FYSS5120 Efficient Numerical Programming - Demo 3

Drop solutions before the demo session to the Nexcloud box (link)
Please indicate clearly your name in the file name.

1. Study the program `aitken_accelerate.py`, which applies Aitken's Δ^2 acceleration to two series, which slowly converge to $\pi/4$ and $\pi^2/6$, respectively. Answer the questions:

   - Let's examine how to a generator function like `leibnitz_pi()` works, and how `itertools.cycle([1,-1])` generates the sequence 1,-1,1,-1....
     For testing, define a simplified function,

     ```
     >>> import itertools
     >>> def test():
     >>>     for i in itertools.cycle([1,-1]):
     >>>        yield i
     ```

     Why does

     ```
     >>> next(test())
     >>> next(test())
     >>> next(test())
     ```

     give the wrong sequence 1,1,1,1..., but

     ```
     >>> tt = test()
     >>> next(tt)
     >>> next(tt)
     >>> next(tt)
     ```

     produces the correct sequence 1,-1,1,-1...?

   - What does `itertools.islice(leibnitz_pi(),N))` do ? How would you write that as a plain `for`-loop?

   - How does the function `aitken(seq)` work?

2. The code `demo3_heat_animation.py` animates heat flow under the assumption that the temperature of every element is the average of its own temperature and of elements next to it, while keeping the outer edges at fixed temperature. That makes 5 elements to average over. Answer the questions:

   - The method `step()` does one step of heat flow, and it's the most important part of the code. Please explain how it works, even without any `for`-loops?

   - The code has the line

```
mid[:] = (mid+above+below+left+right)/5
```

Why doesn't

```
mid = (mid+above+below+left+right)/5
```

work as intended? For the same reason `self.heat_map[:]  = 100.0` works, but `self.heat_map = 100.0` doesn't.

About the Matplotlib animation:
For clarity, the animation is detached from the class `Heat2D`. The only contact point is the generator function `data_gen()`, which progresses the simulation one step using the `.step()` method. The `update()` function uses Matplotlib's data update method `.set_data()`, which is specific to the `.matshow()` plotting routine. Data update methods vary with plot type, the sample code `sine_animation.py` uses `.plot()` method for plotting, and data update uses `.set_ydata(data)`.