Stupid Python Tricks

Kevin Adler - kadler@us.ibm.com - @kadler_ibm
Background
What is it?

“Python is a clear and powerful object-oriented programming language, comparable to Perl, Ruby, Scheme, or Java.”

- Python Wiki

- Elegant syntax
- Easy to use
- Easy to extend
- Embeddable
- Powerful
- Popular
The Zen of Python

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.
- Special cases aren't special enough to break the rules.
- Although practicality beats purity.
- Errors should never pass silently.
- Unless explicitly silenced.
- ... and more at https://www.python.org/dev/peps/pep-0020/
Why Use Python?
High Level Language

- Built-in regular expression support
- No compiling needed
- Great at data storage and manipulation
  - Arrays
  - Hash maps
  - List comprehensions
- Easy to build web applications
Lots of Tools in the Toolbox

• Got a problem? Somebody’s probably solved it already
• Rich standard library built in
• Additional packages on the Python Package Index (PyPI)
  – Over 125,000 projects available
• What tools available?
  – Data parsing: CSV, XML, JSON, HTML, Excel, ...
  – Internet Protocols: HTTP, FTP, TELNET, SMTP, POP3
  – Web services: REST, SOAP, XML-RPC, JSON-RPC, ...
  – Web service wrappers: Twitter, Jenkins, GitHub, ...
  – Message Queuing
  – Image manipulation
  – Data analytics
  – Database access
  – Web application serving
Why Else?

• Simple, straightforward language
• People know it!
  – used heavily in the industry
  – taught in Academia
Who is using Python?
Web Sites Using Python
Python GUI Applications: Virtual Machine Manager

![Image of Virtual Machine Manager interface]

<table>
<thead>
<tr>
<th>Name</th>
<th>CPU usage</th>
<th>Host CPU usage</th>
<th>Memory usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>localhost (QEMU)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f16 Running</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f17 Saved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f18 Paused</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f18-q35 Shutoff</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f19 Shutoff</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f19-32 Shutoff</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f19-migrate Shutoff</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Python GUI Applications: Orange

https://orange.biolab.si/
Other Applications Using Python

WebSphere® Application Server

CIVILIZATION IV

EVE ONLINE

LibreOffice - The Document Foundation
Raspberry Pi

By Evan-Amos - Own work, Public Domain,
https://commons.wikimedia.org/w/index.php?curid=56262833
Raspberry Pi

https://www.raspberry-pi.org/blog/pioneers-summer-camp-2017/raspberry-pi-pioneers-at-google-kings-cross-28717-8/
Icon Explanation

Included with Python (batteries included)
Available from PyPI (batteries not included)
Don’t Copy That Floppy!

- Don’t try to cut and paste these examples
  - Python indentation may mess you up
- Solution: Download them from my GitHub repo
Sending Files as Email

- Built in support for sending email
  - SMTP, ESMTP, LMTP protocols
  - TLS/SSL support
  - Authentication support

- Documentation: https://docs.python.org/3/library/smtplib.html
Sending Files as Email

```python
from sys import argv
import smtplib
from email.mime.text import MIMEText

smtp = smtplib.SMTP('smtp.example.com')

for arg in argv[1:]:
    with open(arg) as file:
        msg = MIMEText(file.read())
        msg['Subject'] = arg
        msg['From'] = 'sysadmin@example.com'
        msg['To'] = 'bugwriter@example.com'
        smtp.send_message(msg)

smtp.quit()
```
Sending file attachments as email

```python
from sys import argv
import smtplib
from email.mime.text import MIMEText
from os.path import basename
from email.mime.multipart import MIMEMultipart
from email.mime.application import MIMEApplication

smtp = smtplib.SMTP('smtp.example.com')

msg = MIMEMultipart()
msg['From'] = 'sysadmin@example.com'
msg['To'] = 'bugwriter@example.com'
msg['Subject'] = 'Application Crashed. Fix now!
msg.attach(MimeText('See attached logs.'))
```
Sending file attachments as email

```python
for arg in argv[1:]:
    with open(arg) as file:
        part = MIMEApplication(file.read())
        part['Content-Disposition'] = \
        'attachment; filename="{}"'.format(basename(arg))

        msg.attach(part)

smtp.send_message(msg)
smtp.quit()
```
Dealing with Zip Files

• Read and Write Zip files
  - Get stored file info
  - Extract or add files to zip archives
  - Supports password-encrypted zip files

• Documentation:
  - [https://docs.python.org/3/library/zipfile.html](https://docs.python.org/3/library/zipfile.html)
Writing Zip Files

```python
from zipfile import ZipFile
from io import BytesIO
from sys import argv
# <snip email setup>
zipbuf = BytesIO()
with ZipFile(zipbuf, 'w') as myzip:
    for arg in argv[1:]:
        myzip.write(arg)
zipbuf.seek(0)
part = MIMEApplication(zipbuf.read())
part['Content-Disposition'] = '
    'attachment; filename="logs.zip"
msg.attach(part)
smtplib.send_message(msg)
```
DIY cPYtoimpf

• Built in support for csv reading/parsing & writing
  - Multiple pre-defined output formats
  - Extensible – generate your own format

• Documentation: https://docs.python.org/3/library/csv.html
DIY cPYtoimpf

from csv import writer, QUOTE_NONNUMERIC
import ibm_db_dbi as db2

conn = db2.connect()
cur = conn.cursor()
cur.execute("select cusnum, lstnam, init, cdtlmt from qiws.qcustcdt where cdtlmt > 100")

with open('qcustcdt.csv', 'w', newline='') as file:
    csvf = writer(file, quoting=QUOTE_NONNUMERIC)
    for row in cur:
        csvf.writerow(row)
<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Initials</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>938472</td>
<td>Henning</td>
<td>G K</td>
<td>5000</td>
</tr>
<tr>
<td>839283</td>
<td>Jones</td>
<td>B D</td>
<td>400</td>
</tr>
<tr>
<td>392859</td>
<td>Vine</td>
<td>S S</td>
<td>700</td>
</tr>
<tr>
<td>938485</td>
<td>Johnson</td>
<td>J A</td>
<td>9999</td>
</tr>
<tr>
<td>397267</td>
<td>Tyron</td>
<td>W E</td>
<td>1000</td>
</tr>
<tr>
<td>389572</td>
<td>Stevens</td>
<td>K L</td>
<td>400</td>
</tr>
<tr>
<td>846283</td>
<td>Alison</td>
<td>J S</td>
<td>5000</td>
</tr>
<tr>
<td>475938</td>
<td>Doe</td>
<td>J W</td>
<td>700</td>
</tr>
<tr>
<td>693829</td>
<td>Thomas</td>
<td>A N</td>
<td>9999</td>
</tr>
<tr>
<td>593029</td>
<td>Williams</td>
<td>E D</td>
<td>200</td>
</tr>
<tr>
<td>192837</td>
<td>Lee</td>
<td>F L</td>
<td>700</td>
</tr>
<tr>
<td>583990</td>
<td>Abraham</td>
<td>M T</td>
<td>9999</td>
</tr>
</tbody>
</table>
DIY cPYtoimpf

```python
from csv import writer, QUOTE_NONNUMERIC

# <snip>

def trim_col(s):
    return s.rstrip() if hasattr(s, 'rstrip') else s

with open('qcustcdt.csv', 'w', newline='') as file:
    csvf = writer(file, quoting=QUOTE_NONNUMERIC)
    for row in cur:
        csvf.writerow([trim_col(col) for col in row])
```

© 2016-2018 IBM Corporation
DIY cPYtoimpf

938472,"Henning","G K",5000
839283,"Jones","B D",400
392859,"Vine","S S",700
938485,"Johnson","J A",9999
397267,"Tyron","W E",1000
389572,"Stevens","K L",400
846283,"Alison","J S",5000
475938,"Doe","J W",700
693829,"Thomas","A N",9999
593029,"Williams","E D",200
192837,"Lee","F L",700
583990,"Abraham","M T",9999
Parsing Arguments with Argparse

- Easily define and parse command line arguments
- Very featureful
  - Positional arguments
  - Short and long arguments
  - Convert to int and other types automatically
  - Built-in help text support
- Documentation:
  [https://docs.python.org/3/library/argparse.html](https://docs.python.org/3/library/argparse.html)
Parsing Arguments with Argparse

```python
from argparse import ArgumentParser
from os import system

parser = ArgumentParser(description='HTTP Admin')

parser.add_argument('--action', required=True, 
                    choices=('start', 'stop', 'restart'), 
                    help='Server Action')

parser.add_argument('--server', default='*ALL', 
                    help='Server to act on')

args = parser.parse_args()
```
Parsing Arguments with Argparse

```python
cmd = {
    'start': 'STRTCPSVR',
    'stop': 'ENDTCPSVR',
    'restart': 'STRTCPSVR',
}[args.action]

cl = "{} SERVER(*HTTP) HTTPSVR({})" \
     .format(cmd, args.server)

if args.action == 'restart':
    cl += ' RESTART(*HTTP)'

system('system "{}"'.format(cl))
```
Parsing Arguments with Argparse

args.py -h

usage: args.py [-h] --action {start,stop,restart} [--server SERVER]

HTTP Admin

optional arguments:
- -h, --help show this help message and exit
--action {start,stop,restart} Server Action
--server SERVER Server to act on
Parsing Arguments with Argparse

```
args.py --action start --server GITWEB
TCP1A0F: HTTP server starting.
```
Parsing JSON

- Encode and decode JSON
- Load from file object or string
- Documentation:
  - https://docs.python.org/3/library/json.html
Reading JSON

```python
import ibm_db_dbi as db2
import json

query = "SELECT JSON_OBJECT('name' : lstnam, 'limit' : cdtlmt) AS object FROM qiws.qcustcdt"
cur.execute(query)

for row in cur:
    obj = json.loads(row[0])
    print("{o[name]}: {o[limit]}".format(o=obj))
```
Reading JSON

Henning : 5000
Jones   : 400
Vine    : 700
Johnson : 9999
Tyron   : 1000
Stevens : 400
Alison  : 5000
Doe     : 700
Thomas  : 9999
Williams: 200
Lee     : 700
Abraham : 9999
Using SQLite

• Access the lightweight database from Python
• Useful for applications that support SQLite but not Db2
• Documentation:
Extending SQLite with Python Functions

```python
import sqlite3

def usd_to_btc(usd_m):
    return round(usd_m * 1000000 / 14_289, 2)

c = sqlite3.connect('my.db')
#                      name, #parms, func
conn.create_function('btc', 1, usd_to_btc)
c = conn.cursor()

c.execute("select movie, gross, btc(gross) \
            from mytable")
for row in c:
    print(row)
```
Extending SQLite with Python Functions

# movie, gross ($M USD), gross (BTC)
('Gone with the Wind', 3.44, 240.74)
('Avatar', 3.02, 211.35)
('Star Wars', 2.85, 199.45)
Honey, can you pick up some batteries?
Package Management

- Python has a package manager: pip (pip3)
- Use pip to install packages from the internet
  - Automatically determines dependencies needed
  - Downloads needed packages from the Python Package Index (pypi.python.org)
  - Installs the packages
- upgrade and uninstall packages as well
- pip can also install local packages (wheels)
- No internet access from IBM i? No problem! Check out devpi
Making Text Tables with Ptable

- Generates and displays “ASCII-art” tables
- Can also generate HTML tables
- Installation
  - pip3 install ptable
- Documentation:
  - https://pypi.python.org/pypi/PrettyTable
  - https://github.com/dprince/python-prettytable
- License: BSD 3-clause
Making a text table

```python
from prettytable import PrettyTable
x = PrettyTable()

x.add_column("City", ["Adelaide", "Brisbane", 
                    "Darwin", "Hobart", "Sydney"])

x.add_column("Area", 
              [1295, 5905, 112, 1357, 2058])

x.add_column("Annual Rainfall", 
              [600.5, 1146.4, 1714.7, 619.5, 1214.8])

print(x)
```
Making a text table

```python
from prettytable import PrettyTable
x = PrettyTable()

x.field_names = ("City", "Area", "Annual Rainfall")

x.add_row(("Adelaide", 1295, 600.5))
x.add_row(("Brisbane", 5905, 1146.4))
x.add_row(("Darwin", 112, 1714.7))
x.add_row(("Hobart", 1357, 619.5))
x.add_row(("Sydney", 2058, 1214.8))

print(x)
```
Making a text table

<table>
<thead>
<tr>
<th>City</th>
<th>Area</th>
<th>Annual Rainfall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adelaide</td>
<td>1295</td>
<td>600.5</td>
</tr>
<tr>
<td>Brisbane</td>
<td>5905</td>
<td>1146.4</td>
</tr>
<tr>
<td>Darwin</td>
<td>112</td>
<td>1714.7</td>
</tr>
<tr>
<td>Hobart</td>
<td>1357</td>
<td>619.5</td>
</tr>
<tr>
<td>Sydney</td>
<td>2058</td>
<td>1214.8</td>
</tr>
</tbody>
</table>
Converting database table to text table

```python
from prettytable import from_db_cursor
import ibm_db_dbi as db2

conn = db2.connect()
cur = conn.cursor()
cur.execute("select cusnum, lstnam, cdtlmt, baldue, cdtdue from qiws.qcustcdt")

print(from_db_cursor(cur))
```
Converting database table to text table

<table>
<thead>
<tr>
<th>CUSNUM</th>
<th>LSTNAM</th>
<th>CDTLMT</th>
<th>BALDUE</th>
<th>CDTDUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>938472</td>
<td>Henning</td>
<td>5000</td>
<td>37.00</td>
<td>0.00</td>
</tr>
<tr>
<td>839283</td>
<td>Jones</td>
<td>400</td>
<td>100.00</td>
<td>0.00</td>
</tr>
<tr>
<td>392859</td>
<td>Vine</td>
<td>700</td>
<td>439.00</td>
<td>0.00</td>
</tr>
<tr>
<td>938485</td>
<td>Johnson</td>
<td>9999</td>
<td>3987.50</td>
<td>33.50</td>
</tr>
<tr>
<td>397267</td>
<td>Tyron</td>
<td>1000</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>389572</td>
<td>Stevens</td>
<td>400</td>
<td>58.75</td>
<td>1.50</td>
</tr>
<tr>
<td>846283</td>
<td>Alison</td>
<td>5000</td>
<td>10.00</td>
<td>0.00</td>
</tr>
<tr>
<td>475938</td>
<td>Doe</td>
<td>700</td>
<td>250.00</td>
<td>100.00</td>
</tr>
<tr>
<td>693829</td>
<td>Thomas</td>
<td>9999</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>593029</td>
<td>Williams</td>
<td>200</td>
<td>25.00</td>
<td>0.00</td>
</tr>
<tr>
<td>192837</td>
<td>Lee</td>
<td>700</td>
<td>489.50</td>
<td>0.50</td>
</tr>
<tr>
<td>583990</td>
<td>Abraham</td>
<td>9999</td>
<td>500.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Creating a spreadsheet with XlsxWriter

- Generates Excel .xlsx files
- Quite featureful:
  - Charts
  - Data validation
  - Full formatting (including conditional formatting)
  - Autofilters
  - ...
- Installation
  - `pip3 install xlsxwriter`
- Documentation
  - `https://pypi.python.org/pypi/XlsxWriter`
  - `https://xlsxwriter.readthedocs.io/`
- License: BSD
Creating a spreadsheet with XlsxWriter

```python
from xlsxwriter import Workbook

with Workbook('test.xlsx') as workbook:
    ws = workbook.add_worksheet()
    ws.write_column('A1', [10, 93, 42, 59, 34])

    chart = workbook.add_chart({'type': 'line'})
    chart.add_series({'values': 'Sheet1!$A$1:$A$5'})

    ws.insert_chart('C1', chart)
```

Creating a spreadsheet with XlsxWriter
Creating a spreadsheet with XlsxWriter

<table>
<thead>
<tr>
<th>Column A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>93</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Graph showing data from Column A.
Converting table to Excel spreadsheet

```python
from xlsxwriter import Workbook
import ibm_db_dbi as db2

cur = db2.connect().cursor()
cur.execute("select cusnum, lstnam, cdtlmt, baldue, cdtdue from qiws.qcustcdt")

headers = [desc[0] for desc in cur.description]

with Workbook('qcustcdt.xlsx') as workbook:
    ws = workbook.add_worksheet()
    ws.write_row('A1', headers)
    for row, data in enumerate(cur, start=1):
        ws.write_row(row, 0, data)
```

Converting table to Excel spreadsheet
Converting table to Excel spreadsheet

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CUSNUM</td>
<td>LSTNAM</td>
<td>CDTLMT</td>
<td>BALDUE</td>
<td>CDTDUE</td>
</tr>
<tr>
<td>2</td>
<td>938472</td>
<td>Henning</td>
<td>5000</td>
<td>37</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>839283</td>
<td>Jones</td>
<td>400</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>392859</td>
<td>Vine</td>
<td>700</td>
<td>439</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>938485</td>
<td>Johnson</td>
<td>9999</td>
<td>3987.5</td>
<td>33.5</td>
</tr>
<tr>
<td>6</td>
<td>397267</td>
<td>Tyron</td>
<td>1000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>389572</td>
<td>Stevens</td>
<td>400</td>
<td>58.75</td>
<td>1.5</td>
</tr>
<tr>
<td>8</td>
<td>846283</td>
<td>Alison</td>
<td>5000</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>475938</td>
<td>Doe</td>
<td>700</td>
<td>250</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>693829</td>
<td>Thomas</td>
<td>9999</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>593029</td>
<td>Williams</td>
<td>200</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>192837</td>
<td>Lee</td>
<td>700</td>
<td>489.5</td>
<td>0.5</td>
</tr>
<tr>
<td>13</td>
<td>583990</td>
<td>Abraham</td>
<td>9999</td>
<td>500</td>
<td>0</td>
</tr>
</tbody>
</table>
Converting table to Excel spreadsheet

with Workbook('qcustcdt.xlsx') as workbook:
    fmt = workbook.add_format({"font_size": 20})
    hdr_fmt = workbook.add_format(
        {'font_size': 20, 'align':'center',border:1})
    red_fmt = workbook.add_format(
        {'font_size': 20, 'bg_color': '#FF0000'})
    ws.conditional_format("D2:D13", {'type': 'cell',
        'criteria': '>', 'value': 'C2*0.5', 'format': red_fmt})

    ws = workbook.add_worksheet()
    ws.set_column(0, len(headers)-1, 16)

    ws.write_row('A1', headers, hdr_fmt)
    ws.set_row(0, 22)

    for rownum, row in enumerate(cur, start=1):
        ws.write_row(rownum, 0, row)
        ws.set_row(rownum, 22, fmt)
Converting table to Excel spreadsheet

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CUSNUM</td>
<td>LSTNAM</td>
<td>CDTLMT</td>
<td>BALDUE</td>
<td>CTDUE</td>
</tr>
<tr>
<td>2</td>
<td>938472</td>
<td>Henning</td>
<td>5000</td>
<td>37</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>839283</td>
<td>Jones</td>
<td>400</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>392859</td>
<td>Vine</td>
<td>700</td>
<td>439</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>938485</td>
<td>Johnson</td>
<td>9999</td>
<td>3987.5</td>
<td>33.5</td>
</tr>
<tr>
<td>6</td>
<td>397267</td>
<td>Tyron</td>
<td>1000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>389572</td>
<td>Stevens</td>
<td>400</td>
<td>58.75</td>
<td>1.5</td>
</tr>
<tr>
<td>8</td>
<td>846283</td>
<td>Alison</td>
<td>5000</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>475938</td>
<td>Doe</td>
<td>700</td>
<td>250</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>693829</td>
<td>Thomas</td>
<td>9999</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>593029</td>
<td>Williams</td>
<td>200</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>192837</td>
<td>Lee</td>
<td>700</td>
<td>489.5</td>
<td>0.5</td>
</tr>
<tr>
<td>13</td>
<td>583990</td>
<td>Abraham</td>
<td>9999</td>
<td>500</td>
<td>0</td>
</tr>
</tbody>
</table>
Using Bottle

• Lightweight framework for building web applications
  - Includes a templating engine
  - Self-hosting web server included
  - Or use with flipflop (also included in OPS) in FastCGI mode
• Need PTF SI60566 or superseding
• See https://ibm.biz/installpythonpackages for more info to install
• https://pypi.python.org/pypi/bottle
views/index.html

<!DOCTYPE HTML>
<html lang="en-US">
<head><title>IBM i Bottle Sample</title></head>
<body>
    <form action="query" method="post">
        <h1><label for="sql">SQL Query</label></h1>
        <textarea rows="4" cols="60" name="sql" />
    </textarea>
    <br /><br />
    <input type="submit" value="Execute" />
    </form>
</body>
</html>
views/query.html

% from prettytable import from_db_cursor
% table = from_db_cursor(rows)

<!DOCTYPE HTML>
<html lang="en-US">
<head><title>IBM i Bottle Query</title></head>
<body>
{{! table.get_html_string() }}
</body>
</html>
Building a Simple Website

```python
from bottle import request, get, post, run, view
import ibm_db_dbi as db2

@get('/

def root():
    return bottle.template('index')

@post('/query')
@view('query')
def query():
    cur = db2.connect().cursor()
    cur.execute(request.forms.get('sql'))
    return {'rows': cur}

run(host='0.0.0.0', port=9000)
```
Website Example

SQL Query

```sql
SELECT * FROM QIWS.QCUSTCDT
```

Execute
Website Example

<table>
<thead>
<tr>
<th>CUSNUM</th>
<th>LSTNAM</th>
<th>INIT</th>
<th>STREET</th>
<th>CITY</th>
<th>STATE</th>
<th>ZIPCOD</th>
<th>CDTLMT</th>
<th>CHGCOD</th>
<th>BALDUE</th>
<th>CDTDUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>938472</td>
<td>Henning</td>
<td>G K</td>
<td>4859 Elm Ave</td>
<td>Dallas</td>
<td>TX</td>
<td>75217</td>
<td>5000</td>
<td>3</td>
<td>37.00</td>
<td>0.00</td>
</tr>
<tr>
<td>839283</td>
<td>Jones</td>
<td>B D</td>
<td>21B NW 135 St</td>
<td>Clay</td>
<td>NY</td>
<td>13041</td>
<td>400</td>
<td>1</td>
<td>100.00</td>
<td>0.00</td>
</tr>
<tr>
<td>392859</td>
<td>Vine</td>
<td>S S</td>
<td>PO Box 79</td>
<td>Broton</td>
<td>VT</td>
<td>5046</td>
<td>700</td>
<td>1</td>
<td>439.00</td>
<td>0.00</td>
</tr>
<tr>
<td>938485</td>
<td>Johnson</td>
<td>J A</td>
<td>3 Alpine Way</td>
<td>Helen</td>
<td>GA</td>
<td>30545</td>
<td>9999</td>
<td>2</td>
<td>3987.50</td>
<td>33.50</td>
</tr>
<tr>
<td>397267</td>
<td>Tyron</td>
<td>W E</td>
<td>13 Myrtle Dr</td>
<td>Hector</td>
<td>NY</td>
<td>14841</td>
<td>1000</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>389572</td>
<td>Stevens</td>
<td>K L</td>
<td>208 Snow Pass</td>
<td>Denver</td>
<td>CO</td>
<td>80226</td>
<td>400</td>
<td>1</td>
<td>58.75</td>
<td>1.50</td>
</tr>
<tr>
<td>846283</td>
<td>Alison</td>
<td>J S</td>
<td>787 Lake Dr</td>
<td>Isle</td>
<td>MN</td>
<td>56342</td>
<td>5000</td>
<td>3</td>
<td>10.00</td>
<td>0.00</td>
</tr>
<tr>
<td>475938</td>
<td>Doe</td>
<td>J W</td>
<td>59 Archer Rd</td>
<td>Sutter</td>
<td>CA</td>
<td>95685</td>
<td>700</td>
<td>2</td>
<td>250.00</td>
<td>100.00</td>
</tr>
<tr>
<td>693829</td>
<td>Thomas</td>
<td>A N</td>
<td>3 Dove Circle</td>
<td>Casper</td>
<td>WY</td>
<td>82609</td>
<td>9999</td>
<td>2</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>593029</td>
<td>Williams</td>
<td>E D</td>
<td>485 SE 2 Ave</td>
<td>Dallas</td>
<td>TX</td>
<td>75218</td>
<td>200</td>
<td>1</td>
<td>25.00</td>
<td>0.00</td>
</tr>
<tr>
<td>192837</td>
<td>Lee</td>
<td>F L</td>
<td>5963 Oak St</td>
<td>Hector</td>
<td>NY</td>
<td>14841</td>
<td>700</td>
<td>2</td>
<td>489.50</td>
<td>0.50</td>
</tr>
<tr>
<td>583990</td>
<td>Abraham</td>
<td>M T</td>
<td>392 Mill St</td>
<td>Isle</td>
<td>MN</td>
<td>56342</td>
<td>9999</td>
<td>3</td>
<td>500.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Rest Your Head on My Pillow

• “The friendly PIL fork”
  - Updated version of the Python Imaging Library
  - jpeg, png, tiff, webp formats and more
  - Variety of image manipulation functions

• Installation
  - export MAX_CONCURRENCY=1
  - export \
    CFLAGS=-I/QOpenSys/QIBM/ProdData/OPS/tools/include
  - pip3 install pillow

• Documentation
  - https://pypi.python.org/pypi/Pillow
  - https://python-pillow.org/

• License: Standard PIL License
Image Manipulation with Pillow

https://commons.wikimedia.org/wiki/File:4-Week-Old_Netherlands_Dwarf_Rabbit.JP
Making Thumbnails

from PIL import Image

img = Image.open('rabbit_full.jpg')
small_size = [dim//2 for dim in img.size]
small_img = img.resize(small_size)
small_img.save('rabbit.jpg')

# or better yet
max_size = (534, 534)
small_img = img.thumbnail(max_size)
small_img.save('rabbit.jpg')
Cropping Pictures

from PIL import Image

img = Image.open('rabbit.jpg')
# upper left x,y; lower right x,y
box = (0, 160, 356, 460)
small_img = img.crop(box)
small_img.save('rabbit_crop.jpg')
Watermarking

from PIL import Image

img = Image.open('rabbit.jpg')
logo = Image.open('ibmi.png')

position = ( 
(img.width - logo.width - 5), 
(img.height - logo.height - 5))

img.paste(logo, position, logo)
img.save('watermarked.jpg')
Interacting with Twitter

- Wrapper around Twitter REST APIs
  - Search
  - Send direct messages
  - Tweet & retweet
  - Favorite
  - Find trends
- Installation
  - pip3 install tweepy
- Documentation
  - https://pypi.python.org/pypi/tweepy
- License: MIT
Using Twitter

```python
import tweepy
from os import environ
KEY = environ['CONSUMER_KEY']
SECRET = environ['CONSUMER_SECRET']
ACCESS_TOKEN = environ['ACCESS_TOKEN']
ACCESS_SECRET = environ['ACCESS_TOKEN_SECRET']

auth = tweepy.OAuthHandler(KEY, SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)
api = tweepy.API(auth)
for result in api.search('@OCEANUserGroup')[:3]:
    print('@' + result.user.screen_name)
    print(result.text)
    print('')
```
Using Twitter
@theGonif
@kadler_ibm @OCEANUserGroup Is it not enough that there's no snow? 😊

@kadler_ibm
Excited to talk about #Python on #IBMi at today's @OCEANUserGroup kickoff, but where's my California sunrise? https://t.co/oAXcAqDHzO

@freschesolution
Join us TOMORROW with @OCEANUserGroup to kick off another great year with #IBM star guests @Steve_Will_IBMi &... https://t.co/iowktrR2rl
Shipping Packages

• Python API for goshippo.com
  – Supports FedEx, UPS, USPS, and more
  – Price Estimation
  – Shipment creation
  – Tracking
  – Customs declaration

• Installation
  – pip3 install shippo

• Documentation
  – https://pypi.python.org/pypi/shippo
  – https://github.com/goshippo/shippo-python-client
  – https://goshippo.com/docs

• License: MIT
Shipping Packages with Shippo

```python
import shippo

from = {
    "street1": "233 S Wacker Dr", 
    "city": "Chicago", "state": "IL"
}

to = {
    "street1": "1302 McKinley Park Rd", 
    "city": "Soudan", "state": "MN"
}

parcel = {
    "length": "5", "width": "5", 
    "height": "5", "distance_unit": "in", 
    "weight": "2", "mass_unit": "lb"
}
```
Shipping Packages with Shippo

```python
shippo.api_key = "<APIKEY>"

shipment = shippo.Shipment.create(
    address_from=from,
    address_to=to,
    parcels=[parcel], async=False
)

for rate in shipment.rates:
    print("{{} {} {}: ${{}}"."format(
        rate["provider"],
        rate["servicelevel"]["name"],
        rate["amount"])))
```
Shippo Output

USPS Priority Mail Express: $29.02
USPS Priority Mail: $6.47
USPS Parcel Select: $6.83
One-time Passcode Generation

- Generate one-time passcodes with ease
- Supports both TOTP and HOTP
- Compatible with Google Authenticator
- Installation
  - `pip3 install pyotp`
- Documentation
  - [https://pypi.python.org/pypi/pyotp](https://pypi.python.org/pypi/pyotp)
  - [https://github.com/pyotp/pyotp](https://github.com/pyotp/pyotp)
- License: BSD
PyOTP – One Time Passcode generator

```python
import pyotp
import time

key = pyotp.random_base32()
print(key)  # XK3I4RJ3OY7M7DAY

totp = pyotp.TOTP(key)

print(totp.now())  # 923442
time.sleep(60)
print(totp.now())  # 593490
```
Generating QR Codes

• Generate QR Codes in Python
• Uses PIL/Pillow under the covers
• Installation
  – pip3 install qrcode
• Documentation
  – https://pypi.python.org/pypi/qrcode
  – https://github.com/lincolnloop/python-qrcode
• License: BSD
Generating QR Codes

```python
import qrcode

qr = qrcode.make(url)
qr.save('qr.png')
```
Generating QR Codes

from bottle import request, response, get, run
import qrcode
import pyotp
import io

@get('/

def root():
    key = request.query.get('key', 'XK3I4RJ30Y7M7DAY')
    totp = pyotp.TOTP(key)
    qr = qrcode.make(totp.provisioning_uri('pyqrcode'))
    imgbuf = io.BytesIO()
    qr.save(imgbuf, format='PNG')
    response.content_type = 'image/png'
    return imgbuf.getvalue()
Generating QR Codes
Reading RSS Feeds

- Supports both RSS and Atom formats
- Data normalization and sanitization
- Installation
  - `pip3 install feedparser`
- Documentation
  - [https://pypi.python.org/pypi/feedparser](https://pypi.python.org/pypi/feedparser)
- License: BSD
Reading RSS Feeds

```python
import feedparser

url = 'http://ibmsystemsmag.com/CMSTemplates/IBMSYSTEMSMag/Feeds/Open-Your-i.aspx'

feed = feedparser.parse(url)

for entry in feed['entries'][:3]:
    print(entry['title'])
    print(entry['link'])
    print()
```
Reading RSS Feeds

IBM i Open Source and the Talent That Follows

Cleared for Takeoff With Node.js on IBM i

IBM Cloud, Watson and Web Services Help Applications Fly
Parsing HTML with Beautiful Soup

• Turn “tag soup” in to something beautiful
• Supports xml and html parsers
• Installation
  – pip3 install beautifulsoup4
• Documentation
  – https://pypi.python.org/pypi/beautifulsoup4
  – https://www.crummy.com/software/BeautifulSoup/
• License: MIT
**Parsing HTML with Beautiful Soup**

- **Monty Python and the Holy Grail credits:**

<table>
<thead>
<tr>
<th>Cast</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graham Chapman</td>
<td>King Arthur / Voice of God / Middle Head / Hiccoughing Guard</td>
</tr>
<tr>
<td>John Cleese</td>
<td>Second Swallow-Savvy Guard / The Black Knight / Peasant 3 / Sir Lancelot the Brave / Taunting French Guard / Tim the Enchanter</td>
</tr>
<tr>
<td>Eric Idle</td>
<td>Dead Collector / Peasant 1 / Sir Robin the Not-Quite-So-Brave-as-Sir Launcelot / First Swamp Castle Guard / Concorde / Roger the Shrubber / Brother Maynard</td>
</tr>
<tr>
<td>Terry Gilliam</td>
<td>Patsy / Green Knight / Old Man from Scene 24 (Bridgekeeper) / Sir Bors / Animator / Gorrilla Hand</td>
</tr>
<tr>
<td>Terry Jones</td>
<td>Dennis's Mother / Sir Bedevere / Left Head / Prince Herbert / Cartoon Scribe (voice)</td>
</tr>
<tr>
<td>Michael Palin</td>
<td>First Swallow-Savvy Guard / Dennis / Peasant 2 / Right Head / Sir Galahad the Pure / Narrator / King of Swamp Castle / Brother Maynard's Brother / Leader of The Knights Who Say NI!</td>
</tr>
</tbody>
</table>
Parsing HTML with Beautiful Soup

```html
<h4 id="cast" class="dataHeaderWithBorder" name="cast"></h4>
<table class="cast_list">
  <tbody>
    <tr>
      <td class="primary_photo"></td>
      <td class="character">Graham Chapman</td>
    </tr>
    <tr class="odd">
      <td class="primary_photo"></td>
      <td class="character">John Cleese</td>
    </tr>
  </tbody>
</table>
```
Parsing HTML with Beautiful Soup

```python
from bs4 import BeautifulSoup
from urllib.request import urlopen

u = 'http://imdb.com/title/tt0071853/fullcredits'
resp = urlopen(u)

soup = BeautifulSoup(resp.read(), 'html.parser')

top_cast = soup.find_all('td', 'itemprop')[:6]
names = [actor.span.string for actor in top_cast]

for name in names:
    print(name)
```
Parsing HTML with Beautiful Soup

Graham Chapman
John Cleese
Eric Idle
Terry Gilliam
Terry Jones
Michael Palin
Using Plac for Argument Parsing

- Parsing command line options the easy way
- "main" function become command line arguments
- Installation
  - pip3 install plac
- Documentation
  - https://pypi.python.org/pypi/plac
  - http://micheles.github.io/plac/
- License: BSD
import ibm_db_dbi as db2
from prettytable import from_db_cursor

def main(port: ("Local port", 'option')):
    "NETSTAT in Python using plac"
    sql = 'SELECT CONNECTION_TYPE, LOCAL_ADDRESS, LOCAL_PORT, JOB_NAME FROM QSYS2.NETSTAT_JOB_INFO'
    params = []
    if port:
        sql += ' WHERE LOCAL_PORT = ?'
        params.append(port)
    cur = db2.connect().cursor()
    cur.execute(sql, params)
    print(from_db_cursor(cur))

if __name__ == '__main__':
    import plac; plac.call(main)
Even Simpler Parsing with Plac

netstat.py -h

usage: netstat.py [-h] [-port PORT]

NETSTAT in Python using plac

optional arguments:
  -h, --help    show this help message and exit
  -port PORT    Local port
## Even Simpler Parsing with Plac

```
netstat.py -p 2010
```

<table>
<thead>
<tr>
<th>CONNECTION_TYPE</th>
<th>LOCAL_ADDRESS</th>
<th>LOCAL_PORT</th>
<th>JOB_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPV4</td>
<td>0.0.0.0</td>
<td>2010</td>
<td>576538/QTMHHTTP/ADMIN</td>
</tr>
<tr>
<td>IPV4</td>
<td>0.0.0.0</td>
<td>2010</td>
<td>576977/QTMHHTTP/ADMIN</td>
</tr>
<tr>
<td>IPV6</td>
<td>::</td>
<td>2010</td>
<td>576538/QTMHHTTP/ADMIN</td>
</tr>
<tr>
<td>IPV6</td>
<td>::</td>
<td>2010</td>
<td>576977/QTMHHTTP/ADMIN</td>
</tr>
</tbody>
</table>
More Python Resources

• Python 3 std library:
  https://docs.python.org/3.4/library/index.html
  - re – regular expression support
  - hashlib – generate md5, sha1, sha2, ... hashes
  - tempfile – create temporary file and directories
  - pprint - “pretty print” data
  - glob – Unix-style path expansion
  - socket – direct socket handling

• Python Package Index: https://pypi.python.org/pypi
Python Getting Started Resources

- Official Tutorial: https://docs.python.org/3.4/tutorial/
- Learn Python in Y Minutes: https://learnxinyminutes.com/docs/python3/
- Python on IBM i Examples: http://ibm.biz/pythonexamplesonibmi
- Download these Examples: http://ibm.biz/spt-ocean-2018
Questions?
How do I get it?

- Python 3 delivered in 5733-OPS Option 2 in June 2015
- Python 2 is also available in 5733-OPS Option 4 if you *really* need it
- 5733-OPS is a “skip-ship” LPO that is licensed for IBM i 7.1+
- Initially only Option 1 was defined (Node.js v1) all the others are placeholders, to be defined later and delivered via PTF
- [http://ibm.biz/getting-ops](http://ibm.biz/getting-ops)

To get Python 3, you must install 5733-OPS *BASE and Option 2, and then install the enablement PTFs and any requisites.

- The easiest way is via the Open Source Group PTF
  - 7.3: SF99225
  - 7.2: SF99223
  - 7.1: SF99123
But Wait, There's More

- We also include many optional Python packages:
  - *ibm_db* package, DB2 interface
  - *itoolkit* package, IBM i toolkit wrapper for XMLService
  - *flipflop* package, FastCGI gateway
  - *bottle* package, lightweight web framework
- Each PTF just lays down the “wheel”, you still need to install it.
  eg.
  - `cd /QOpenSys/QIBM/ProdData/OPS/Python-pkgs/ibm_db`
  - `pip3 install ibm_db-*cp34*.whl`
- See [https://ibm.biz/installpythonpackages](https://ibm.biz/installpythonpackages) for more info
Python 3 Programs

• Programs are added to /QOpenSys/usr/bin
  – SSH is recommended, QP2TERM or QSH as last resort
• Programs:
  – python3 – main runtime/interpreter
  – pip3 – package manager: Installs local wheels, downloads wheels and dependencies from the Python Package Index (PyPI)
  – pydoc3 – documentation tool, can be used to show documentation for keywords, topics, functions, modules, packages, and more
  – 2to3 – converts Python 2 source code to Python 3
Power Systems Social Media

https://facebook.com/IBMPowerSystems

https://twitter.com/IBMPowerSystems

https://www.linkedin.com/company/ibm-power-systems

http://www.youtube.com/c/ibmpowersystems

https://www.ibm.com/blogs/systems/topics/servers/power-systems/
More to Follow:

<table>
<thead>
<tr>
<th>Blogs</th>
<th>Twitter</th>
<th>#Hashtags</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Systems Magazine You and i (Steve Will)</td>
<td>@IBMSystems</td>
<td>#PowerSystems</td>
</tr>
<tr>
<td>IBM Systems Magazine i-Can (Dawn May)</td>
<td>@COMMONug</td>
<td>#IBMi</td>
</tr>
<tr>
<td>IBM Systems Magazine: iDevelop (Jon Paris and Susan Gantner)</td>
<td>@IBMChampions</td>
<td>#IBMAIX</td>
</tr>
<tr>
<td>IBM Systems Magazine: iTalk with Tuohy</td>
<td>@IBMSystemsISVs</td>
<td>#POWER8</td>
</tr>
<tr>
<td>IBM Systems Magazine: Open your i (Jesse Gorzinski)</td>
<td>@LinuxIBMMag</td>
<td>#LinuxonPower</td>
</tr>
<tr>
<td>Trevor Perry Blog</td>
<td>@OpenPOWERorg</td>
<td>#OpenPOWER</td>
</tr>
<tr>
<td>IBM DB2 for i (Mike Cain)</td>
<td>@AIXMag</td>
<td>#HANAonPower</td>
</tr>
<tr>
<td>IBM DB2 Web Query for i (Doug Mack)</td>
<td>@IBMiMag</td>
<td>#ITinfrastructure</td>
</tr>
<tr>
<td>Modern-i-zation (Tim Rowe)</td>
<td>@ITJungleNews</td>
<td>#OpenSource</td>
</tr>
<tr>
<td></td>
<td>@SAPonIBM</td>
<td>#HybridCloud</td>
</tr>
<tr>
<td></td>
<td>@SiDforIBMi</td>
<td>#BigData</td>
</tr>
<tr>
<td></td>
<td>@IBMAIXeSupp</td>
<td>#IBMiOSS</td>
</tr>
</tbody>
</table>
Special notices

This document was developed for IBM offerings in the United States as of the date of publication. IBM may not make these offerings available in other countries, and the information is subject to change without notice. Consult your local IBM business contact for information on the IBM offerings available in your area.

Information in this document concerning non-IBM products was obtained from the suppliers of these products or other public sources. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. Send license inquiries, in writing, to IBM Director of Licensing, IBM Corporation, New Castle Drive, Armonk, NY 10504-1785 USA.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

The information contained in this document has not been submitted to any formal IBM test and is provided “AS IS” with no warranties or guarantees either expressed or implied. All examples cited or described in this document are presented as illustrations of the manner in which some IBM products can be used and the results that may be achieved. Actual environmental costs and performance characteristics will vary depending on individual client configurations and conditions.

IBM Global Financing offerings are provided through IBM Credit Corporation in the United States and other IBM subsidiaries and divisions worldwide to qualified commercial and government clients. Rates are based on a client’s credit rating, financing terms, offering type, equipment type and options, and may vary by country. Other restrictions may apply. Rates and offerings are subject to change, extension or withdrawal without notice.

IBM is not responsible for printing errors in this document that result in pricing or information inaccuracies.

All prices shown are IBM’s United States suggested list prices and are subject to change without notice; reseller prices may vary.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Some measurements quoted in this document may have been estimated through extrapolation. Users of this document should verify the applicable data for their specific environment.
Special notices (cont.)

IBM, the IBM logo, ibm.com AIX, AIX (logo), AIX 5L, AIX 6 (logo), AS/400, BladeCenter, Blue Gene, ClusterProven, DB2, ESCON, i5/OS, i5/OS (logo), IBM Business Partner (logo), IntelliStation, LoadLeveler, Lotus, Lotus Notes, Notes, Operating System/400, OS/400, PartnerLink, PartnerWorld, PowerPC, pSeries, Rational, RISC System/6000, RS/6000, THINK, Tivoli, Tivoli (logo), Tivoli Management Environment, WebSphere, xSeries, z/OS, zSeries, Active Memory, Balanced Warehouse, CacheFlow, Cool Blue, IBM Systems Director VMControl, pureScale, TurboCore, Chiphopper, Cloudscape, DB2 Universal Database, DS4000, DS6000, DS8000, EnergyScale, Enterprise Workload Manager, General Parallel File System, GPFS, HACMP, HACMP6000, HASM, IBM Systems Director Active Energy Manager, iSeries, Micro-Partitioning, POWER, PowerExecutive, PowerVM, PowerVM (logo), PowerHA, Power Architecture, Power Everywhere, Power Family, POWER Hypervisor, Power Systems, Power Systems (logo), Power Systems Software, Power Systems Software (logo), POWER2, POWER3, POWER4, POWER4+, POWER5, POWER5+, POWER6, POWER6+, POWER7, System i, System p, System p5, System Storage, System z, TME 10, Workload Partitions Manager and X-Architecture are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks owned by IBM in other countries.

A full list of U.S. trademarks owned by IBM may be found at: http://www.ibm.com/legal/copytrade.shtml.

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.

AltiVec is a trademark of Freescale Semiconductor, Inc.

AMD Opteron is a trademark of Advanced Micro Devices, Inc.

InfiniBand, InfiniBand Trade Association and the InfiniBand design marks are trademarks and/or service marks of the InfiniBand Trade Association.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, 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.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Microsoft, Windows and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries or both.

NetBench is a registered trademark of Ziff Davis Media in the United States, other countries or both.

SPECint, SPECfp, SPECjbb, SPECweb, SPECAppServer, SPEC OMP, SPECviewperf, SPECCapc, SPECCapc, SPECjvm, SPECmail, SPECmap and SPECsfs are trademarks of the Standard Performance Evaluation Corp (SPEC).

The Power Architecture and Power.org wordmarks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

TPC-C and TPC-H are trademarks of the Transaction Performance Council (TPC).

UNIX is a registered trademark of The Open Group in the United States, other countries or both.

Other company, product and service names may be trademarks or service marks of others.