FRESCHE SOLUTIONS

Introduction to Python for IBM i

Mike Pavlak – IT Strategist

NHMUG
New Hampshire
Midrange User Group

Corporate Overview

40+ years in business

- Management and employee owned
- 300+ employees
- 18 yr average employee tenure
- Named one of Canada's top small and medium employers

22,000+ customers

- 8 offices around the world
- North America, Europe, Latin America and Asia Pacific
- Over 200 partners/resellers

Application Management and Modernization Experts

- 15+ years application optimization, integration, modernization expertise
- 40 years of IBM i product development

Innovative Products and Services

- Web, mobile and GUI modernization
- Analysis & productivity
- Application & database modernization

FRESCHE SOLUTIONS | BCD looksoftware QUADRANT SOFTWARE

Agenda

- A little about Python
- Why use Python
- How to install/determine if installed
  - IDE
- Syntax 101
  - Variables
  - Strings
  - Functions

Acknowledgements

- Kevin Adler
- Tony Cairns
- Jesse Gorzinski
- Google
- Memegenerator
- Corn chips and salsa
- Parrots
- And, of course, spam

Before you freak out

- Why isn't Mike talking about PHP?
  - Zend was the PHP company
  - Rogue Wave is the Open Source company
  - Fresche Solutions is the IBM i solutions company!

A little about Python

What is it, really?

- General purpose language
- Easy to get started
- Simple syntax
- Great for integrations (glue between systems)
- Access to C and other APIs
- Infrastructure first, but applications, too

Historically…

- Python was conceptualized by Guido Van Rossum in the late 1980's
- Rossum published the first version of Python code (0.9.0) in February of 1991 at the CWI(Centrum Wiskunde & Informatica) in the Netherlands, Amsterdam
- Python is derived from the ABC programming language, which is a general purpose language that was also developed at CWI.
- Rossum chose the name "Python" since he was a 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 and as leading "commitor".

The Python programming language https://www.python.org/

99,953 commits | 9 branches | 331 releases | 356 contributors

Branch: master | New pull request | Find file | Clo

haypo committed on GitHub bpo-31234: Enhance test_thread.test_forkinthread() (#3516) ··· Latest commit a15d155 5 hou

.github	Create PULL_REQUEST_TEMPLATE.md (GH-3404)	6 days ago
Doc	bpo-31421: Document how IDLE runs tkinter programs. (#3513)	10 hours ago

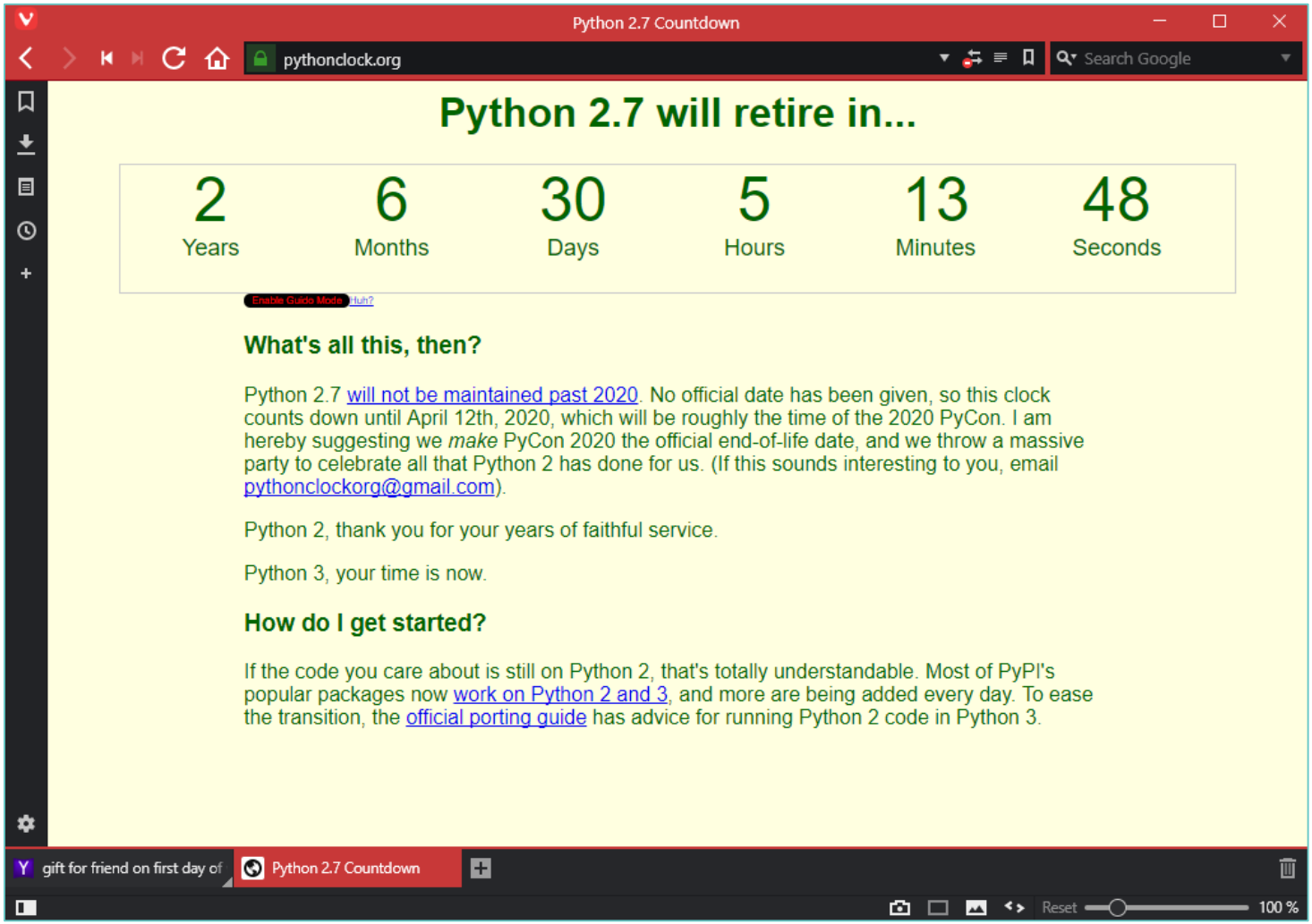
Look 4 this

Python lineage

- Python 1 – 1994
- Python 2 – 2000 (Not dead yet…)
  - 2,7 – 2010
- Python 3 – 2008
  - 3.5 – 2015
  - 3.6.2 – July 2017
  - 3.7 ➔ ETA July 2018

Release version	Release date
Python 3.4.7	2017-08-09
Python 3.5.4	2017-08-08
Python 3.6.2	2017-07-17
Python 3.6.1	2017-03-21
Python 3.4.6	2017-01-17
Python 3.5.3	2017-01-17
Python 3.6.0	2016-12-23

Python 2 or 3?


Python 2.7 Countdown
pythonclock.org
Search Google
Python 2.7 will retire in...
2 Years
6 Months
30 Days
5 Hours
13 Minutes
48 Seconds
Enable Guido Mode
Huh?
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.
gift for friend on first day of
Python 2.7 Countdown
Reset
100 %

What's the diff?

- Example:
  - Python 2 print statement replaced by function
    - Python 2 – print "Hello World!"
    - Python 3 – print("Hello World!")
- *Many more differences, tho…*
- *Which one?*
  - Correct answer: It depends…
    - Many existing libraries are Python 2
    - But 90%+ are also Python 3 compliant, or on their way

Got Python?

I.T. can make you smile

Details at Developerworks

Updated July 2017, Thanks Jesse!

- https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/IBM%20i%20Technology%20Updates/page/Open%20Source%20Technologies

Python

Python is a popular high-level programming language. It is easily extensible through the use of third-party packages and often allows powerful function to be written with few lines of code. Python caters to multiple programming styles (object oriented, procedural, etc) and the code tends to be readable and maintainable.

Python is now being delivered and packaged for IBM i. It is available through the following options:

- Option 2 - Python 3.4
- Option 4 - Python 2.7

The following add-ons are also available via separate PTFs:

Package	Option 2 PTF	Option 4 PTF	Description
ibm_db	SI57253	SI60567	DB2 for i connector - Allows native access to DB2 for i.
itoolkit	SI57254	SI60568	Toolkit for IBM i - allows access to system resources through program calls, SQL queries, CL commands, shell commands, and more.
flipflop	SI57255		FastCGI gateway
bottle	SI57256		Lightweight web framework.

Add-on packages are located in `/QOpenSys/QIBM/ProdData/OPS/Python-pkgs`. They must first be installed in order to be available. See Installing shipped add-ons.

Open Source Solutions for i Group PTF

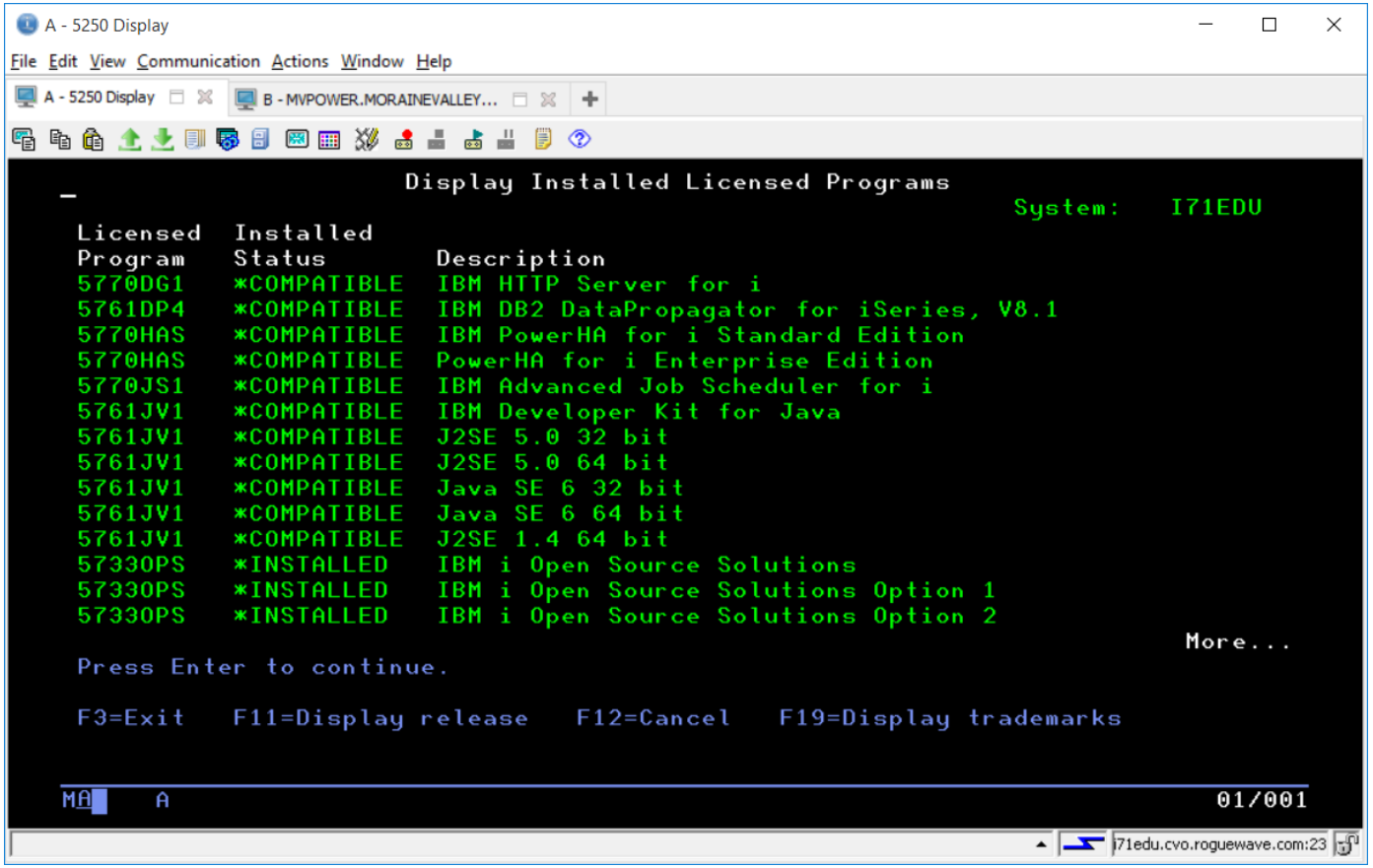
IBM i	Group PTF	Level
7.3	SF99225	5
7.2	SF99223	5
7.1	SF99123	5

Open Source Technologies on IBM i

	SAMBA on IBM i
5733-OPS Option 1	Node.js v1
5733-OPS Option 2	Python 3
5733-OPS Option 3	CHROOT
5733-OPS Option 4	Python 2
5733-OPS Option 5	Node.js v4
5733-OPS Option 6	Git
5733-OPS Option 7	Tools
5733-OPS Option 8	Orion
5733-OPS Option 9	cloud-init
5733-OPS Option 10	Node.js v6
5733-OPS Option 11	Nginx
5733-OPS Option 12	TBD
5733-OPS Option 13	TBD
5733-OPS Option 14	TBD
5733-OPS Option 15	TBD

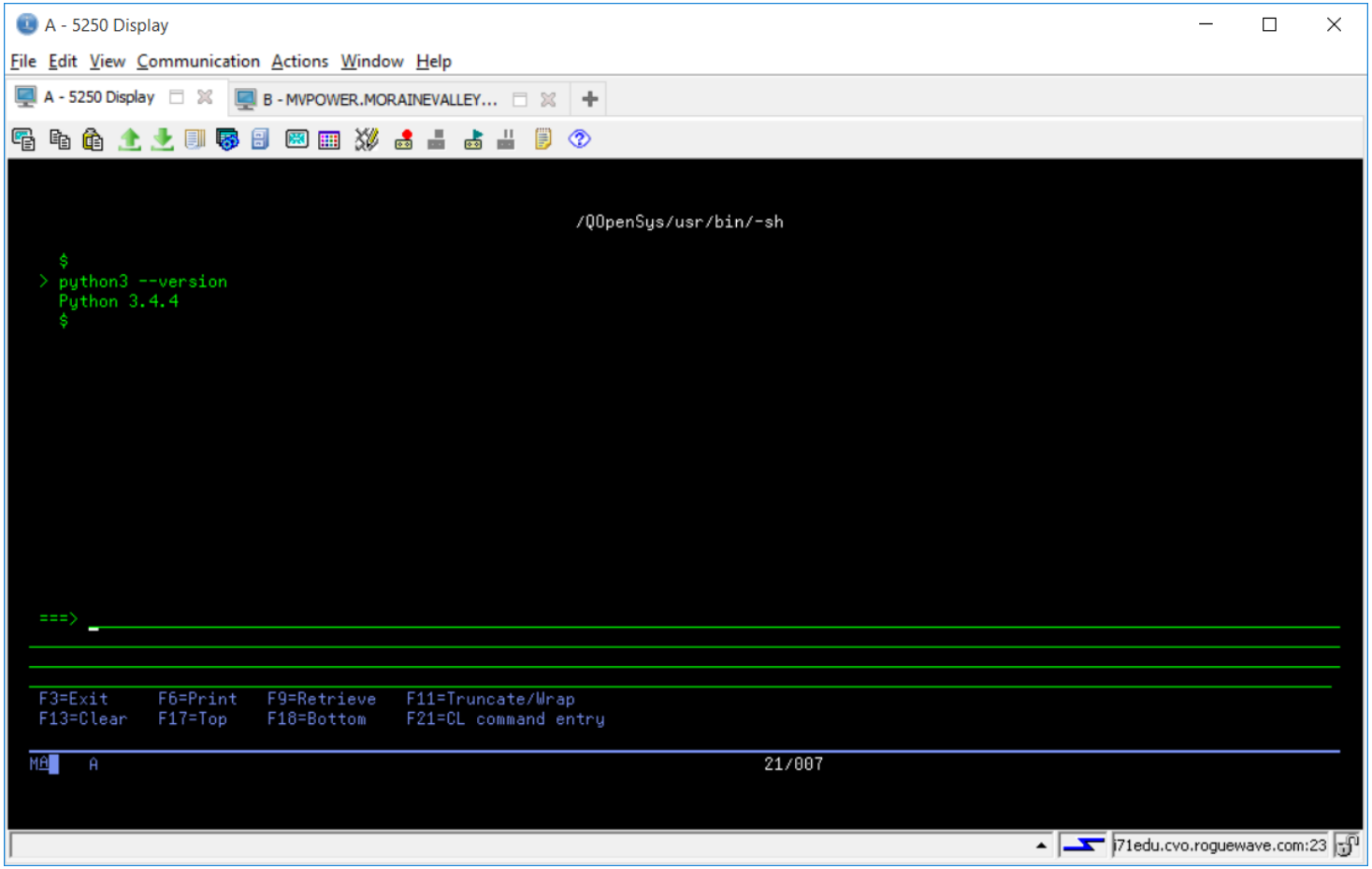
Need licensed program

- 5733OPS Base and option 2 or 4

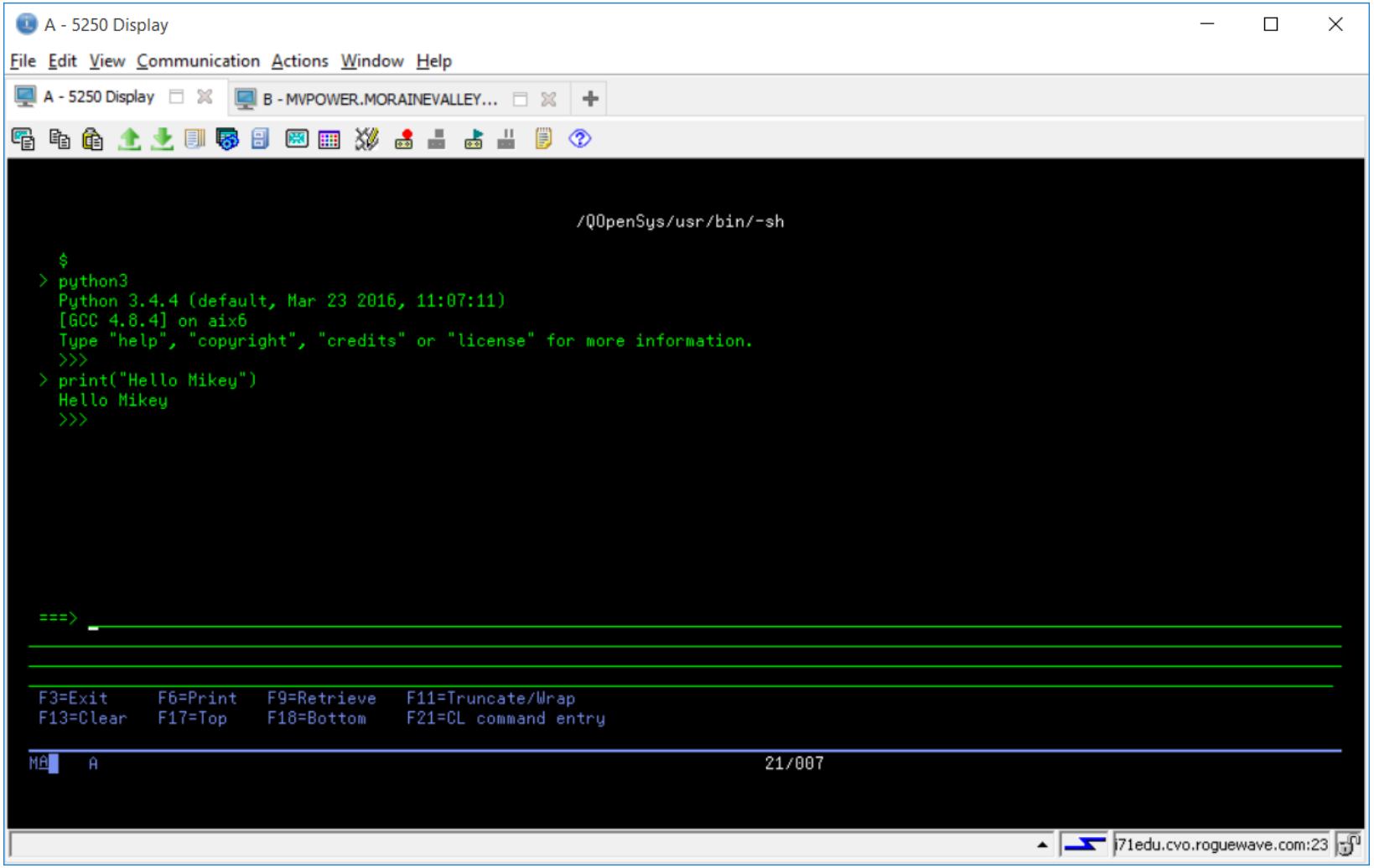


Python in action

- Command line via green screen



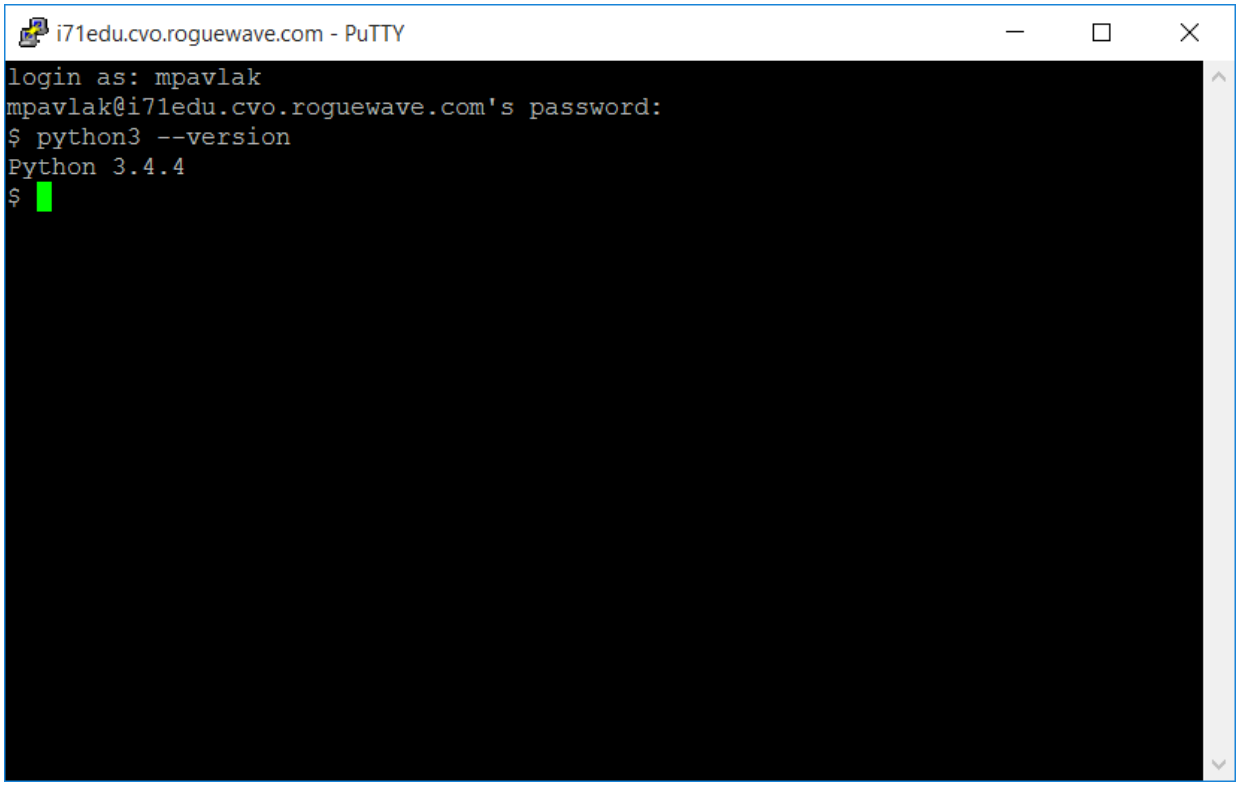
Hello World


A - 5250 Display
File Edit View Communication Actions Window Help
A - 5250 Display
B - MVPOWER.MORAINEVALLEY...
/QOpenSys/usr/bin/-sh
$
> 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
>>>
===>
F3=Exit F6=Print F9=Retrieve F11=Truncate/Wrap
F13=Clear F17=Top F18=Bottom F21=CL command entry
MA A 21/007
i71edu.cvo.roguewave.com:23

Most prefer SSH

- Command line via SSH terminal
  - Recommended strongly by Jesse!

http://ibmsystemsmag.com/blogs/open-your-i/



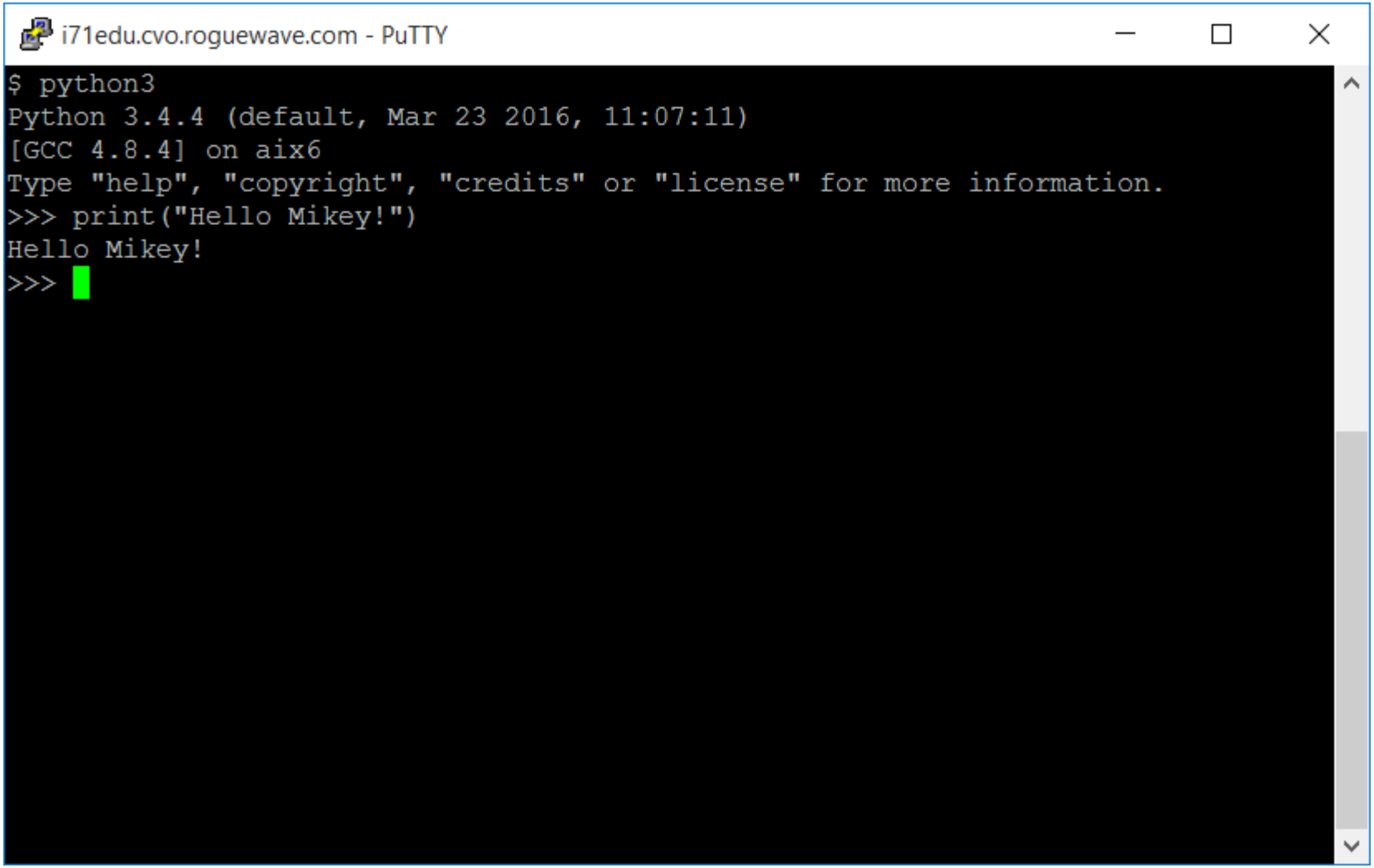
Eight Reasons to Embrace SSH

In my previous post, I gave a brief introduction to the concept of a shell and focused on SSH connectivity. Often, when we think of a command-entry interface to our IBM i system, we think of a 5250 emulator. Perhaps we also know QSHELL as an interface to run open source or other commands in the root (/) or /QOpenSys filesystems.

Read More

Posted: August 29, 2017 | 0 Comments

Hello World, again…

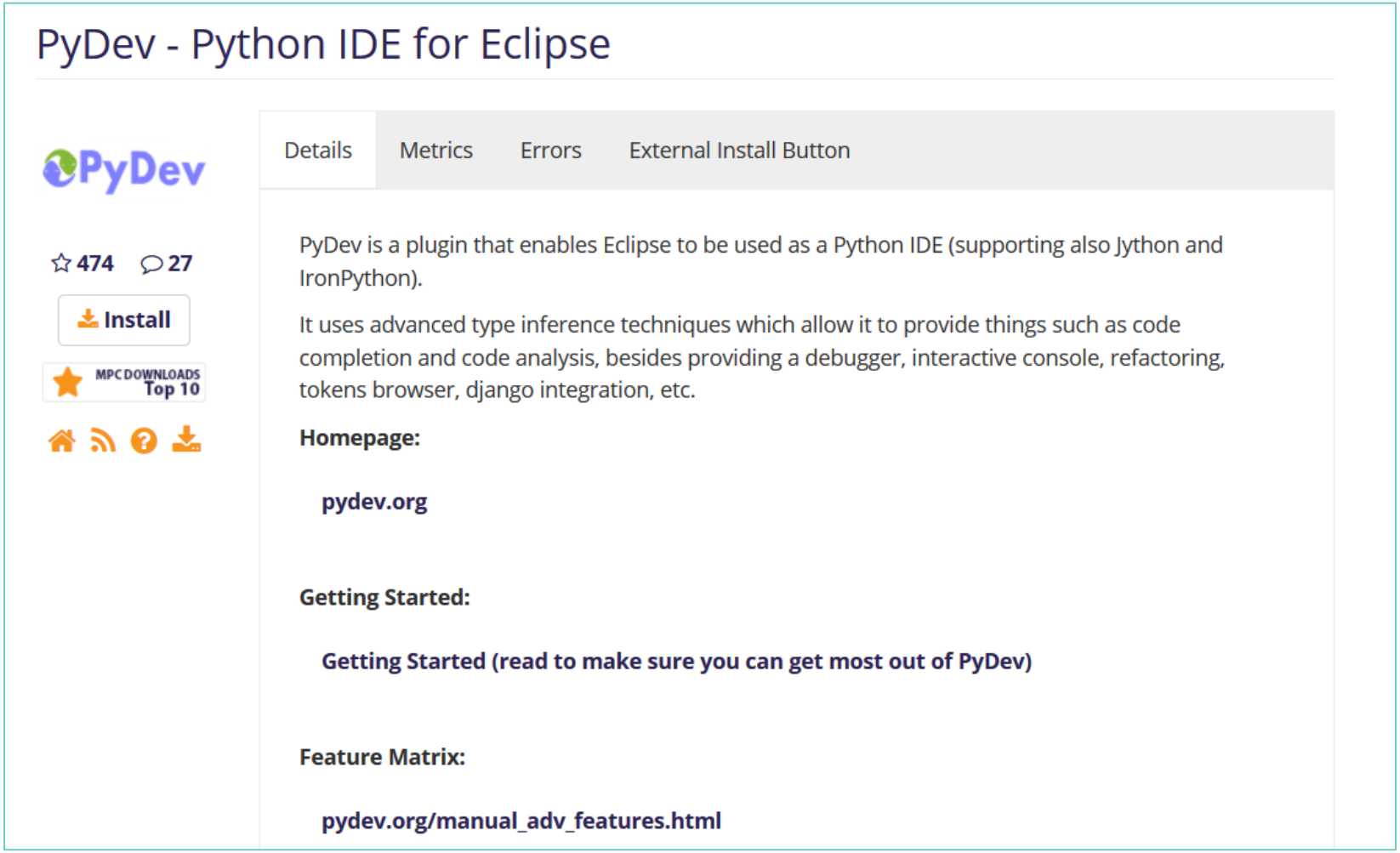


IDE

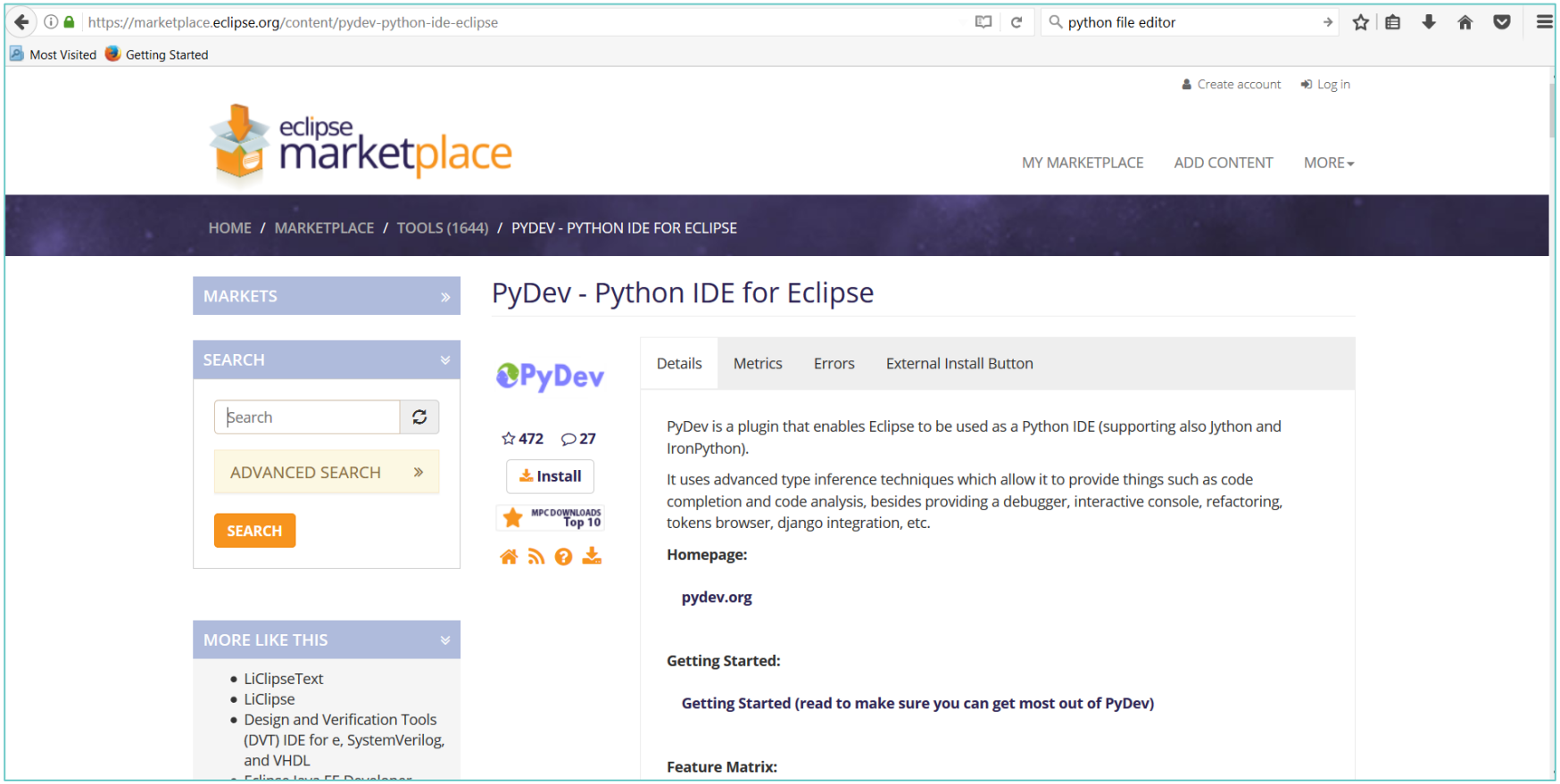
Zend Studio

- 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

Eclipse

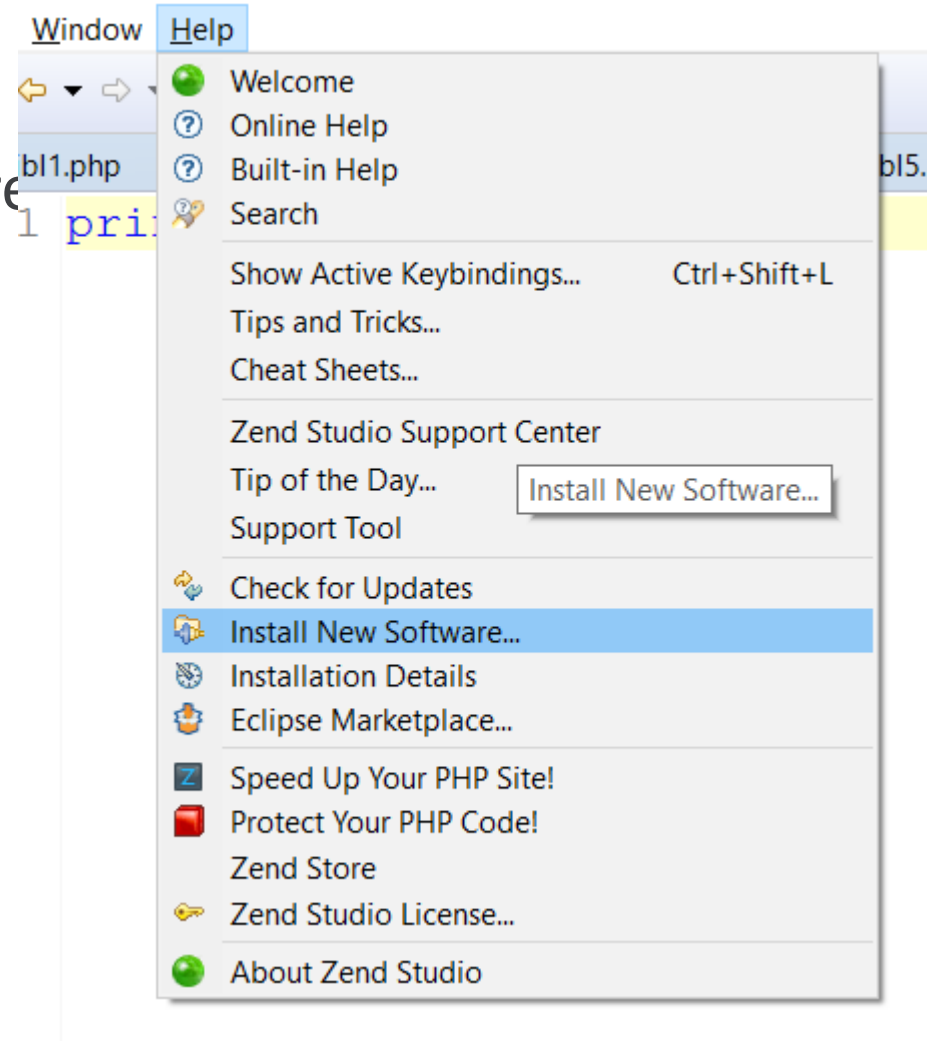

PyDev - Python IDE for Eclipse
PyDev
☆ 474
27
Install
MPC DOWNLOADS Top 10
Details
Metrics
Errors
External Install Button
PyDev is a plugin that enables Eclipse to be used as a Python IDE (supporting also Jython and IronPython).
It uses advanced type inference techniques which allow it to provide things 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

Download PyDev from Eclipse



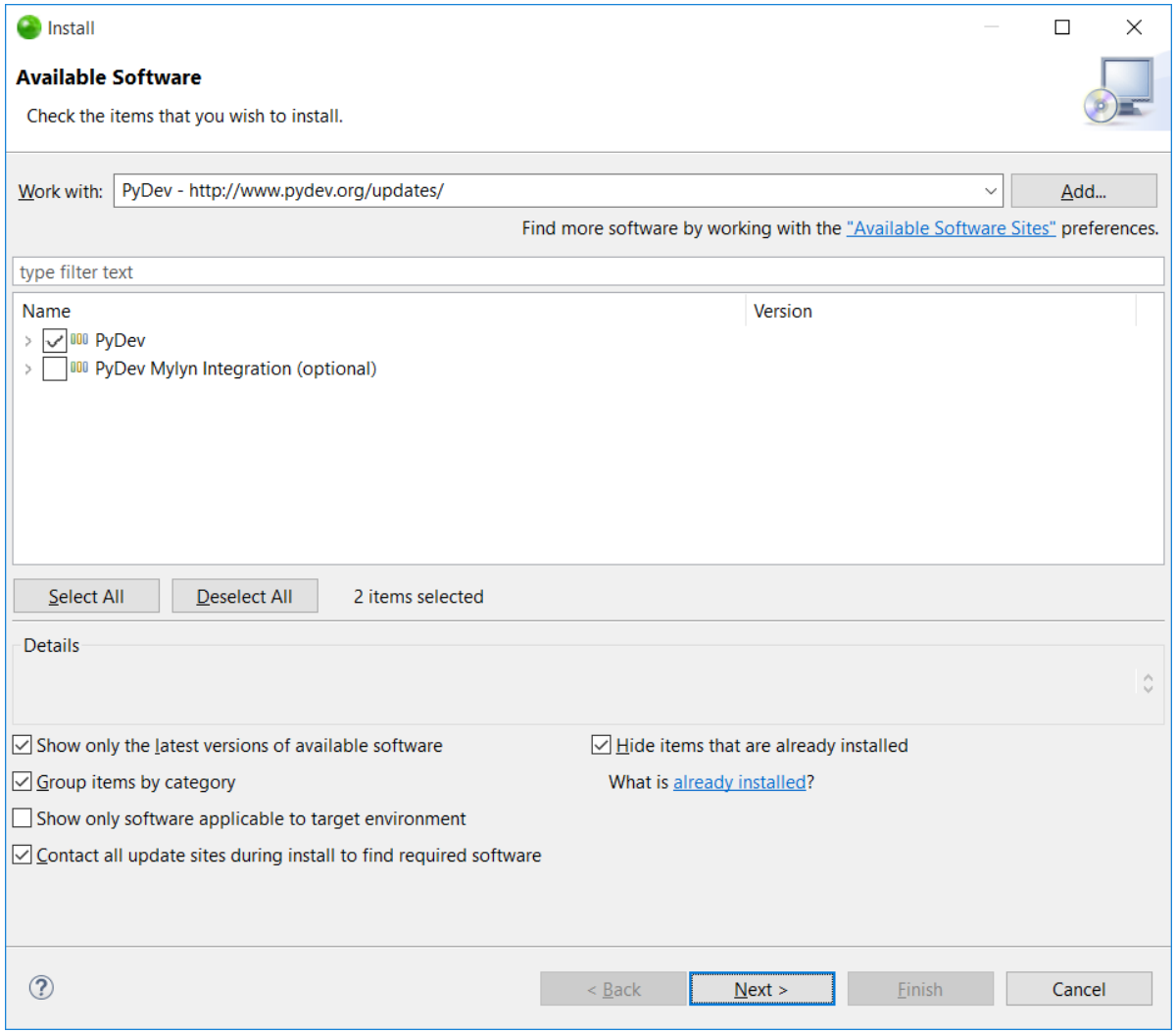
Capture URL

- Help➔
  - Install New Software
  - Follow prompts



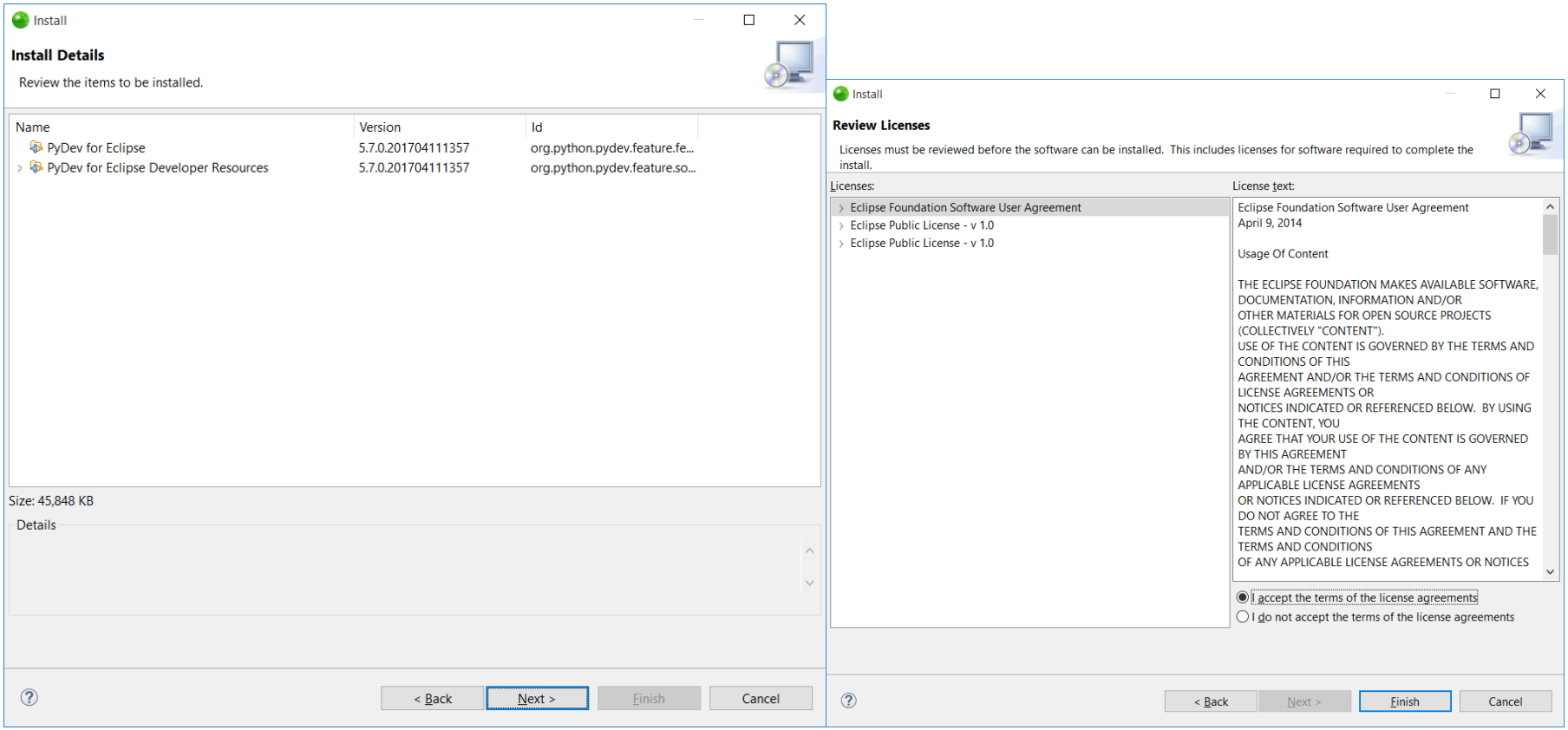
Editor for Eclipse

- Select what you like
  - Click next

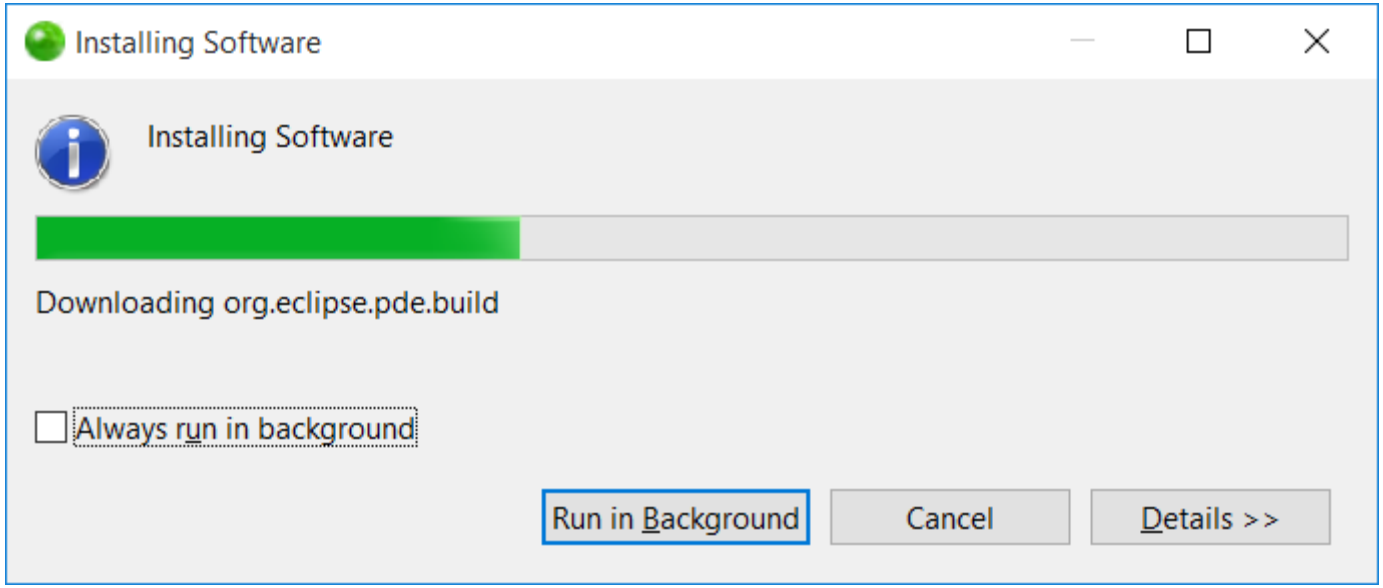


Confirm versions

- Click next again
  - Then accept EULA



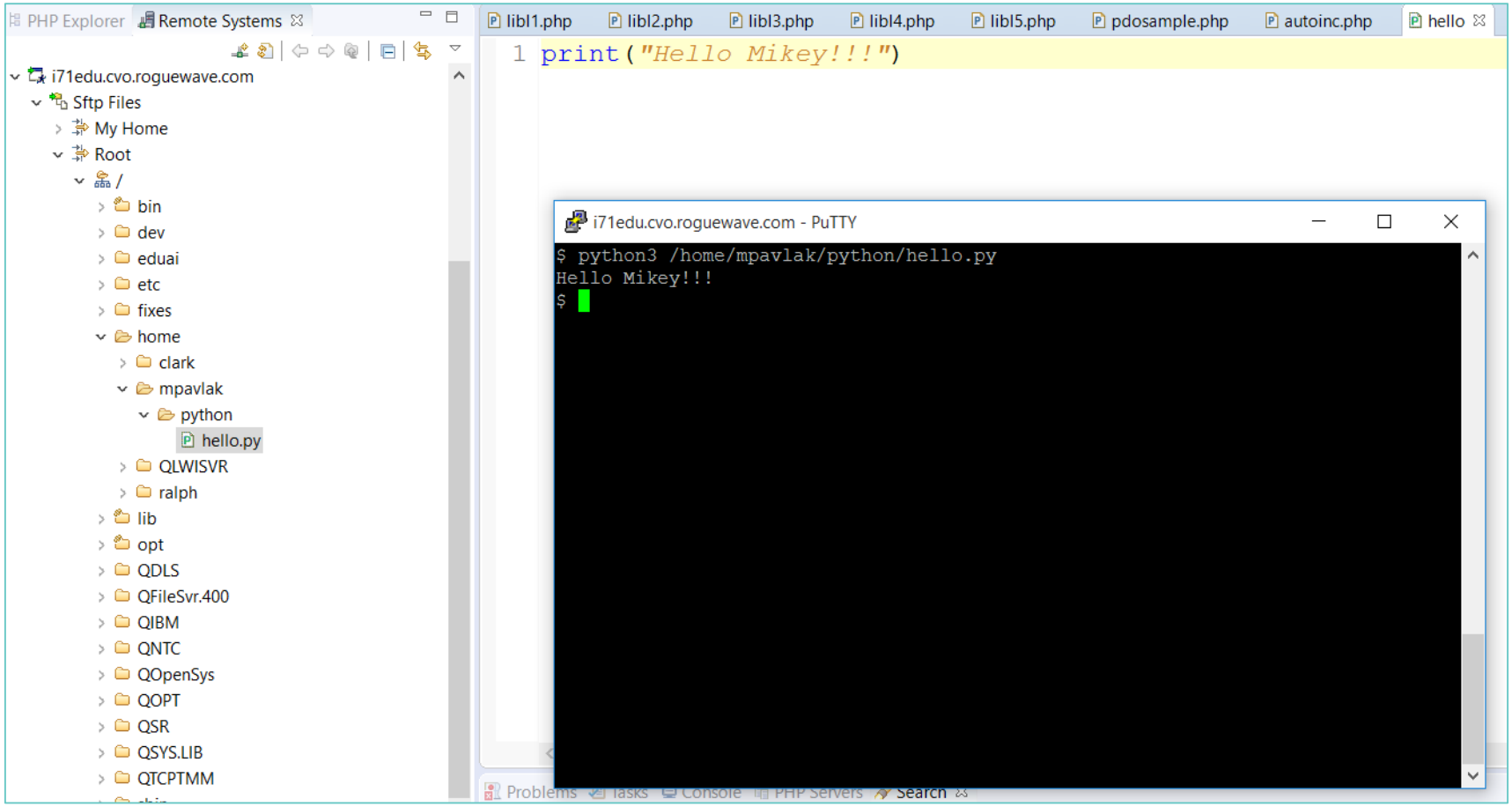
Watch the pretty status bar



skwib.com

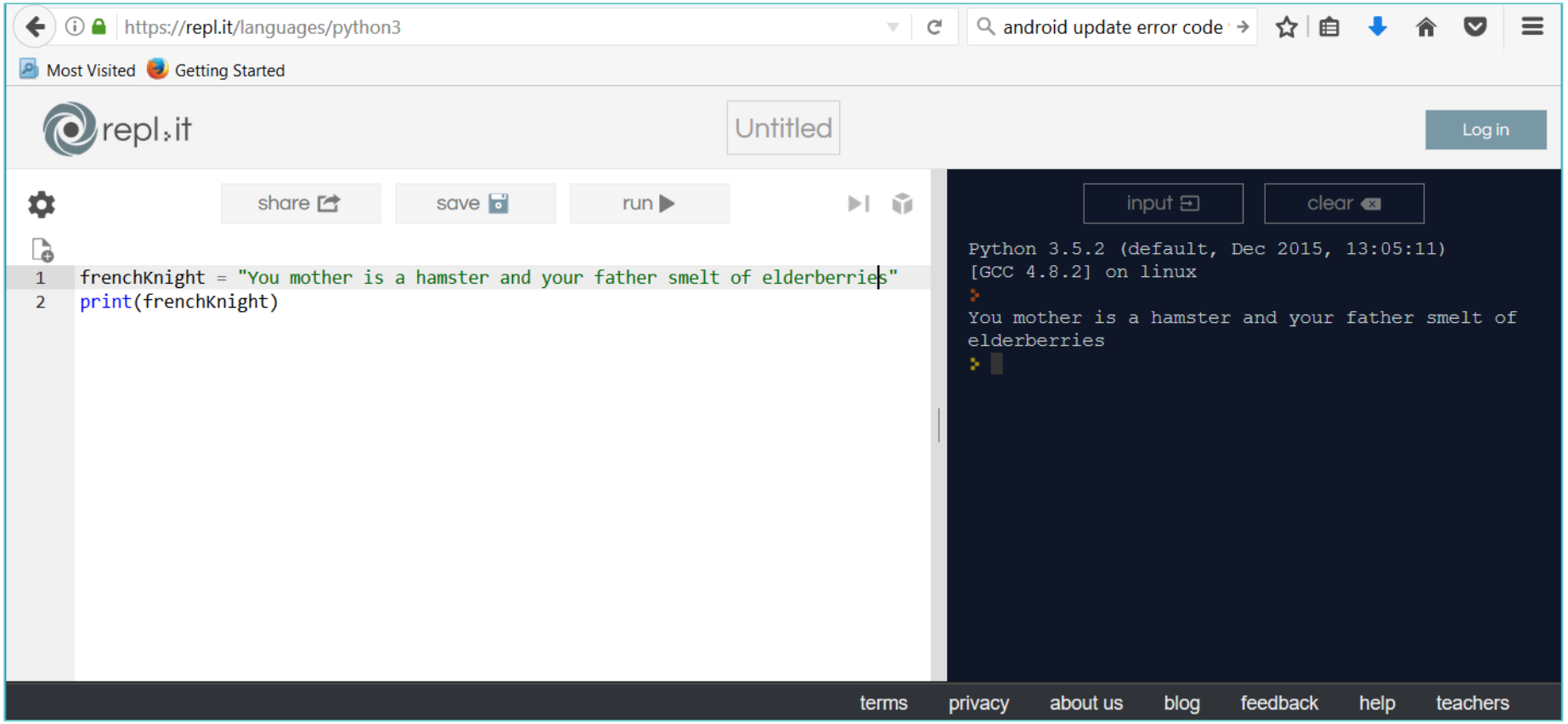
Python in Eclipse (i.e. Zend Studio)

- I bet RDi works, too!



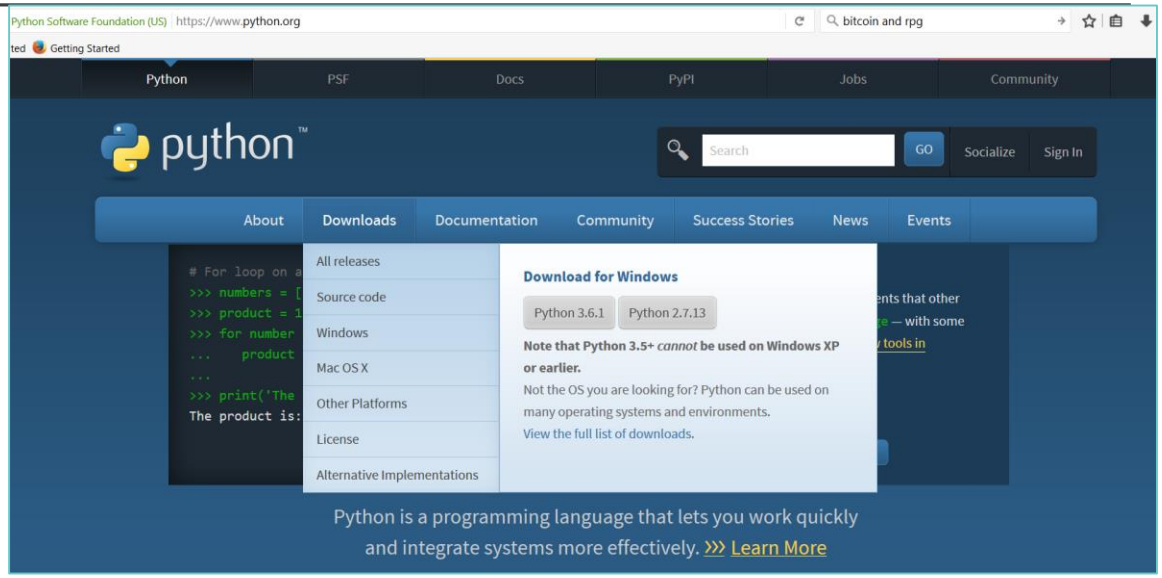
Alternatives to IBM i when learning

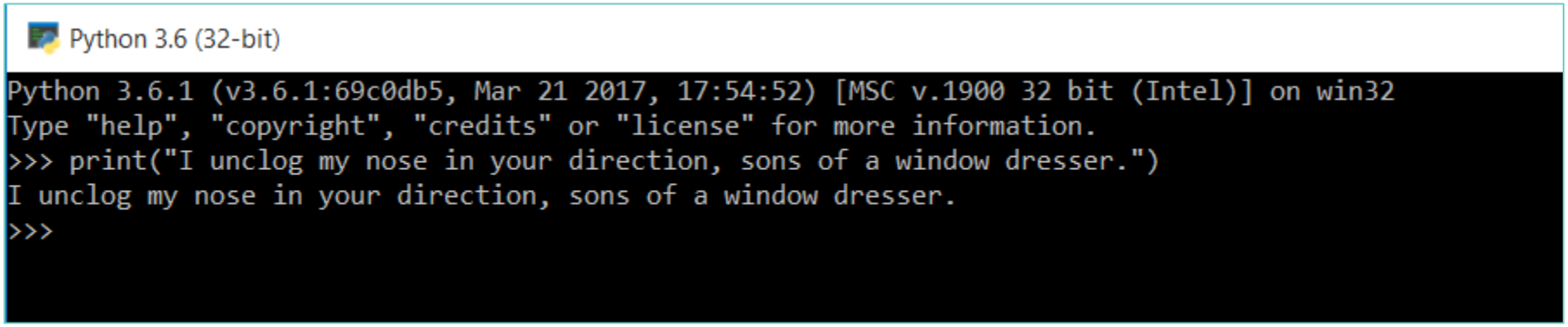
- What's that? The boss won't let you install Python on the IBM i?



Desktop education at it's finest

- How about your PC?
- Head to Python.org site:
  - Download
  - Install
  - Viola!





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

```
1 #
2 # Hello World???
3 #
4 print("Hello Mikey!!!")
```

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

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

```
1 #
2 # Hello World???
3 #
4 #print("Hello Mikey!!!")
5 #
6 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 !== sin-tax

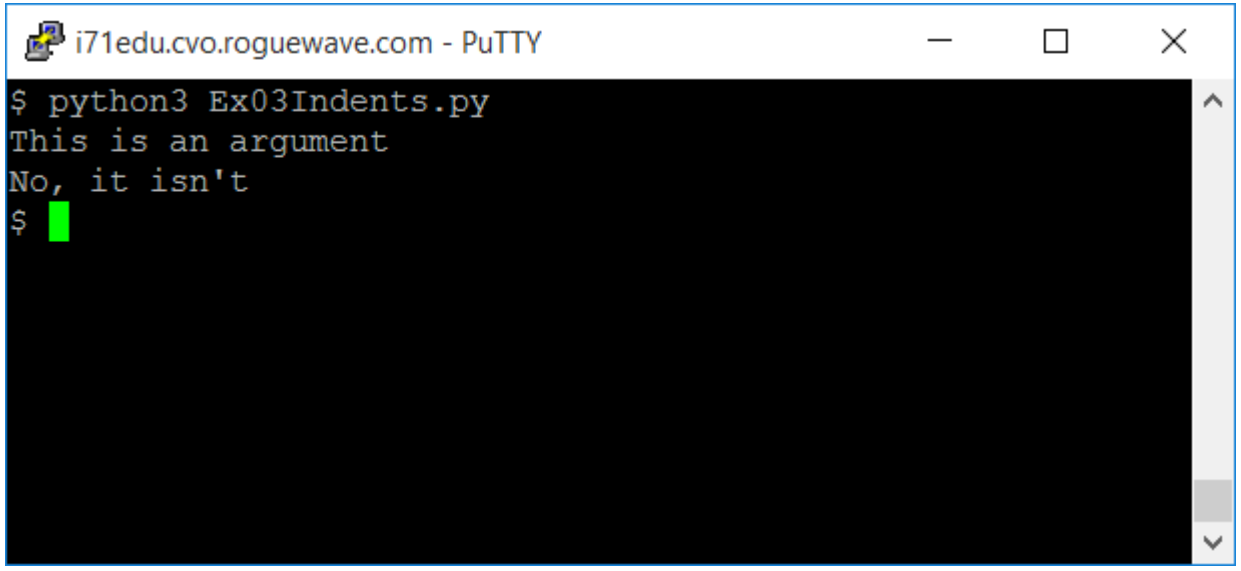
(eh, Cook county?)

How it is written

- Indentation means EVERYTHING
  - Don’t use tab (unless good editor)
  - 4 spaces – No more, no less
  - Mismatched indents can cause failures. Good luck finding…
  - Mismatched spaces and tabs will cause failures
- No scope terminators like other languages (WTF?)
- Colon introduces start block, then indent
- Much more readable than other languages
- Get a good editor!!!

Indentation

```
1  #
2  #Indentation example
3  #
4  count = 0
5  argument = True
6  while count < 2:
7      if argument:
8          print ("This is an argument")
9      else:
10             print ("No, it isn't ")
11     argument = False
12     count = count+1
```



Operators – Similar to other C derivatives

- 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

Syntax

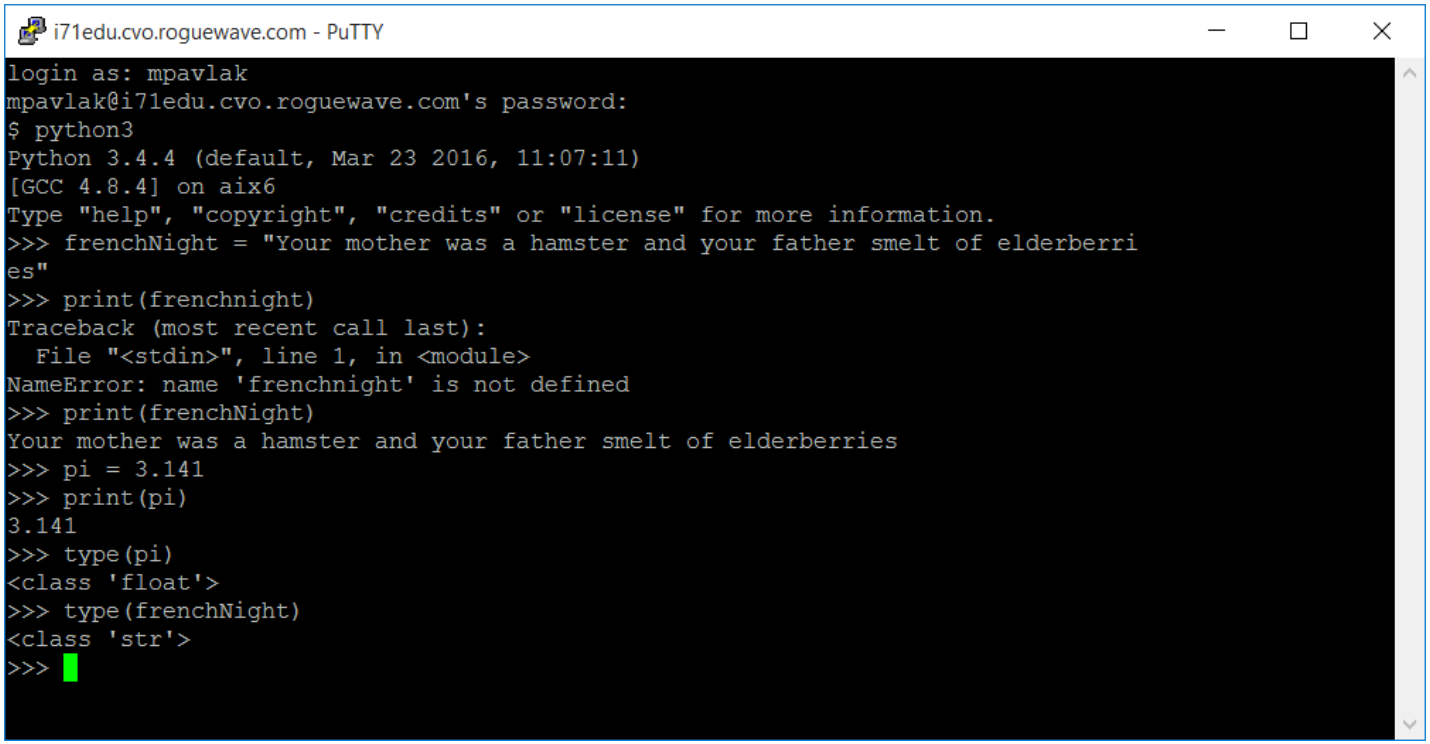
Variables

Data types – yeah…about that…

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

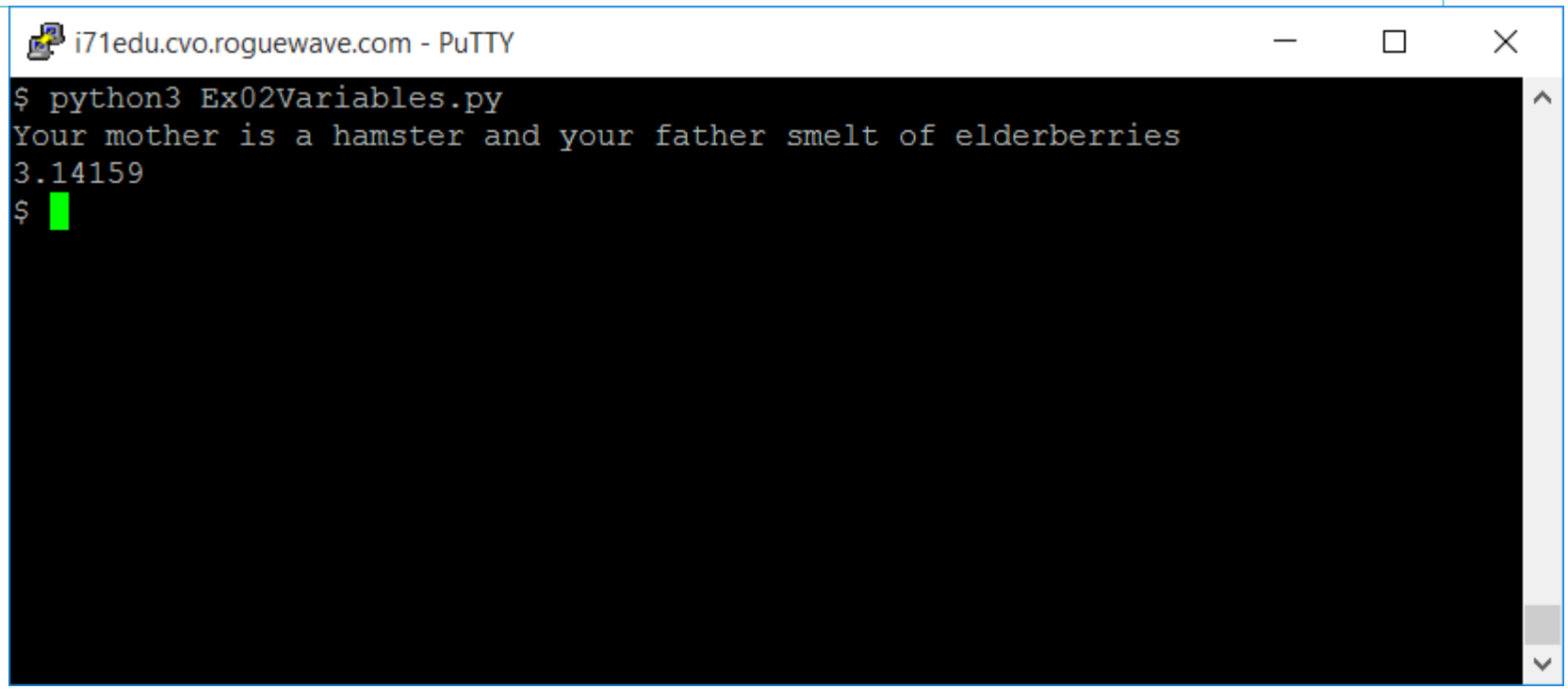
Variables on the fly

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



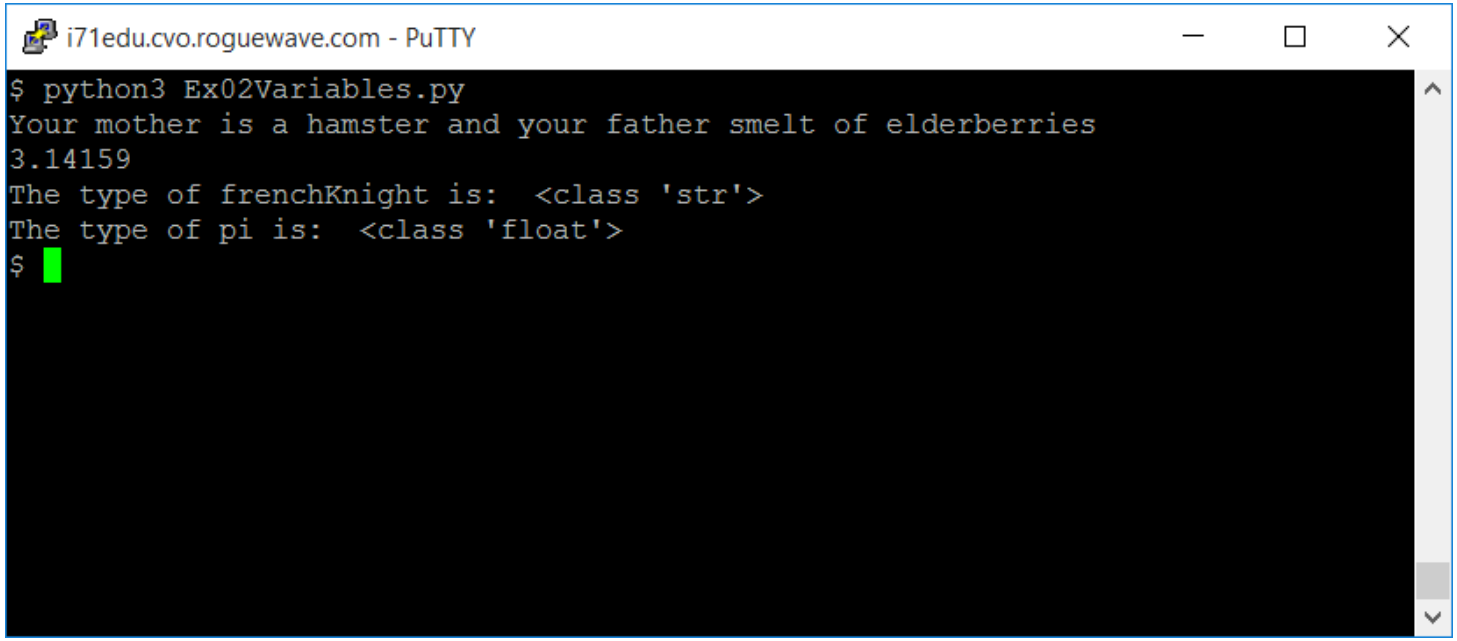
Variables in a file

```
1 #
2 # Variables are defined on the fly...
3 #
4 frenchKnight = "Your mother is a hamster and your father smelt of elderberries"
5 pi = 3.14159
6
7 print(frenchKnight)
8 print(pi)
```



Data type?

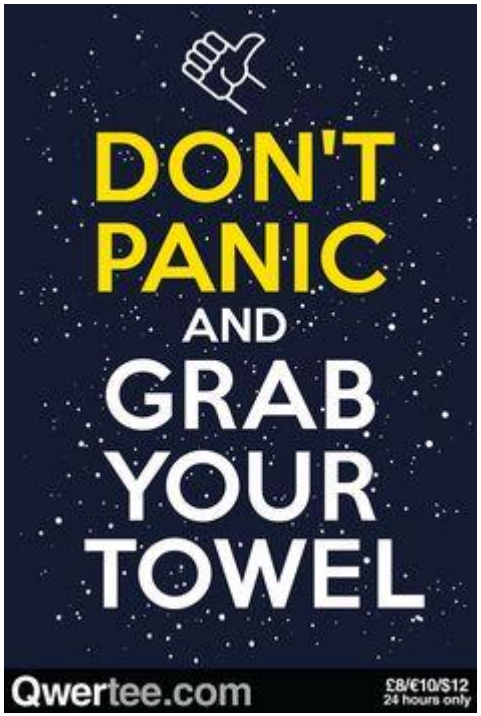
```
1  #
2  # Variables are defined on the fly...
3  #
4  frenchKnight = "Your mother is a hamster and your father smelt of elderberries"
5  pi = 3.14159
6
7  print(frenchKnight)
8  print(pi)
9
10 print("The type of frenchKnight is: ", type(frenchKnight))
11 print("The type of pi is: ", type(pi))
```



FRESCHE

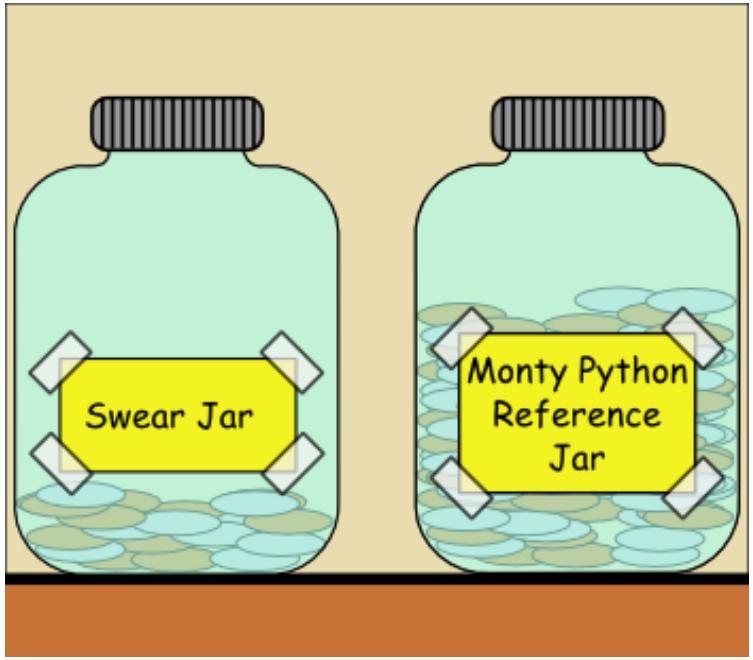
Every variable is implemented as a class

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

- Numbers – 3 Data types
  - Integer 1,2,42
  - Float 3.14159
  - Complex: <real> + <imaginary> (not used much…)

Strings

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

String formatting

- Interpolation, of sorts

```
#
# String example
#

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

Lists

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

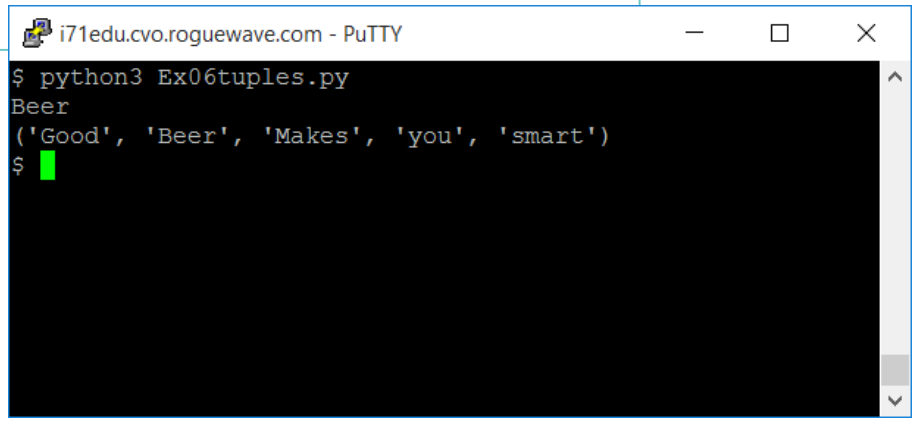
```
1 #
2 # List ExampleService
3 #
4 mylist = ["Rock Bottom", "Gordon Biersch", "BJ's", "Granite City"]
5 print(mylist[1])
6
7 print(mylist[0:2])
8
9 print(mylist)
```

```
i71edu.cvo.roguewave.com - PuTTY
$ 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

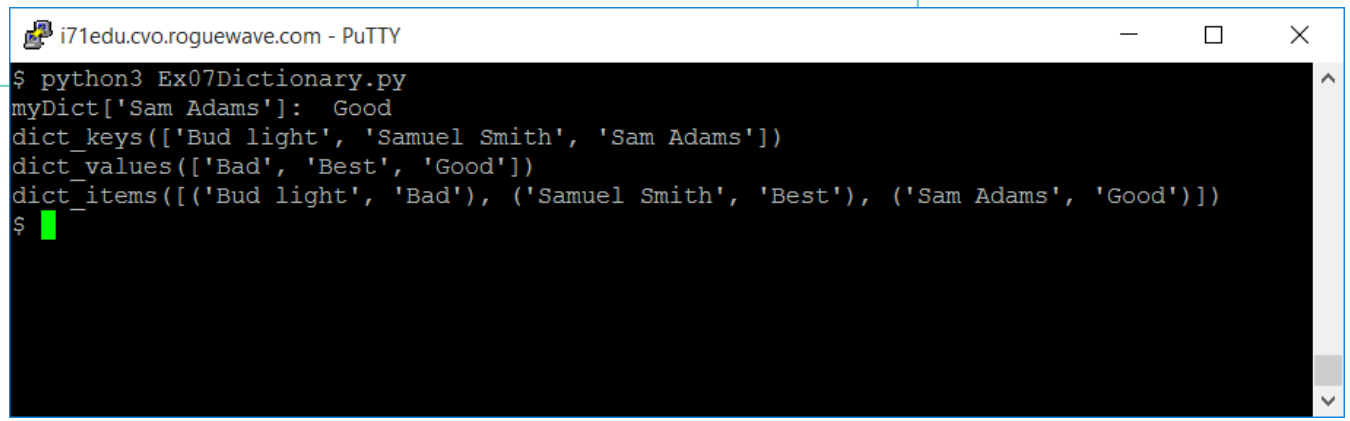
```
1 #
2 # Tuples Examples
3 #
4
5 mytuple = ("Good", "Beer", "Makes", "you", "smart")
6 print(mytuple[1])
7 print(mytuple)
```



Dictionary

- Again, like lists but more like hash or PHP Assoc. Array
- Mutable
- Key value pairs

```
1  #
2  # Dictionary Examples
3  #
4
5  myDict = {"Sam Adams":"Good", "Samuel Smith":"Best", "Bud light": "Bad"}
6
7  print("myDict['Sam Adams']: ", myDict["Sam Adams"])
8
9  print(myDict.keys())
10 print(myDict.values())
11 print(myDict.items())
```

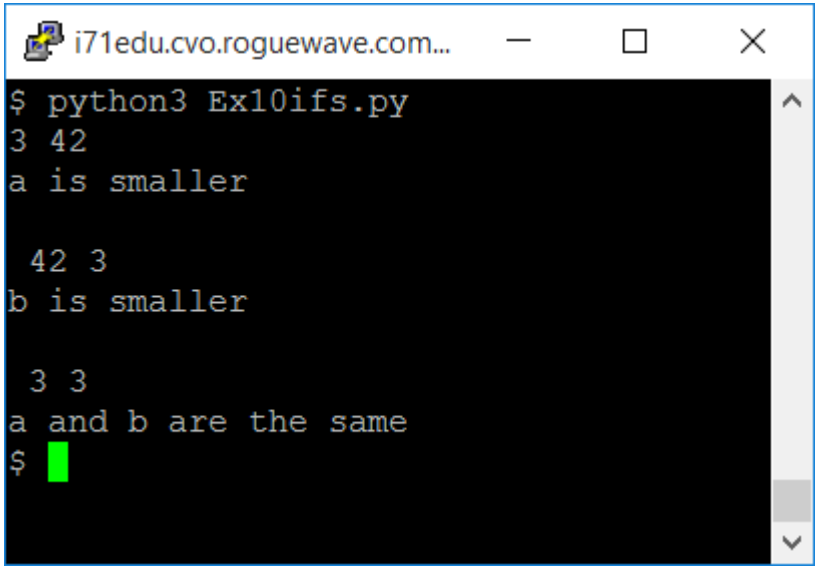


Syntax

Control Structures

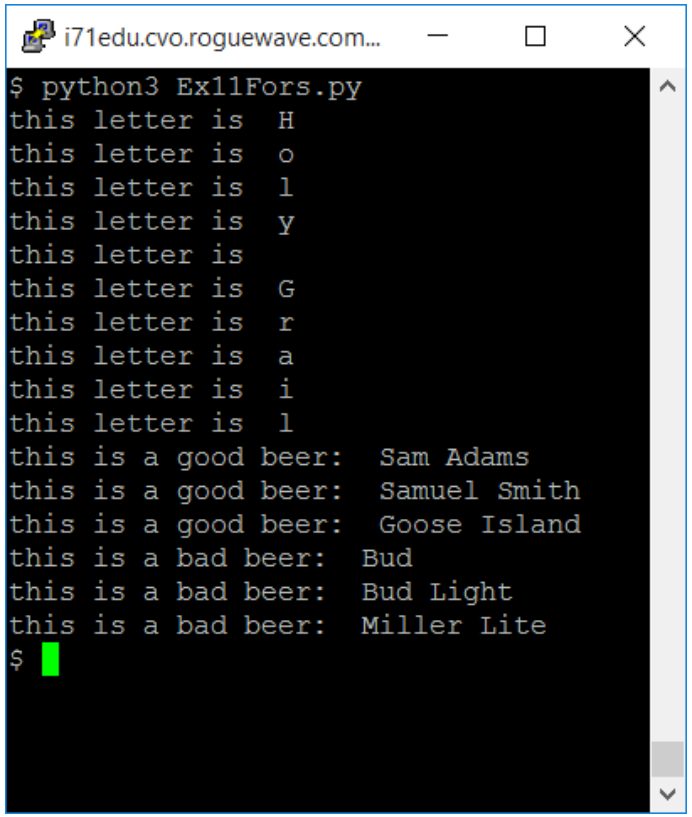
ifs

```
1  #
2  # If examples
3  #
4  a,b = 3,42
5  print(a,b)
6  if a < b:
7      print("a is smaller")
8
9  a,b = 42,3
10 print("\n",a,b)
11 if a < b:
12     print("a is smaller")
13 else:
14     print("b is smaller")
15
16 a,b = 3,3
17 print("\n",a,b)
18 if a < b:
19     print("a is smaller")
20 elif a > b:
21     print("b is smaller")
22 else:
23     print("a and b are the same")
```



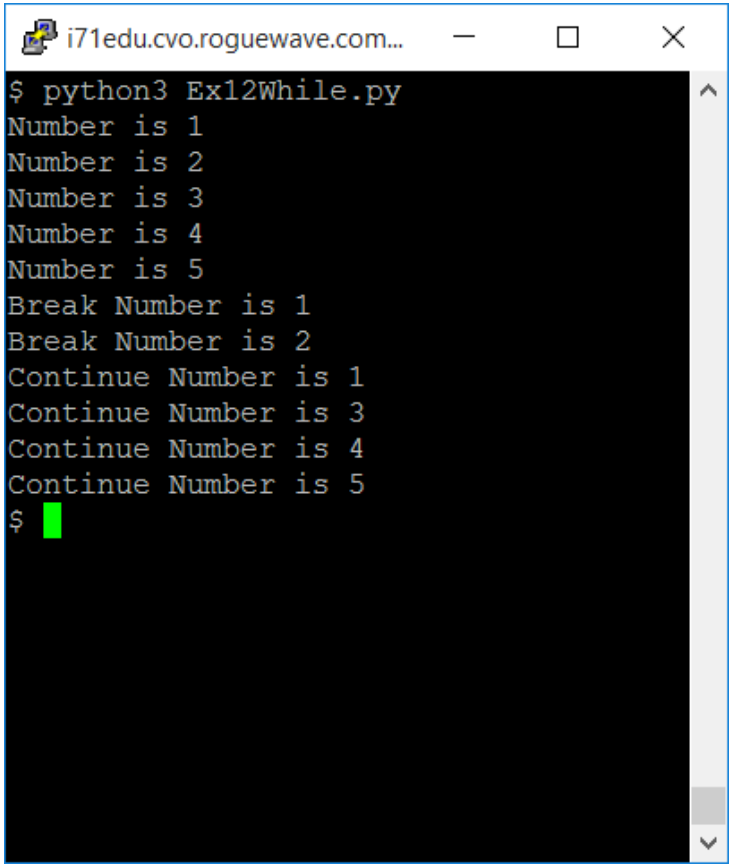
for loop

```
1#
2 # For Loop Examples
3 #
4
5 myString = "Holy Grail"
6 for letter in  myString:
7     print("this letter is ", letter)
8
9 beers = ["Sam Adams", "Samuel Smith", "Goose Island"]
10 for beer in  beers:
11     print("this is a good beer: ", beer)
12
13 badBeers = ["Bud", "Bud Light", "Miller Lite"]
14 for index in range(len(beers)):  #iterates 0 thru 2
15     print("this is a bad beer: ", badBeers[index])
```



while loop

```
1  #
2  # While Loop Examples
3  #
4
5  count, limit = 0,5
6  while count < limit:
7      count = count+1
8      print("Number is", count)
9
10 count = 0
11 while count < limit:
12     count = count+1
13     if count==3:
14         break
15     print("Break Number is", count)
16
17
18 count = 0
19 while count < limit:
20     count = count+1
21     if count==2:
22         continue
23     print("Continue Number is", count)
```

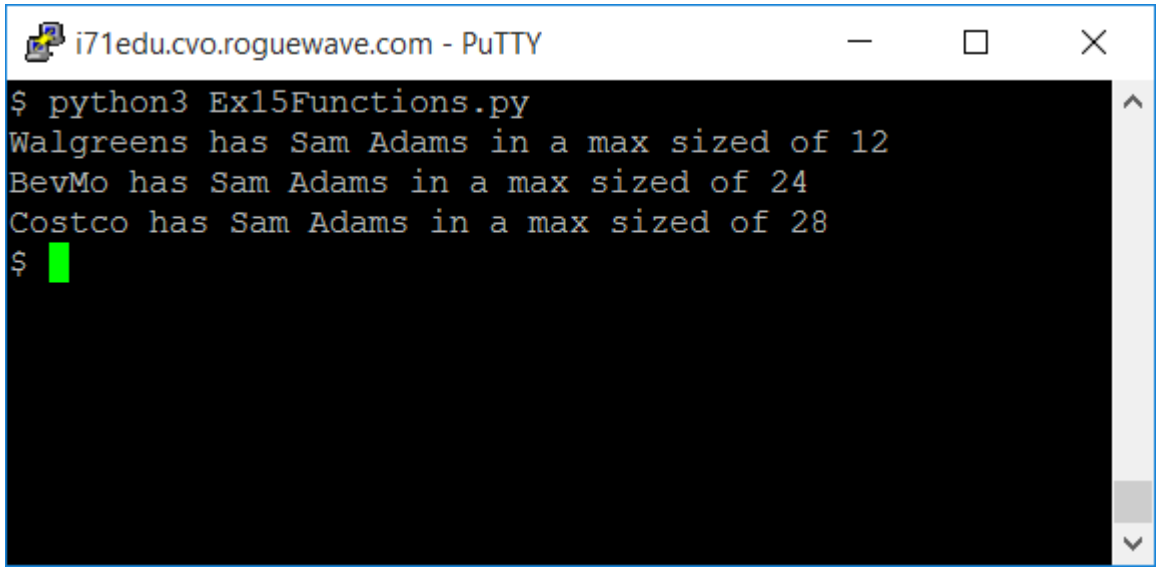


Syntax

Functions

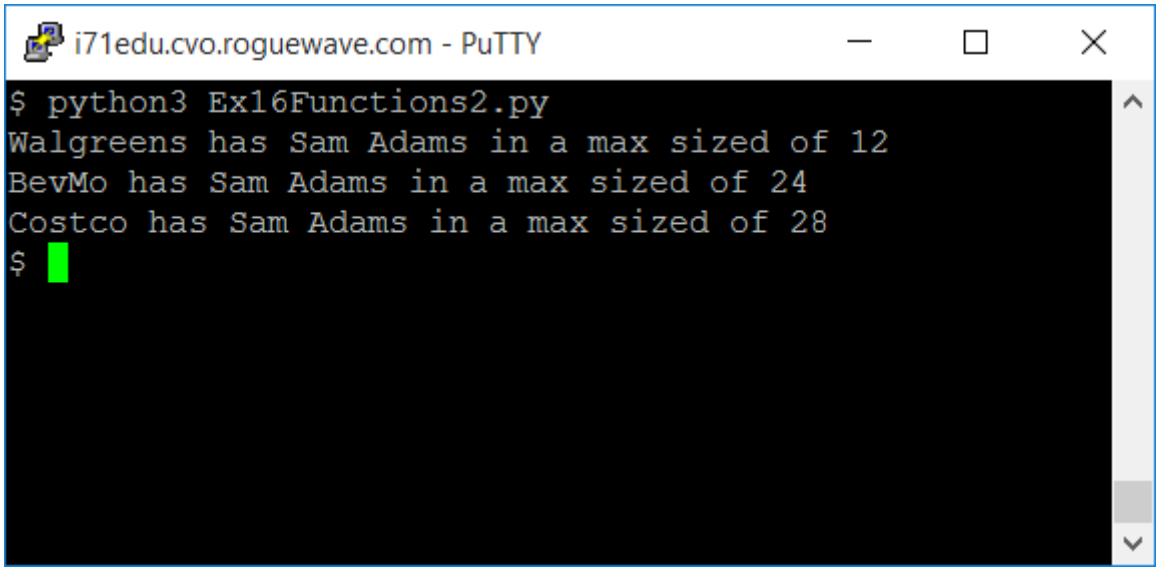
Basic functions

```
1  #
2  # Function Examples
3  #
4
5  def printBeer(store, beer, size):
6      print(store + " has " + beer + " in a max sized of " + str(size) )
7
8  myBeer = "Sam Adams"
9  printBeer("Walgreens", myBeer, 12)
10 printBeer("BevMo", myBeer, 24)
11 printBeer("Costco", myBeer, 28)
```



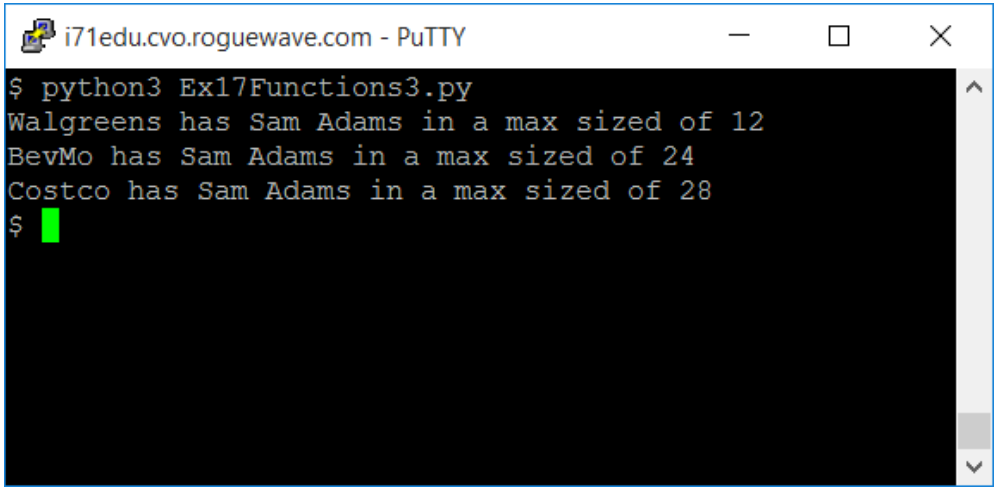
Functions with defaults

```
1  #
2  # Function Examples
3  #
4
5  def printBeer(store, beer, size=24):
6      print(store + " has " + beer + " in a max sized of " + str(size) )
7
8  myBeer = "Sam Adams"
9  printBeer("Walgreens", myBeer, 12)
10 printBeer("BevMo", myBeer)
11 printBeer("Costco", myBeer, 28)
```



Functions with Keyword arguments

```
1 #
2 # Function Examples
3 #
4
5 def printBeer(store, beer, size):
6     print(store + " has " + beer + " in a max sized of " + str(size) )
7
8 myBeer = "Sam Adams"
9 printBeer("Walgreens", myBeer, 12)
10 printBeer(beer=myBeer, size=24, store="BevMo")
11 printBeer(beer=myBeer, store="Costco", size=28)
```

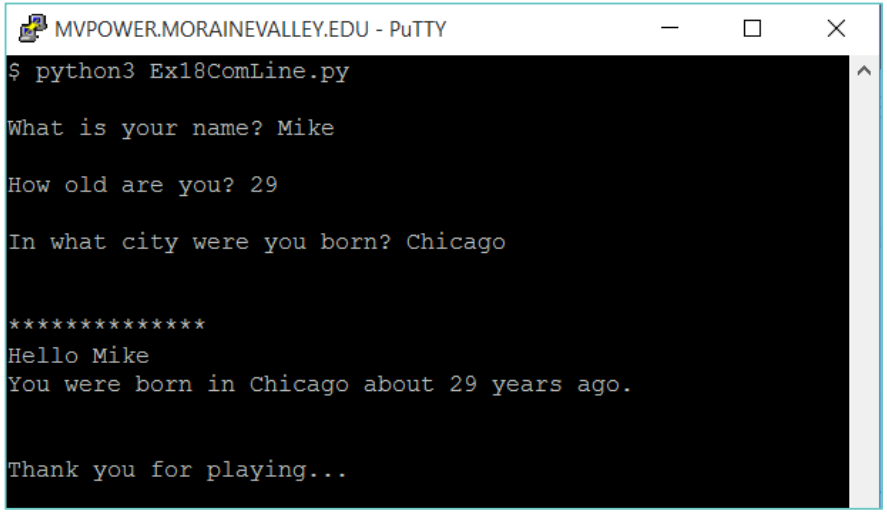


Command Line

Input form command line

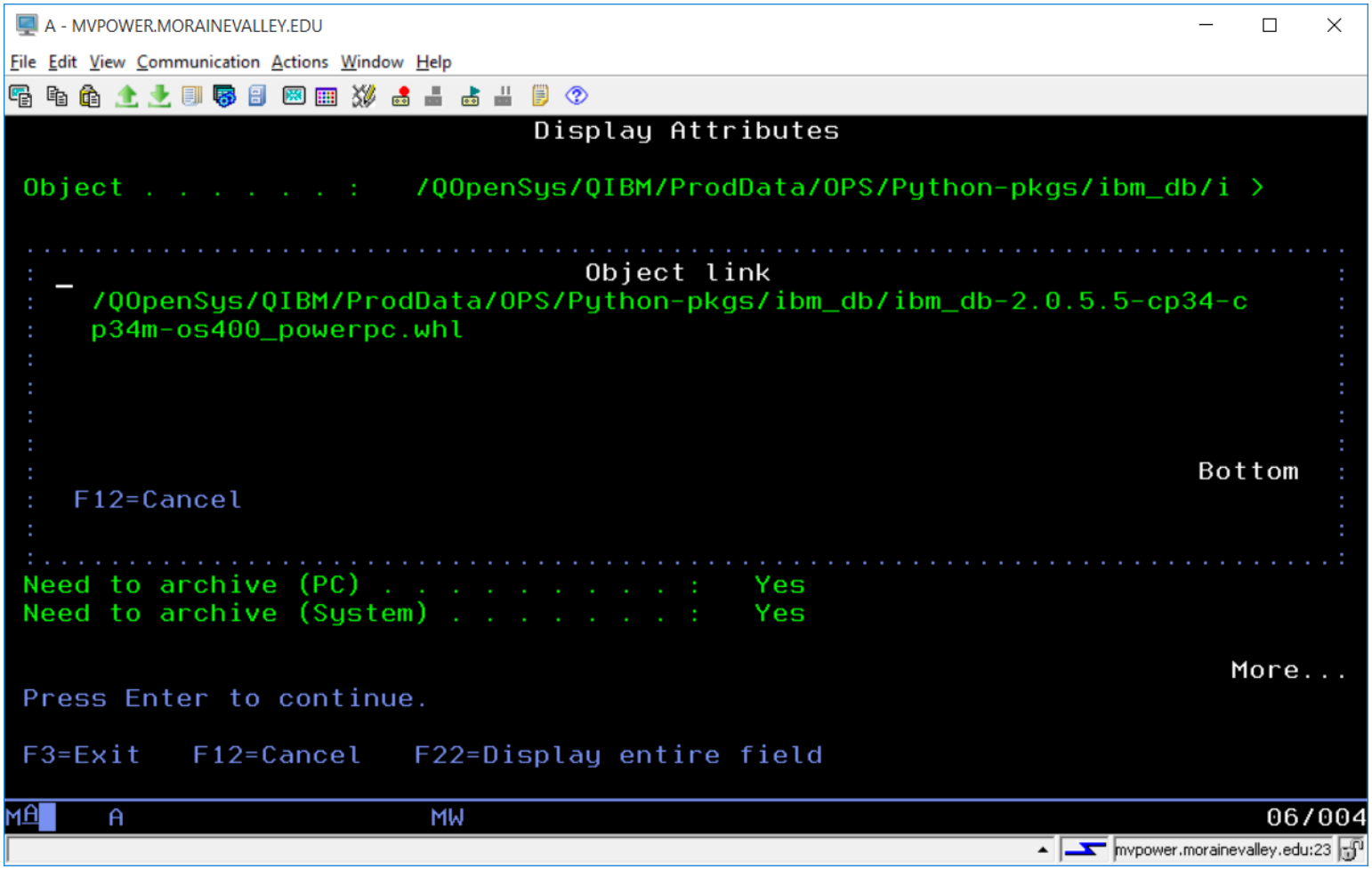
- "Talk" to the script…

```
1  # Get input from user and then embed in string
2  from pip._vendor.distlib.compat import raw_input
3
4  name = raw_input("\nWhat is your name? ")
5  age = raw_input("\nHow old are you? ")
6  city = raw_input("\nIn what city were you born? ")
7  print("\n\n**************")
8  print("Hello %s" % (name))
9  print("You were born in %s about %s years ago." % (city, str(age)))
10 print("\n\nThank you for playing...\n\n")
```



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:

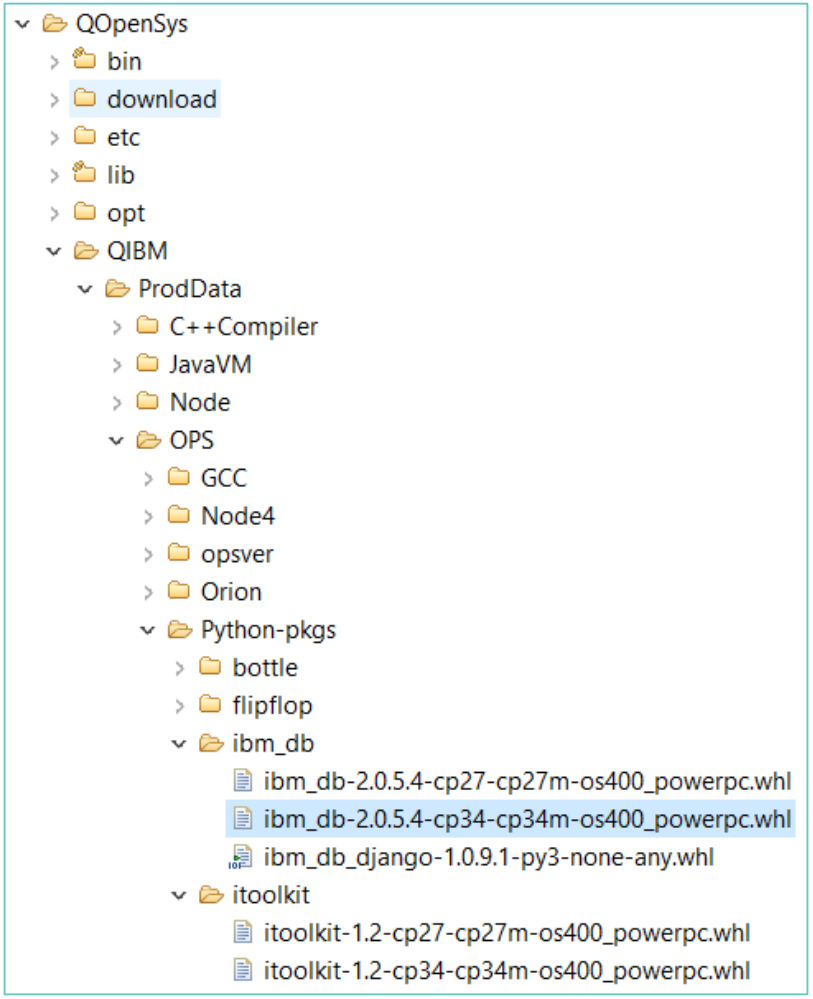
```
pip3 install /QOpenSys/QIBM/ProdData/OPS/Python-pkgs/itoolkit/itoolkit-*-cp34m-*.whl
```

To install FastCGI gateway support:

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

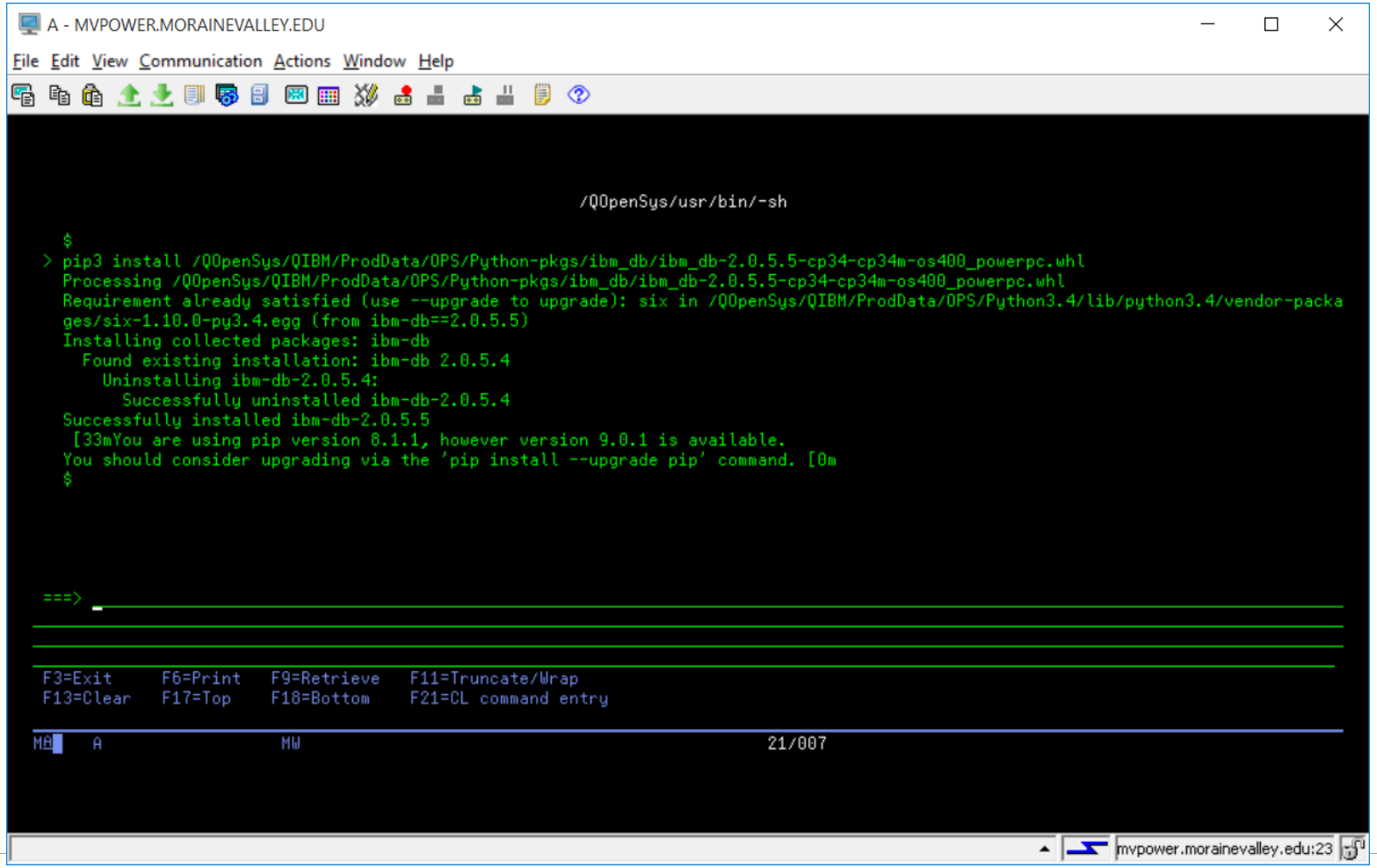
Find the connector

- YMMV
- With wheels



Run the pip install

- pip == Python installer program



What version of the DB2 Extension?

```
1 import ibm_db_dbi as dbi
2
3 print(dbi.__version__)
```

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

Steps for simple database Access

- Import the class
- Connect (with or without options
- Open the cursor
- Set the SQL
- Read

Simple database access

```
1  import ibm_db_dbi as dbi
2  conn = dbi.connect()
3  sql = "SELECT COMPANY, COUNTRY FROM samples.SP_CUST where country = 'US'"
4  c01 = conn.cursor()
5  c01.execute(sql)
6  #Bring it in as tuple
7  print("\n\n******Tuple*****\n\n")
8
9  for row in c01.fetchall():
10     print(row)
11 c01.close()
12 conn.close()
13 print("\n\n******End******\n\n")
```

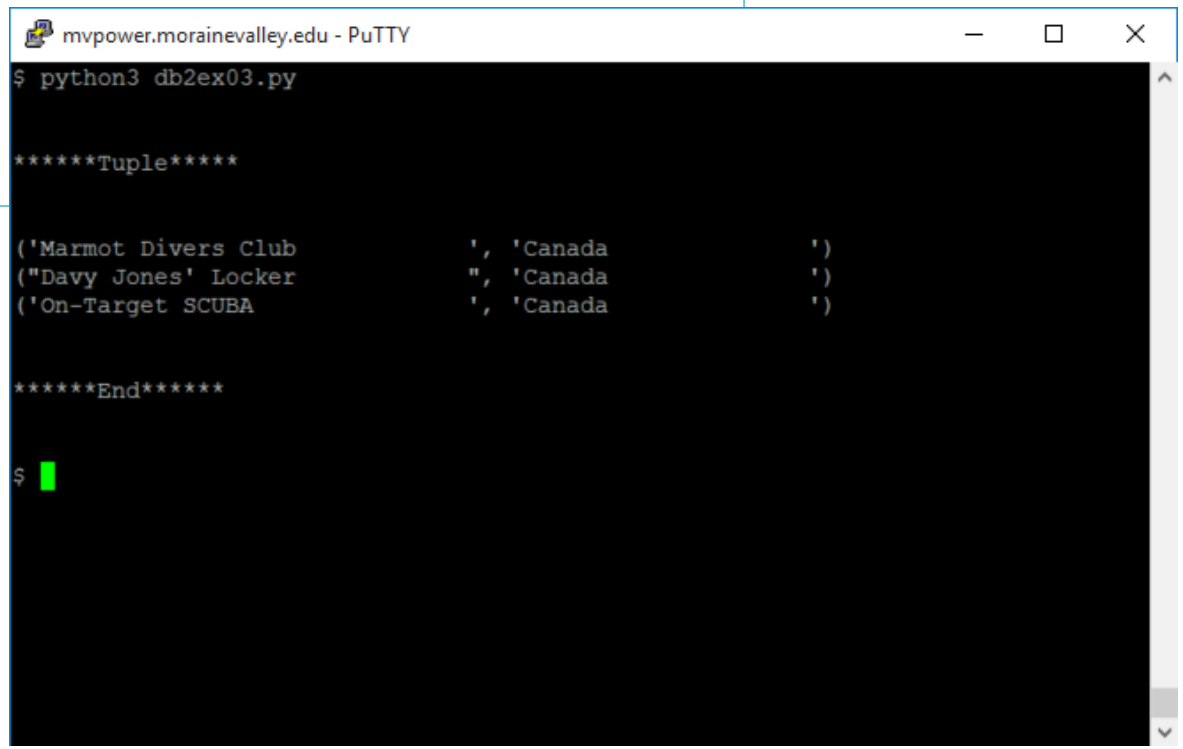
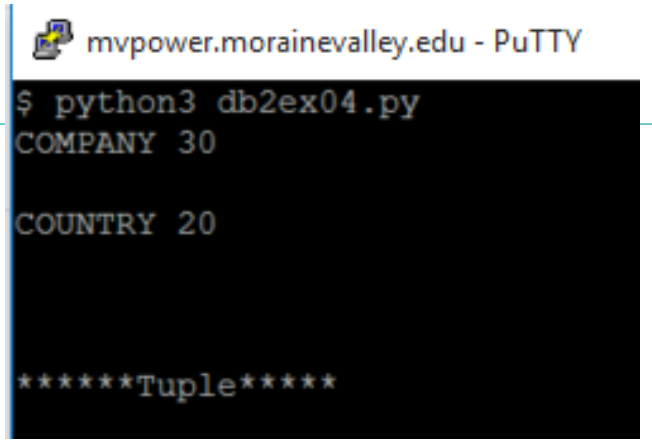


Table info

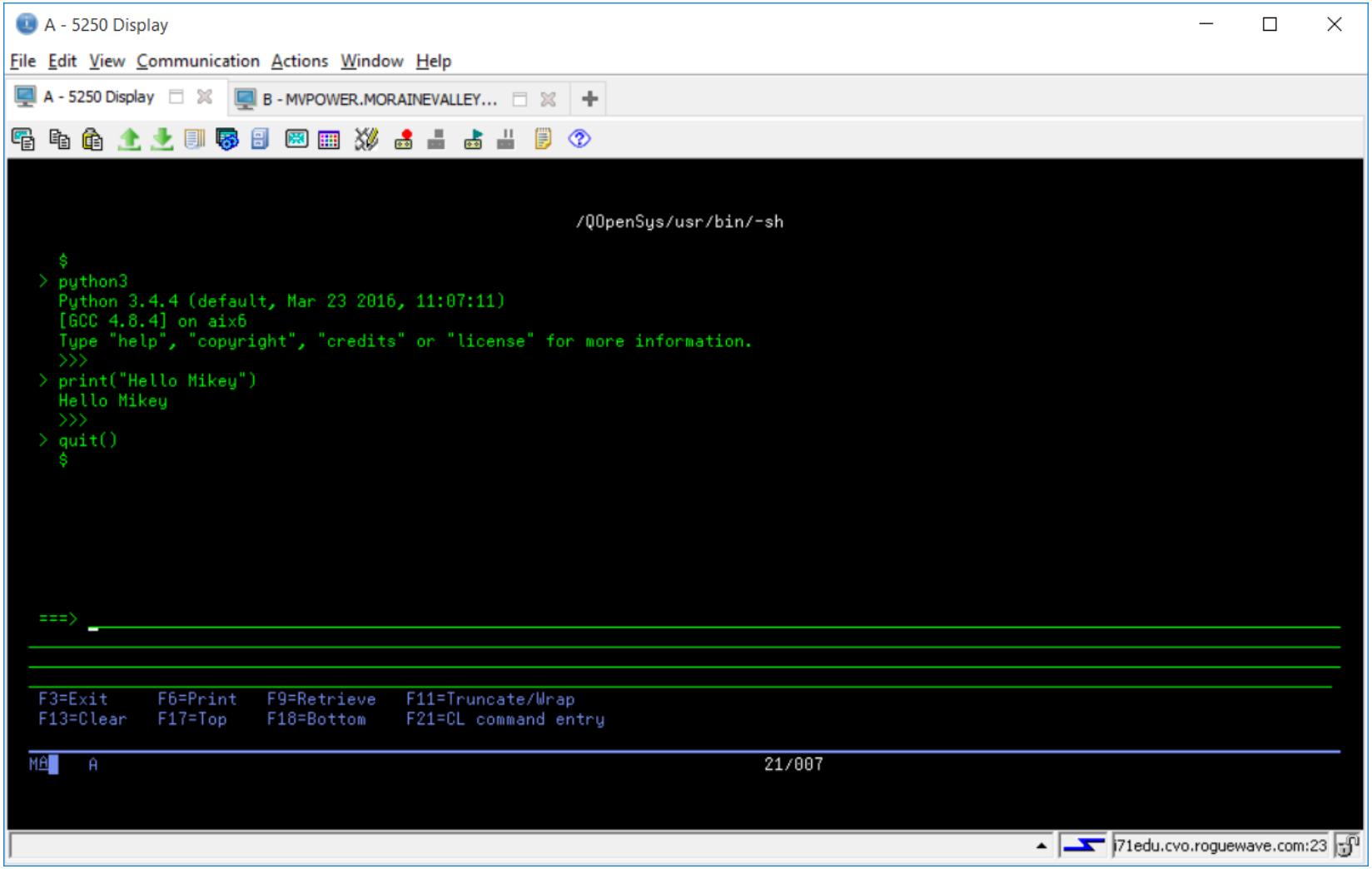
```
1  import ibm_db_dbi as dbi
2  conn = dbi.connect()
3  sql = "SELECT COMPANY, COUNTRY FROM ZENDSVR6.SP_CUST where country = 'Canada'"
4  c01 = conn.cursor()
5  c01.execute(sql)
6  desc = c01.description
7  print(desc[0][0], desc[0][4], "\n")
8  print(desc[1][0], desc[1][4], "\n")
9
10 #Bring it in as tuple
11 print("\n\n******Tuple*****\n\n")
12 for row in c01.fetchall():
13     print(row)
14 c01.close()
15 conn.close()
16 print("\n\n******End******\n\n")
```



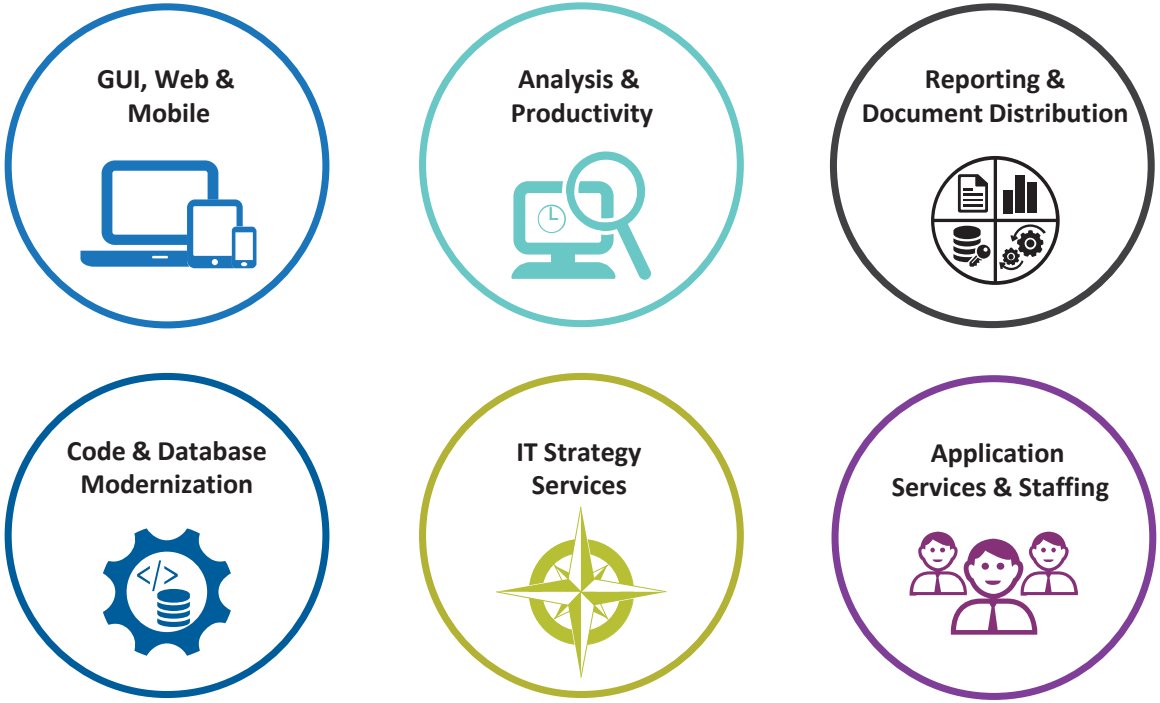
Summary – Why Python

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

End the session



End-to-End Application Modernization Solutions



THANK YOU

Mike.Pavlak@freschesolutions.com

FRESCHE SOLUTIONS