Installing Python 3.6 via Miniconda

We will be using Python 3.6 in this class. The easiest way to install this is through the Anaconda distribution, which is platform independent so avoids many potential tricky IT issues. However, the Anaconda installer takes much space and a while to download, since it installs 150 packages at once. I would recommend instead using the **Miniconda** installer, which installs the basic Python environment, and you can later install the specific packages you need. (However, if you have a lot of disk space and don't mind the large download, you can skip the rest of this section and following the full Anaconda installation instructions here.)

To install using Miniconda (skip this if you already installed the full Anaconda by following instructions above), download the installer for your Operating System (OS) here: https://conda.io/miniconda.html.

Then run the installer and keep the default options. (Choose the default location for copying files and to add this location to your "PATH" environment variable.)

- For windows, simply double click the .exe file downloaded, and choose the two options above.
- For Mac or Linux, you would open a Terminal and type in the following, replacing the capitalized terms with the correct location. Then, go through the user agreement and type yes for everything.

```
cd PATH_WHERE_YOU_DOWNLOADED_THE_FILE_STARTING_WITH_SLASH
bash NAME_OF_FILE_YOU_DOWNLOADED
```

For example, depending on where you downloaded the installer, this might be

```
cd /home/user/pengshi/Downloads/
bash Miniconda3-latest-MacOSX-x86_64.sh
```

After this, launch the **Anaconda Prompt** if you are using Windows and type the following command to install the packages we will frequently use. If you are using Mac or Linux, you would have to close the terminal where you installed and start a new one, then type in the following commands into the terminal.

```
conda install numpy scipy matplotlib pandas jupyter ipython spyder
```

In the above command, the first two words are standard syntax. Starting from `numpy`, we are listing a set of Python packages to install. Each package is separated by a space. This is the Syntax for installing future Python packages. The first four packages, numpy, scipy, matplotlib and pandas and bread and butter for manipulating data sets. Jupyter is what is used to create notebooks similar to the one you are reading now. IPython is an interactive shell that is better than the original "Python" shell. Spyder is a Interactive Development Environment (IDE), which is an editor for code, that I recommend (similar to RStudio for R), but there are many IDEs that you can use instead. Here is a webpage suggesting the pros and cons of popular IDEs for Python.

To start running Python, type one of `ipython`, `spyder`, or `jupyter notebook` into the Anaconda prompt (on Windows) or any terminal (on Linux or Mac). The first activates the IPython shell. You can begin typing Python commands and seeing the output. The second is the graphical interface. The third launches a web browser. And you can start a new Python 3 shell and the interface looks like what you see now. You can test your installation by running the code.

```
[2]: print ("My first Python code")
```

```
My first Python code
```

Installing Graphviz (Optional)

This is a software package you need in order to visualize decision trees using the code provided in the course notes.

In the Anaconda prompt in Windows and in any termina in Mac or Linux, type the following set of commands:

```
conda install -c conda-forge graphviz python-graphviz
```

To test whether this works, go into IPython or Jupyter notebook and type the following, and see if there's an error message.

```
from graphviz import Graph
g=Graph('test')
g.node('a')
g.node('b')
g.edge('a','b')
g.view()
```

Installing Gurobi

Later in the course, we will need to install Gurobi, which is the most powerful software in the world currently for solving large-scale linear optimization and mixed-integer optimization problems. (We will learn about these topics in the second half of the course.)

Step 1 (Installing Gurobi via conda):

In Anaconda prompt (Windows) or in a terminal (Mac or Linux), type:

```
conda install -c http://conda.anaconda.org/gurobi gurobi
```

Type yes for everything to install.

Step 2 (Obtaining license):

To use Gurobi, you need to obtain a **free academic license** for your computer. To do so, following steps 2-4 of the instructions on the following webpage, under the heading "***To obtain a free named-user academic license***": http://www.gurobi.com/academia/for-universities. Note that after registering on the Gurobi website using your USC.edu email address, you also need to run the command `grbgetkey` followed by the license key, as indicated in step 4 of the above instructions. **You must be using the USC network while obtaining the license key, as this is an academic license.** After obtaining the license, you can use Gurobi anywhere you want (it does not require Internet connection) for one year.

Step 3: (Test import):

In Anaconda prompt (Windows) or terminal (Mac or Linux), start a Python shell using

```
python
```

Once the shell starts, type the following and see it there is an error.

```
import gurobipy
```

If there is no error, then you move on to the following session to test solving a LP in Gurobi within Jupyter notebook. If there is an error, try to Google the error message and "Gurobi" together to see if you can figure out what's wrong. Come to office hour (Friday 3:30-5:30pm) or email the professor for help.

If the error message is `Segmentation Fault` and you are using a Mac, go to the following step.

Step 4: (Only for Mac users who obtained the Segmentation Fault Error above)

Unfortunately, there is a bug with the latest version of Anaconda for Python 3, which makes it unable to use Gurobi. This bug has not been resolved since Fall 2017. See discussion here.

The simplest work around unfortunately is to create a new Anaconda environment that uses Python 2.7 and install Gurobi in that environment. If you follow the instructions below, you can avoid the Python 2.7 conflicting with the Python 3.6 that you already have. **DO NOT DOWNLOAD ANACONDA 2.7 FROM THE WEBSITE AND INSTALL THE PACKAGE THAT WAY. IT WILL MESS UP YOUR SYSTEM.**

Creating a new environment using Python 2.7: Type in a terminal the following three lines, answering `yes` whenever it prompts you. The first line adds the Gurobi package to the conda search path. The second creates a new environment called `py27` (feel free to change the name), with Python version 2.7, as well as the packages you need, which are jupyter, pandas, scipy, numpy, and gurobi. The third line activates the environment you created. The last line updates a `pyzmq` package (otherwise jupyter notebook would have a `IOLoop` error).

```
conda config --add channels http://conda.anaconda.org/gurobi
conda create -n py27 python=2.7 jupyter pandas scipy numpy gurobi
source activate py27
conda install -c conda-forge pyzmq
```

If the first line results in a "Permission denied error", then you can replace the above three lines with

```
conda create -n py27 python=2.7 jupyter pandas scipy numpy
source activate py27
conda install -c conda-forge pyzmq
conda install -c http://conda.anaconda.org/gurobi gurobi
```

Working with environments: When you start a new terminal, you are in the default environment with Python 3.6 without gurobi. When you activate the environment using `source activate py27`, the prefix `py27` will appear in your terminal, showing that you are now working in the `py27` environment. You must start Jupyter notebook when you see this, by typing `jupyter notebook` in the terminal, otherwise you will be using the Python 3 Jupyter (not the Python 2 Jupyter), and Guorbi will not work. To go back to the original Python 3.6 environment (without Gurobi), you can type

```
source deactivate
```

Testing Gurobi again in Python 2.7: Now, activate the `py27` environment by typing `source activate py27` in terminal, and go back to step 3 to test. Then proceed to the next section to test a bigger chunk of code. (When you test the LP in the next section, make sure you start jupyter notebook in the new environment, by typing:

```
source activate py27
jupyter notebook
```

Email the professor if you have problems.

Testing Gurobi

Run the following code and make sure you get the same output. The code solves the Gemstone Tool Company LP from DMD section 7.3.

Maximize :	$130W + 100P$	
subject to:	$1.5W + P \leq 27$	(Constraint on steel)
	$W + P \leq 21$	(Constraint on molding)
	$0.3W + 0.5P \leq 9$	(Constraint on assembly)
	$W \leq 15$	(Limit on demand for wrenches)
	$P \leq 16$	(Limit on demand for pliers)
	$W, P \geq 0$	(Non-negativity)

[13]:
```
import gurobipy as grb

mod=grb.Model()                          # Defining model
W=mod.addVar(lb=0,name='W')             # Defining variable W with lower bound (lb) 0.
P=mod.addVar(lb=0,name='P')
c={}
c[1]=mod.addConstr(1.5*W+P<=27,'Steel')  # Adding each constraint and naming it (the na
c[2]=mod.addConstr(W+P<=21,'Molding')
c[3]=mod.addConstr(0.3*W+0.5*P<=9,'Assembly')
c[4]=mod.addConstr(W<=15,'W-Demand')
c[5]=mod.addConstr(P<=16, 'P-Demand')

# Setting the objective function and specifying that we are maximizing it
mod.setObjective(130*W+100*P,sense=grb.GRB.MAXIMIZE)

# Calling the Gurobi solver to solve the model we inputed.
mod.optimize()

print('\n-------RESULTS--------------')
print('Optimal Objective: ',mod.ObjVal)
print ('Optimal W=',W.x)
print ('Optimal P=',P.x)
print('Shadow prices')
for ind in c:
    print ('\tconstraint c[{0:d}]: {1:.2f}'.format(ind,c[ind].PI))
```

```
Optimize a model with 5 rows, 2 columns and 8 nonzeros
Coefficient statistics:
  Matrix range     [3e-01, 2e+00]
  Objective range  [1e+02, 1e+02]
  Bounds range     [0e+00, 0e+00]
  RHS range        [9e+00, 3e+01]
Presolve removed 2 rows and 0 columns
Presolve time: 0.03s
Presolved: 3 rows, 2 columns, 6 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.7000000e+03   3.250000e+00   0.000000e+00      0s
```

```
       3    2.4600000e+03   0.000000e+00   0.000000e+00      0s

Solved in 3 iterations and 0.05 seconds
Optimal objective  2.460000000e+03

-------RESULTS--------------
Optimal Objective:  2460.0
Optimal W= 12.0
Optimal P= 9.0
Shadow prices
        constraint c[1]: 60.00
        constraint c[2]: 40.00
        constraint c[3]: 0.00
        constraint c[4]: 0.00
        constraint c[5]: 0.00
```