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

© 2018 IBM Corporation
Why Else?

- Simple, straightforward language
- People know it!
  - used heavily in the industry
  - taught in Academia
Who is using Python?
Web Sites Using Python

- YouTube
- Mozilla
- Bitbucket
- Instagram
Python GUI Applications: Orange

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

LibreOffice
The Document Foundation

WebSphere
Application Server
Raspberry Pi

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

© 2018 IBM Corporation
Raspberry Pi

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
- http://ibm.biz/spt-nhmug-2018
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](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'] = f'attachment; filename="{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
Writing Zip Files

```python
from zipfile import ZipFile
from io import BytesIO
from sys import argv
from os.path import basename

# <snip email setup>
zipbuf = BytesIO()

with ZipFile(zipbuf, 'w') as myzip:
    for arg in argv[1:]:
        myzip.write(arg, basename(arg))

zipbuf.seek(0)
part = MIMEApplication(zipbuf.read())
part['Content-Disposition'] = 'attachment; filename="logs.zip"
msg.attach(part)
smtp.send_message(msg)
```

© 2018 IBM Corporation
DIY cPYtoimpf

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

• Documentation: https://docs.python.org/3/library/csv.html
from csv import writer, QUOTE_NONNUMERIC
import ibm_db_dbi as db2

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

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

# <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])
<table>
<thead>
<tr>
<th>ID</th>
<th>First Name</th>
<th>Last Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>938472</td>
<td>G K</td>
<td>Henning</td>
<td>5000</td>
</tr>
<tr>
<td>839283</td>
<td>B D</td>
<td>Jones</td>
<td>400</td>
</tr>
<tr>
<td>392859</td>
<td>S S</td>
<td>Vine</td>
<td>700</td>
</tr>
<tr>
<td>938485</td>
<td>J A</td>
<td>Johnson</td>
<td>9999</td>
</tr>
<tr>
<td>397267</td>
<td>W E</td>
<td>Tyron</td>
<td>1000</td>
</tr>
<tr>
<td>389572</td>
<td>K L</td>
<td>Stevens</td>
<td>400</td>
</tr>
<tr>
<td>846283</td>
<td>J S</td>
<td>Alison</td>
<td>5000</td>
</tr>
<tr>
<td>475938</td>
<td>J W</td>
<td>Doe</td>
<td>700</td>
</tr>
<tr>
<td>693829</td>
<td>A N</td>
<td>Thomas</td>
<td>9999</td>
</tr>
<tr>
<td>593029</td>
<td>E D</td>
<td>Williams</td>
<td>200</td>
</tr>
<tr>
<td>192837</td>
<td>F L</td>
<td>Lee</td>
<td>700</td>
</tr>
<tr>
<td>583990</td>
<td>M T</td>
<td>Abraham</td>
<td>9999</td>
</tr>
</tbody>
</table>
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
Parsing Arguments with Argparse

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

cl = f"{cmd} SERVER(*HTTP) HTTPSVR({args.server})"

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

system(f'system "{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(f'{obj["name"]}: {obj["limit"]}')
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Henning</td>
<td>5000</td>
</tr>
<tr>
<td>Jones</td>
<td>400</td>
</tr>
<tr>
<td>Vine</td>
<td>700</td>
</tr>
<tr>
<td>Johnson</td>
<td>9999</td>
</tr>
<tr>
<td>Tyron</td>
<td>1000</td>
</tr>
<tr>
<td>Stevens</td>
<td>400</td>
</tr>
<tr>
<td>Alison</td>
<td>5000</td>
</tr>
<tr>
<td>Doe</td>
<td>700</td>
</tr>
<tr>
<td>Thomas</td>
<td>9999</td>
</tr>
<tr>
<td>Williams</td>
<td>200</td>
</tr>
<tr>
<td>Lee</td>
<td>700</td>
</tr>
<tr>
<td>Abraham</td>
<td>9999</td>
</tr>
</tbody>
</table>
Using SQLite

- Access the lightweight database from Python
- Useful for applications that support SQLite but not Db2
- Documentation:
import sqlite3

conn = sqlite3.connect('my.db')

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

conn.create_function('btc', 1, usd_to_btc)

cur = conn.cursor()

cur.execute("select movie, gross, btc(gross) from mytable")

for row in cur:
    print(row)
# 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)
More Python 3 “batteries included”

• Python 3 std library: https://docs.python.org/3.6/library/index.html
  - json – json parsing and generation
  - xml – Element tree (simple), dom (DOM API), and sax (SAX parser)
  - 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
  - sqlite3 – sqlite support and database adapter
Honey, can you pick up some 9V and lithium-ion 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
  https://devpi.net/docs/devpi/devpi/stable/%2Bd/index.html
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
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
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)
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>
from prettytable import from_db_cursor
import ibm_db_dbi as db2
from sys import argv

conn = db2.connect()
cur = conn.cursor()
cur.execute(" .join(argv[1:])")

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

./from_db.py select cusnum, lstnam, cdtlmt, baldue, cdtdue from qiws.qcustcdt
<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
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

<table>
<thead>
<tr>
<th>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>1</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>93</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>34</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]
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 rownum, row in enumerate(cur, start=1):
        ws.write_row(rownum, 0, row)
Converting table to Excel spreadsheet

<table>
<thead>
<tr>
<th>CUSNUM</th>
<th>LSTNAM</th>
<th>CDTLMT</th>
<th>BALDUE</th>
<th>CTDTUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>938472</td>
<td>Henning</td>
<td>5000</td>
<td>37</td>
</tr>
<tr>
<td>3</td>
<td>839283</td>
<td>Jones</td>
<td>400</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>392859</td>
<td>Vine</td>
<td>700</td>
<td>439</td>
</tr>
<tr>
<td>5</td>
<td>938485</td>
<td>Johnson</td>
<td>9999</td>
<td>3987.5</td>
</tr>
<tr>
<td>6</td>
<td>397267</td>
<td>Tyron</td>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>389572</td>
<td>Stevens</td>
<td>400</td>
<td>58.75</td>
</tr>
<tr>
<td>8</td>
<td>846283</td>
<td>Alison</td>
<td>5000</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>475938</td>
<td>Doe</td>
<td>700</td>
<td>250</td>
</tr>
<tr>
<td>10</td>
<td>693829</td>
<td>Thomas</td>
<td>9999</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>593029</td>
<td>Williams</td>
<td>200</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>192837</td>
<td>Lee</td>
<td>700</td>
<td>489.5</td>
</tr>
<tr>
<td>13</td>
<td>583990</td>
<td>Abraham</td>
<td>9999</td>
<td>500</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()
# set column widths
ws.set_column(0, len(headers)-1, 20)

ws.write_row('A1', headers, hdr_fmt)
# set row height
ws.set_row(0, 22)

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

<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</td>
<td>0</td>
</tr>
<tr>
<td>839283</td>
<td>Jones</td>
<td>400</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>392859</td>
<td>Vine</td>
<td>700</td>
<td>439</td>
<td>0</td>
</tr>
<tr>
<td>938485</td>
<td>Johnson</td>
<td>9999</td>
<td>3987.5</td>
<td>33.5</td>
</tr>
<tr>
<td>397267</td>
<td>Tyron</td>
<td>1000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>389572</td>
<td>Stevens</td>
<td>400</td>
<td>58.75</td>
<td>1.5</td>
</tr>
<tr>
<td>846283</td>
<td>Alison</td>
<td>5000</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>475938</td>
<td>Doe</td>
<td>700</td>
<td>250</td>
<td>100</td>
</tr>
<tr>
<td>693829</td>
<td>Thomas</td>
<td>9999</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>593029</td>
<td>Williams</td>
<td>200</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>192837</td>
<td>Lee</td>
<td>700</td>
<td>489.5</td>
<td>0.5</td>
</tr>
<tr>
<td>583990</td>
<td>Abraham</td>
<td>9999</td>
<td>500</td>
<td>0</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
  − yum install libjpeg-turbo-devel libpng-devel
  − 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.JPG
Making Thumbnails

```python
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

```python
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')
Using Flask

- Most popular Python microframework
- Built-in templating engine (Jinja2)
- Python WSGI compliant
  - Built-in development server
  - gunicorn or CherryPy are great for production
  - Use flipflop for FastCGI
templates/index.html

<!DOCTYPE HTML><html lang="en-US">
<head><title>IBM i Flask 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 />
<input type="submit" value="Execute" />
</form>
</body>
</html>
<!DOCTYPE HTML>
<html lang="en-US">
<head><title>IBM i Flask Query</title></head>
<body>
{{ cursor | cursor_to_table | safe }}
</body>
</html>
Building a Simple Website

```python
from flask import *
import ibm_db_dbi as db2
import prettytable

app = Flask(__name__)

@app.template_filter('cursor_to_table')
def cursor_to_table(cursor):
    return prettytable.from_db_cursor(cursor).get_html_string()
```
Building a Simple Website

```python
@app.route('/

def root():
    return render_template('index.html')

@app.route('/query', methods=['POST'])
def query():
    cur = db2.connect().cursor()
    cur.execute(request.form.get('sql'))
    return render_template('query.html', cursor=cur)
```
Website Example

SQL Query

```
SELECT * FROM QIWS.QCUSCSTCDT
```
<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>

© 2018 IBM Corporation
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
import tweepy
from config import *

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)

for tweet in tweepy.Cursor(api.search, q="#NHMUG").items(4):
    print('@' + tweet.user.screen_name + ':
' + tweet.text + '"
')
    print()
Using Twitter

@rbrownpa:
Great sessions on python on #IBMi @ #NHMUG with @kadler_ibm user groups are the place to #learn https://t.co/tXxJc7hGeQ

@jbuck_imPower:
RT @rbrownpa: Have a great #IBMi #NHMUG conference. https://t.co/GPBI5EBL2w

@rbrownpa:
Have a great #IBMi #NHMUG conference. https://t.co/GPBI5EBL2w

@kadler_ibm:
Thunder and lightning! 😊😊😊
#NHMUG must know how much I love thunderstorms, so they had one brought in just for me! https://t.co/cjdnDcULJT
Arbitrary Web Services with Requests

- “The only non-GMO HTTP library for Python”
  - High-level HTTP library
  - Designed around PEP 20 idioms
    - Beautiful is better than ugly
    - Explicit is better than implicit
    - Simple is better than complex
    - Complex is better than complicated
    - Readability counts

- Installation
  - pip3 install requests

- Documentation
  - http://docs.python-requests.org

- License: Apache2
Using Requests

```python
import requests

payload = {
    'title': 'Python is great!',
    'userId': 1,
    'body': '''Python is super cool.
You should really check it out!

https://www.python.org/'''
}

r = requests.post('https://jsonplaceholder.typicode.com/posts', json=payload)
print(f"Post id: {r.json()]['id']}")
```
Mechanizing Web Sites with MechanicalSoup

- Website “mechanization” library
- Based on Mechanize, but for Python 3
- Glues together
  - Requests for HTTP
  - BeautifulSoup for HTML parsing
- Installation
  - pip3 install MechanicalSoup
- Documentation
  - http://mechanicalsoup.readthedocs.io
- License: MIT
Using MechanicalSoup

```python
import mechanicalsoup

browser = mechanicalsoup.StatefulBrowser()

browser.open("http://httpbin.org/")

browser.follow_link("forms")

browser.select_form('form[action="/post"]')
```
Using MechanicalSoup

browser["custname"] = "Kevin";
browser["custtel"] = "507-555-1985"
browser["custemail"] = "me@example.com"
browser["size"] = "medium"
browser["topping"] = ("bacon", "cheese")
browser["comments"] = "Deliver to my hammock"
response = browser.submit_selected()
print(response.json())
Using MechanicalSoup

Customer name: 

Telephone: 

E-mail address: 

Pizza Size
- Small
- Medium
- Large

Pizza Toppings
- Bacon
- Extra Cheese
- Onion
- Mushroom

Preferred delivery time: - - : - - - -
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://github.com/pyotp/pyotp

• License: BSD
import pyotp
import time

key = pyotp.random_base32()
print(key) # XK3I4RJ30Y7M7DAY

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://pypi.python.org/pypi/qrcode)
  - [https://github.com/lincolnloop/python-qrcode](https://github.com/lincolnloop/python-qrcode)
- License: BSD
Generating QR Codes

```python
import qrcode
qr = qrcode.make(url)
qr.save('qr.png')
```
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
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://pypi.python.org/pypi/shippo)
  - [https://github.com/goshippo/shippo-python-client](https://github.com/goshippo/shippo-python-client)
  - [https://goshippo.com/docs](https://goshippo.com/docs)

- License: MIT
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:
    provider = rate["provider"]
    name = rate["servicelevel"]["name"]
    amount = rate["amount"]
    print(f'{provider} {name}: ${amount}')
```
Shippo Output

USPS Priority Mail Express: $29.02
USPS Priority Mail: $6.47
USPS Parcel Select: $6.83
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(s)</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-Sawy 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 / Gorilla 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-Sawy 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>
<table class="simpleTable simpleCreditsTable">
  <thead>
    <tr>
      <th class="primary_photo"></th>
      <th class="character"></th>
    </tr>
  </thead>
  <tbody>
    <tr class="odd">
      <td class="primary_photo">
        <a href="/name/nm0001037/?ref=ttfc_fc_cl_t1" itemprop="url">
          <span class="itemprop" itemprop="name">Graham Chapman</span>
        </a>
      </td>
      <td class="ellipses">...</td>
    </tr>
    <tr class="even">
      <td class="primary_photo">
        <a href="/name/nm0000092/?ref=ttfc_fc_cl_t2" itemprop="url">
          <span class="itemprop" itemprop="name">John Cleese</span>
        </a>
      </td>
      <td class="ellipses">...</td>
    </tr>
  </tbody>
</table>
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
Reading RSS Feeds

- Supports both RSS and Atom formats
- Data normalization and sanitization
- Installation
  - pip3 install feedparser
- Documentation
  - 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
Writing a NETSTAT Shell Utility

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

sql = 'SELECT * FROM QSYS2.NETSTAT_JOB_INFO'
conn = db2.connect()
cur = conn.cursor()
cur.execute(sql)

print(from_db_cursor(cur))
```
Adding Arguments with argparser

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

parser = ArgumentParser(description='NETSTAT')
parser.add_argument('--port', type=int, 
    help='Show only local PORT entries')
args = parser.parse_args()

sql = 'SELECT * FROM QSYS2.NETSTAT_JOB_INFO'
params = []
if args.port is not None:
    sql += ' WHERE LOCAL_PORT = ?'
    params.append(args.port)
```
Writing a netstat shell utility

$ netstat.py -h

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

NETSTAT

optional arguments:

  -h, --help    show this help message and exit
  --port PORT   Show only local PORT entries
Writing a netstat shell utility

$ netstat.py

+-----------------+----------------+-------------+------------------+---------------+-----------
| CONNECTION_TYPE  | REMOTE_ADDRESS  | REMOTE_PORT | REMOTE_PORT_NAME  | LOCAL_ADDRESS  | LOCAL_PORT |
+-----------------+----------------+-------------+------------------+---------------+-----------
|       IPV4      |   127.0.0.1    |    35087    |       None       |   127.0.0.1   |   52647    |
|       IPV6      |       ::       |      0      |       None       |       ::      |     22     |
|       IPV4      |    0.0.0.0     |      0      |       None       |    0.0.0.0    |    2012    |
|       IPV4      |    0.0.0.0     |      0      |       None       |   9.5.39.48   |    427     |
|       IPV4      |    0.0.0.0     |      0      |       None       |    0.0.0.0    |     25     |

$ netstat.py –port 445

+-----------------+----------------+-------------+------------------+---------------+-----------
| CONNECTION_TYPE  | REMOTE_ADDRESS  | REMOTE_PORT | REMOTE_PORT_NAME  | LOCAL_ADDRESS  | LOCAL_PORT |
+-----------------+----------------+-------------+------------------+---------------+-----------
|       IPV6      |       ::       |      0      |       None       |       ::      |    445     |
|       IPV4      |    0.0.0.0     |      0      |       None       |    0.0.0.0    |    445     |

© 2018 IBM Corporation
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>
Python Getting Started Resources

- Official Tutorial: https://docs.python.org/3.6/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-nhmug-2018
Questions?
For More Information:

<table>
<thead>
<tr>
<th>Some Links You Need</th>
<th>Twitter</th>
<th>Hashtags</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM i Home Page:  <a href="http://www.ibm.com/systems/i">www.ibm.com/systems/i</a></td>
<td>@IBMSystems</td>
<td>#IBMi30</td>
</tr>
<tr>
<td>IBM 30th Anniversary:  <a href="http://ibmi30.mybluemix.net/">http://ibmi30.mybluemix.net/</a></td>
<td>@COMMONug</td>
<td>#PowerSystems</td>
</tr>
<tr>
<td>IBM Systems Magazine IBM i Edition:</td>
<td>@IBMChampions</td>
<td>#IBMi</td>
</tr>
<tr>
<td><a href="http://ibmsystemsmag.com/ibmi/">http://ibmsystemsmag.com/ibmi/</a></td>
<td>@IBMSystemsISVs</td>
<td>#IBMAIX</td>
</tr>
<tr>
<td>Support Life Cycle:</td>
<td>@IBMiMag</td>
<td>#POWER8</td>
</tr>
<tr>
<td><a href="https://www-01.ibm.com/software/support/ibmi/lifecycle/">https://www-01.ibm.com/software/support/ibmi/lifecycle/</a></td>
<td>@ITJungleNews</td>
<td>#LinuxonPower</td>
</tr>
<tr>
<td>License Topics:</td>
<td>@SAPonIBMi</td>
<td>#OpenPOWER</td>
</tr>
<tr>
<td><a href="https://www-01.ibm.com/support/docview.wss?uid=nas8N1022087">https://www-01.ibm.com/support/docview.wss?uid=nas8N1022087</a></td>
<td>@SiDforIBMi</td>
<td>#HANAonPower</td>
</tr>
</tbody>
</table>

#IBMi30
#PowerSystems
#IBMi
#IBMAIX
#POWER8
#LinuxonPower
#OpenPOWER
#HANAonPower
#ITinfrastructure
#OpenSource
#HybridCloud
#BigData
## For More Information:

### Blogs

<table>
<thead>
<tr>
<th>IBM Blogs:</th>
<th>IBM Champion’s Blogs:</th>
</tr>
</thead>
<tbody>
<tr>
<td>You and i (Steve Will)</td>
<td>iDevelop (Jon Paris and Susan Gantner)</td>
</tr>
<tr>
<td>i Can (Dawn May)</td>
<td>iTalk with Tuoy</td>
</tr>
<tr>
<td>Open your i (Jesse Gorzinski)</td>
<td></td>
</tr>
<tr>
<td>IBM DB2 for i (Mike Cain)</td>
<td></td>
</tr>
<tr>
<td>IBM DB2 Web Query for i (Doug Mack)</td>
<td><a href="http://ibmsystemsmag.com/blogs/idvelop/">http://ibmsystemsmag.com/blogs/idvelop/</a></td>
</tr>
<tr>
<td><a href="http://ibmsystemsmag.com/blogs/you-and-i/">http://ibmsystemsmag.com/blogs/you-and-i/</a></td>
<td></td>
</tr>
<tr>
<td><a href="http://ibmsystemsmag.com/blogs/i-can/">http://ibmsystemsmag.com/blogs/i-can/</a></td>
<td></td>
</tr>
<tr>
<td><a href="http://ibmsystemsmag.com/blogs/open-your-i/">http://ibmsystemsmag.com/blogs/open-your-i/</a></td>
<td></td>
</tr>
<tr>
<td><a href="http://db2fori.blogspot.co.uk/">http://db2fori.blogspot.co.uk/</a></td>
<td></td>
</tr>
<tr>
<td><a href="http://db2webqueryi.blogspot.co.uk/">http://db2webqueryi.blogspot.co.uk/</a></td>
<td></td>
</tr>
</tbody>
</table>