Breaking out of PASE:

Access the rest of IBM i from Python

Kevin Adler
kadler@us.ibm.com
@kadler_ibm

Jesse Gorzinski
jgorzins@us.ibm.com
@IBMJesseG

Quick intro

“Getting Started” Resources

- Information on installing/managing general open source
  - http://ibm.biz/ibmi-rpms
- Examples repository on GitHub
  - http://github.com/IBM/ibmi-oss-examples
  -

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/

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!!
- Additional packages on the Python Package Index (PyPI)
  - Over 212,000 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

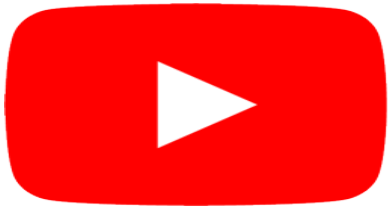
Why Else?

- Simple, straightforward language
- People know it!
  - used heavily in the industry
  - taught in Academia

Origin of the "Python" name?

- "Now, it's quite simple to defend yourself against a man armed with a banana. First of all you force him to drop the banana; then, second, you eat the banana, thus disarming him. You have now rendered him helpless."
- "This parrot is no more! He has ceased to be!"
- "I'm sorry to have kept you waiting, but I'm afraid my walk has become rather sillier recently."
- "I don't like Spam©!"
- "That rabbit's dynamite!"

Web Sites Using Python

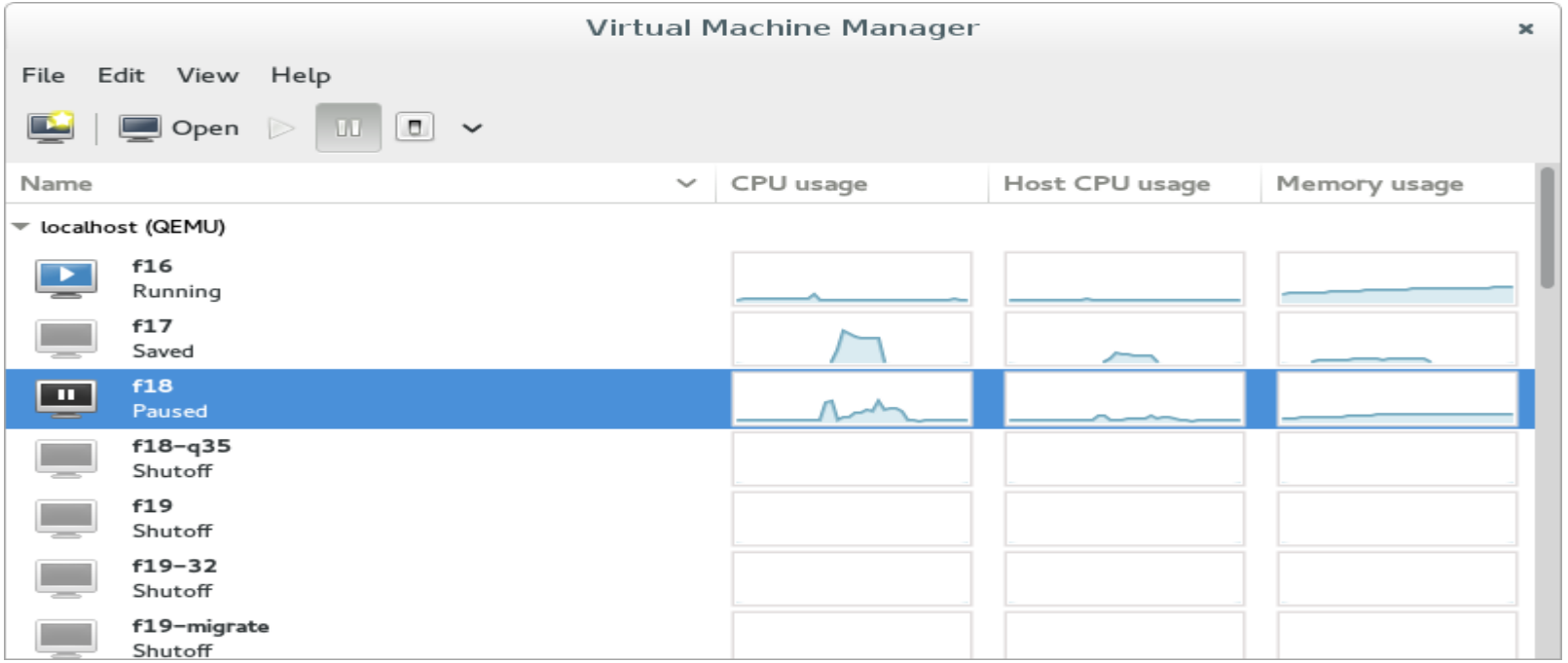


YouTube

moz://a

Bitbucket

Python GUI Applications: VM Manager

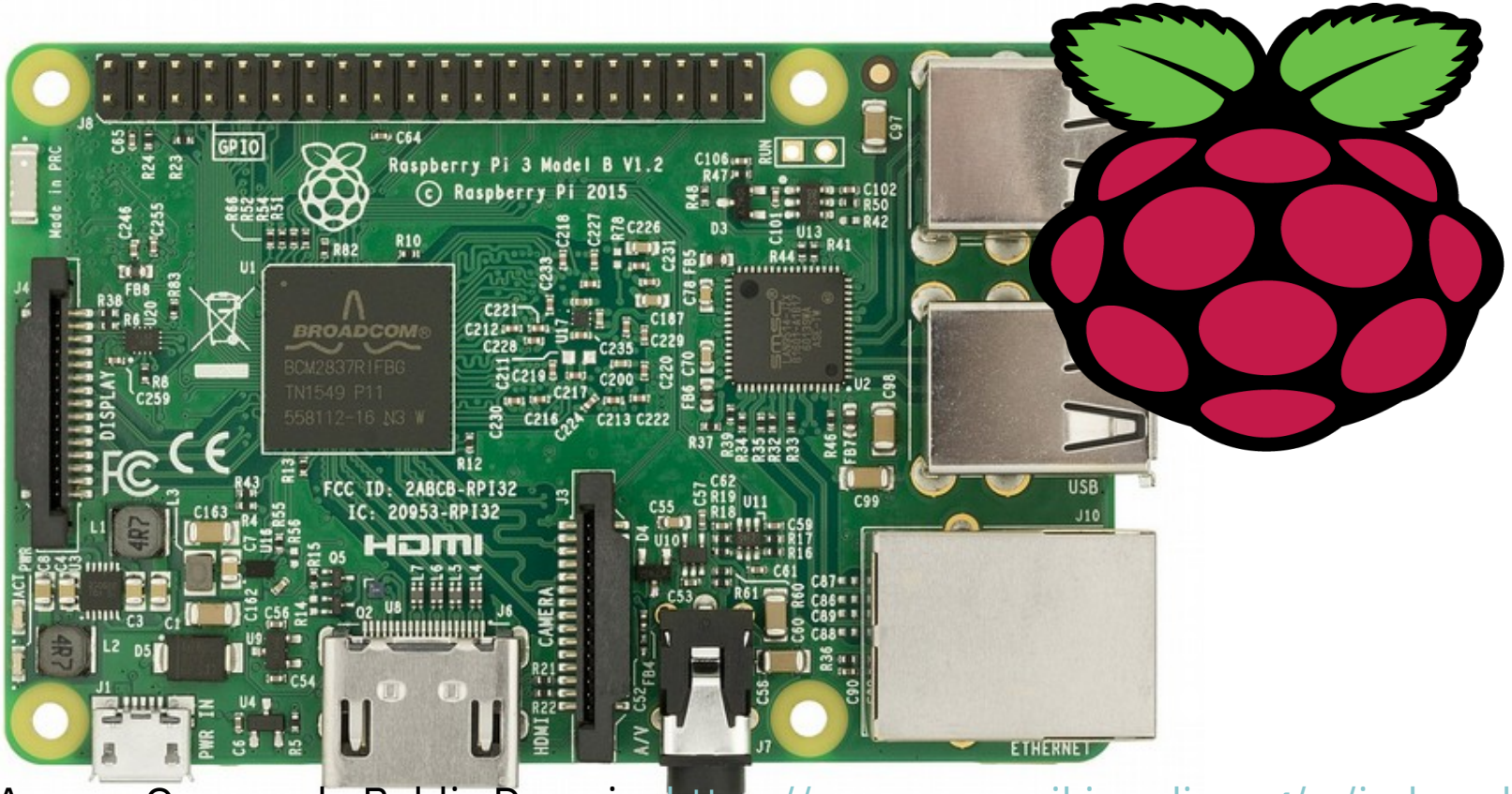


Other Applications Using Python

WebSphere®
Application Server

LibreOffice
The Document Foundation

Raspberry Pi



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

Python success stories!

- https://www.python.org/about/success/

- 41 real-life success stories across multiple domains

Assistive Technologies

- Python in The Blind Audio Tactile mapping System
- A Custom Image Viewing Game for an Autistic Child

Code Generation

- Cog: A Code Generation Tool Written in Python

Computer Graphics

- Industrial Light & Magic Runs on Python

Configuration

- D-Link Australia Uses Python to Control Firmware Updates
- Python as Technology Enabler for TTTech's Development Software

Language Basics

Language basics

- Indentation matters!

- Typing
  - Java and other "traditional" compiled languages are statically typed
  - Python is dynamically typed

- Don't need to declare variables before using them

```
x = 7
```

- New objects constructed without 'new' keyword

```
thing = MyClass()
```

Language basics (sequences)

- Lists
  - Declared like this:
    [0, 1, 2, 3, 4]
  - Mutable
  - Can access items with positive or negative index
  - List comprehensions can be used (we'll learn about these)

- Tuples
  - Declared like this:
    (0, 1, 2, 3, 4)
  - Immutable
  - Can access items with positive or negative index

Language basics (data structures)

- Dictionaries
  - Key/value pairs
  - Declared like this:

    {'this':7, 'that': 12, 'the other thing': 42}
  - Mutable
  - Can access items via key

- Sets
  - Declared like this:

    set("Jesse G")

    set(['N', 'H', 'M', 'U', 'G'])
  - Mutable, but can only contain immutable objects
  - Methods for set operations (supersets, intersections, etc)

Database Access with ibm_db

Database Access Just a Pip Away

- ibm_db project: https://github.com/ibmdb/python-ibmdb
- Cross-platform, open source project sponsored by IBM
- Uses IBM i SQL CLI interface in PASE to access the database from IBM i.
  - Can be used with Db2 Connect from other platforms.
- The ibm_db_dbi module complies with the Python Database API version 2 specified in PEP-249: https://www.python.org/dev/peps/pep-0249/
- The lower-level ibm_db module is IBM-specific and more of a pain to use. **(Not recommended)**
- Install:
  - `yum install python3-ibm_db`

Dive In the Water's Fine

```
import ibm_db_dbi

conn = ibm_db_dbi.connect()
cursor = conn.cursor()

cursor.execute('select * from qiws.qcustcdt')
for row in cursor:
    print(row)

conn.disconnect()
```

Dive In the Water's Fine

```
import ibm_db_dbi as db2

conn = db2.connect()
cursor = conn.cursor()

cursor.execute('select * from qiws.qcustcdt')
for row in cursor:
    print(row)

conn.disconnect()
```

Dive In the Water's Fine

```
from ibm_db_dbi import connect


conn = connect()
cursor = conn.cursor()


cursor.execute('select * from qiws.qcustcdt')
for row in cursor:
    print(row)


conn.disconnect()
```

Where’s the Fetch?

- Cursor objects have 3, mostly unused fetch methods…
  - fetchone()
  - fetchmany(count)
  - fetchall()
- Because cursor objects are also iterators!

Dive In the Water's Fine

```
cur.execute('select * from qiws.qcustcdt')

one_row = cur.fetchone()
print(row)

five_rows = cur.fetchmany(5)
for row in five_rows:
    print(row)

remaining_rows = cur.fetchall()
for row in remaining_rows:
    print(row)
```

Connect (to the) Matrix

```
# *LOCAL w/ myuser
conn = db2.connect(user="myuser", password="Passw0rd")

# MyRDB w/ current user
conn = db2.connect(database="MyRDB")

# Same as above, but using connection string
conn = db2.connect("DATABASE=*LOCAL;UID=myuser;PWD=Passw0rd")
conn = db2.connect("DATABASE=MyRDB")
```

Execute All the Things!

```
cursor.execute("create table mytable(i int, d double, c char(10)")

cursor.execute("insert into mytable values(4, 32.5, 'foo')")

cursor.execute("update mytable set i=5 where i=4")

cursor.execute("delete from mytable where i<4")

cursor.execute(ANY SQL HERE)
```

Don’t Inject that SQL

```
sql = f"update mytable set phone='{phn}' where id={id}"
cursor.execute(sql)
```

Don’t Inject that SQL

```
sql = f"update mytable set phone='{phn}' where id={id}"

cursor.execute(sql)


phn = "876-5309', salary=200000 where id=1234 --"

id = 1234


cursor.execute(f"update mytable set phone='876-5309',
salary=200000 where id=1234 --' where id=1234")
```

Parameterize It

~~sql = f"update mytable set phone='{phn}' where id={id}"~~

```
sql = f"update mytable set phone=? where id=?"
cursor.execute(sql, (phn, id))
```

```
cursor.execute("""create or replace procedure myproc(in x int, out
y int, inout z int)
language sql
begin
  set y = x + z; set z = x * 5; set x = 10;
end""")


cursor.execute("call myproc(?,?,?)", (19,23,41))


# Where are output parameters?
```

```
cursor.execute("""create or replace procedure myproc(in x int, out
y int, inout z int)
language sql
begin
  set y = x + z; set z = x * 5; set x = 10;
end""")


(x, y, z) = cursor.callproc("myproc", (19, 23, 41))


print(x, y, z) # 19 60 95
```

Cursor objects

```
cur.callproc("proc_returns_3_rs")


rs = True
while rs:
    for row in cur:
        print(row)

    rs = cur.nextset()
```

```
cur.execute("insert into t(c) values('Huh?')")

conn.close()

conn = db2.connect();
cur = conn.cursor()

cur.execute("select * from t where c = 'Huh?'")
rows = cur.fetchall()

assert len(rows) == 1
```

```
cur.execute("insert into t(c) values('Aha!')")
conn.commit()
conn.close()

conn = db2.connect();
cur = conn.cursor()

cur.execute("select * from t where c = 'Aha!'")
rows = cur.fetchall()

assert len(rows) == 1
```

```
conn.set_autocommit(True)
cur.execute("insert into t(c) values('Aha!')")
conn.close()

conn = db2.connect();
cur = conn.cursor()

cur.execute("select * from t where c = 'Aha!'")
rows = cur.fetchall()

assert len(rows) >= 1
```

```
cur.execute("select 1 from sysibmi.sysdummy1")
print(cur.fetchone()[0])
```

```
cur.execute("select 1 from sysibmi.sysdummy1")
print(cur.fetchone()[0])

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/QOpenSys/pkgs/lib/python3.6/site-packages/ibm_db-2.0.5.6-
py3.6-os400-powerpc64.egg/ibm_db_dbi.py", line 1387, in execute
    self._prepare_helper(operation)
  File "/QOpenSys/pkgs/lib/python3.6/site-packages/ibm_db-2.0.5.6-
py3.6-os400-powerpc64.egg/ibm_db_dbi.py", line 1250, in
_prepare_helper
    raise self.messages[-1]
ibm_db_dbi.ProgrammingError: ibm_db_dbi::ProgrammingError: SYSDUMMY1
in SYSIBMI type *FILE not found. SQLSTATE=42704 SQLCODE=-204
```

```
try:
    cur.execute("select 1 from sysibmi.sysdummy1")
    print(cur.fetchone()[0])
except db2.ProgrammingError e:
    print(e)
```

```
cur.callproc('qcmdexc', ('ADDLIBLE QIWS',))
cur.execute("select * from qcustcdt")
for row in cur:
    print(row)
```

```
cur.callproc('qcmdexc', ('ADDLIBLE QIWS',))
cur.execute("select * from qcustcdt")
for row in cur:
    print(row)


# ibm_db_dbi::ProgrammingError: QCUSTCDT in KADLER type *FILE not
found. SQLSTATE=42704 SQLCODE=-204
```

```
from ibm_db import SQL_ATTR_DBC_SYS_NAMING, SQL_TRUE

options = { SQL_ATTR_DBC_SYS_NAMING: SQL_TRUE }
conn = db2.connect(conn_options=options)
cur = conn.cursor()

cur.callproc('qcmdexc', ('ADDLIBLE QIWS',))
cur.execute("select * from qcustcdt")
for row in cur:
    print(row)
```

```
cur.execute('create table qtemp.foo(i int)')
cur.execute('insert into qtemp.foo values(1)')
```

```
cur.execute('create table qtemp.foo(i int)')

cur.execute('insert into qtemp.foo values(1)')


# ibm_db_dbi::ProgrammingError: Statement Execute Failed: FOO in
QTEMP not valid for operation. SQLSTATE=55019 SQLCODE=-7008
```

```
from ibm_db import SQL_ATTR_TXN_ISOLATION, SQL_TXN_NO_COMMIT

conn.set_option({ SQL_ATTR_TXN_ISOLATION: SQL_TXN_NO_COMMIT })
cur = conn.cursor()

cur.execute('create table qtemp.foo(i int)')
cur.execute('insert into qtemp.foo values(1)')
```

```
import ibm_db_dbi as db2
from ibm_db import SQL_ATTR_DBC_SYS_NAMING, SQL_TRUE
from ibm_db import SQL_ATTR_TXN_ISOLATION, SQL_TXN_NO_COMMIT
options = {
    SQL_ATTR_TXN_ISOLATION: SQL_TXN_NO_COMMIT,
    SQL_ATTR_DBC_SYS_NAMING: SQL_TRUE,
}
conn = db2.connect()
conn.set_option(options)
```

Cursor description

- description – read-only attribute
  - “2d” sequence describing the columns
  - each index contains a 7-item sequence for that column in this order:
    - name – column name
    - type_code – ibm_db_dbi type object
    - display_size – how many characters needed to display
    - internal_size – how many bytes needed to store
    - precision – total digits for NUMERIC/DECIMAL columns
    - scale – digits after decimal for NUMERIC/DECIMAL/TIMESTAMP columns
    - null_ok – is the column nullable?

Example: Converting table to text table

```
from prettytable import from_db_cursor
import ibm_db_dbi as db2

conn = db2.connect()

cur = conn.cursor()
cur.execute("select * from qiws.qcustcdt")
print(from_db_cursor(cur))
```

Example: Converting table to text table

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

Example: Converting table to Excel spreadsheet

```
from xlsxwriter import Workbook
import ibm_db_dbi as db2

conn = db2.connect()
cur = conn.cursor()
cur.execute("select * from qiws.qcustcdt")

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

with Workbook('qcustcdt.xlsx') as workbook:
    worksheet = workbook.add_worksheet()

    worksheet.write_row('A1', headers)
    for rownum, row in enumerate(cur, start=1):
        worksheet.write_row(rownum, 0, row)
```

Example: Converting table to Excel spreadsheet

	A	B	C	D	E	F	G	H	I	J	K
1	CUSNUM	LSTNAM	INIT	STREET	CITY	STATE	ZIPCOD	CDTLMT	CHGCOD	BALDUE	CDTDUE
2	938472	Henning	G K	4859 Elm	Dallas	TX	75217	5000	3	37	0
3	839283	Jones	B D	21B NW	Clay	NY	13041	400	1	100	0
4	392859	Vine	S S	PO Box 7	Broton	VT	5046	700	1	439	0
5	938485	Johnson	J A	3 Alpine V	Helen	GA	30545	9999	2	3987.5	33.5
6	397267	Tyron	W E	13 Myrtle	Hector	NY	14841	1000	1	0	0
7	389572	Stevens	K L	208 Snow	Denver	CO	80226	400	1	58.75	1.5
8	846283	Alison	J S	787 Lake	Isle	MN	56342	5000	3	10	0
9	475938	Doe	J W	59 Archer	Sutter	CA	95685	700	2	250	100
10	693829	Thomas	A N	3 Dove C	Casper	WY	82609	9999	2	0	0
11	593029	Williams	E D	485 SE 2	Dallas	TX	75218	200	1	25	0
12	192837	Lee	F L	5963 Oak	Hector	NY	14841	700	2	489.5	0.5
13	583990	Abraham	M T	392 Mill S	Isle	MN	56342	9999	3	500	0
14											
15											

Database Access with pyodbc

Can you ODBC me?

- pyodbc https://github.com/mkleehammer/pyodbc
- Python wrapper around ODBC
- Conforms to PEP-249 interface – same API as ibm_db_dbi*
- Works anywhere with an ODBC driver manager (macOS, Windows, Linux, PASE)
- Same interface for multiple drivers (Db2, MySQL, MS SQL, PostgreSQL, ...)
- Use IBM i Access ODBC driver on Linux and Windows... and now in PASE!
- Install:
  - `yum install python3-devel unixODBC-devel`
  - `pip3 install pyodbc`

Dive In the Water's Fine, Redux

```
import pyodbc

conn = pyodbc.connect("DSN=*LOCAL")
cursor = conn.cursor()

cursor.execute('select * from qiws.qcustcdt')
for row in cursor:
    print(row)

conn.disconnect()
```

DSN

- Data Source Names (DSN) are basically a collection of configuration options
  - What system (hostname)
  - What options (sytem naming, database, etc)
- System DSNs available to everyone
  - `/QOpenSys/etc/odbc.ini`
- User DSNs specific to a user
  - `~/.odbc.ini`

IBM i Access ODBC Driver for PASE

- Available from IBM i Access download site (soon!!)
- Provides System DSN *LOCAL out of the box
  - Needs 7.3 PTF SI68113 or 7.2 PTF SI69058 to use without a password
- Install with yum
  - `yum install ibm-iaccess-1.1.0.11-0.ibmi7.2.ppc64.rpm`

Configuring IBM i Access Driver

- Other configuration options documented here
  - https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_74/rzaik/connectkeywords.htm
- Useful options
  - CommitMode – default isolation level *(*CHG)*
  - TrueAutoCommit – whether ODBC autocommit changes isolation level to *NONE *(yes)*
  - Naming – whether to use SQL or system naming *(SQL)*
  - DefaultLibraries – sets the connection's library list

```
cursor.execute("""create or replace procedure myproc(in x int, out
y int, inout z int)
language sql
begin
  set y = x + z; set z = x * 5; set x = 10;
end""")


(x, y, z) = cursor.callproc("myproc", (19, 23, 41))


print(x, y, z) # 19 60 95
```

```
cursor.execute("""create or replace procedure myproc(in x int, out
y int, inout z int)
language sql
begin
  set y = x + z; set z = x * 5; set x = 10;
end""")


(x, y, z) = cursor.callproc("myproc", (19, 23, 41))


print(x, y, z) # 19 60 95
```

Note: pyodbc doesn't support callproc or output parameters

Calling ILE with itoolkit

itoolkit – an XMLSERVICE wrapper

- itoolkit project: https://ibm.biz/itoolkitpython
- Python interface to XMLService: https://ibm.biz/xmlservice
- Let's you call
  - RPG programs and service programs
  - CL commands
  - PASE programs and shell scripts
  - SQL database access
- `yum install python3-itoolkit`
- `pip3 install itoolkit`

What Can We Do?

- Commands
  - iCmd – Call CL command (even with output parameters)
  - iCmd5250 – call CL command and get screen output
- Programs
  - iPgm – Call program
  - iSrvPgm – Call service program
- iSh – call PASE program or shell script
- iXml – anything else

Which Transport is Right for You?

- DirectTransport
- DatabaseTransport
- HttpTransport
- SshTransport

Requires itoolkit 1.6 or higher.
Previous transport classes still supported, but deprecated.

DatabaseTransport

- Stored procedure call over database connection
- Any* PEP-249 connection object can be used
  - ibm_db_dbi
  - pyodbc
- No configuration needed using ibm_db_dbi when used locally
- Can use pyodbc with IBM i Access driver
- Local and remote connections supported
  - Using ibm_db_dbi remotely requires Db2 Connect license

HttpTransport

- Calls using HTTP REST API
- Requires XMLSERVICE configured as a FastCGI endpoint in Apache
- Uses Apache TLS configuration for security
- Local and remote connections supported

DirectTransport

- No configuration needed
- Fastest call, runs in current job
- Some things don't work from chroot
- Currently broken with 64-bit Python (ie yum version)
- Local connection only

SshTransport

- Calls over SSH using Paramiko package
- Requires `xmlservice-cli` package installed on the target system
- Uses SSH for security
- Local and remote connections supported

Connecting

```
# DirectTransport example
from itoolkit import *
from itoolkit.transport import DirectTransport

itransport = DirectTransport()

# HttpTransport example
from itoolkit import *
from itoolkit.transport import HttpTransport

itransport = HttpTransport(url, user, password)
```

Connecting

```
# DatabaseTransport examples
from itoolkit import *
from itoolkit.transport import DatabaseTransport

# using ibm_db
import ibm_db_dbi
itransport = DatabaseTransport(ibm_db_dbi.connect())

# using pyodbc
import pyodbc
itransport = DatabaseTransport(pyodbc.connect("DSN=myDSN"))
```

Basics

```
from itoolkit import *
from itoolkit.transport import DatabaseTransport
import ibm_db_dbi
itransport = DatabaseTransport(ibm_db_dbi.connect())
itool = iToolKit()
itool.add(iFoo('op1’, ...))
itool.add(iBar('op2', ...))
itool.call(itransport)
op1 = itool.dict_out('op1')
if 'success' in op1:
    print (op1['success'])
else:
    print (op1['error'])
```

Calling Standard CL Commands

```
itool.add(iCmd('addlible', 'ADDLIBLE KADLER'))


itool.call(itransport)

rtvjoba = itool.dict_out('rtvjoba')

if 'error' in rtvjoba:
    print("Encountered an error")
```

Output Parameters from CL Commands

```
itool.add(iCmd('rtvjoba', \
               'RTVJOBA USRLIBL(?) SYSLIBL(?) CCSID(?N) OUTQ(?)'))
itool.call(itransport)
rtvjoba = itool.dict_out('rtvjoba')
if 'success' in rtvjoba:

    print(rtvjoba['row'][0]['USRLIBL'])
    print(rtvjoba['row'][1]['SYSLIBL'])
    print(rtvjoba['row'][2]['CCSID'])
    print(rtvjoba['row'][3]['OUTQ'])
```

Output Parameters, Sensible

```
itool.add(iCmd('rtvjoba', \
               'RTVJOBA USRLIBL(?) SYSLIBL(?) CCSID(?N) OUTQ(?)'))
itool.call(itransport)
rtvjoba = itool.dict_out('rtvjoba')
if 'success' in rtvjoba:
    rtvjoba = { k: v for d in rtvjoba['row'] for k, v in d.items() }
    print(rtvjoba['USRLIBL'])
    print(rtvjoba['SYSLIBL'])
    print(rtvjoba['CCSID'])
    print(rtvjoba['OUTQ'])
```

Command Display Output

```
itool.add(iCmd5250('wrkactjob_key', 'WRKACTJOB'))


itool.call(itransport)
wrkactjob = itool.dict_out('wrkactjob_key')
print(wrkactjob['wrkactjob_key'])
```

Command Display Output

```
                                    Work with Active Jobs                                                    Page    1
 5770SS1 V7R2M0 140418                                                                  DBCSB3P2  05/09/18  11:18:59 CDT
 Reset . . . . . . . . . . . . . . . . . :   *NO
 Subsystems  . . . . . . . . . . . . . . :   *ALL
 CPU Percent Limit . . . . . . . . . . . :   *NONE
 Response Time Limit . . . . . . . . . . :   *NONE
 Sequence  . . . . . . . . . . . . . . . :   *SBS
 Job name  . . . . . . . . . . . . . . . :   *ALL
 CPU %  . . . :      .0          Elapsed time . . . . . . . :   00:00:00          Active jobs . . . . . . :    233
                                     Current                            --------Elapsed---------
Temporary
 Subsystem/Job  User        Number   User        Type Pool Pty     CPU   Int    Rsp  AuxIO   CPU%  Function       Status   Threads    Storage
 QBATCH         QSYS        799117   QSYS        SBS    2   0        .4                 0     .0                  DEQW          2          3
   QDFTJOBD     JENKINS     846920   JENKINS     BCH    2  50        .0                 0     .0  CMD-QSH         EVTW          1          4
   QDFTJOBD     REDIS       846925   REDIS       BCH    2  50        .0                 0     .0  CMD-QSH         TIMW          1          4
   QZSHSH       JENKINS     846921   JENKINS     BCI    2  50     244.9                 0     .0  JVM-jenkins.wa  THDW         81        520
   QZSHSH       KADLER      846858   KADLER      BCI    2  50      14.6                 0     .0  PGM-python3     SELW          1         20
   QZSHSH       KADLER      849046   KADLER      BCI    2  50       1.0                 0     .0  PGM-python3     SELW          1         46
   QZSHSH       KADLER      849091   KADLER      BCI    2  50        .9                 0     .0  PGM-python3     SELW          1         45
   QZSHSH       KADLER      849106   KADLER      BCI    2  50        .9                 0     .0  PGM-python3     SELW          1         44
   QZSHSH       KADLER      849121   KADLER      BCI    2  50        .9                 0     .0  PGM-python3     SELW          1         45
   QZSHSH       REDIS       846926   REDIS       BCI    2  50      57.1                 0     .0  PGM-redis-serv  SELW          4
```

Calling an ILE program

```
itool.add(iPgm('my_key','MYPGM')
    .addParm(iData('a', '1a', 'a'))
    .addParm(iDS('ds')
        .addData(iData('b', '10i0', '1'))
        .addData(iData('c', '12p2', '3.33'))))
itool.call(itransport)
results = itool.dict_out('my_key')
print(results['a'])
print(results['ds']['b'])
print(results['ds']['c'])
```

Calling a Service Program

```
itool.add(iSrvPgm('my_key','MYSRVPGM', 'myfunction')
    .addParm(iData('a', '1a', 'a'))
    .addParm(iDS('ds')
        .addData(iData('b', '10i0', '1'))
        .addData(iData('c', '12p2', '3.33'))))
itool.call(itransport)
results = itool.dict_out('my_key')
print(results['a'])
print(results['ds']['b'])
print(results['ds']['c'])
```

Specifying the Program Library

```
itool.add(iPgm('my_key','MYPGM', {'lib': 'KADLER'})
    .addParm(iData('a', '1a', 'a'))
    .addParm(iDS('ds')
        .addData(iData('b', '10i0', '1'))
        .addData(iData('c', '12p2', '3.33'))))
itool.call(itransport)
```

Example: Calling an ILE program from Python

```
itool.add(iCmd('addlible', 'ADDLIBLE KADLER'))
itool.add(iPgm('my_key','MYPGM')
    .addParm(iData('a', '1a', 'a'))
    .addParm(iDS('ds')
        .addData(iData('b', '10i0', '1'))
        .addData(iData('c', '12p2', '3.33'))))
itool.call(itransport)
```

iSh

```
itool.add(iSh('mysh', 'curl http://google.com'))
itool.call(itransport)
mysh = itool.dict_out('mysh')
print(tools['mysh'])
```

Advanced Toolkit

- Scenario: RPG program to get back a list of objects that haven't been saved
- Two output parameters
  - NumObjs – number of objects returned in second parm (10i0)
  - Objects – array of 1000 object structures
    - Name – object name (10a)
    - Library – object library (10a)

Advanced Toolkit

```
itool.add(iPgm('unsaved', 'QUNSAVED')
 .addParm(iData('NumObjs','10i0',''))
 .addParm(iDS('Objects', {'dim':'1000'})
  .addData(iData('Name','10a',''))
  .addData(iData('Library','10a',''))))

data = itool.dict_out('unsaved')
print(len(data['Objects']))
for obj in data['Objects']
    print(obj)
```

Advanced Toolkit

```
1000
{'Name': 'OBJ001', 'Library': 'QGPL'}
{'Name': 'OBJ002', 'Library': 'QGPL'}
{'Name': 'OBJ003', 'Library': 'QGPL'}
{'Name': 'OBJ004', 'Library': 'QGPL'}
{'Name': '', 'Library': ''}
...
{'Name': '', 'Library': ''}
{'Name': '', 'Library': ''}
```

Advanced Toolkit

```
itool.add(iPgm('unsaved', 'QUNSAVED')
 .addParm(iData('NumObjs','10i0',''))
 .addParm(iDS('Objects', {'dim':'1000'})
  .addData(iData('Name','10a',''))
  .addData(iData('Library','10a',''))


data = itool.dict_out('unsaved')
num_objs = data['NumObjs']
print(num_objs)
for obj in data['Objects'][:num_objs]
    print(obj)
```

Advanced Toolkit

```
4
{'Name': 'OBJ001', 'Library': 'QGPL'}
{'Name': 'OBJ002', 'Library': 'QGPL'}
{'Name': 'OBJ003', 'Library': 'QGPL'}
{'Name': 'OBJ004', 'Library': 'QGPL'}
```

Advanced Toolkit

```
itool.add(iPgm('unsaved', 'QUNSAVED')
 .addParm(iData('NumObjs', '10i0', '', {'enddo':'count'}))
 .addParm(iDS('Objects', {'dim':'1000', 'dou':'count'})
  .addData(iData('Name','10a',''))
  .addData(iData('Library','10a',''))))
data = itool.dict_out('unsaved')
print(len(data['Objects']))
for obj in data['Objects']
    print(obj)
```

Advanced Toolkit

```
4
{'Name': 'OBJ001', 'Library': 'QGPL'}
{'Name': 'OBJ002', 'Library': 'QGPL'}
{'Name': 'OBJ003', 'Library': 'QGPL'}
{'Name': 'OBJ004', 'Library': 'QGPL'}
```

Advanced Toolkit

- Scenario: System API call that takes in a large buffer with variable sized output data and needs to know the size of the buffer passed in
- One output parameter
  - Buffer – API defined data structure with a header containing numerous fields, followed by an array of data structures
- One input parameter
  - Len – size of Buffer

Advanced Toolkit

```
itool.add(iPgm('apicall', 'QAPICALL')
 .addParm(
  iDS('API0001', { 'len': 'buflen' })
  .addData(iData('Fld1','4b',''))
  # ...
  .addData(iData('NumResults','10i0','', {'enddo':'count'}))
  .addData(iDS('results', {'dim':'1000', 'dou':'count'})
    .addData(iData('Res1','3i0','')))
    # ...
 .addParm(iData('BufSize','10i0','', { 'setlen': 'buflen' })))
```

Resources

- ibm_db links
  - Code: https://github.com/ibmdb/python-ibmdb
  - Wiki: https://github.com/ibmdb/python-ibmdb/wiki
  - Forum: https://groups.google.com/forum/#!forum/ibm_db
  - Python DB API (PEP 249): https://www.python.org/dev/peps/pep-0249/
  - `yum install python3-ibm_db`
- itoolkit links
  - Code: https://bitbucket.org/litmis/python-itoolkit
  - Docs: http://python-itoolkit.readthedocs.io/en/latest/
  - XMLSERVICE: http://yips.idevcloud.com/wiki/index.php/XMLService/XMLService
  - `yum install python3-itoolkit`

IBM TSS Support

- Git
- Jenkins
- rsync
- Node.js
- Apache Tomcat
- WordPress
- Python

- For more resources, see my blog post: http://ibmsystemsmag.com/blogs/open-your-i/december-2018/a-game-changer-for-open-source-support/

Questions?

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 inquires, 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.

Revised September 26, 2006

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, HACMP/6000, 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 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.
Linux is a registered trademark of Linus Torvalds in the United States, other countries or both.
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, SPECjAppServer, SPEC OMP, SPECviewperf, SPECapc, SPEChpc, SPECjvm, SPECmail, SPECimap 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 Processing Council (TPPC).
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.

Revised December 2, 2010