Best Practices Using the IBM ILOG CPLEX Python 3 API
Agenda

- Background
- Why should I use the CPLEX Python API?
- Why Python 3?
- Python 3 Considerations
- Using Multiple Versions Simultaneously
- Example with matplotlib
- Q&A
Background

- The CPLEX Optimizer is a high-performance mathematical programming solver for LP, MIP, and QP
- Available commercially since 1997
- Implemented in the C programming language
- Python interface available since 2009 (v12.0)
- Has kept up with the Python 2.x line (e.g., 2.4, 2.5, … , 2.7)
- Currently, v12.6.0 supports Python 2.6 and/or 2.7 depending on platform
- For more, see recent “CPLEX Optimization Modeling using Python” talk
- **New** support for Python 3.4 in upcoming v12.6.1 release!
Why should I use the CPLEX Python API?

- Python is a high-level programming language that emphasizes readability and ease-of-use
- Available on many platforms
- Open Source
- Python is popular (according to which index you look at, it can range from 4th to 8th)
- Used in the scientific community (e.g., Scientific Computing, Bioinformatics, GIS and Mapping, etc.)
- Active User Communities (e.g., mailing lists, stackoverflow, github, etc.)
- Many excellent libraries / Interoperability (e.g., SQLAlchemy, Flask, matplotlib)
- As an alternative to the compiled languages
  - Get things done in very few lines of code
  - Also great for testing (even C libraries)
  - Quick feedback loop
- The reference implementation (CPython) has nice support for extensions written in C
  - Makes it nice for hooking up with CPLEX
  - You get the performance benefits
Why should I use the CPLEX Python API? continued

- The CPLEX Python API consists of a thin layer on top of the C Callable Library
  - Makes it more friendly and natural to use from Python
  - If you're familiar with the C Callable Library it should be fairly obvious how the C functions are mapped to the Python objects and methods
  - Once you're familiar with the Python API you can use it in place of the CPLEX interactive to read model files, query, and solve them
Why Python 3?

- Python 2 introduced in 2000 (quite old)
- Python 2.x is essentially bug-fix only at this point
- However, Python 2.x still installed by default on many (all?) Linux distros and on Mac OS
- Python 3 introduced in 2008 (been around for awhile)
  - Some of the improvements backported to Python 2.6 and 2.7
- Adoption of Python 3 has taken (is taking) a long time, but seems to be gaining traction (i.e., many libraries support it)
- Increasing requests from customers and forum users
- Good tools/libraries for dealing with the differences between Python 2 and 3 (more on this to come)
Python 3 Considerations

- **Major differences between Python 2 and Python 3:**
  - Better Unicode support
  - Unification of int/long
  - print function
  - Integers using “true” division
  - Functions like range() return an iterable
- **Python 2 code may or may not run on Python 3. It may or may not give different results.**
- **One code base or two?**
  - 2to3
  - six/modernize/future
- **Does my favorite Python library work on Python 3?**
  - E.g., Python modeling languages such as pyomo and puLP do support Python 3
Python 3 Considerations continued

Example:  Python 2.x (won't work with Python 3.x):
for i in xrange(3):
    print u'Hello, World'
print 'floor div:', 3 / 2
print 'true div:', 3 / float(2)

Python 3.x (will run in Python 2.x, but diff results):
for i in range(3):
    print('Hello, World')
print('floor div:', 3 // 2)
print('true div:', 3 / 2)

Python >=2.6 or 3.x (works in either, same results):
from __future__ import division
from __future__ import print_function
import six
from six.moves import range
for i in range(3):
    print(six.text_type('Hello, World'))
print('floor div:', 3 // 2)
print('true div:', 3 / 2)
Python 3 Considerations continued

- We use one code base for the CPLEX Python API
- Whether you’re using it from Python 2 or 3, it should behave and look the same
Using Multiple Versions Simultaneously

- When using a single version of the CPLEX Python API, the preferred method is to use `setup.py`
  - `$ pythonX.X setup.py install`
  - After doing this, you can import `cplex`, and away you go
- If you need to support multiple versions of CPLEX, use `PYTHONPATH`
  - `$ PYTHONPATH=<COSInstallDir>/cplex/python/VERSION/platform pythonX.X yourscript.py`
- For fine-grained control try `pyvenv` (virtualenv for Python 2.x)
  - Allows the programmer to use many different and isolated Python environments on one system
  - Doesn't pollute the system installation
  - Can pin exact versions of dependencies
  - Particularly useful for web applications
Using Multiple Versions Simultaneously continued

$ pyenv-3.4 cplexenv # create it
$ cd cplexenv
$ source ./bin/activate # activate the virtual environment
(cplexenv) $ which python
/path/to/cplexvenv/bin/python
(cplexenv) $ python --version
Python 3.4.2
(cplexenv) $ wget https://bootstrap.pypa.io/ez_setup.py
(cplexenv) $ python ./ez_setup.py
(cplexenv) $ easy_install pip
(cplexenv) $ cd <COSInstallDir>/cplex/python/3.4/x86-64_linux
(cplexenv) $ pip install .
(cplexenv) $ cd /path/to/cplexenv
(cplexenv) $ python
Python 3.4.2 (default, Oct 11 2014, 17:14:25)
[GCC 4.6.3] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cplex
>>> print(cplex.Cplex().get_version())
12.6.1.0
>>> quit() # or Ctrl+D
(cplexenv) $ deactivate
$ # we're back to our regular environment
Example with matplotlib

Why matplotlib? What is matplotlib?

- To give an example of the wonderful Python libraries that are available
- “matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.” (see http://matplotlib.org for more)
- It allows you to create pretty pictures easily
- MATLAB-like interface
Example with matplotlib continued

How to install matplotlib?

- To install, continuing with our cplexvenv from before:
  - (cplexvenv) $ pip install matplotlib

NOTES:

- Pretty amazing when it works (fetches source code, compiles if necessary, installs, handles dependencies)
- matplotlib requires some C libraries to be installed on the system (e.g., libfreetype-dev, libpng-dev, etc.)
- Depending on platform, it may be easier to use the (pre-compiled) official installer, possibly combined with pyvenv --system-site-packages
import cplex
import sys
from pylab import *

class MyCallback(cplex.callbacks.MIPInfoCallback):
    def __call__(self):
        if self.has_incumbent():
            self.incobjval.append(
                self.get_incumbent_objective_value())
            self.bestobjval.append(
                self.get_best_objective_value())

def main():
    cpx = cplex.Cplex()
    cb = cpx.register_callback(MyCallback)
    cb.incobjval, cb.bestobjval = [], []
    cpx.read(sys.argv[1])
    cpx.solve()
    # plot obj value
    size = len(cb.incobjval)
    plot(range(size), cb.incobjval)
    plot(range(size), cb.bestobjval)
    # save to PNG file
    savefig('cpxplot.png')

if __name__ == '__main__':
    main()
Example with matplotlib continued

(cplexvenv) $ python cpxplot.py caso8.mps
Example with matplotlib continued

(cplexvenv) $ python cpxplot.py noswot.mps
Summary

- Why Python is compelling
- Differences between Python 2 and 3, and how to handle them
- The CPLEX Python API can now be used in Python 3
- Using the CPLEX Python API itself, in either version, should behave and look the same
- Working with multiple versions of the CPLEX Python API
- Example of Python interoperability with matplotlib
  - One of the most compelling reasons to use Python is that you can lean on existing libraries to do almost anything
Resources

- CPLEX Optimizer


- Announcement Letter for IBM ILOG CPLEX Optimization Studio V12.6.1

- CPLEX Python API Reference Manual
  - https://ibm.biz/BdEfzj

- Python
  - https://www.python.org/

- Matplotlib
  - http://matplotlib.org/
Legal Disclaimer

• © IBM Corporation 2014. All Rights Reserved.
• The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.
• References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.
• If the text contains performance statistics or references to benchmarks, insert the following language; otherwise delete:
Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.
• If the text includes any customer examples, please confirm we have prior written approval from such customer and insert the following language; otherwise delete:
All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.
• If you reference Intel® and/or any of the following Intel products in the text, please mark the first use and include the following; otherwise delete:
Intel, Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
• If you reference Microsoft® and/or Windows® in the text, please mark the first use and include the following, as applicable; otherwise delete:
Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
• If you reference Adobe® in the text, please mark the first use and include the following; otherwise delete:
Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
• If you reference Java™ in the text, please mark the first use and include the following; otherwise delete:
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
• If you reference UNIX® in the text, please mark the first use and include the following; otherwise delete:
UNIX is a registered trademark of The Open Group in the United States and other countries.
• If you reference Linux® in your presentation, please mark the first use and include the following; otherwise delete:
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Other company, product, or service names may be trademarks or service marks of others.
• If the text/graphics include screenshots, no actual IBM employee names may be used (even your own), if your screenshots include fictitious company names (e.g., Renovations, Zeta Bank, Acme) please update and insert the following; otherwise delete: All references to [insert fictitious company name] refer to a fictitious company and are used for illustration purposes only.
Backup Material

- Using the CPLEX Python API Interactively
Using the CPLEX Python API Interactively

- The documentation is generated from the source code
  - You can use help() to read it
- You can introspect on the fly using dir()
  - To find parameters
  - To find the values in a “constant class” (e.g., Cplex.problem_type)
- Use the readline module for tab completion
  - Or use IDLE (e.g., from Windows)
- Run a script and drop into the Python interactive (use -i)
  - E.g., Build a model and query it (get_stats(), get_histogram())
Using the CPLEX Python API Interactively continued

- Example:

```python
>>> import cplex
>>> help(cplex)
Help on package cplex

NAME
cplex — The CPLEX Python API

DESCRIPTION
This package contains....
```
Using the CPLEX Python API Interactively continued

>>> cpx = cplex.Cplex()

>>> dir(cpx)
['MIP_starts', 'SOS', ..., 'advanced', 'cleanup', 'conflict',
'copy_vmconfig', 'del_vmconfig', 'feasopt', 'get_dettime',
'get_num_cores', 'get_problem_name', 'get_problem_type',
'get_stats', 'get_time', 'get_version', 'get_versionnumber',
'has_vmconfig', 'indicator_constraints', 'linear_constraints',
'objective', 'order', 'parameters', 'populate_solution_pool',
'presolve', 'problem_type', 'quadratic_constraints', 'read',
'read_copy_vmconfig', 'register_callback', 'set_error_stream',
'set_log_stream', 'set_problem_name', 'set_problem_type',
'set_results_stream', 'set_warning_stream', 'solution', 'solve',
'start', 'unregister_callback', 'variables', 'write']
Using the CPLEX Python API Interactively continued

```python
>>> dir(cpx.parameters)
[...'advance', 'barrier', 'clocktype', 'conflict',
'dettimelimit', 'distmip', 'emphasis', 'feasopt', 'get_changed',
'help', 'lpmethod', 'mip', 'output', 'parallel', 'preprocessing',
'qpmethod', 'randomseed', 'read', 'read_file', 'reset', 'sifting',
'simplex', 'solutiontarget', 'threads', 'timelimit', 'tune',
'tune_problem', 'tune_problem_set', 'tuning_status', 'workdir',
'workmem', 'write_file']
```
Using the CPLEX Python API Interactively continued

```python
>>> import readline
>>> readline.parse_and_bind('tab: complete')
>>> cpx.get # hit tab twice
  cpx.get_dettime(     cpx.get_problem_type(     cpx.get_version(      cpx.get_num_cores(     cpx.get_stats(                    cpx.get_versionnumber(     cpx.get_problem_name(     cpx.get_time(      
```
Using the CPLEX Python API Interactively continued

Assume we have the following saved as loadmodel.py:

```python
import sys
import cplex
cpx = cplex.Cplex()
cpx.read(sys.argv[1])
```

Now, we can run like this (then inspect, solve, whatever):

```
$ python -i loadmodel.py caso8.mps
Specified objective sense: MAXIMIZE
Selected objective name: COST
Selected RHS name: RHS
Selected bound name: BOUNDS
>>> print(cpx.get_stats())
Problem name : src/caso8.mps
Objective sense : Maximize
```