Introduction to Python on IBM i
Agenda

• A little about Python
• Why use Python?
• How to install/determine if installed
• Syntax101
  – Variables
  – Strings
  – Functions
  – Command Line
• DB2 Data Access
Acknowledgements

- Kevin Adler
- Tony Cairns
- Jesse Gorzinski
- Google
- Memegenerator
- Corn chips & salsa
- Clean socks
- and, of course,
  - spam

"I AM SPEECHLESS, WITHOUT WORDS, AS SILENT AS A EUROPEAN SPARROW."

THANK YOU. THANK YOU. THANK YOU VERY MUCH.
Before you freak out

• Why isn’t Mike talking about PHP?
  – Zend WAS the PHP company
  – Rogue Wave IS the Open Source company
    • Support for 300+ Open Source projects
    • And, of course, PHP
A little about Python
What is it really?

- General purpose programming language
- Easy to get started
- Simple syntax
- Great for integrations
- Access to C and other API
- Infrastructure first, but applications too.

Thanks: Tahani Alamanie
Historically...

- Python was conceptualized by Guido Van Rossum in the late 1980s.
- Rossum published the first version of Python code (0.9.0) in February 1991 at the CWI (Centrum Wiskunde & Informatica) in the Netherlands, Amsterdam.
- Python is derived from ABC programming language, which is a general-purpose programming language that had been developed at the CWI.
- Rossum chose the name "Python", since he was a big fan of Monty Python's Flying Circus.
- Python is now maintained by a core development team at the institute, although Rossum still holds a vital role in directing its progress.

Thanks: Tahani Alamanie
Python lineage

• Python 1 – 1994
• Python 2 – 2000 (Not dead yet...)
  – 2.7 - 2010
• Python 3 – 2008
  – 3.5 – 2015
  – 3.6.1 – March 2017
Python 2 or 3?

Python 2.7 will retire in...

<table>
<thead>
<tr>
<th>2</th>
<th>9</th>
<th>22</th>
<th>7</th>
<th>34</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Years</td>
<td>Months</td>
<td>Days</td>
<td>Hours</td>
<td>Minutes</td>
<td>Seconds</td>
</tr>
</tbody>
</table>

What's all this, then?

Python 2.7 will not be maintained past 2020. No official date has been given, so this clock counts down until April 12th, 2020, which will be roughly the time of the 2020 PyCon. I am hereby suggesting we make PyCon 2020 the official end-of-life date, and we throw a massive party to celebrate all that Python 2 has done for us. (If this sounds interesting to you, email pythonclockorg@gmail.com).

Python 2, thank you for your years of faithful service.

Python 3, your time is now.

How do I get started?

If the code you care about is still on Python 2, that's totally understandable. Most of PyPi's popular packages now work on Python 2 and 3, and more are being added every day. To ease the transition, the official porting guide has advice for running Python 2 code in Python 3.
What’s the diff?

• Example:
• Python 2 print statement replaced by function:
  – Python2 – print "Hello World!"
  – Python3 - print("Hello World!")
• Many more differences, tho...
Why use it?
Python==hot

<table>
<thead>
<tr>
<th>Change</th>
<th>Programming Language</th>
<th>Ratings</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Java</td>
<td>14.639%</td>
<td>-6.32%</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>7.002%</td>
<td>-6.22%</td>
</tr>
<tr>
<td></td>
<td>C++</td>
<td>4.751%</td>
<td>-1.95%</td>
</tr>
<tr>
<td></td>
<td>Python</td>
<td>3.548%</td>
<td>-0.24%</td>
</tr>
<tr>
<td></td>
<td>C#</td>
<td>3.457%</td>
<td>-1.02%</td>
</tr>
<tr>
<td></td>
<td>Visual Basic .NET</td>
<td>3.391%</td>
<td>+1.07%</td>
</tr>
<tr>
<td></td>
<td>JavaScript</td>
<td>3.071%</td>
<td>+0.73%</td>
</tr>
<tr>
<td></td>
<td>Assembly language</td>
<td>2.859%</td>
<td>+0.98%</td>
</tr>
<tr>
<td></td>
<td>PHP</td>
<td>2.693%</td>
<td>-0.30%</td>
</tr>
<tr>
<td></td>
<td>Perl</td>
<td>2.602%</td>
<td>+0.28%</td>
</tr>
</tbody>
</table>

- RPG is 45, COBOL is 25, ugh...
Some reasons folks use Python

• Open source is free, right?
  – Yes, no... well – sort of...
  – No charge LP on IBM i – 5733OPS

• Only scripting language to support multiple inheritance

• Multi-threading
  – Stateful, not Async like Node
  – But same motivation as Node, non-blocking

• Applications develop faster
  – Extensive standard library that is constantly evolving
  – Frameworks such as Bottle

```python
from multiprocessing.dummy import Pool as ThreadPool
pool = ThreadPool(4)
results = pool.map(my_function, my_array)
```
Industries using Python

• Technology
  – Geospatial, Machine Learning
  – Honeywell - Build automation

• Manufacturing
  – D-Link updates – Multithreaded
  – Phillips – Robot programming and sequencing

• Entertainment
  – ILM – Scripting CGI Intensive films
  – Video gaming

• Transportation
  – Airports – Frequentis for weather info
    • US, Denmark, Iceland, Hong Kong, etc.
Companies using Python

- YouTube - Backend
- Facebook – (see next slide)
- Dropbox
- NASA – Weather and more
- IBM
- Mozilla
- Instagram
- Red
- Quora
  - Why python?

"Adam D’Angelo, wrote a lot of Python for Quora
Updated Sep 12, 2014 · Upvoted by Charlie Cheever, One of the founders of Quora and Scott Danzig, avid Quora user since 2012, and a Quora Top Writer

Python was a language that Charlie and I both knew reasonably well (though I know it a lot better now than I did when we started). We also briefly considered C#, Java, and Scala. The biggest issues with Python are speed and the lack of typechecking."
Facebook in detail

- Top three languages
  - Hack (fork of PHP)
  - C++
  - Python
- Infrastructure management
  - Network switch setup
  - Core services (DNS, etc.)
  - Server imaging, burnin-in.
- Platform services: Job Engine, team workflow management

https://code.facebook.com/posts/1040181199381023/python-in-production-engineering/
Got Python?
Details at Developerworks…


---

Open Source Technologies on IBM i

<table>
<thead>
<tr>
<th>Package</th>
<th>Option 2 PTF</th>
<th>Option 4 PTF</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>em_db</td>
<td>572135</td>
<td>5803567</td>
<td>DB2 for i</td>
</tr>
<tr>
<td>iserlet</td>
<td>572154</td>
<td>580356B</td>
<td>Isolated for IBM i allows access to system resources through program calls, SQL, queries, CL commands, shell commands, and more</td>
</tr>
<tr>
<td>ftprep</td>
<td>572156</td>
<td>580356B</td>
<td>FastCGI gateway</td>
</tr>
<tr>
<td>iotile</td>
<td>572156</td>
<td>580356B</td>
<td>Lightweight web framework</td>
</tr>
</tbody>
</table>

Add-on packages are located in /gcppxya/yten/rxrodata/rxsm/python-pkgs. They must be installed in order to be available. See Installing add-ons.
Which one?

- Python 3 is LPP option 2
- Python 2 is LPP option 4
- Correct answer: It depends...
  - Many existing libraries are Python 2
  - But 90%+ are also Python 3 compliant, or on their way
- Python 2.7 will retire in less than 3 years (EOL==2020)
Need Licensed Programs

- 5733OPS Base and option 2 or 4
Python in action

- Command line via green screen (CALL QP2TERM)
Hello world?

```bash
$ python3
Python 3.4.4 (default, Mar 23 2016, 11:07:11)
[GCC 4.8.4] on aix6
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello Mikey")
Hello Mikey
```
Most prefer SSH

- Command line via SSH terminal
Hello World, again?

```
$ python3
Python 3.4.4 (default, Mar 23 2016, 11:07:11)
[GCC 4.8.4] on aix6
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello Mikey!")
Hello Mikey!
```
IDE?
Zend Studio

• No, you don’t need to buy Zend Studio
• Use Orion, etc.
• But if you have Studio or RDi...
  – Consider something from Eclipse.org
  – I grabbed PyDev
**PyDev - Python IDE for Eclipse**

<table>
<thead>
<tr>
<th>Details</th>
<th>Metrics</th>
<th>Errors</th>
<th>External Install Button</th>
</tr>
</thead>
</table>

PyDev is a plugin that enables Eclipse to be used as a Python IDE (supporting also Jython and IronPython).

It uses advanced type inference techniques which allow it to provide things such as code completion and code analysis, besides providing a debugger, interactive console, refactoring, tokens browser, django integration, etc.

**Homepage:**

pydev.org

**Getting Started:**

Getting Started (read to make sure you can get most out of PyDev)

**Feature Matrix:**

pydev.org/manual_adv_features.html
Alternatives to IBM i when learning

- What's that? The boss won't let you install Python?
  - Consider repl.it
Alternatives to IBM i when learning

• How about your PC?
• Goto the Python site:
  – Download
  – Install
  – Viola!

Python 3.6 (32-bit)

Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("I unclog my nose in your direction, sons of a window dresser.")
I unclog my nose in your direction, sons of a window dresser.
>>>
Download PyDev from Eclipse

PyDev - Python IDE for Eclipse

PyDev is a plugin that enables Eclipse to be used as a Python IDE (supporting also Jython and IronPython).

It uses advanced type inference techniques which allow it to provide things such as code completion and code analysis, besides providing a debugger, interactive console, refactoring, tokens browser, django integration, etc.

Homepage:

pydev.org

Getting Started:

Getting Started (read to make sure you can get most out of PyDev)

Feature Matrix:
Capture URL

- Follow prompts
Editor for Eclipse

- Select what you like
- Next
Confirm versions

- Next
Accept terms and EULA

- Finish
Watch the pretty status bar

[Image of a software installation window with a progress bar, a checkbox for "Always run in background", and buttons to "Run in Background", "Cancel", and "Details >>"]
Python in Eclipse (i.e. Zend Studio)

- I bet RDi works, too!
Hello world….again…

- Create a file like Ex01hello.py
- Open the file
- Key up some code and click save

```python
# Hello World???
#
print("Hello Mikey!!!")
```

```
$ python3 /home/mpavlak/python/Ex01hello.py
Hello Mikey!!!
$`
Hello world....again...

- Change the file
- Click save
- Back to qp2term & F9

```python
# Hello World???
#
# print("Hello Mikey!!!")
#
print("\n\nHello Mikey!!! \nTry the spam! \n\n")
```

```
> python3 /home/mpavlak/python/Ex01hello.py
Hello Mikey!!!
$
> python3 /home/mpavlak/python/Ex01hello.py

Hello Mikey!!!
Try the spam!
$
```
Syntax
How is it written

• Indentation means EVERYTHING
  – Don’t use tab
  – 4 spaces is the best practice
  – Mismatched indents can cause failures. Good luck finding...
  – Mismatched spaces and tabs will cause failures
• No need for scope terminators like other languages
• Colon introduces start block, then indent
• Much more readable than other languages
• Get a good editor!!!
# Indentation

```python
# Indentation example
#
count = 0
terminate = True
while count < 2:
    if terminate:
        print("This is an argument")
    else:
        print("No, it isn't ")
    terminate = False
    count = count + 1
```

```
$ python3 Ex03Indents.py
This is an argument
No, it isn't
```
Operators – Similar to Java, PHP, C, etc.

- Comparison
  - Assignment =
  - Comparison ==
  - Inequality !=
  - Less than <
  - Greater than >
  - Less than or equal to <=
  - Greater than or equal to >=

- Mathematical
  - Addition +
  - Multiplication *
  - Division /
  - Floor division //
  - Modulus %
  - Exponentiation **

- Booleans
  - And
  - Or
  - Not

\[ Vi = Ao \sqrt{5 \left[ \left( \frac{Qc}{Po} + 1 \right)^{\frac{2}{7}} - 1 \right]} \]

*The air speed velocity of an unladen swallow.*
Variables
Data Types – yeah…about that

- **Int**
  - Integer of unlimited size
- **Float**
  - System defined precision
- **Complex**
  - Complex with real and imaginary parts
- **Bool**
  - TRUE & FALSE
Built in types

• Str
  – Character string composed of Unicode
• Bytes and bytearray
  – Sequences of bytes
• List and tuple (list/array/data structure)
• Range
  – Start, end, step
• Set & frozenset
  – Unordered set of terms
• Dict
  – Associative array (dictionary, hash map)
Variables on the fly

- Case sensitive
- camelCase
- Who are you? `type()`

```python
>>> print(frenchNight = "Your mother was a hamster and your father smelt of elderberries")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'frenchNight' is not defined
>>> print(frenchNight)
Your mother was a hamster and your father smelt of elderberries
>>> pi = 3.141
>>> print(pi)
3.141
>>> type(pi)
<class 'float'>
>>> type(frenchNight)
<class 'str'>
```
Variables in a file

```python
# Variables are defined on the fly...

frenchKnight = "Your mother is a hamster and your father smelt of elderberries"
pi = 3.14159

print(frenchKnight)
print(pi)
```

```
$ python3 Ex02Variables.py
Your mother is a hamster and your father smelt of elderberries
3.14159
```

# Variables are defined on the fly...

```python
frenchKnight = "Your mother is a hamster and your father smelt of elderberries"
pi = 3.14159

print(frenchKnight)
print(pi)

print("The type of frenchKnight is: ", type(frenchKnight))
print("The type of pi is: ", type(pi))
```

```
$ python3 Ex02Variables.py
Your mother is a hamster and your father smelt of elderberries
3.14159
The type of frenchKnight is: <class 'str'>
The type of pi is: <class 'float'>
```

© 2017 Rogue Wave Software, Inc. All Rights Reserved.
Every variable is implemented as a class!

RUN AWAY!!!
And now for something completely different
It’s OK....

• Monty Python references are not only acceptable...
  – They are encouraged!
• Documentation is littered with references
• Examples are well covered
Back to Variables

- Numbrs - 3 Data Types
  - Int 1,2,42
  - Float 3.14159
  - Complex: <real> + <imaginary> (not used much...)

© 2017 Rogue Wave Software, Inc. All Rights Reserved.
Strings

- Immutable objects, cannot change value
- Can reassign. (dynamic typing)
- Single or Double quotes, OK (even triple...)
- Index starts at 0
String formatting

- Interpolation, of sorts

```python
# String example
#
# count = 0
while count < 6:
    string1 = "I have {} dead parrots!".format(count)
    print(string1)
    count = count + 1
print("\nThank you for shopping!")
```

```bash
$ python3 Ex04Strings.py
I have 0 dead parrots!
I have 1 dead parrots!
I have 2 dead parrots!
I have 3 dead parrots!
I have 4 dead parrots!
I have 5 dead parrots!
Thank you for shopping!
$```

© 2017 Rogue Wave Software, Inc. All Rights Reserved.
Lists

- Ordered group, similar to array
- Different data types, ok
- Multi-dimensional (sub lists)
- Mutable (changeable)

```python
# List ExampleService
#
mylist = ['Rock Bottom', 'Gordon Biersch', 'BJ’s', 'Granite City']

print(mylist[1])

print(mylist[0:2])

print(mylist)
```

```bash
$ python3 Ex05Lists.py
Gordon Biersch
['Rock Bottom', 'Gordon Biersch']
['Rock Bottom', 'Gordon Biersch', 'BJ’s', 'Granite City']
```
Tuples

• Similar to lists
• Immutable (don’t change once created)
• Use parenthesis instead of brackets

```python
# Tuples Examples
#
mytup = ("Good", "Beer", "Makes", "you", "smart")
print(mytup[1])
print(mytup)
```

```
$ python3 Ex06tuples.py
Beer
('Good', 'Beer', 'Makes', 'you', 'smart')
```
Dictionary

- Again, like lists but more like hash table
- Mutable
- Key value pairs

```python
# Dictionary Examples

myDict = {"Sam Adams": "Good", "Samuel Smith": "Best", "Bud light": "Bad"}

print("myDict['Sam Adams']": "", myDict['Sam Adams'])

print(myDict.keys())
print(myDict.values())
print(myDict.items())
```

```
$ python3 Ex07Dictionary.py
myDict['Sam Adams']: Good
dict_keys(['Bud light', 'Samuel Smith', 'Sam Adams'])
dict_values(['Bad', 'Best', 'Good'])
dict_items([('Bud light', 'Bad'), ('Samuel Smith', 'Best'), ('Sam Adams', 'Good')])
```
Control Structures
# Ifs

```python
# If examples

a, b = 3, 42
print(a, b)
if a < b:
    print("a is smaller")

a, b = 42, 3
print("\n", a, b)
if a < b:
    print("a is smaller")
else:
    print("b is smaller")

a, b = 3, 3
print("\n", a, b)
if a < b:
    print("a is smaller")
elif a > b:
    print("b is smaller")
else:
    print("a and b are the same")
```

```
$ python3 Ex10ifs.py
3 42
a is smaller

42 3
b is smaller

3 3
a and b are the same
```

# For Loop Examples

```python
# This example demonstrates the use of for loops in Python.

myString = "Holy Grail"
for letter in myString:
    print("this letter is ", letter)

beers = ["Sam Adams", "Samuel Smith", "Goose Island"]
for beer in beers:
    print("this is a good beer: ", beer)

badBeers = ["Bud", "Bud Light", "Miller Lite"]
for index in range(len(beers)):
    print("this is a bad beer: ", badBeers[index])
```

The output of the program is as follows:

```
this letter is o
this letter is l
this letter is y
this letter is G
this letter is r
this letter is a
this letter is i
this letter is l
this is a good beer: Sam Adams
this is a good beer: Samuel Smith
this is a good beer: Goose Island
this is a bad beer: Bud
this is a bad beer: Bud Light
this is a bad beer: Miller Lite
```
# While Loop Examples

```python
# count, limit = 0, 5
while count < limit:
    count = count + 1
    print("Number is", count)

count = 0
while count < limit:
    count = count + 1
    if count == 3:
        break
    print("Break Number is", count)

count = 0
while count < limit:
    count = count + 1
    if count == 2:
        continue
    print("Continue Number is", count)
```
Functions
Basic functions

```python
# Function Examples
#

def printBeer(store, beer, size):
    print(store + " has " + beer + " in a max sized of " + str(size))

myBeer = "Sam Adams"
printBeer("Walgreens", myBeer, 12)
printBeer("BevMo", myBeer, 24)
printBeer("Costco", myBeer, 28)
```

```bash
$ python3 Ex15Functions.py
Walgreens has Sam Adams in a max sized of 12
BevMo has Sam Adams in a max sized of 24
Costco has Sam Adams in a max sized of 28
```

Functions with defaults

```python
# # Function Examples
#

def printBeer(store, beer, size=24):
    print(store + " has " + beer + " in a max sized of " + str(size) )

myBeer = "Sam Adams"
printBeer("Walgreens", myBeer, 12)
printBeer("BevMo", myBeer)
printBeer("Costco", myBeer, 28)
```

```
$ python3 Ex16Functions2.py
Walgreens has Sam Adams in a max sized of 12
BevMo has Sam Adams in a max sized of 24
Costco has Sam Adams in a max sized of 28
$  
```
Functions with keyword arguments

```python
# Function Examples
#
def printBeer(store, beer, size):
    print(store + " has " + beer + " in a max sized of " + str(size))

myBeer = "Sam Adams"
printBeer("Walgreens", myBeer, 12)
printBeer(beer=myBeer, size=24, store="BevMo")
printBeer(beer=myBeer, store="Costco", size=28)
```

```bash
$ python3 Ex17Functions3.py
Walgreens has Sam Adams in a max sized of 12
BevMo has Sam Adams in a max sized of 24
Costco has Sam Adams in a max sized of 28
$ ```
Command Line
Input from command line

• Talk with the script

```python
# Get input from user and then embed in string
from pip._vendor.distlib.compat import raw_input

name = raw_input("\nWhat is your name? ")
age = raw_input("\nHow old are you? ")
city = raw_input("\nIn what city were you born? ")
print("\n\n**************************")
print("Hello %s" % (name))
print("You were born in %s about %s years ago." % (city, str(age)))
print("\n\nThank you for playing...")
```

```
$ python3 Ex18ComLine.py
What is your name? Mike
How old are you? 29
In what city were you born? Chicago

**************************
Hello Mike
You were born in Chicago about 29 years ago.

Thank you for playing...
```
Database
Locate the package or "wheel"
Install commands

Installing shipped add-ons

5733-OPS Option 2 and Option 4 come with several add-on packages (shipped via separate PTFs). Installation of these add-ons is easy, just use the applicable command.

If you're on a recent PTF level, all the packages should now be in wheel format (*.whl). Previous versions used egg format (*.egg). If you want to know the nitty-gritty details of why wheels are better than eggs and why we switched, click this link. Otherwise, just know that wheels are better in every way except name.

New way, with wheels:

(for Python 3)

To install the native DB2 connector:

```
pip3 install /QOpenSys/QIBM/ProdData/OPS/Python-pkgs/ibm_db/ibm_db-*-cp34m--*.whl
```

To install the DB2 Django interface:

```
pip3 install --no-deps /QOpenSys/QIBM/ProdData/OPS/Python-pkgs/ibm_db/ibm_db_django-*-py3--*.whl
```

To install the Toolkit for IBM I:

```
pip3 install /QOpenSys/QIBM/ProdData/OPS/Python-pkgs/itoolkit/itoolkit-*-cp34m--*.whl
```

To install FastCGI gateway support:

```
pip3 install /QOpenSys/QIBM/ProdData/OPS/Python-pkgs/flipflop/flipflop-*-py34--*.whl
```
Find the connector

- YMMV
- With wheels
Run the pip install...

```
$ pip3 install /QOpenSys/QIBM/ProdData/OPS/Python-pkgs/ibm_db/ibm_db-2.0.5.5-cp34-cp34m-os400_powerpc.whl
Processing /QOpenSys/QIBM/ProdData/OPS/Python-pkgs/ibm_db/ibm_db-2.0.5.5-cp34-cp34m-os400_powerpc.whl
Requirement already satisfied (use --upgrade to upgrade): six in /QOpenSys/QIBM/ProdData/OPS/Python3.4/lib/python3.4/vendor-packages/six-1.10.0-py3.4.egg (from ibm-db==2.0.5.5)
Installing collected packages: ibm-db
  Found existing installation: ibm-db 2.0.5.4
  Uninstalling ibm-db-2.0.5.4:
    Successfully uninstalled ibm-db-2.0.5.4
Successfully installed ibm-db-2.0.5.5
```

[j3a] You are using pip version 8.1.1, however version 9.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command. [0m

$
What version of the DB2 Extension?

```python
import ibm_db_dbi as dbi
print(dbi.__version__)
```

```
> python3 /home/mpavlak/python/db2/db2ex01.py
2.0.5.5
>$
```
Simple database access

• Import the class
• Connect (with or without options
• Open the cursor
• Set the SQL
• Read
Simple database access

```python
import ibm_db_dbi as dbi
conn = dbi.connect()
sql = "SELECT COMPANY, COUNTRY FROM samples.SP_CUST where country = 'US'"
c01 = conn.cursor()
c01.execute(sql)
#Bring it in as tuple
print("\n\n*****Tuple*****\n\n")

for row in c01.fetchall():
    print(row)
c01.close()
conn.close()
print("\n\n*****End*****\n\n")
```
Table information

```python
import ibm_db_dbi as dbi
conn = dbi.connect()
sql = "SELECT COMPANY, COUNTRY FROM ZENDSVR6.SP_CUST where country = 'Canada'"
c01 = conn.cursor()
c01.execute(sql)
desc = c01.description
print(desc[0][0], desc[0][4], "\n")
print(desc[1][0], desc[1][4], "\n")

#Bring it in as tuple
print("\n\n******Tuple*****\n\n")
for row in c01.fetchall():
    print(row)
c01.close()
conn.close()
print("\n\n******End*****\n\n")
```

mvpower.morainevalley.edu - PuTTY

```bash
$ python3 db2ex04.py
COMPANY 30
COUNTRY 20

******Tuple*****
```
Summary – Why Python?

- Lot’s of libraries
- Make it easy to do stuff
- OPC / OPO
- Education
End the session

```
$ python3
>>> Python 3.4.4 (default, Mar 23 2016, 11:07:11)
   [GCC 4.6.4] on aix6
   Type "help", "copyright", "credits" or "license" for more information.
   >>>
   >>> print("Hello Mikey")
   Hello Mikey
   >>>
   >>> quit()
   $ 
```

```bash
>>> -
```

F3=Exit   F6=Print   F9=Retrieve   F11=Truncate/Wrap
F13=Clear F17=Top   F18=Bottom   F21=CL command entry

MO1 A 21/087
```
ATTEND
Connect with experts. Become an authority. Register now: zendcon.com/register-now

EXPLORE
Share, inspire, educate. Network

SPONSOR
Spotlight your best in enterprise PHP, open source & frameworks: sponsors@zendcon.com

Visit zendcon.com
Who was that masked man?

Mike Pavlak – Solution Consultant, Rogue Wave Software

Mike.Pavlak@roguewave.com

Thank You!