University of Oldenburg
Python Programming in Energy Science, WiSe 2023/24
Dr. Jonas Schulte, jonas.schulte@iwes.fraunhofer.de

Assignment 4

Allowed library imports: `numpy`, `matplotlib`

E12. **(10 points)** Array calculations

Do not use loops over the time dimension for this exercise!

- Create a numpy array called `t` that represents $N = 5000$ time values between 0 and 60 seconds (both included).
- Create another single numpy array called `data`, which represents $\sin(\omega t)$, $\cos(\omega t)$ for the following choices of angular frequency: $\omega = 2\pi/60\text{s}, 2\pi/30\text{s}, 2\pi/100\text{s}, 2\pi/120\text{s}$.
- Make one nice plot showing all of the lines (both functions, all frequencies) with a descriptive legend.
- Now add two more functions to the data: $\sin^2$ and $\cos^2$, by extending the corresponding dimension in the array. Repeat the plotting, but now showing *only* the newly added quadratic functions.
- Sum all $\sin$ and $\sin^2$ functions, for all frequencies, into one new numpy array. Similarly sum all cos and $\cos^2$ results into another array. Plot both, and also their sum, into one plot.

E13. **(10 points)** Data creation

Do not use loops for this exercise!

- Create a numpy array that represents dimensions `(hour, minute, index, data)`, for a single day of measurements (hours: 0-23, minutes:0-59), where the `index` indicates 5 different weather stations and the `data` 3 data entries per location, filled with zeros of type `np.float64`. Print the `shape` and the `dtype` of your array. How many elements are contained in the array?
- Assume the 3 data entries represent solar irradiance in W/m^2, temperature in Kelvin and wind speed in m/s at 10 m height, respectively. Set all solar irradiance values to 250 W/m^2, then replace all values for the time between 9 PM and 5 AM to zero. Check that this worked as intended, by printing appropriate slices.
- Create another numpy array representing dimensions `(minute, index)` with random values between 3 and 8 m/s. Set all wind speed data for all stations to that data, such that it repeats every hour. Check that this worked as intended, by printing appropriate slices.
- Fill the temperature data with random values between 5 and 20 degrees Celsius (translate that into Kelvin). Station 2 has some problem during nights, so replace all temperatures by `np.nan` between 9 PM and 5 AM. Station 0 and 1 do not work correctly at high wind speeds, so replace the temperature by `np.nan` whenever the wind speed is above or equal 6 m/s for those stations. Again, check that this worked as intended.

E14. **(10 points)** Wind farm data – part I

The provided data file `results_farm.csv.gz` has a single header line, stating the column names:

```
State,turbine,YAW,WD,AMB_WS,WS,AMB_P,P,GY,NY,P75,P90,EFF,WLOSS
```

The `State` column contains integer numbers, indexing 30-minute time average data (the time stamp itself has been removed, since it is not needed here).

- Read the file `results_farm.csv.gz` into a `numpy.ndarray`, ignoring the first row (i.e., the above header line). Print the first 10 rows of data. Note that you do not need to extract the *gz* file before reading it with *numpy*.
- Select the subset of the data for which the produced power is zero. What is the maximal ambient wind speed `AMB_WS` in this subset? What the minimal wake-corrected wind speed `WS` in another subset with power greater than zero? The data unit in both cases is m/s.

- How many turbines are in the wind farm? Create a separate numpy array for each turbine (use at most a single loop). Then, find out how much mean power *P* was produced by each turbine with respect to the states. The unit of `P` in the data is kW. How high is the annual energy production (AEP) in GWh based on this data for the whole farm?

E15. **(10 points)** Wind farm data – part II

- Read the data from E14 into a single numpy array, as described there.
- Create a simple line plot, showing the mean ambient power `AMB_P` and the mean produced power `P` wrt. the states, as a function of turbine index. Include axes labels, a legend and a title.
- Replace all values of the wind turbine efficiency `EFF` by `np.nan` if they are zero (or below). Then create a 2D `pcolormesh` plot that shows the variable `EFF` as a function of `turbine` on x-axis and `state` on the y-axis, with coordinate (0, 0) in the lower left corner. Make the plot as pretty as you can, e.g., add axes labels, a title, and a color bar. Follow the lecture on matplotlib, but please, no red lines between the data squares.
- Similarly, create two beautiful `pcolormesh` plots showing the wind speed `WS` and the produced power `P`. Use different color maps for each of the latter three 2D-plots, and always show color bars. You may also use other ways of visualizing this 2D data than `pcolormesh`, if you prefer (but please not just a bunch of single lines).

E16. **(10 points)** Topography data

The file `RoedeserBerg_large.pts.gz` contains (x,y,z) topography data of a location near Kassel, Germany.

Solve the whole exercise without using loops.

- Read the data contained in the above file into memory.
- Select the subset of points whose horizontal distance from the position $p_0 = (513196.5, 5689695.0)$ is less than radius $r = 5$ km.
- Store the selected data into a new file `RoedeserBerg.pts.gz` in the same format as the original file. How many points are stored in the file?
- Visualize the selected topography data using `matplotlib` or a sub-package. Additionally store the resulting graphic into a png file `RoedeserBerg.png`