# Coding Projects

## Project 1 - Game of life

Game of life is a cellular automaton devised by John Conway in 70's:
http://en.wikipedia.org/wiki/Conway's_Game_of_Life

The game consists of two dimensional orthogonal grid of cells. Cells are in two possible states, alive or dead. Each cell interacts with its eight neighbours, and at each time step the following transitions occur:
- Any live cell with fewer than two live neighbours dies, as if caused by underpopulation
- Any live cell with more than three live neighbours dies, as if by overcrowding
- Any live cell with two or three live neighbours lives on to the next generation
- Any dead cell with exactly three live neighbours becomes a live cell

The initial pattern constitutes the seed of the system, and the system is left to evolve according to rules. Deaths and births happen simultaneously.

In a git repository implement the Game of Life using Numpy. Try first 32x32 square grid and cross-shaped initial pattern:

Try also other grids and initial patterns (e.g. random pattern). Try to avoid for loops. For visualization you ca use Matplotlib: import matplotlib.pyplot as plt plt.imshow(array)

Make a pip package out of it including dependencies. Add CI using Travis, testing installation from pip and running one game of 1000 iterations checking that it matches a pre known pattern.

References: http://science-it.aalto.fi/wp-content/uploads/sites/2/2016/06/numpy_exercises.pdf

## Project 2 - The Traveling Salesperson Problem

Consider the *Traveling Salesperson Problem*:

> *Given a set of cities and the distances between each pair of cities, what is the shortest possible tour that visits each city exactly once, and returns to the starting city?*

Create a package that tries to find a good solution for the traveling salesperson problem which takes a matrix with the distances between the points and outputs the best solution found. Try it out on the 10 largest Swedish cities.
Make a pip package out of it including dependencies. Add CI using Travis, testing installation from pip and running the 10 cities example.

References: http://nbviewer.jupyter.org/url/norvig.com/ipython/TSP.ipynb

## Project 3 - Finding exoplanets

Using the data from the Kepler mission (https://www.kaggle.com/keplersmachines/kepler-labelled-time-series-data/downloads/exoTest.csv) see how you fare at finding exoplanets. Develop an algorithm that can detect exoplanets using the training set and then see how many of them do you detect on the test set.
A full description of the dataset is available at https://www.kaggle.com/keplersmachines/kepler-labelled-time-series-data

## Project 4 - Matrix Orthogonalization

Develop a package that does a QR decomposition of a matrix using the orthogonal Householder transformation. You can find details of the algorithm here http://www.cs.cornell.edu/~bindel/class/cs6210-f09/lec18.pdf.

Make a pip package out of it including dependencies. Add CI using Travis, testing installation from pip and running on a random matrix.

Reference - http://www.maths.lth.se/na/courses/NUMA21/exercises/

## Project 5 - Enigma machine

The Enigma machine is a well known device to encrypt messages, primarily used by the Germans in the second world war and famously cracked by the Allied forces. It would be interesting to create a Python version of the Enigma machine, that can encrypt and decrypt messages. You can find plenty of information online about the Enigma machine. Of course, you could also implement more contemporary encryption methods.

Reference - https://web.stanford.edu/~schmit/cme193/project.html

## Project 6 - Try out  a Kaggle competition

Enter the Kaggle competition “Titanic: Machine Learning from Disaster.” using Python to make your predictions. https://www.kaggle.com/c/titanic. See how well you do against your peers.

## Project 7 - Your own project

Please contact the teachers for approval before starting on your own project. Your project should be in line with the other projects in terms of difficulty and apply the concepts of the course.

## Honor code

You are encouraged to work together and discuss both exercises and the project. However, the project should be your own work and all the code in the git repository you share with us should be written by yourself. Of course, you are free, and encouraged, to search online to solutions to problems you run into (though not full solutions to the exercises), and use those. A simple example would be to look up the syntax of a for loop.

Adapted from: https://web.stanford.edu/~schmit/cme193/