Projekt part 1

Step-by-step

* GUI
  + Graph generation (interactive process)
    - Read from file -> location input af user
      * Error handling, eg. not open, incomplete lines
      * Conform to standard:
        + Non-empty line represents edge of the graph and is identified by two integers separated by a comma
    - Random
      * User input num of sites
      * Random connected
  + Initial color pattern
    - All 1
    - All 0
    - All random
  + Generer graph
    - Verify all vertices connected
      * Hvis alle connected -> fortsæt til update procedure
      * Ellers start forfra

=========================== BRUGEREN KAN SE ===============================

Hvordan grafen ser ud, valg af update procudere og valg af number of iterations.

* + Update procedure
    - User define num of iterations
    - Ordered
    - MaxViolation
    - MonteCarlo
  + Quit eller kør simulation igen.
* Simulationen
  + Local\_metric
    - Iterate over alle noder/sites/vertices og give en frustration score
  + Global\_metric
    - Sum af local frustration score
  + Visualiser simulerationen med visualizer\_rndgraph.py
    - Update function
      * Implementer update function // ændre farver
      * Displayes til brugeren
      * Track frustration - > skal lagres løbende
      * Når num of iterations er fuldført, så slut.
* Reporting
  + Når simulationen er slut, så skal brugeren præsenteres for en graph, der viser hvordan hver iteration har påvirket global\_metric (total frustration)

Et billede, der indeholder tekst, diagram, skærmbillede, Kurve

Automatisk genereret beskrivelse

* **HUSK** kun afrapportering af en update funktion
  + quit eller ny simulation?