Python I: Intro to Python Modeling

Webinar Overview

Part 1. Introduction

An overview of Python, why you should consider using Python and getting started

Part 2. Getting Started

Introductory examples, key concepts, best practices and getting help

Part 3: Extended Example

Practical aspects of modeling, solving, and analyzing solutions

Gurobi and community resources to help you

Part 4. Questions & Answers Session

Part 1. Introduction

Python, Gurobi and Jupyter Notebook

Python in a Nutshell

- Python is powerful **programming language**
  - Very easy to learn
  - Runs on every major operating system
  - Suitable for beginners and professional developers
  - Clean, simple and compact syntax
  - Open Source community: Tons of ready-made libraries and frameworks

- Python is also a great choice for **mathematical modeling**
  - Many popular packages for scientific computing
  - Jupyter Notebooks: Interactive graphical environment
  - Key language features look similar to mathematical notations
  - Language extensibility

Why Python?

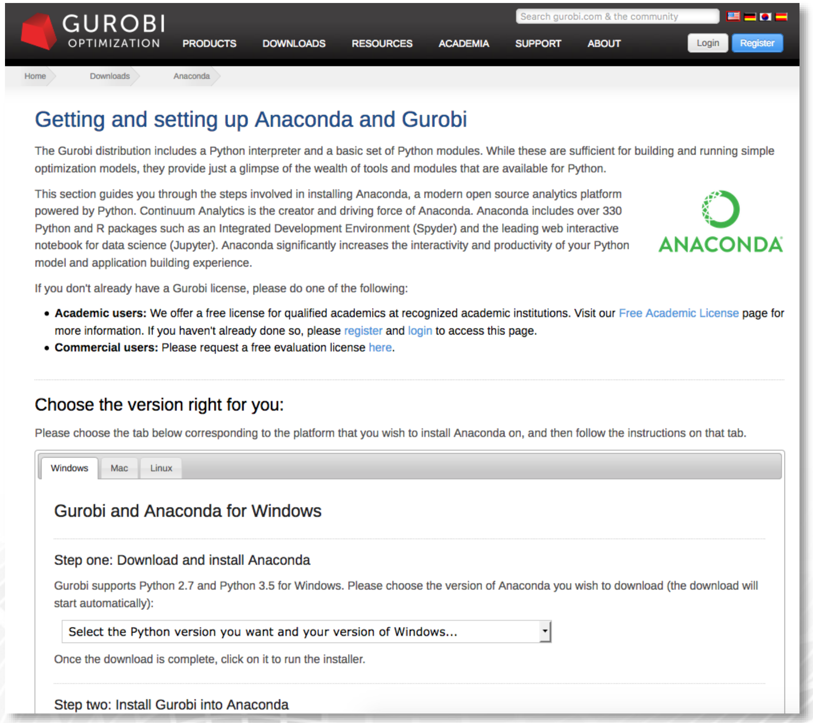
- Different Gurobi customers use different interfaces
  - Python
  - Java
  - C++
  - .NET (C#, Visual Basic, …)
  - MATLAB,
  - R
  - ...

- Gurobi is committed to all of these interfaces

- This presentation focuses on Python since
  - Python is simple to learn for new Gurobi users
  - Python has special features that make it easy to build and maintain optimization models

Python Installation

- The Gurobi Optimizer installation includes a small Python 2.7 distribution with a basic set of packages.
- Gurobi Optimizer works with multiple Python distributions (including http://www.python.org).
- In this presentation, we use **Anaconda Python**
  - Download from http://www.gurobi.com/downloads/get-anaconda
  - Windows, Mac, Linux; 32/64-bit
  - All basic packages in the default installation
  - Easy-to-use package management (**conda**)
  - Gurobi provides/maintains Anaconda packages
  - Recommended version: Python 2.7
  - Gurobi will support Python 3.6 in the next release



Basic Concepts

- Python is an "interpreted language"
  - No need for a manual compiling step before running a program.
  - Performance optimization is done in the background (byte-code compilation/execution).

- Two basic ways to use Python
  - Put your code in a text file (e.g. `example.py`) and run the command "`python example.py`".
  - Start the Python Interactive Shell (command "`python`") and enter commands line by line.

```
C:\Anaconda3\python.exe
Python 3.4.4 |Anaconda 2.5.0 (64-bit)| (default, Feb 16 2016, 09:54:04) [MSC v.1600 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World")
Hello World
>>>
```

- In this webinar we will furthermore use a rich graphical interactive environment instead of the command line shell: **Jupyter Notebooks**.

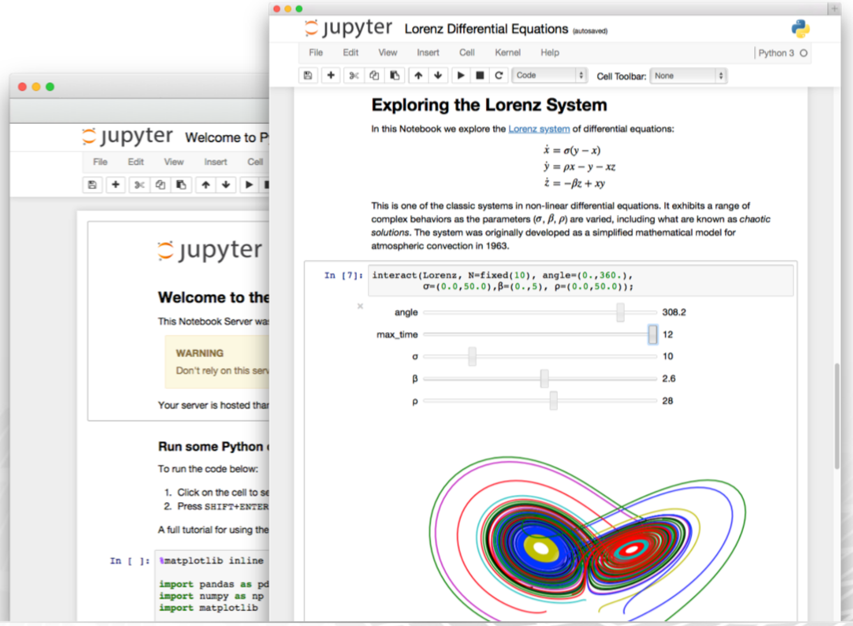
Jupyter Notebook

- The Jupyter Notebook is a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text.

- Free software with an active open source community
- Born out of the IPython Project in 2014
- Already included in Anaconda Python

- Simply change to your working directory and run `jupyter notebook` on the command-line

- Learn more at http://jupyter.org/



Connecting Python and Gurobi

- The Python API is a first-class citizen of Gurobi
- To use Gurobi from Python you need to install the **`gurobipy`** module (included in the Gurobi installation).
- Automatic installation with Anaconda:

```
conda config --add channels http://conda.anaconda.org/gurobi
conda install gurobi
```

```
(webinar) C:\>conda install gurobi
Fetching package metadata .............
Solving package specifications: ..........

Package plan for installation in environment C:\Anaconda3\envs\webinar:

The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    gurobi-7.0.2               |           py27_0        15.1 MB  gurobi

The following NEW packages will be INSTALLED:

    gurobi: 7.0.2-py27_0 gurobi

Proceed ([y]/n)? y

Fetching packages ...
gurobi-7.0.2-p 100% |###############################| Time: 0:00:28 552.34 kB/s
Extracting packages ...
[      COMPLETE      ]|##################################################| 100%
Linking packages ...
[      COMPLETE      ]|##################################################| 100%

(webinar) C:\>
```

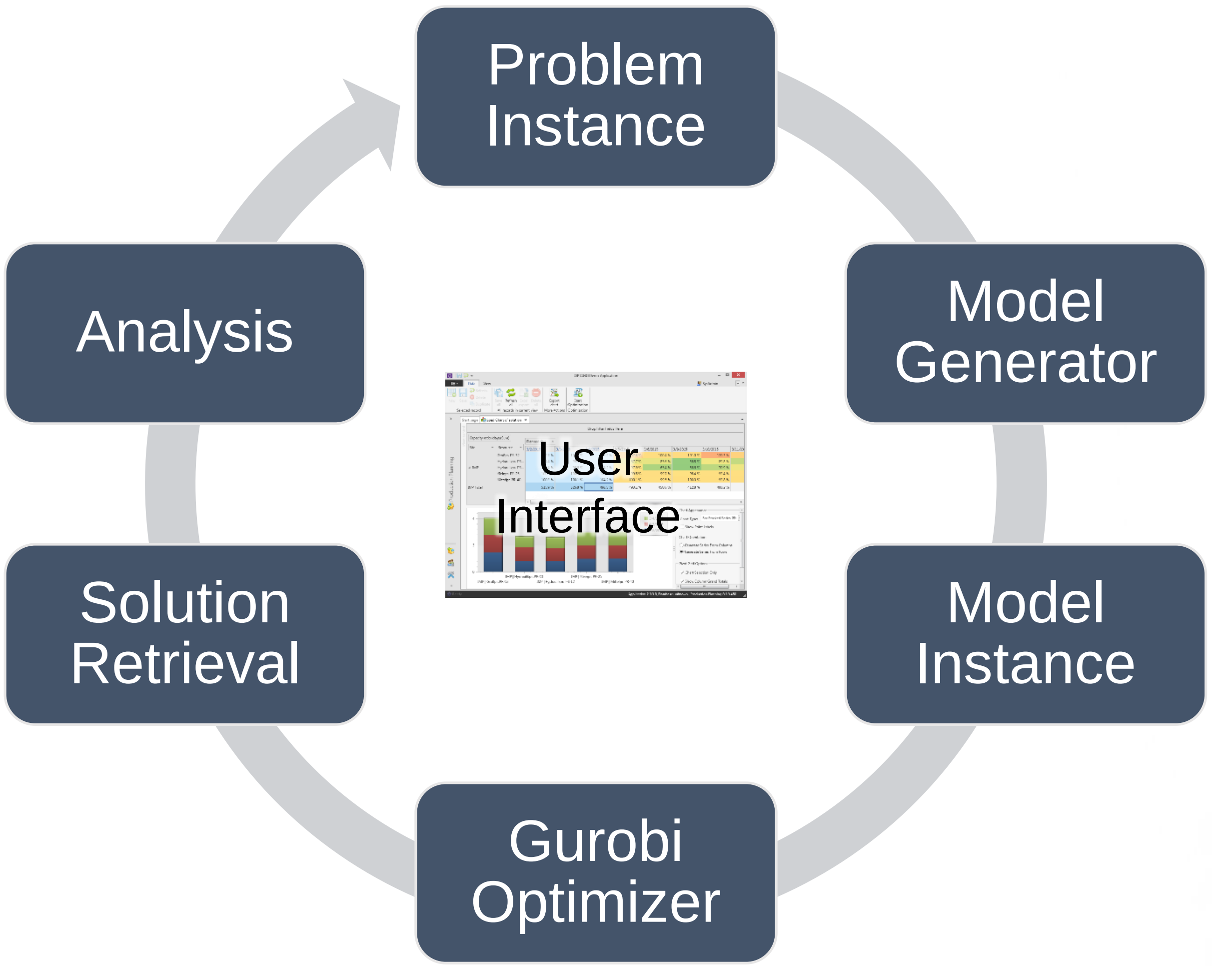
http://www.gurobi.com/documentation/current/quickstart_windows/installing_the_anaconda_py.html

- Manual installation:
  - Change to GUROBI_HOME directory
  - Run `python setup.py install`

```
C:\gurobi702\win64>python setup.py install
running install
running build
running build_py
running install_lib
copying build\lib\gurobipy\gurobipy.pyd -> C:\Anaconda3\Lib\site-packages\gurobipy
copying build\lib\gurobipy\__init__.py -> C:\Anaconda3\Lib\site-packages\gurobipy
byte-compiling C:\Anaconda3\Lib\site-packages\gurobipy\__init__.py to __init__.pyc
running install_egg_info
Writing C:\Anaconda3\Lib\site-packages\gurobipy-7.0.2-py2.7.egg-info

C:\gurobi702\win64>
```

Optimization Workflow



- The real-world problem instance is usually derived from a specific planning problem.
- A configurable model generator is used to build the model instance using data sources.
- Gurobi is used to find an optimal solution of the model instance.
- The solution is transferred back to the planning system for further analysis.
- The cycle repeats until a satisfying real-world solution has been found.

Image: http://optano.net/en/production-planning/

Part 2. Getting Started

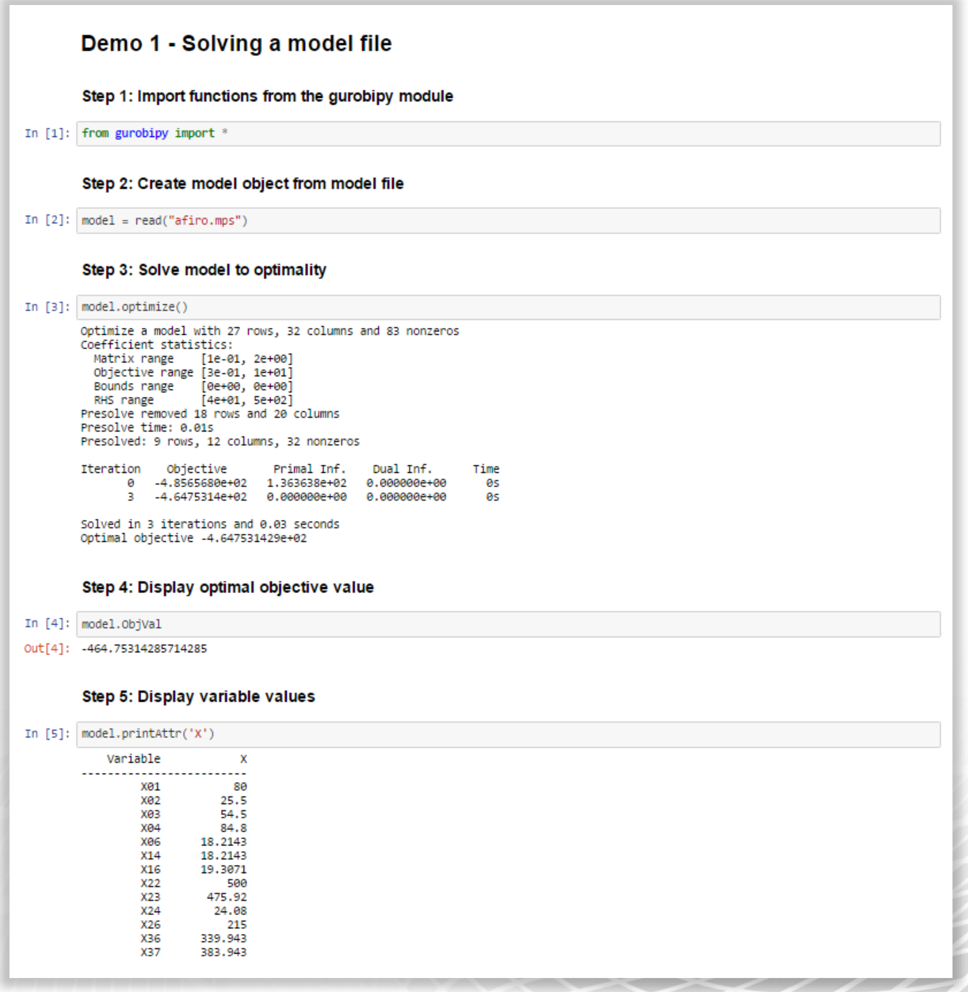
Building and Solving Your First Models

Live Demo 1: Solving a Model File

- Goals:
  - Make yourself familiar with Jupyter Notebook
  - Solve a model file in MPS format

- Example file: **`afiro.mps`** (included in the examples directory of the Gurobi installation)

- Steps:
  - Import the **`gurobipy`** module
  - Create model object from model file
  - Solve model to optimality
  - Print solution values



Demo 1 - Solving a model file

Step 1: Import functions from the gurobipy module

In [1]:
```
from gurobipy import *
```

Step 2: Create model object from model file

In [2]:
```
model = read("afiro.mps")
```

Step 3: Solve model to optimality

In [3]:
```
model.optimize()
```

```
Optimize a model with 27 rows, 32 columns and 83 nonzeros
Coefficient statistics:
  Matrix range     [1e-01, 2e+00]
  Objective range  [3e-01, 1e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [4e+01, 5e+02]
Presolve removed 18 rows and 20 columns
Presolve time: 0.01s
Presolved: 9 rows, 12 columns, 32 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0   -4.8565680e+02   1.363638e+02   0.000000e+00      0s
       3   -4.6475314e+02   0.000000e+00   0.000000e+00      0s

Solved in 3 iterations and 0.03 seconds
Optimal objective -4.647531429e+02
```

Step 4: Display optimal objective value

In [4]:
```
model.ObjVal
```

Out[4]:
```
-464.75314285714285
```

Step 5: Display variable values

In [5]:
```
model.printAttr('X')
```

```
    Variable            X
-------------------------
         X01           80
         X02         25.5
         X03         54.5
         X04         84.8
         X06      18.2143
         X14      18.2143
         X16      19.3071
         X22          500
         X23       475.92
         X24        24.08
         X26          215
         X36      339.943
         X37      383.943
```

Interactive Demo 1: Details

- The **`gurobipy`** module provides access to all Gurobi Optimizer functionality through Python API
- `read()` is a global function which returns an object of type `Model`
- `Model.optimize()` and `Model.printAttr()` are methods (functions on the `Model` object)
- You can choose any name for the `Model` object variable:

```
model = read('afiro.mps')
model.optimize()
model.printAttr('X')
```

or

```
m = read('afiro.mps')
m.optimize()
m.printAttr('X')
```

Basic Gurobi Concepts

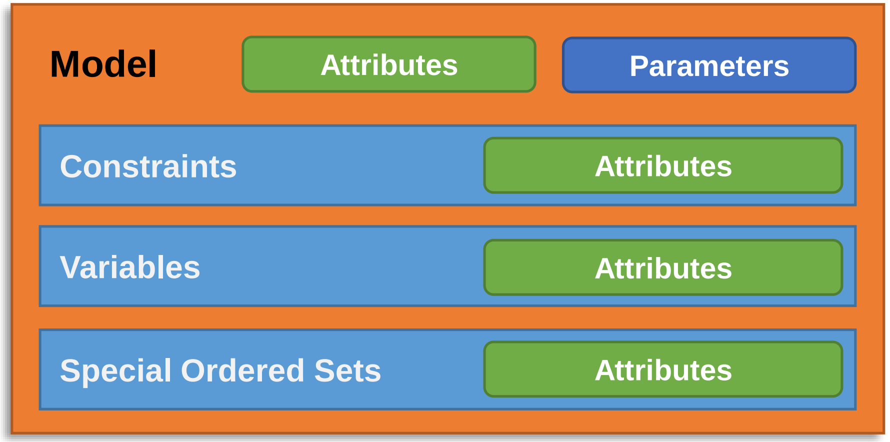
- **Parameters** control the operation of the Gurobi solvers. They must be modified before the optimization begins.

  http://www.gurobi.com/documentation/current/parameters.html

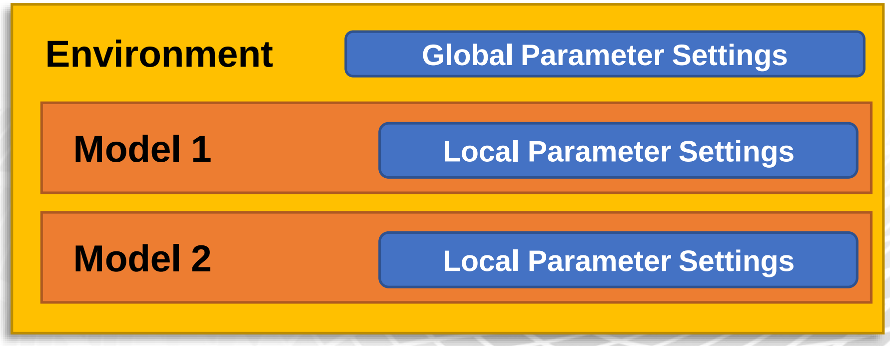
- **Attributes** are the primary mechanism for querying and modifying properties of a Gurobi model.

  http://www.gurobi.com/documentation/current/refman/attributes.html

  Attributes can be associated with
  - the model as a whole (e.g. the objective function value)
  - variables (e.g. lower bounds)
  - constraints (e.g. the right-hand side)
  - SOSs (e.g. IIS membership)



- **Environments** are the containers for models and global parameter settings. When a model is created inside an environment, it creates a local copy of all global parameter settings.
- The Python API automatically provides a default environment.



Python API Fundamentals #1

- Important global functions
  - `setParam()` set (global) parameters for new models in this session
  - `read()` create model object from a model file

- Primary objects
  - `Model` the model
  - `Var` a variable
  - `Constr` a constraint

- Parameter/attribute examples:
  - `setParam('TimeLimit', 60)` Set global parameter
  - `model.TimeLimit = 60` Set local parameter
  - `variable.VarName = 'TotalCosts'` Set variable attribute
  - `constraint.RHS = 0` Set constraint attribute

- Other important methods:
  - `Model.setObjective(expression)` Set objective function
  - `Model.optimize()` Start optimization
  - `Model.reset()` Reset model state

Python API Fundamentals #2

- Model summary information methods
  - `Model.printAttr()` display nonzero attribute values
  - `Model.printStats()` display sizes, coefficient statistics, etc.
  - `Model.printQuality()` display info about solution quality

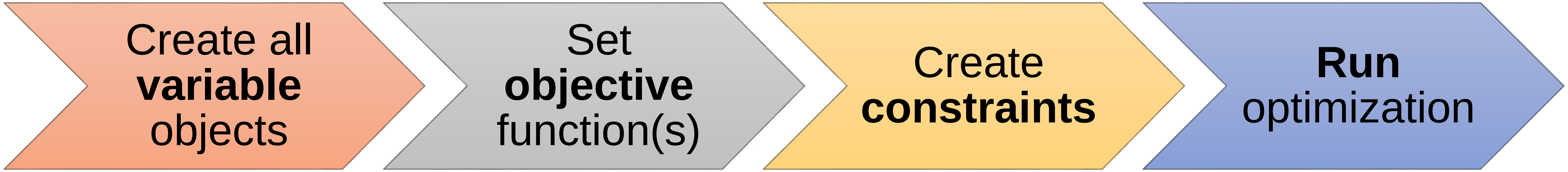
- Model introspection methods
  - `Model.getVars()` get variables as a list of Var objects
  - `Model.getConstrs()` get constraints as a list of Constr objects
  - `Model.getRow()` get LHS expression for a given constraint
  - `Model.getCol()` get list of constraints for a given variable

- Python control statements
  - ```
    for i in <some list>:
      <do something for each i here>
    ```
  - ```
    if <condition>:
      <do something if condition is true here>
    ```

  - Must indent body of each statement (at least one space or tab)

Model Generator Best Practices

- For performance reasons we recommend the following structure when building a model instance with the Gurobi Python API:



`addVar()`
`addVars()`

`setObjective()`

`addConstr()`
`addConstrs()`

`optimize()`

- The Python API supports (linear and quadratic) expressions similar to the mathematical notation:

$$2x + \frac{1}{2}y \leq 10 \quad \Longleftrightarrow$$

```
2*x + (1/2)*y <= 10
```

Creating and Solving Your First Model #1

- Simple example:
  - You want to decide about three activities (do or don‘t do) and aim for maximum value
  - You need to choose at least activity 1 or 2 (or both)
  - The total time limit is 4 hours
    - Activity 1 takes 1 hour
    - Activity 2 takes 2 hours
    - Activity 3 takes 3 hours
  - Activity 3 is worth twice as much as 1 and 2

$$
\begin{aligned}
\max \quad & x + y + 2z \\
\text{s.t.} \quad & x + 2y + 3z \leq 4 \\
& x + y \geq 1 \\
& x, y, z \in \{0, 1\}
\end{aligned}
$$

- This can be modelled as a linear mixed-integer problem
  - Binary variables x,y,z for activities 1,2,3
  - Linear constraint for time limit
  - Linear constraint for condition (1 or 2)

Creating and Solving Your First Model #2

- Open a new Jupyter Notebook
- Follow the Best Practices
  - Create decision variables
  - Set objective function
  - Create linear expressions and use them to create constraints
  - Call optimize()
- Print out results

This model is the mip1 example that you can find for all APIs in the **`examples`** directory of the Gurobi installation.

```
# Create a new model
m = Model()

# Add variables
x = m.addVar(vtype=GRB.BINARY, name="x")
y = m.addVar(vtype=GRB.BINARY, name="y")
z = m.addVar(vtype=GRB.BINARY, name="z")

# Set objective function
m.setObjective(x + y + 2*z, GRB.MAXIMIZE)

# Add constraints
c1 = m.addConstr(x + 2*y + 3*z <= 4)
c2 = m.addConstr(x + y >= 1)

# Solve model
m.optimize()
```

Live Demo 2: Creating and Solving Your First Model

GUROBI OPTIMIZATION

jupyter Demo 2 - Creating and solving your first model Last Checkpoint: 7 minutes ago (autosaved)

File Edit View Insert Cell Kernel Help | Python [webinar]

CellToolbar

Demo 2 - Creating and solving your first model

$$
\begin{aligned}
\max \quad & x + y + 2z \\
\text{s.t.} \quad & x + 2y + 4z \leq 4 \\
& x + y \geq 1 \\
\\
& x, y, z \in \{0, 1\}
\end{aligned}
$$

Step 1: Import functions from the gurobipy module

In [1]:
```
from gurobipy import *
```

Step 2: Create empty model

In [2]:
```
m = Model()
```

Step 3: Create activitiy variables

In [3]:
```
x = m.addVar(vtype=GRB.BINARY, name="x")
y = m.addVar(vtype=GRB.BINARY, name="y")
z = m.addVar(vtype=GRB.BINARY, name="z")
```

Getting Help

- Use the `help()` function with no arguments to get general help

- You can get help on specific Gurobi objects
  - `help(Model)` Gurobi model object (or `help(m)` if `m` is a model)
  - `help(Var)` Gurobi variable object (or `help(v)` if `v` is a variable)
  - `help(Constr)` Gurobi constraint object (or `help(c)` if `c` is a constraint)
  - `help(GRB.attr)` Gurobi attributes

- You can get help on specific functions or methods
  - Ex: `help(read)`
  - Ex: `help(Model.printAttr)`

- You can get help on any parameter
  - Ex: `paramHelp('Method')`

Deciphering Errors

- Errors are raised when something goes wrong
  - Trace specifies where error occurred
  - Last line specifies error type and may provide additional details

- Ex: `IndentationError` raised when block spacing is incorrect (must indent before `if`)

```
for v in model.getVars():
if v.x != 0:
  File "<stdin>", line 2
    if v.x != 0:
     ^
IndentationError: expected an indented block
```

- `<stdin>` corresponds to interactive shell

Part 3. Extended Example

Factory Planning

Iterating in Python and Aggregate Sums

- Loops iterate over collections of elements (list, dictionary, …)
  - Useful when representing the for-all modeling construct
  - Example:

```
for c in cities:
    print c # must indent all loop statements
```

- List comprehension efficiently builds lists via set notation
  - `penaltyarcs = [a for a in arcs if cost[a] > 1000]`
- `quicksum()` is direct replacement for Python's `sum()` function
  - Gurobi feature that is more efficient when working with `Var` objects
  - `obj = quicksum(cost[a]*x[a] for a in arcs)`

- Combining loops with sums is ideal when building constraints

Example: $$\sum_{j \in J} x_{i,j} \leq 5 \ \forall i \in I$$

can be built with

```
for i in I:
  m.addConstr(quicksum(x[i,j] for j in J) <= 5)
```

- Convenient batch creation of variables and constraints has been introduced as part of Gurobi 7.0. Example:

```
m.addConstrs((x.sum(i,'*') <= 5) for i in I))
```

Factory Planning

- Example from our website: http://www.gurobi.com/resources/examples/factory-planning-I

- In production planning problems, choices must be made about how many of **what products to produce using what resources (variables)** in order to **maximize profits or minimize costs (objective function)**, while meeting a **range of constraints**. These problems are common across a broad range of manufacturing situations.

- We will develop the mathematical model, the Python Implementation and a nice tabular output of the result all within a single Jupyter Notebook.

Live Demo 3: Interactive Model Development

GUROBI OPTIMIZATION

Demo 3 - Factory Planning

Source: http://www.gurobi.com/resources/examples/factory-planning-I

Problem Description

A factory makes seven products (Prod 1 to Prod 7) using a range of machines including:

- Four grinders
- Two vertical drills
- Three horizontal drills
- One borer
- One planer

Each product has a defined profit contribution per unit sold (defined as the sales price per unit minus the cost of raw materials). In addition, the manufacturing of each product requires a certain amount of time on each machine (in hours). The contribution and manufacturing time value are shown below. A dash indicates the manufacturing product for the given product does not require that machine.

	PROD 1	PROD 2	PROD 3	PROD 4	PROD 5	PROD 6	PROD 7
Contribution to profit	10	6	8	4	11	9	3
Grinding	0.5	0.7	-	-	0.3	0.2	0.5
Vertical drilling	0.1	0.2	-	0.3	-	0.6	-
Horizontal drilling	0.2	-	0.8	-	-	-	0.6
Boring	0.05	0.03		0.07	0.1	-	0.08
Planing	-	-	0.01	-	0.05	-	0.05

In each of the six months covered by this model, one or more of the machines is scheduled to be down for maintenance and as a result will not be available to use for production that month. The maintenance schedule is as follows:

Month	Machine
January	One Grinder
February	Two Horizontal Drills
March	One borer
April	One vertical drill
May	One grinder and one vertical drill
June	One horizontal drill

There limitations to how many of each product can be sold in a given month. These limits are shown below:

Month	PROD 1	PROD 2	PROD 3	PROD 4	PROD 5	PROD 6	PROD 7
January	500	1000	300	300	800	200	100
February	600	500	200	0	400	300	150
March	300	600	0	0	500	400	100
April	200	300	400	500	200	0	100
May	0	100	500	100	1000	300	0
June	500	500	100	300	1100	500	60

Up to 100 units of each product may be stored in inventory at a cost of $0.50 per unit per month. At the start of January there is no product inventory. However, by the end of June there should be 50 units of each product in inventory.

The factory produces product six days a week using two eight-hour shifts per day. It may be assumed that each month consists of 24 working days. Also, for the purposes of this model, there are no production sequencing issues that need to be taken into account.

What should the production plan look like? Also, recommend any price increases and identify the value of acquiring any new machines.

Model Formulation

Sets

Let T be a set of time periods (months), where $t_0 \in T$ is the first month and $t_e \in T$ the last month.

Let P be a set of products and M be a set of machines.

Parameters

- For each product $p \in P$ and each type of machine $m \in M$ we are given the time $f_{p,m}$ (in hours) the product $p \in P$ needs to be manufactured on the machine $m \in M$.
- For each month $t \in T$ and each product $p \in P$ we are given the upper limit on sales of $l_{t,p}$ for that product in that month.
- For each product $p \in P$ we are given the profit k_p.
- For each month $t \in T$ and each machine $m \in M$ we are given the number of available machines $q_{t,m}$.
- Each machine can work g hours a month.
- There can be z products of each type stored in each month and storing costs r per product per month occur.

The capacity constraints ensure that per month the time all products needs on a certain kind of machines is lower or equal than the available hours for that machine in that month multiplied by the number of available machines in that month. Each product needs some machine hours on different machines. Each machine is down in one or more months due to maintenance, so the number of available machines varies per month. There can be multiple machines per machine type.

$$\sum_{p \in P} f_{p,m} \cdot b_{t,p} \leq g \cdot q_{t,m} \quad \forall t \in T, \forall m \in M$$

Python Implementation

Import `gurobipy` module:

In [1]:
```
from gurobipy import *
```

Data definition

Define sets P, M and T:

In [2]:
```
products = ["Prod1", "Prod2", "Prod3", "Prod4", "Prod5", "Prod6", "Prod7"]
machines = ["grinder", "vertDrill", "horiDrill", "borer", "planer"]
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun"]
```

Values for parameter k_p (profit contribution per product $p \in P$):

In [3]:
```
profit_contribution = { "Prod1" : 10, "Prod2" : 6, "Prod3" : 8, "Prod4" : 4,
                        "Prod5" : 11,  "Prod6" : 9, "Prod7" : 3 }
```

Use our Gurobi Online Resources

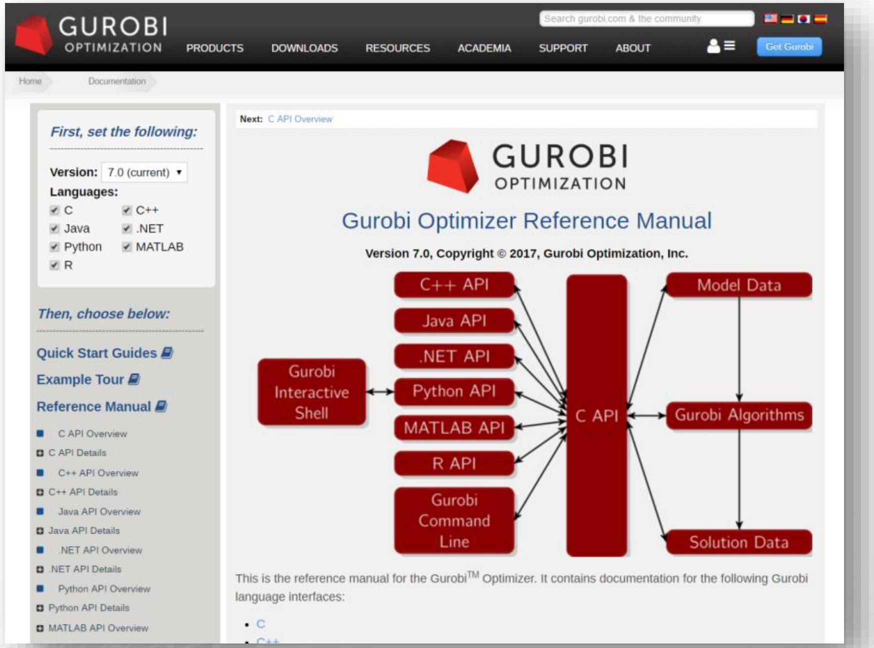
- The Gurobi Reference Manual is freely available at

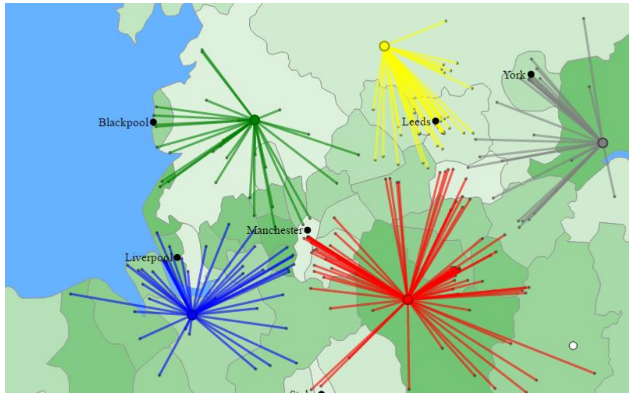
  http://www.gurobi.com/documentation/current/refman/index.html

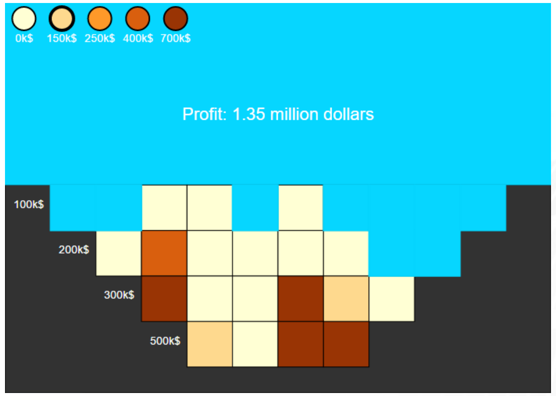
  It contains detailed descriptions of our algorithms and APIs.

- We provide API examples for all major programming languages as well as interactive examples at

  http://www.gurobi.com/resources/examples/example-models-overview







Outlook

- Interactive examples for commonly faced business problems
  - http://www.gurobi.com/resources/examples/example-models-overview
- Learn the powerful new Python features of Gurobi 7
  - Easily create multi-indexes variables using `Model.addVars()`
  - Build advanced expressions and constraints with `Model.addConstrs()`
- More control over when Gurobi environments are created/released
  - Default environment not created until first used
  - Released with new `disposeDefaultEnv()` method
- Build you own model!
- Connect with the community in our discussion group (http://groups.google.com/group/gurobi)

Thank you for joining us

- Please register at gurobi.com and then visit http://www.gurobi.com/downloads/get-anaconda to try Gurobi and Python for yourself

- There will be two more Python webinars
  - Python II: Advanced Algebraic Modeling with Python and Gurobi
  - Python III: Optimization and Heuristics
  - Learn more at http://www.gurobi.com/company/events/webinars-python

- For questions about pricing please contact either sales@gurobi.com or sales@gurobi.de.

- A recording of the webinar will be available in roughly one week.