General constraints for code submissions Please adhere to these rules to make our and your life easier! We will deduct points if your solution does not fulfill the following:

- If not stated otherwise, we will use exclusively Python 3.6.
- If not stated otherwise, we expect a Python script, which we will invoke exactly as stated on the exercise sheet.
- Your solution exactly returns the required output (neither less nor more) – you can implement a `--verbose` option to increase the verbosity level for developing.
- Add comments and docstrings, so we can understand your solution.
- (If applicable) The `README` describes how to install requirements or provides addition information.
- (If applicable) Add required additional packages to `requirements.txt`. Explain in your `README` what this package does, why you use that package and provide a link to it's documentation or GitHub page.
- (If applicable) All prepared unittests have to pass.
- (If applicable) You can (and sometimes have to) reuse code from previous exercises.

Now that you have learned about hyperparameter optimization techniques such as Bayesian optimization (BO) you will implement this loop yourself.

1. **Bayesian Optimization for HPO** [14 points]

   We provide you with a rough structure of the BO loop using a Gaussian Process. You will implement the remaining parts to **minimize** a synthetic 1D function.

   (a) Implement the acquisition functions *Expected Improvement* as presented in the lecture (use `NumPy` and `SciPy` wherever possible for efficiency). Keep in mind that you will use `scipy.minimize` to optimize the acquisition function. [4pt.]

   Your implementation should satisfy the test in `test_expected_improvement.py`.

   (b) Implement the acquisition functions *Lower Confidence Bound*[1] as presented in the lecture (use `NumPy` and `SciPy` wherever possible for efficiency). Keep in mind that you will use `scipy.minimize` to optimize the acquisition function. [4pt.]

   Your implementation should satisfy the test in `test_lower_confidence_bound.py`.

   (c) Implement Grid Search and Random Search. Your implementation should satisfy the test in `test_random_grid_search.py`. [4pt.]

   (d) Compare your implementations of BO against Random Search and Grid Search for at most 50 function evaluations[2]. [2pt.]

[1] Similar to *Upper Confidence Bound*, but for minimizing an objective value.

[2] Hint: Your implementations of BO should perform better than Random Search.