Value Calculator Version 3: Estimate missing land values

Some of the land values are missing! Let's estimate the missing value of a cell with value 0, using the values of its neighbors.

Once you have calculated the missing land values, display the updated grid.

IMPORTANT: Do not modify the original grid, but create a new grid where no cell has missing values.

Study the examples below to see how the `fill_gaps` function should work:

Position (0, 0) of our grid is highlighted in blue.

0	6	1
1	4	7
4	0	3

We can get the average value of the neighbor cell prices using the **`find_neighbors`** function.

```
[6, 1, 4]
```

Position (0, 0) in the new grid is set to that average (assume repeating value for 3.666):

3.6	6	1
1	4	7
4	0	3

Position (2, 1) of our grid is highlighted in blue.

3	6	1
1	4	7
4	0	3

We can average the values from our **`find_neighbors`** function to fill the gap.

```
[1, 4, 7, 4, 3]
```

Position (2, 1) of our new grid becomes:

3	6	1
1	4	7
4	3.8	3

You can safely assume that if a cell value is missing (the value is given as 0), then none of the neighboring cell values are also missing. So you will always take averages over non-zero values.

What to do

- Implement the `fill_gaps` function. This function will do the following:

    - Use the `deepcopy` function from the `copy` module to create a new grid which is a copy of the original grid.
    - Estimates the missing land values within the original grid and updates the new grid with the missing land values.
    - Returns the new grid.
- You must use the given template means you should not change the names of the functions, its parameters or the object it returns.

You must use the following template:

```
def create_grid(filename: str) -> list[list[int]]:
    """
    Create a grid of land values from a file
    """
    # Implemented in Version 1

def display_grid(grid: list[list[int]]) -> None:
    """
    Display a grid of land values
    """
    # Implemented in Version 1

 def find_neighbor_values(grid: list[list[int]], row: int, col: int) -> list[int]:
    """
    Find the neighbors of a cell
    """
    # Implemented in Version 2

 def fill_gaps(grid: list[list[int]]) -> list[list[int]]:
    """
    Fill the gaps in the grid
    Creates a new grid that is a copy of the original grid
    Call find_neighbor_values function to find the neighbors of each cell.
    Find the average of their values and round it to the nearest integer
    Use the average values to fill in the missing values in the new grid.
    Return the new grid
    Do NOT modify the original grid!
    """
    # TODO: Implement this function
    pass

def main() -> None:
    """
    Main program.
    """
    grid = create_grid("data_0.txt")
```

```
    print("SimCity land values:")
    display_grid(grid)
    print("\nCalculated SimCity land values:")
    new_grid = fill_gaps(grid)
    display_grid(new_grid)

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

Hints

- Import the `copy` module in order to use the `deepcopy` function.

Program name

Save your program as `simcity3.py`.

Demo

In this demo, `data_1.txt` is used.

https://asciinema.org/a/YXxXjyRiVAiVXd853d2P9hVsI

Testing

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

Good news: we wrote the unit tests for this version as well: test_simcity3.py

To test your `fill_gaps()` function, simply run the unit tests in your terminal:

```
$ python -m pytest test_simcity3.py
```

All tests should pass.

You should also manually test your program with different input files found in the introduction document.

- Run your program with `python simcity3.py` with `data_0.txt` Your program should print:

```
Sim City land values:
       1        0        3        4
       5        6        7        8
       9       10       11       12
      13       14       15       16

Calculated Sim City land values:
       1        4        3        4
       5        6        7        8
       9       10       11       12
      13       14       15       16
```

- Run your program with `python simcity3.py` with `data_1.txt` Your program should print:

```
Sim City land values:
   76000        0    54000    16000    83000
   27000    49000    62000        0    31000
       0    48000    53000    22000    19000
   71000    37000    63000    41000        0
   83000    25000        0    16000    59000

Calculated Sim City land values:
   76000    53600    54000    16000    83000
   27000    49000    62000    42500    31000
   46400    48000    53000    22000    19000
   71000    37000    63000    41000    31400
   83000    25000    36400    16000    59000
```

- Run your program with `python simcity3.py` with `data_2.txt` Your program should print:

```
Sim City land values:
   94000    64000    30000        0    14000    92000
   37000    49000    50000    29000    35000        0
       0    88000    85000    96000    60000    22000
   13000    44000    73000        0    45000    53000
   20000    33000    67000    71000    82000        0
   36000        0    62000    55000    44000    75000
```

```
Calculated Sim City land values:
   94000    64000    30000    31600    14000    92000
   37000    49000    50000    29000    35000    44600
   46200    88000    85000    96000    60000    22000
   13000    44000    73000    72375    45000    53000
   20000    33000    67000    71000    82000    59800
   36000    43600    62000    55000    44000    75000
```

- Run your program with `python simcity3.py` with `data_3.txt` Your program should print:

```
Sim City land values:
   24000    57000    50000    43000
   38000        0    16000    62000
   51000    25000    49000        0
       0    76000    19000    34000

Calculated Sim City land values:
   24000    57000    50000    43000
   38000    38750    16000    62000
   51000    25000    49000    36000
   50667    76000    19000    34000
```

Submitting

Submit `simcity3.py` via eClass.