About the lessons:

Mathematical Optimization is a high school course in 5 units, comprised of a total of 56 lessons. The first three units are non-Calculus, requiring only a knowledge of Algebra; the last two units require completion of Calculus AB. All of the units make use of the Julia programming language to teach students how to apply basic coding techniques to solve complex and relevant mathematical problems.

These lessons in Mathematical Optimization were written in 2014 by Julia Roberts, a math teacher at Cupertino High School in the Fremont Union High School District, in conjunction with Dr. Mykel Kochenderfer, professor of Aeronautics and Astronautics at Stanford University, through a grant from the National Science Foundation. They were edited, updated, and turned into Julia notebooks in 2015 by Renee Trochet, a math teacher at Eastside College Preparatory in East Palo Alto, also in conjunction with Dr. Kochenderfer and through a grant from the NSF.

The goals of these lessons are:

- to increase exposure of high school students to current topics of interest in mathematics, including optimization and the use of programming as a tool
- to increase the number and variety of students interested in STEM fields
- to provide students with better preparation for future studies in STEM fields.

The lessons are divided into five units with the following titles:

1. Introduction
2. Unbounded Optimization Without Calculus
3. Linear Programming
4. Unbounded Optimization With Calculus (including multivariable Calculus)
5. Constrained Optimization With Calculus

The first three units require only a solid understanding of Algebra and access to a computer with the Julia language installed. The last two units require differential and integral Calculus (Calculus AB). The entire course (units 1-5) is estimated to take about 23 weeks, though many lessons can be taught as stand-alone or partial units. Accompanying each unit is an overview document which provides information about estimated timing, prerequisite knowledge and equipment.

Special Thanks To:

- Mykel Kochenderfer and the Stanford Intelligent Systems Laboratory (SISL) at Stanford
- National Science Foundation (Award #1265721)
- Industry Initiatives for Science and Math Education (IISME) program/Debra Dimas
- Stanford Office of Science Outreach/Kaye Storm
- Leonard Brzezinski

Resource Acknowledgements:

- Text: Optimization Concepts and Applications in Engineering (Belegundu and Chandrupatla, 2011)
- Hard resources: AA222 lecture notes (Youngjun Kim and Rachael Tompa, Stanford); AA222 class notebooks and syllabus (Dr. Mykel Kochenderfer, Stanford)

- Software resources: Microsoft Office, Adobe Acrobat, Julia Studio (forio.com), LightTable, Google Chrome
- Web resources:
  - Jupyter.org – interactive Julia notebooks
  - Wikipedia.org – historical and topical research, graphics
  - Fooplot.com – a superior online grapher Julialang.org – essential source for all things Julia, from information to language
  - Forio.com – IDE and online sandbox for Julia
  - Lighttable.com – IDE for Julia
  - Google.com – I used "google" as a verb, can you do that?
  - Learnxinyminutes.com – good brief Julia tutorial
  - Cs.cmu.edu (Carnegie Mellon) – actually explains conjugate gradients
  - Maths.surrey.ac.uk (University of Surrey) – golden ratio/Fibonacci number resources
  - Cc.gatech.edu – excellent summary of POMDP applications
  - GitHub – notebook storage and version control

Additional Notes:

- Like most first editions, this one may be prone to error, particularly in the keys to practice problems. Please report any issues, feedback or fixes to julia_roberts@fuhsd.org or reneet@eastside.org.
- These presentations may be used with Julia notebooks based on the same curriculum. The more conceptual lessons (as well as early lessons, before programming is introduced) exist as presentations only.
- Although the Julia language is used, the curriculum can be modified to a language of choice. All the tasks are simple enough to be executed in any language.
- Timing suggestions in the overview guides are based on what I would plan for above-average students who have passed at least Algebra 2 (first three units) or Calc AB (last two units). Students with less prior knowledge or who are slower to learn will require more time. Enthusiasm about coding will also play a significant role in timing. Please let me know at the address above about any significant differences you have noted between reality and the guide.
- 23 weeks is so awkward, isn't it? With some stretching a full-year course could be filled, and with some compression/moving practice problems to homework it could be shoved into a semester (but only for high-performing students). The first "half," comprising units 1-3, is estimated to take about 13 weeks and the second half would take about 10.