Exercise 5 *Filtrations and Barcodes*
Let $Q = \{(0,0),(-1,11),(9,1),(11,10)\}$ and $W = \{(0,10),(2.5,1),(5.5,5),(7.5,0),(10,9)\}$. Plot Q and W, and determine their Rips and Čech filtrations, e.g. by visual inspection. In both cases extract the (qualitative) barcode of the Rips filtration. Use ripser to check your result.

Exercise 6 *Geometric Information in Barcodes*

Download the 2d datasets at
https://github.com/micbl/TDAworkshop/blob/main/Exercises/exercise6_2d_shapes.zip

The data is sampled from three different underlying shapes A, B, C with sample sizes $n \in \{10, 50, 100, 500, 1000\}$.

(a) Calculate the barcode of the datasets with ripser.

(b) What do you think the underlying shapes look like?

(c) Plot the data and discuss your findings.

Exercise 7 *Topology Layer*
provided by Sebastian Damrich
based on https://github.com/bruel-gabrielsson/TopologyLayer

TopologyLayer can be used to differentiate through the computation of topological features with persistent homology. We provide examples on images, which despite their high dimensionality (pixel number) are easy to grasp for humans by looking at them. Moreover, changing just a few pixels in an image can drastically change a global property such as topology. Image generation algorithms typically do not consider such global properties but only care about the individual pixel values. Complementing this local approach, differentiating through persistent homology with TopologyLayer yields gradients on the pixels indicating how those pixels would have to be changed to enhance a desired topological feature. This change is then performed by gradient descent. In this exercise we will have a look at toy examples of topological reconstruction in image data based on these ideas.

Preparation
Download the folder `TDA_workshop` at
https://heibox.uni-heidelberg.de/d/75edfd01000a4e0ca4b3/
password: pershomology

The two examples are in `noisy_image.py` and `reg_lin_regression.py`.

The file `environment.yml` is now used to set up a python installation that will run the examples. For this do:

1. Install Anaconda, e.g. https://docs.anaconda.com/anaconda/install/
2. Restart terminal
3. `cd TDA_workshop`
4. `conda env create -f environment.yml`
5. `conda activate topolayer`
6. `pip install git+https://github.com/bruel-gabrielsson/TopologyLayer.git`
7. `cd TopologyLayer`

Noisy Image
`noisy_image.py` creates a corrupted image of a circle, then updates the image to recreate the known topological features of a circle (not more than one connected component, exactly one 1D feature) while trying to stay close to the corrupted image. By varying the parameters `w_sum_features_0d`, `w_spurious_features_1d`, `w_main_feature_1d` (in particular, setting them to zero or inverting the sign) different topological properties of the image can be induced. Play around with these parameters by repeating the following steps.

8. Update the weights to your liking
9. `python noisy_image.py`
10. Check out the created .png's

Regular linear regression
`reg_lin_regression.py` is from the original repository (which also contains other examples in case you are looking for more). Here, we regress a linear model whose parameters form an image. This image is supposed to show a circle (if there were no noise). The least squares solution produces a very noisy circle, but regularising with known topological features of a circle (not more than one connected components, not more than one 1D feature) yields a much cleaner reconstruction.

11. `python reg_lin_regression.py`
12. Check out the created .png's