General constraints for code submissions

- The program can be called via `python src/main.py`
- You will have to provide a usage, callable via `python src/main.py -h`
- Your code should be commented and readable for the tutors
- All functions and classes are documented with a docstring.
- Provide a README detailing how to install requirements and run your program and (if necessary) an installation script if your program requires any other packages.
- Programs are to be submitted in python 3.5 or newer.
- You can use any of the packages that were listed in the exercise requirements.
- If you add new packages not introduced in the exercise sheets, you will have to give a detailed description what you used it for and provide a link to it's documentation and github page.
- We don't accept ipython notebook submissions.

This final project is necessary for the oral exam, which means you are not allowed to work in groups. The purpose of this project is that you get hands-on experience on most topics of the course and you can present and explain the results of your work. To this end, please submit your code, plots, tables etc. to our prepared final project repositories to which all organizers of the lecture have access. In the first 15 minutes of the exam, you will have the chance to explain your approach and the results. It is important that your evaluation builds the basis for discussion and scientifically analyzes which are the important aspects and characteristics of your approach—your presentation has to present your findings in a convincing manner. Please also make sure to carefully keep track of intermediate results. To give the project presentation some structure, you will have to prepare a few presentation slides. Your slides should consist of a motivation slide, slides detailing your approach (2-3) as well as slides for your results (2-3). You are allowed to submit at most 5 slides. Don't go overboard with your slides. They are intended to make your presentation coherent.

Performance Optimization/Analysis of *lpg*

Your final task is to optimize/analyze the performance of the AI planner lpg on the provided instances. How you optimize *lpg* is up to you. You could use algorithm configuration and/or algorithm selection. In the end, you should convince us that you indeed optimized the performance of *lpg*. To this end, you could consider the following tasks:

- Measure the default performance of *lpg*;
- Plot the performance (as presented in the lecture);
- Guess whether it is a heterogeneous or an homogeneous instance set, to apply PIAC or AC
- Apply algorithm configuration to determine a well-performing configuration;
- Determine the importance of *lpg*'s parameters;
- Optimize an algorithm schedule of configurations of *lpg*;
- Apply algorithm selection to select well-performing configurations of *lpg*;
- Plot the performance of the optimized configuration vs the default configuration;
- Report the performance (PAR1, PAR10, number of timeouts).
- Construct an EPM from observed data and predict well performing configurations.

Please note that you do not necessarily have to apply all these methods – pick the ones that you think are most appropriate.

In your repositories we will upload the following:

- An instance set consisting of AI planning instances.
- A file listing all training and test instances.
- The binary of *lpg* and a file detailing it's parameter configuration space (params.pcs).
- A wrapper that is capable of calling *lpg* and parsing its output.
- A csv file consisting of instance features for all training and test instances.
- We provide three machines (each equipped with two Intel Xeon E5-2650v2 8-core CPUs, 20 MB L2 cache and 64 GB of (shared) RAM per machine, running Ubuntu 14.04 LTS 64 bit) which you can work remotely on and test / evaluate your implementation. (Details later on the HisInOne mailing list.)

Furthermore, you can of course use all scripts and tools you already know from the exercises; however, you are not limited to them.

You should respect the following constraints:

- **Metric:**
  - The final performance has to be measured in terms of CPU-time on the provided machines.
  - To make all results comparable, the final results reported in your presentation have to be obtained on the provided machines. Any preliminary/test runs don't have to be executed on the provided machines.
- **Time Constraints:**
  - Your code should run no longer than 14400 seconds (without validation).
  - *lpg* should not run longer than 100 seconds to solve an instance.

We provide a google spreadsheet[1] in which you can upload your current progress. This sheet will not be monitored by us but gives you the opportunity to compare your results. This might help you identify early if your approach is working well or not.

This assignment is due on 18.03 (23:59 GMT). Submit your solution for the tasks by uploading a PDF to your BitBucket repository. The PDF of your slides has to include your name. Teamwork is not allowed.

[1] `https://goo.gl/VVPQ7E`