Introduction to Python on IBM i

RogueWave SOFTWARE

Accelerating Great Code

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

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?

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

TIOBE Index for May 2017

May Headline: the pack is closing in on Java and C

May 2017	May 2016	Change	Programming Language	Ratings	Change
1	1		Java	14.639%	-6.32%
2	2		C	7.002%	-6.22%
3	3		C++	4.751%	-1.95%
4	5	↑	Python	3.548%	-0.24%
5	4	↓	C#	3.457%	-1.02%
6	10	⇈	Visual Basic .NET	3.391%	+1.07%
7	7		JavaScript	3.071%	+0.73%
8	12	⇈	Assembly language	2.859%	+0.98%
9	6	↓	PHP	2.693%	-0.30%
10	9	↓	Perl	2.602%	+0.28%

- 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

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

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

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

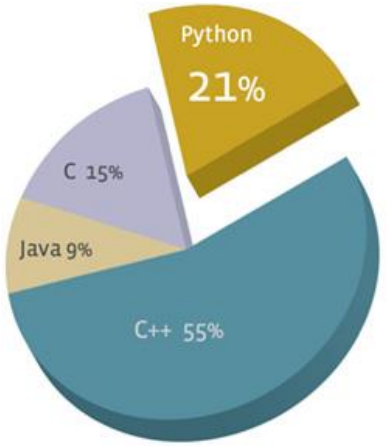
Updated Sep 12, 2014 · Upvoted by Charlie Cheever, One of the founders of Quora and Scott Danzig, avid Quora user since 2012, and a Quora Top Writer

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

- Millions of lines of code, thousands of libraries and binaries
- 2016 to date: average 5,000 commits per month, 1,000+ committers
- 5 percent Py3 (as of May 2016)

- 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

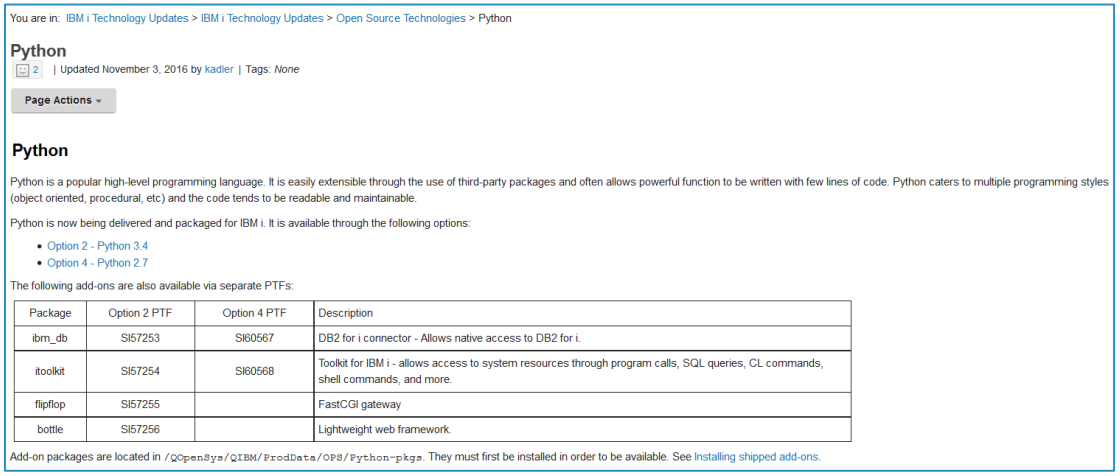


https://code.facebook.com/posts/1040181199381023/python-in-production-engineering/

Got Python?

Details at Developerworks…

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



You are in: IBM i Technology Updates > IBM i Technology Updates > Open Source Technologies > Python

Python

2 | Updated November 3, 2016 by kadler | Tags: *None*

Page Actions

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 Technologies on IBM i

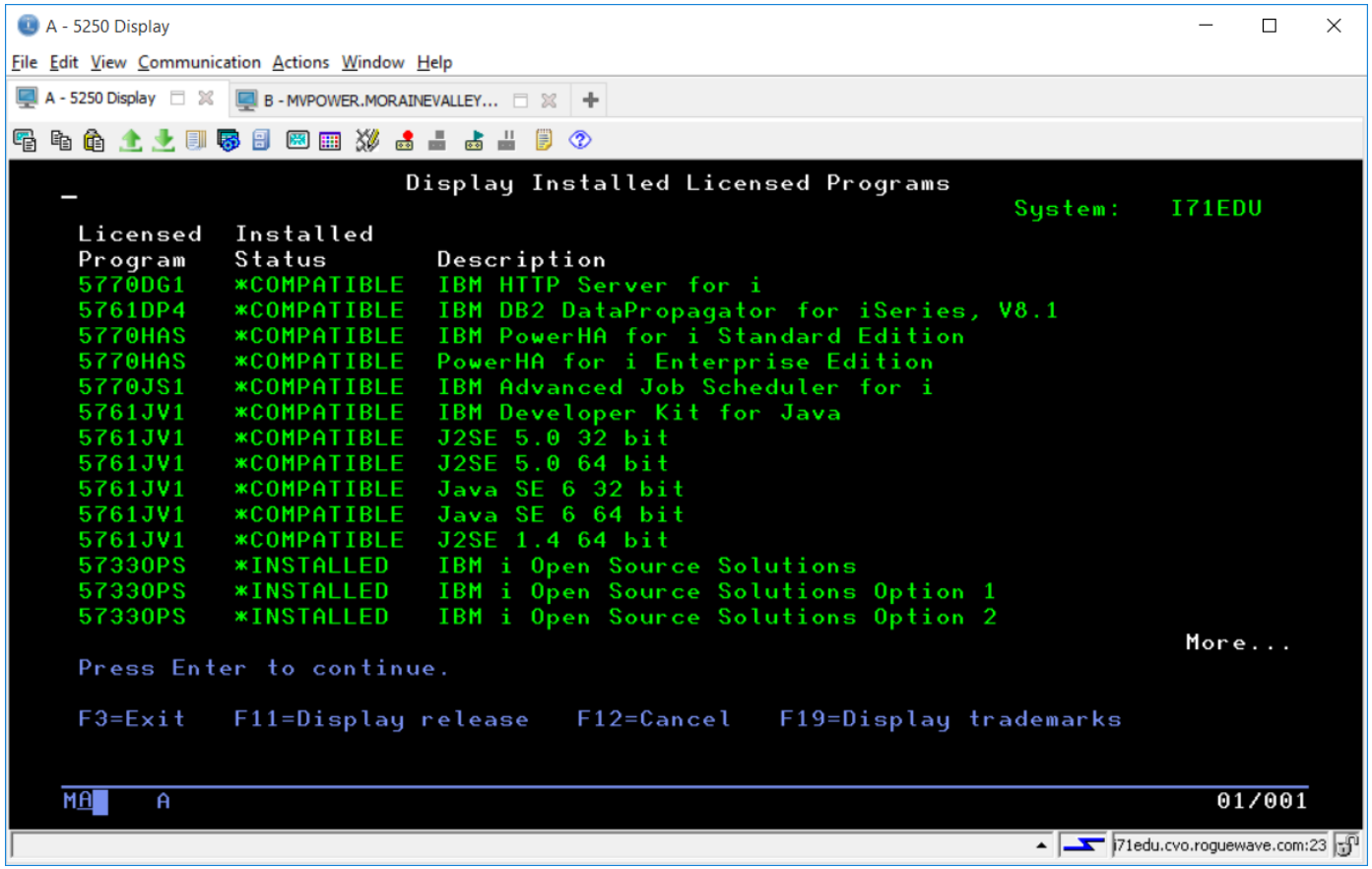
	SAMBA on IBM i
5733-OPS Option 1	Node.js v1
5733-OPS Option 2	Python 3
5733-OPS Option 3	CHROOT
5733-OPS Option 4	Python 2
5733-OPS Option 5	Node.js v4
5733-OPS Option 6	Git
5733-OPS Option 7	Tools
5733-OPS Option 8	Orion
5733-OPS Option 9	cloud-init
5733-OPS Option 10	Node.js v6
5733-OPS Option 11	TBD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

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)

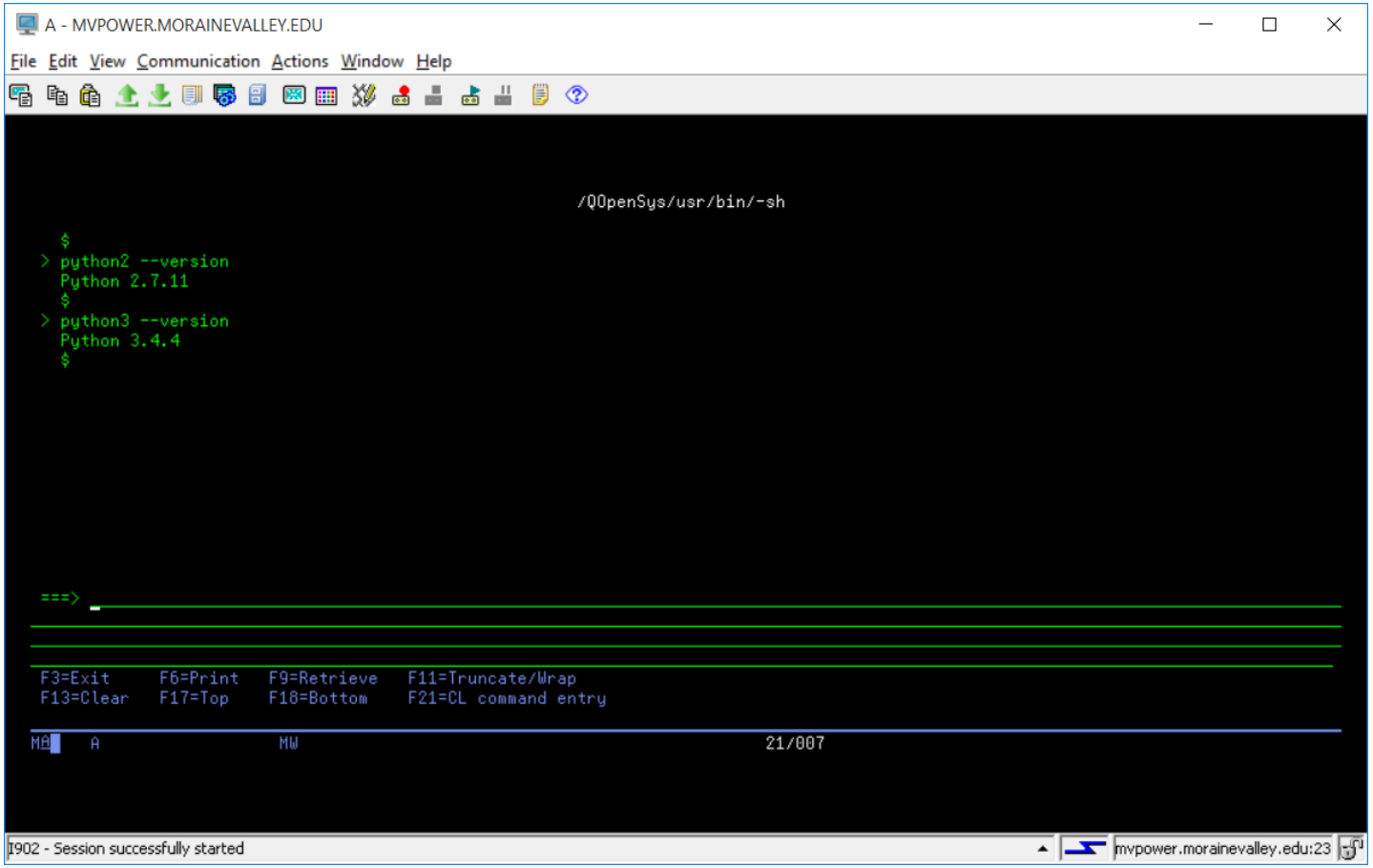
Need Licensed Programs

- 5733OPS Base and option 2 or 4

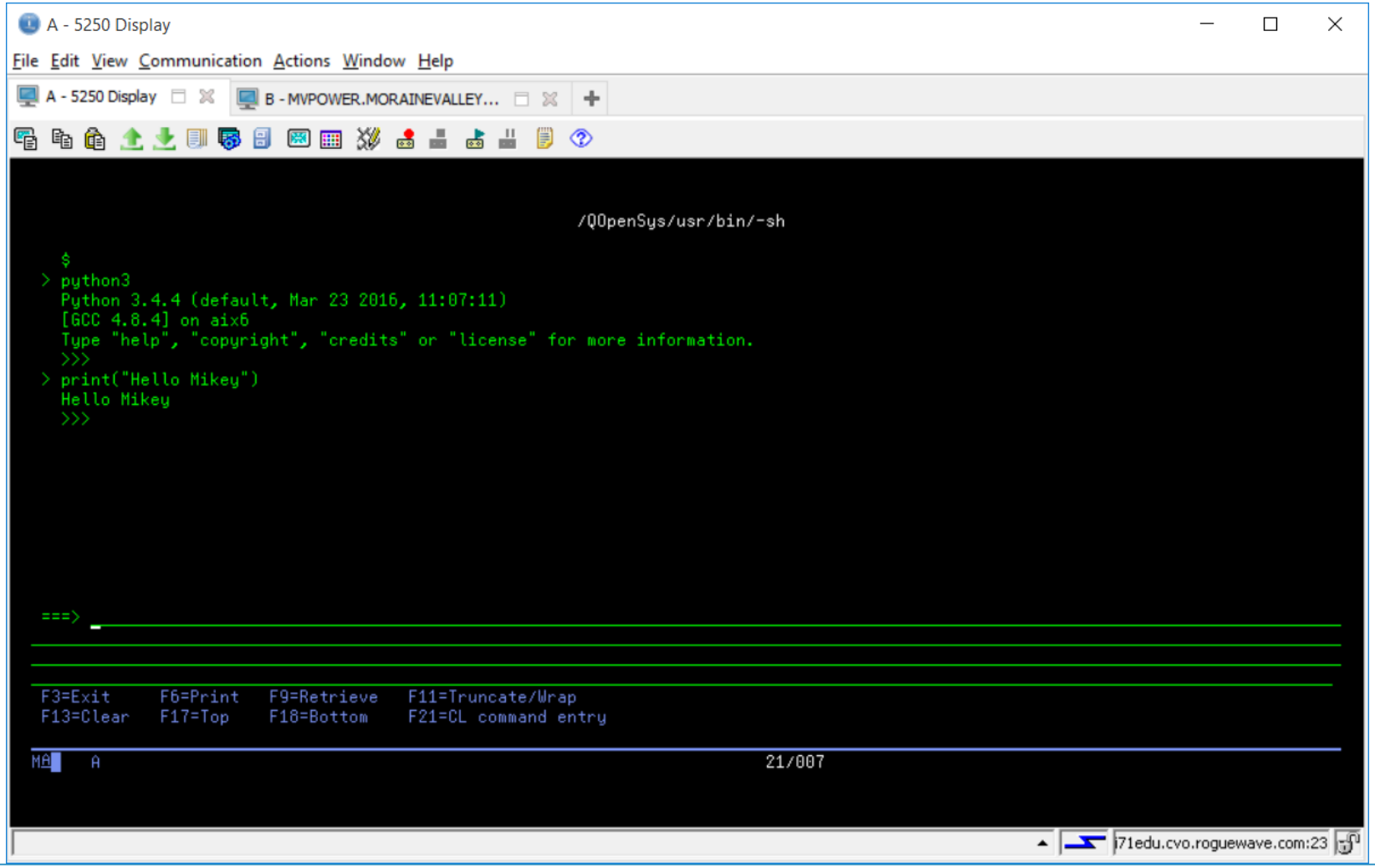


Python in action

- Command line via green screen (CALL QP2TERM)

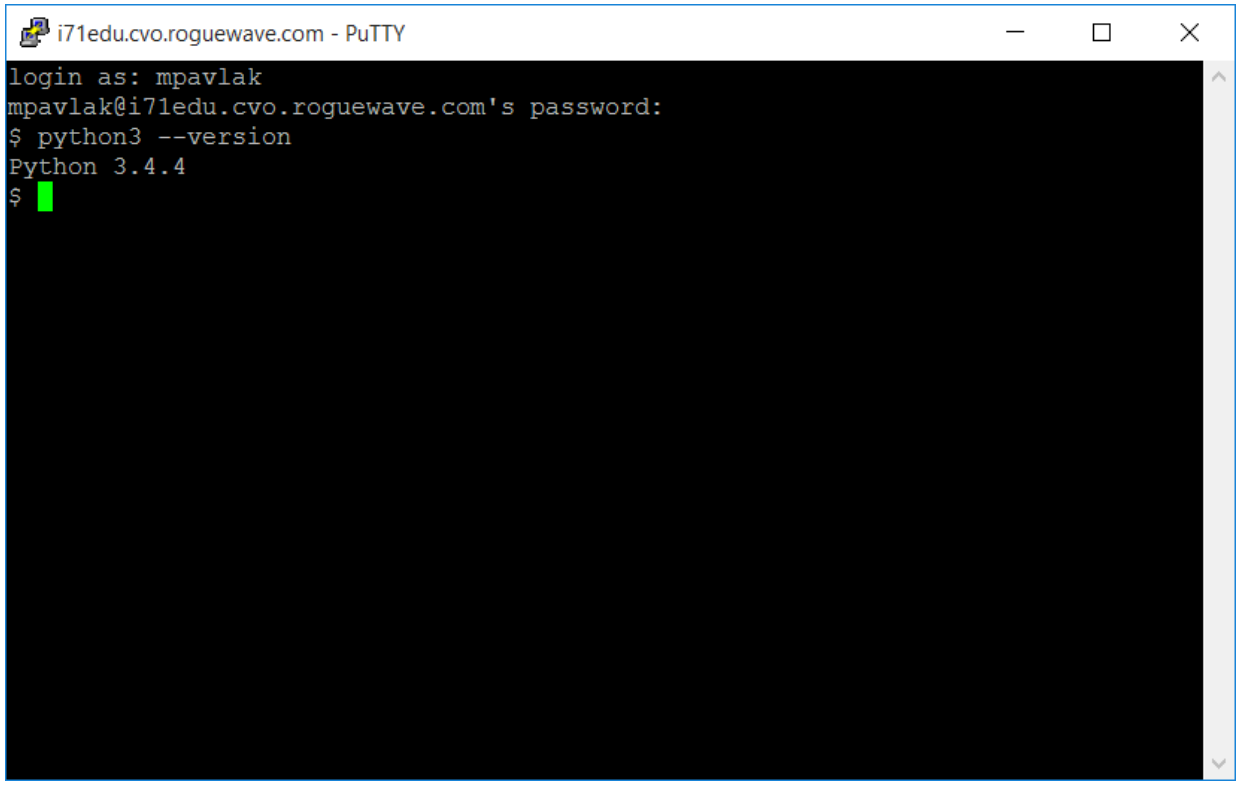


Hello world?

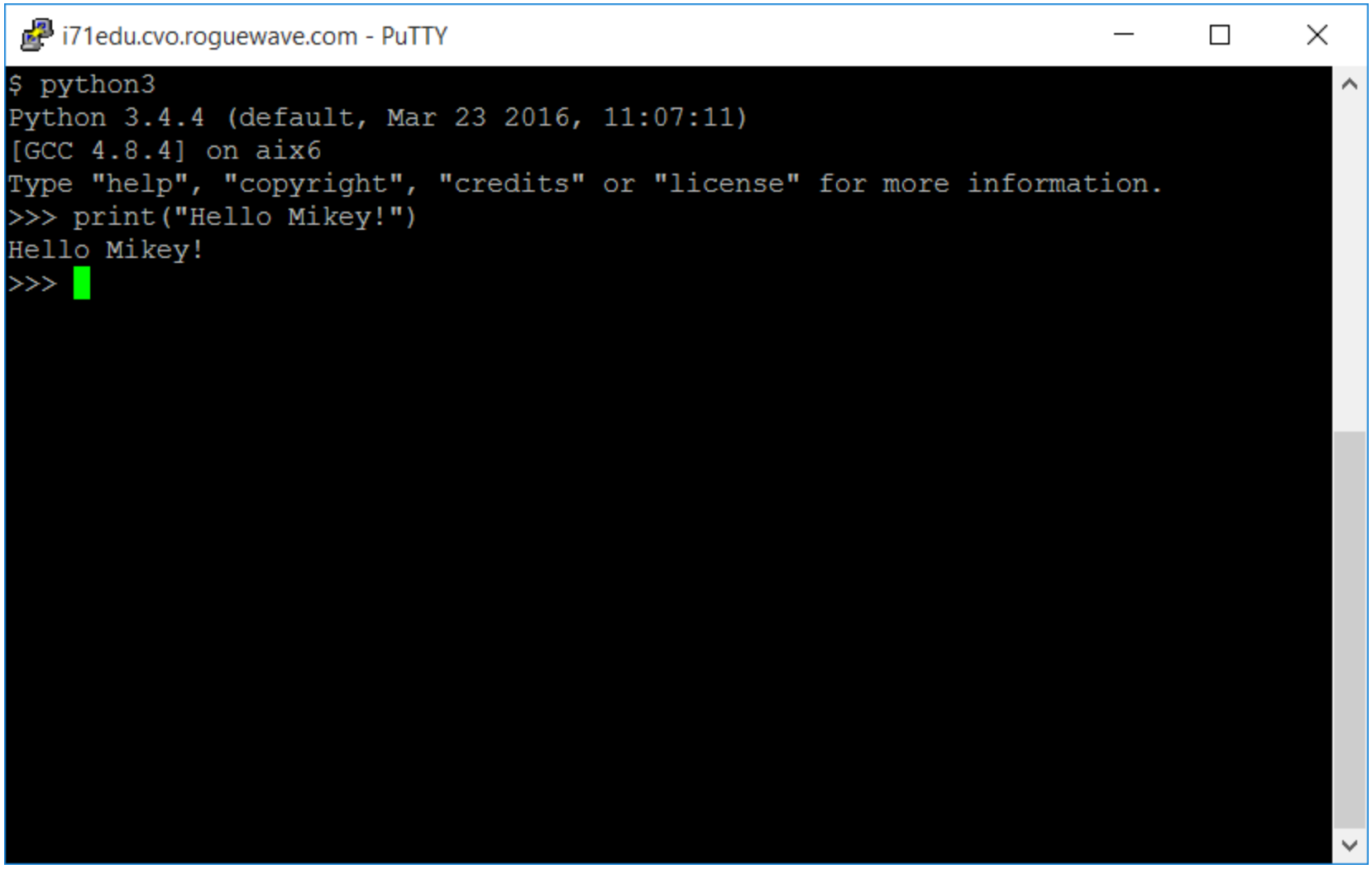


Most prefer SSH

- Command line via SSH terminal



Hello World, again?

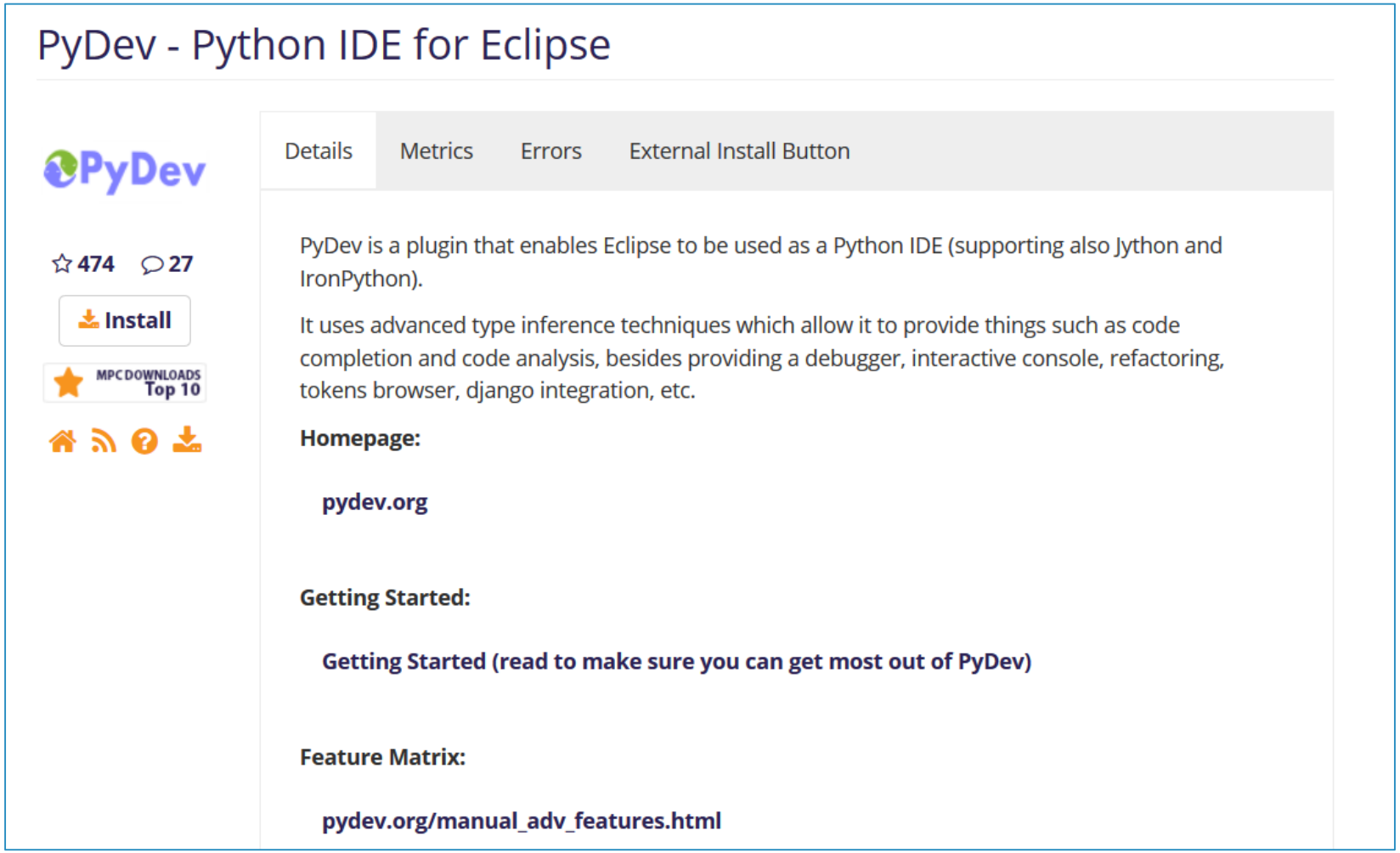


IDE?

Zend Studio

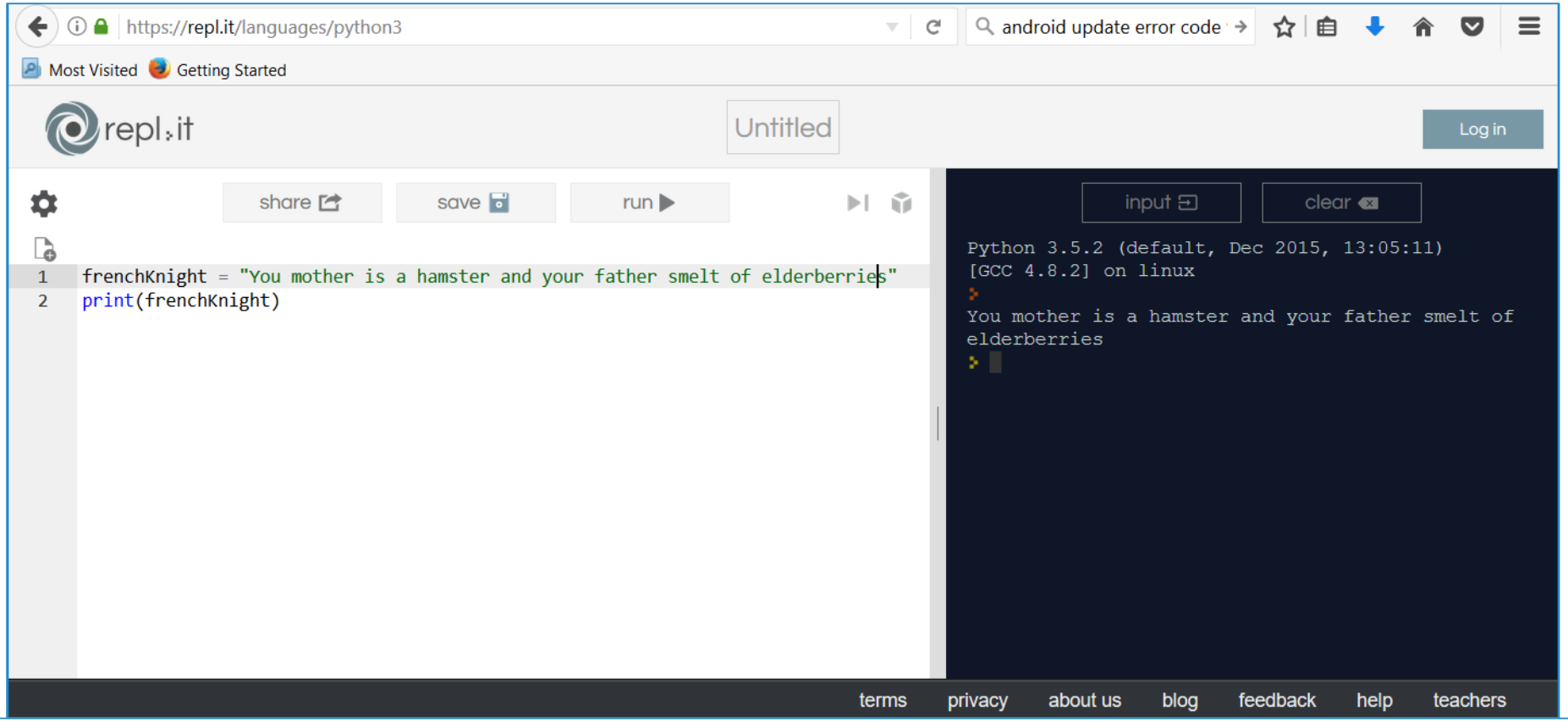
- No, you don’t need to buy Zend Studio
- Use Orion, etc.
- But if you have Studio or RDi…
  - Consider something from Eclipse.org
  - I grabbed PyDev

Eclipse



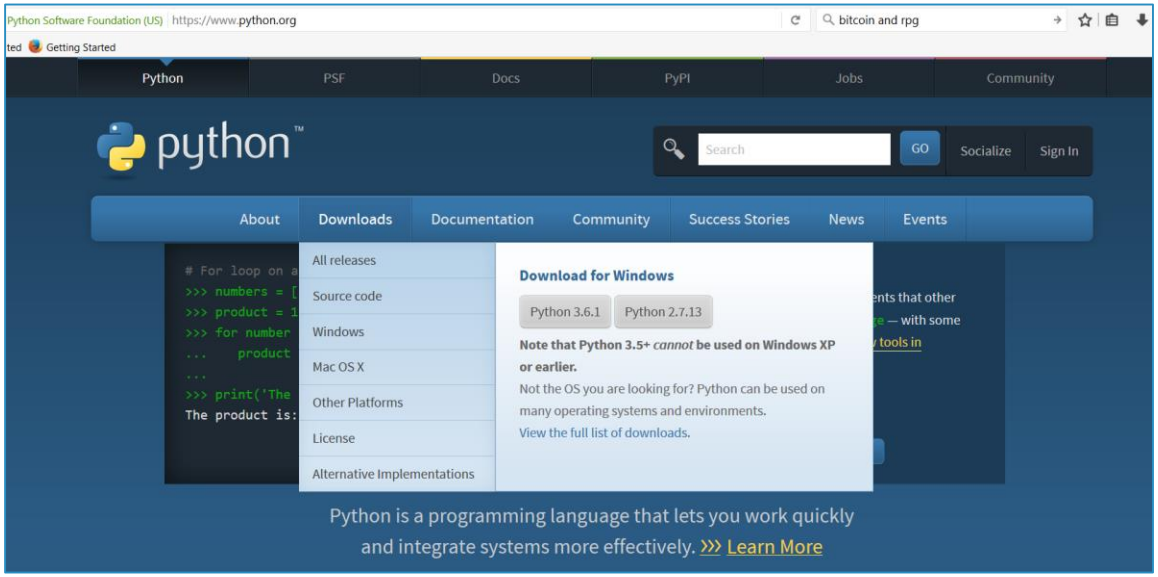
Alternatives to IBM i when learning

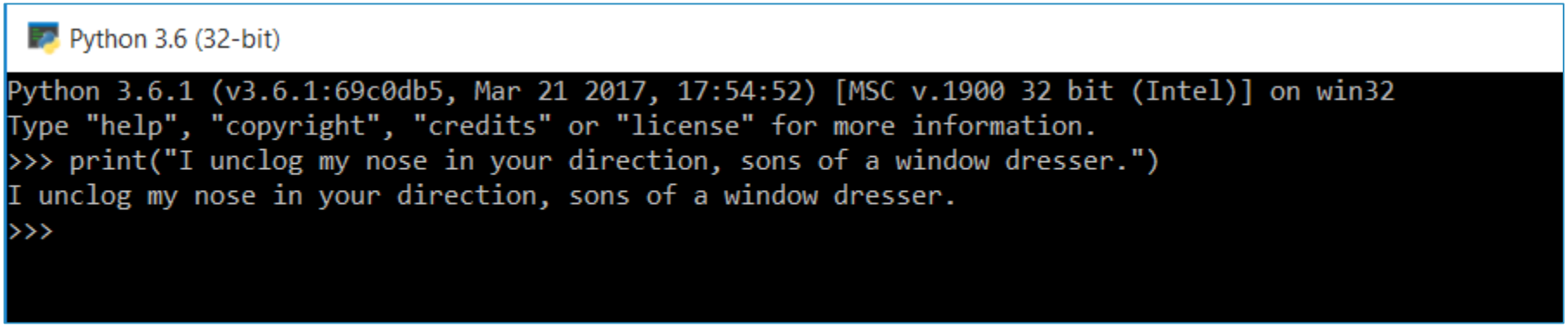
- What's that? The boss won't let you install Python?
  - Consider repl.it



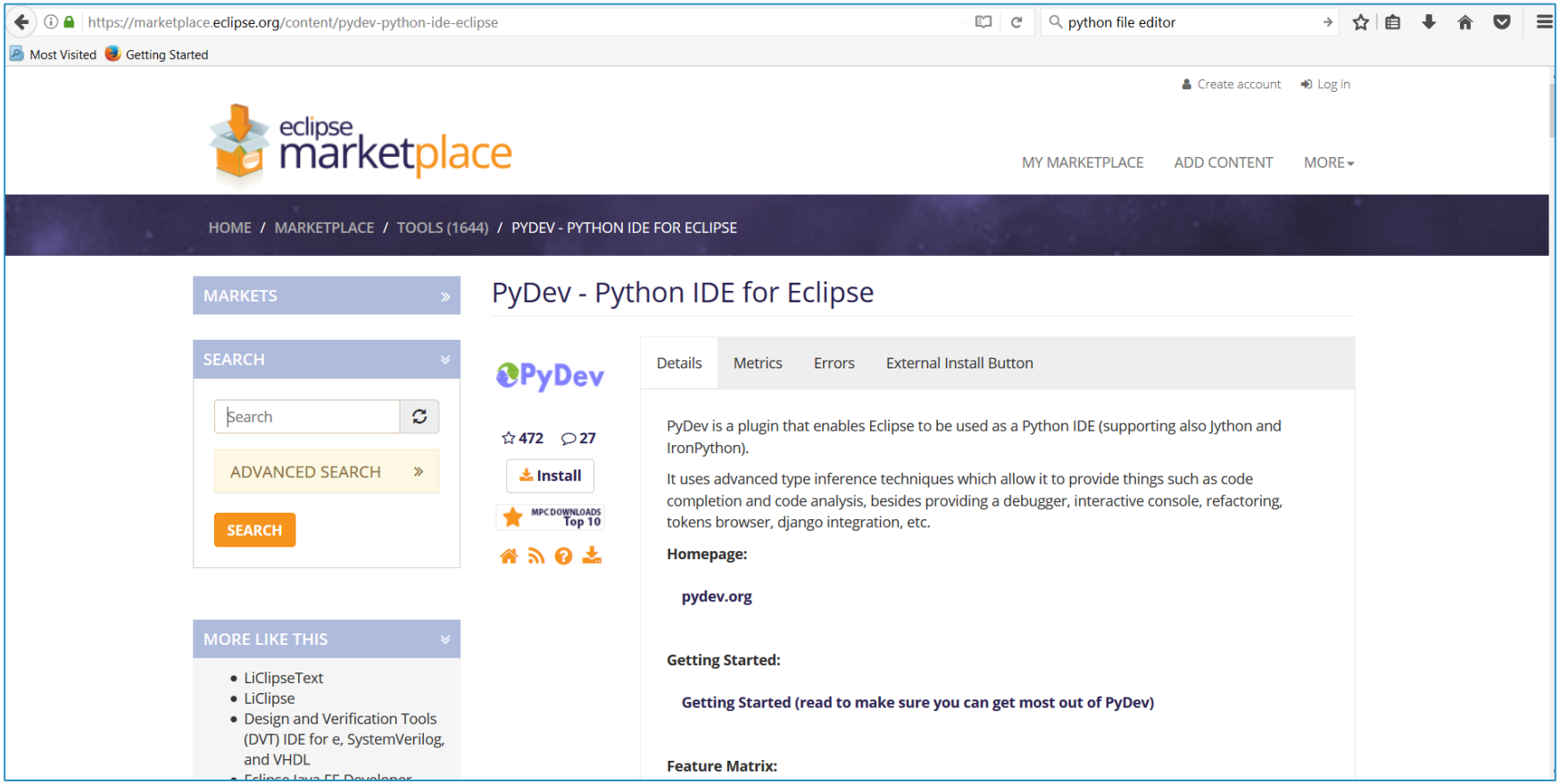
Alternatives to IBM i when learning

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



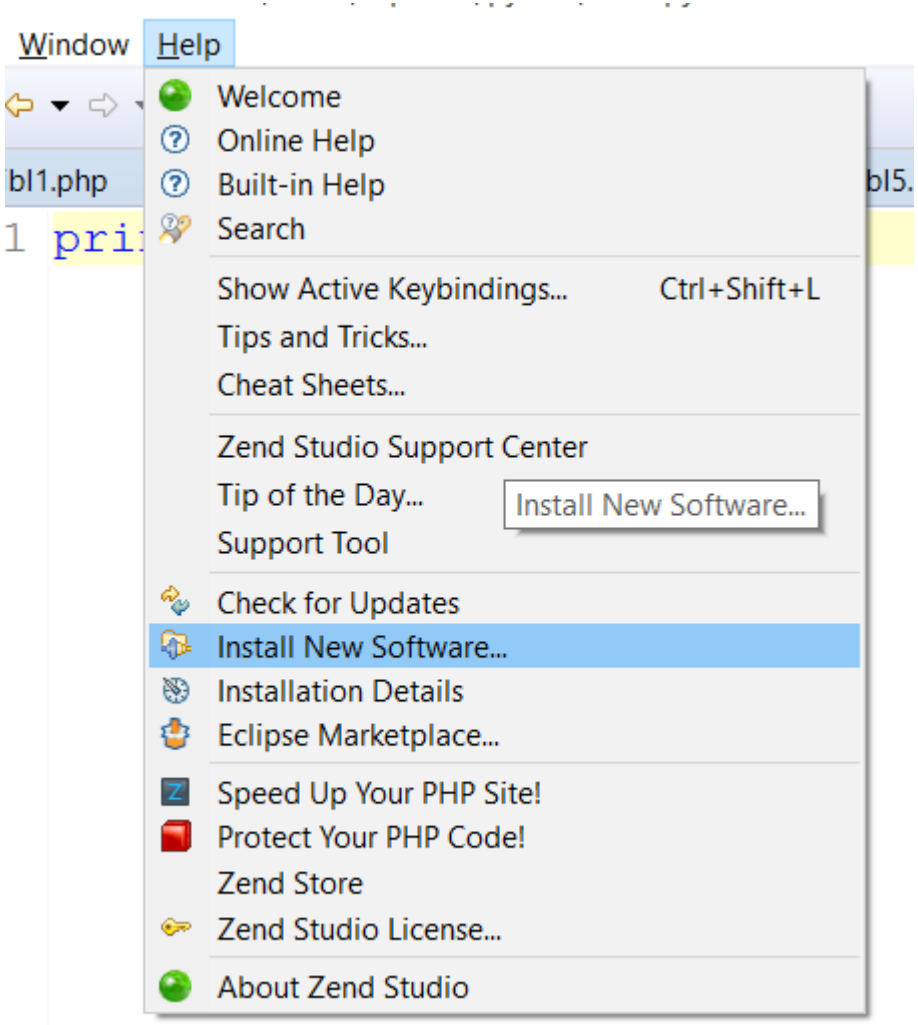


Download PyDev from Eclipse



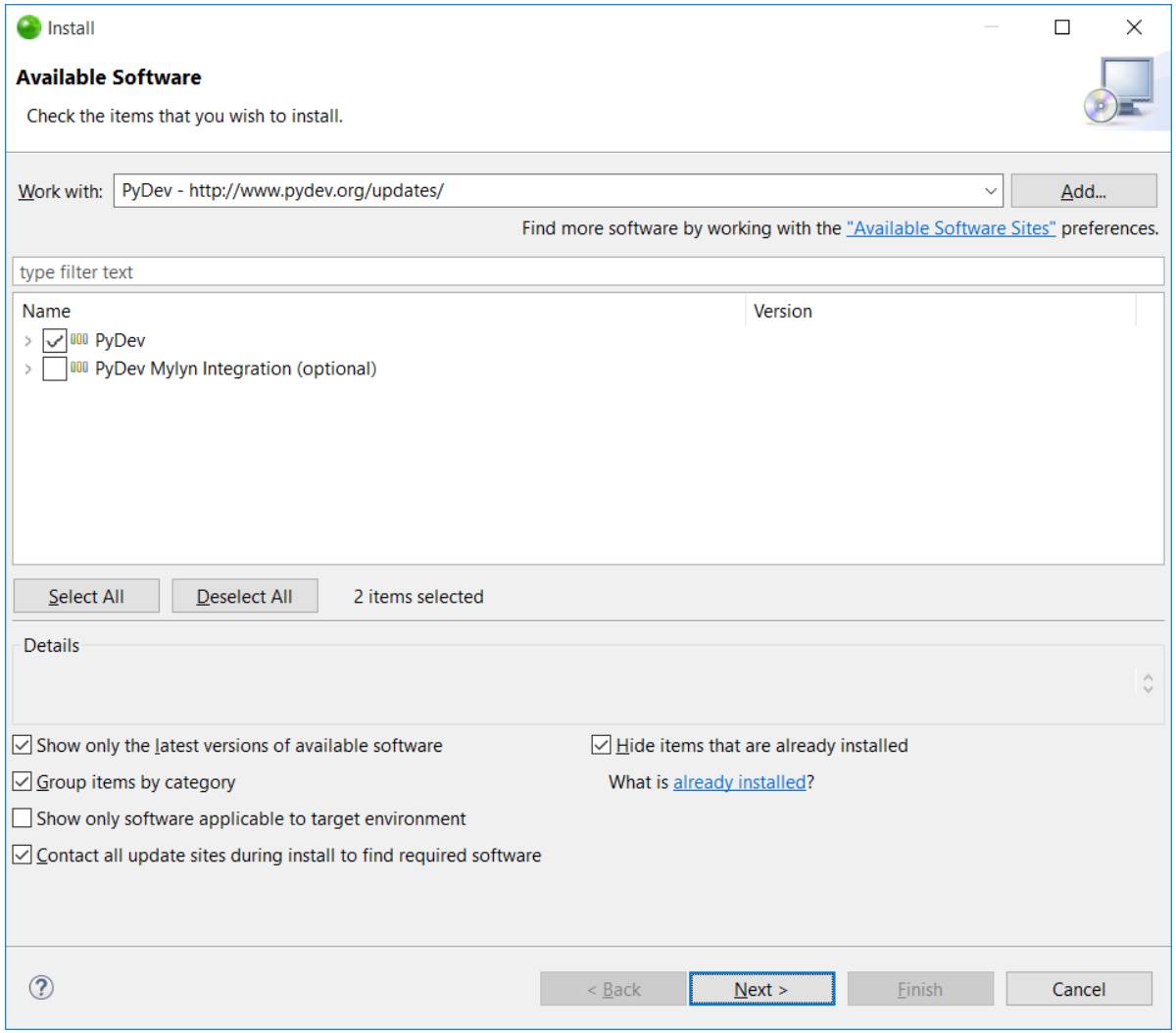
Capture URL

- Follow prompts



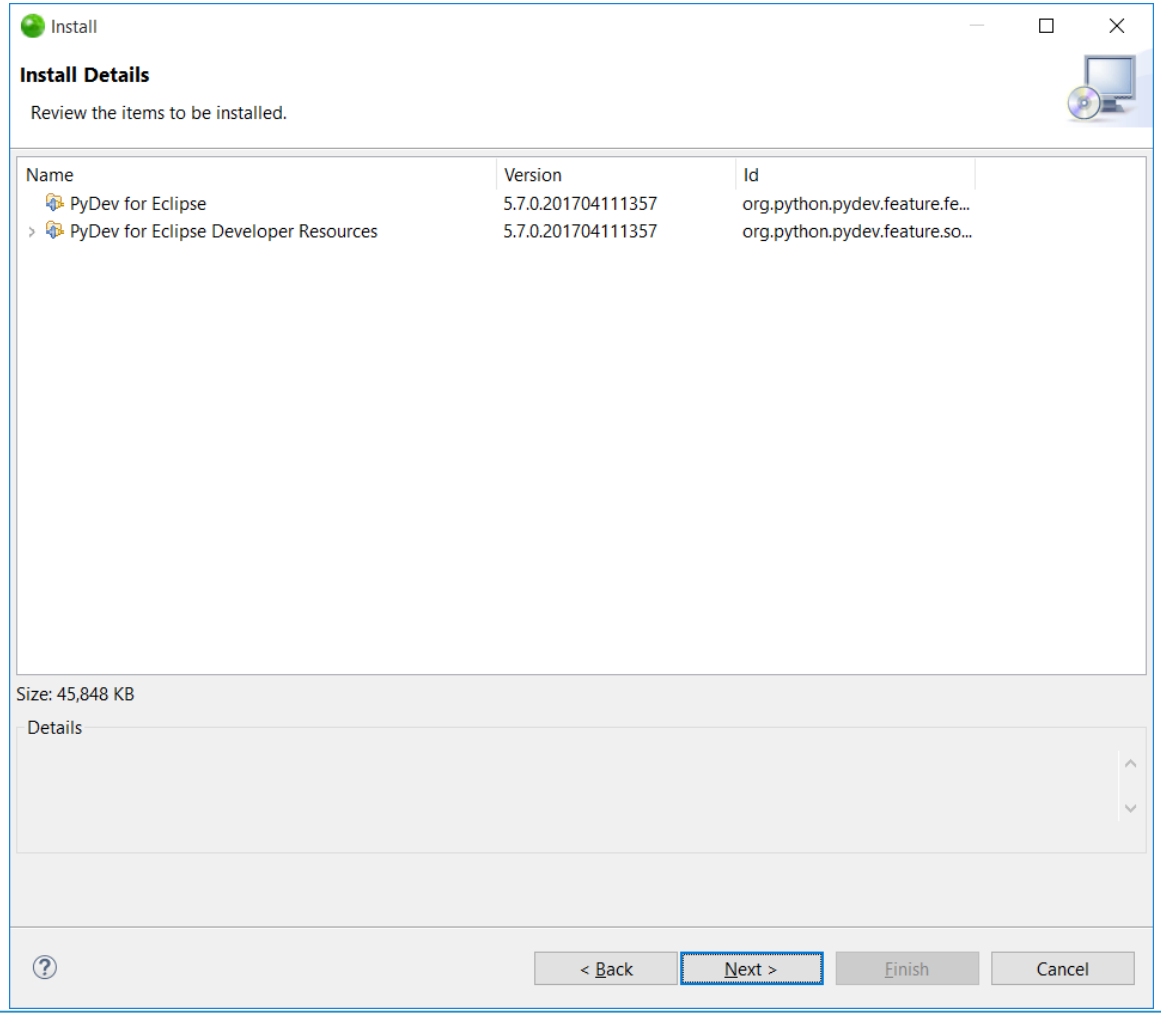
Editor for Eclipse

- Select what you like
- Next



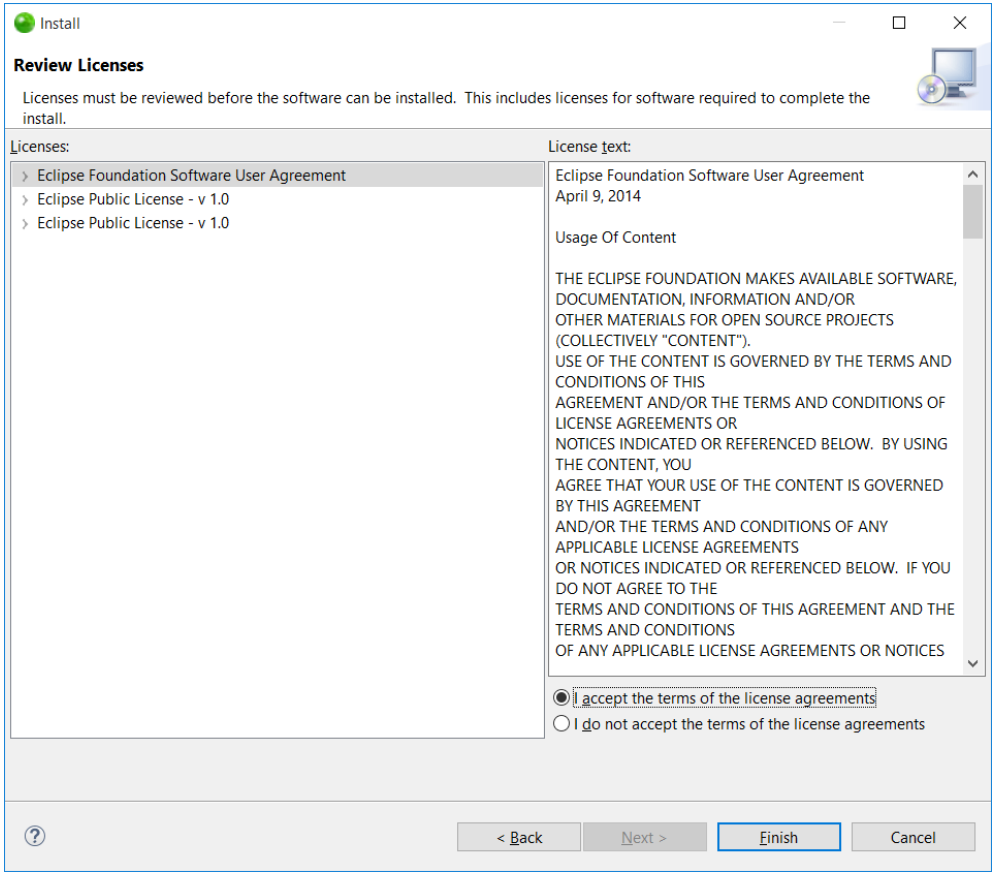
Confirm versions

- Next



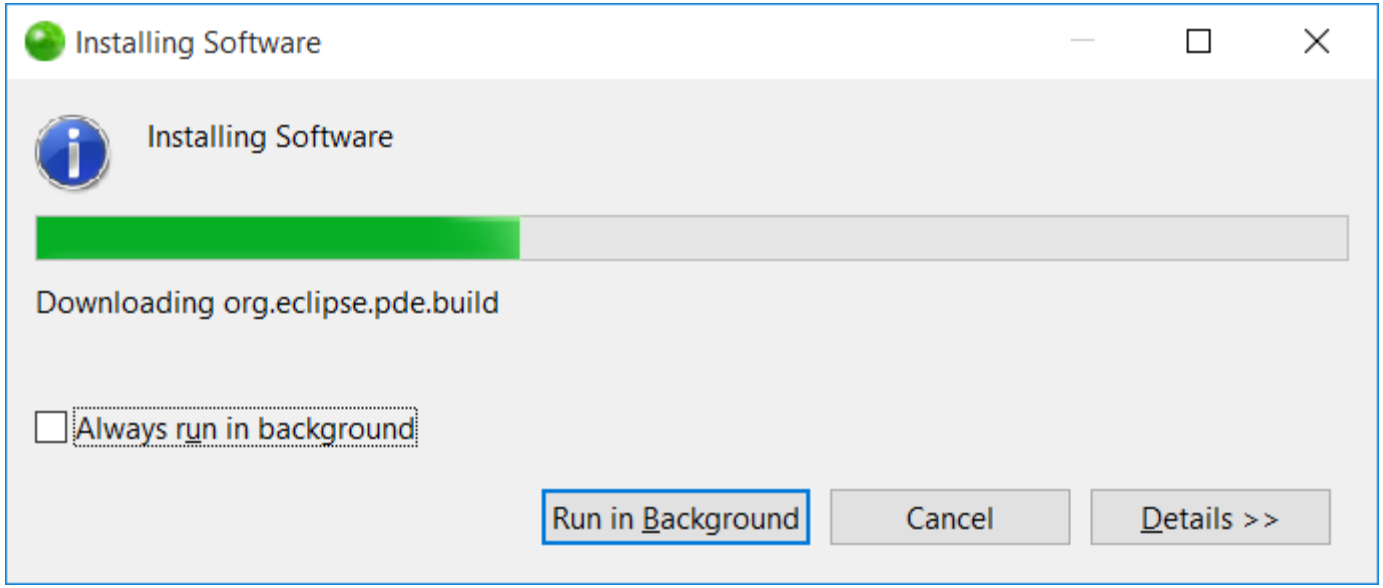
Accept terms and EULA

- Finish



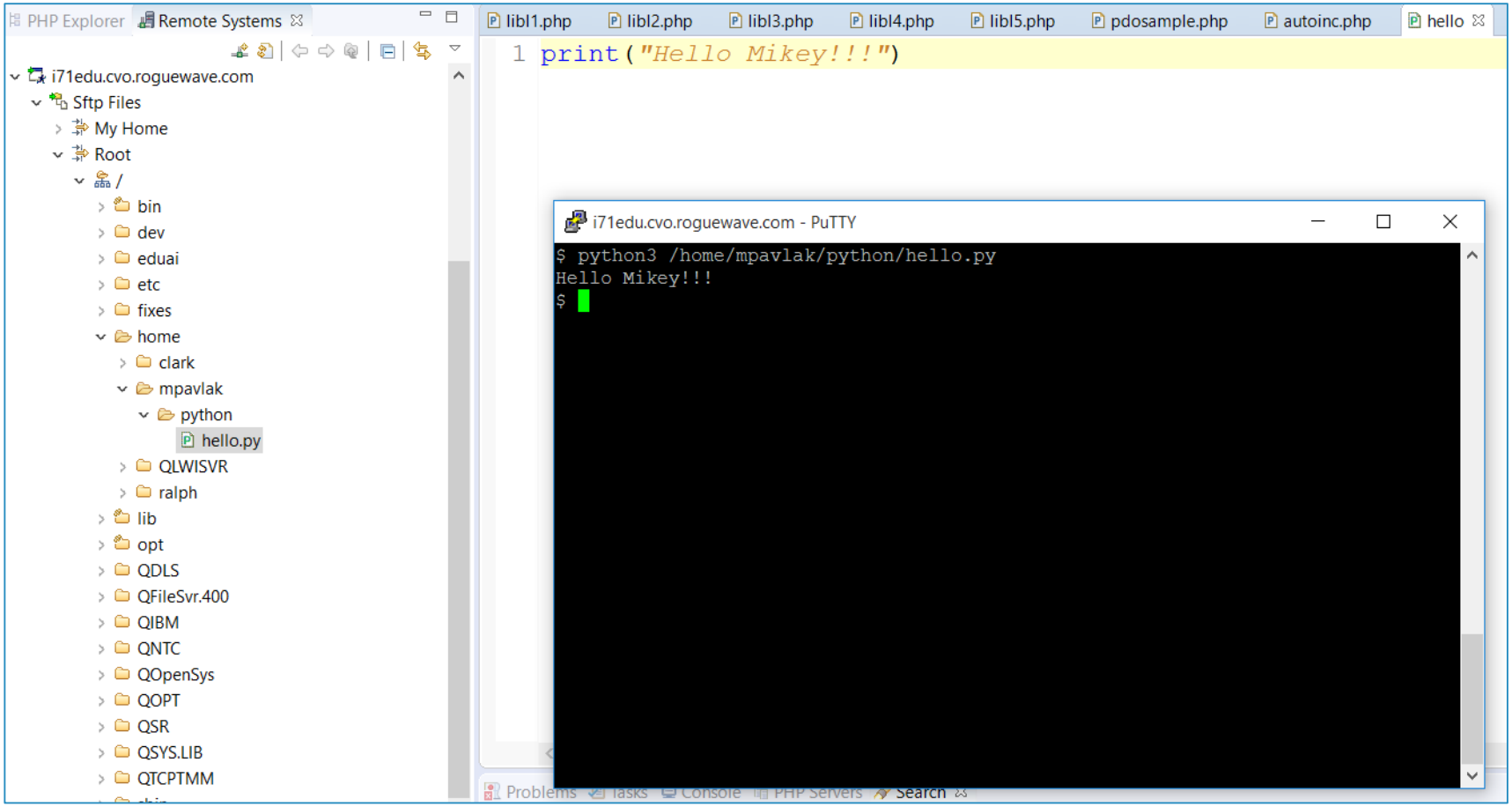
Watch the pretty status bar

skwib.com



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

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

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

Hello world….again…

- 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!


  $
```

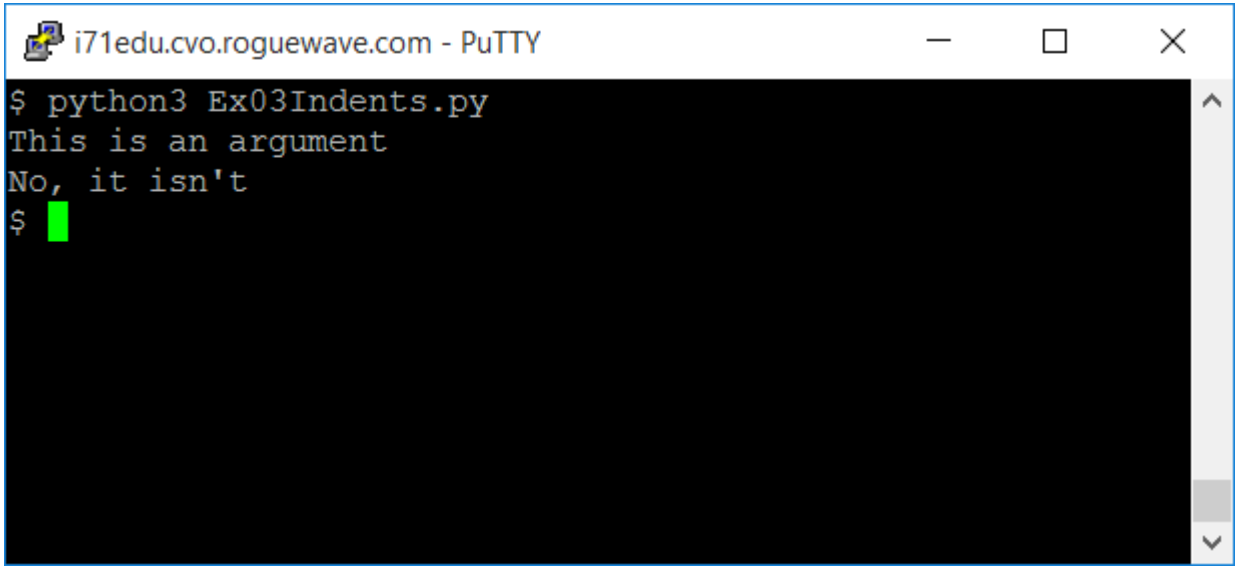
Syntax

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

```
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 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

The air speed velocity

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

Of an unladen swallow.

bigredbubbles5

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

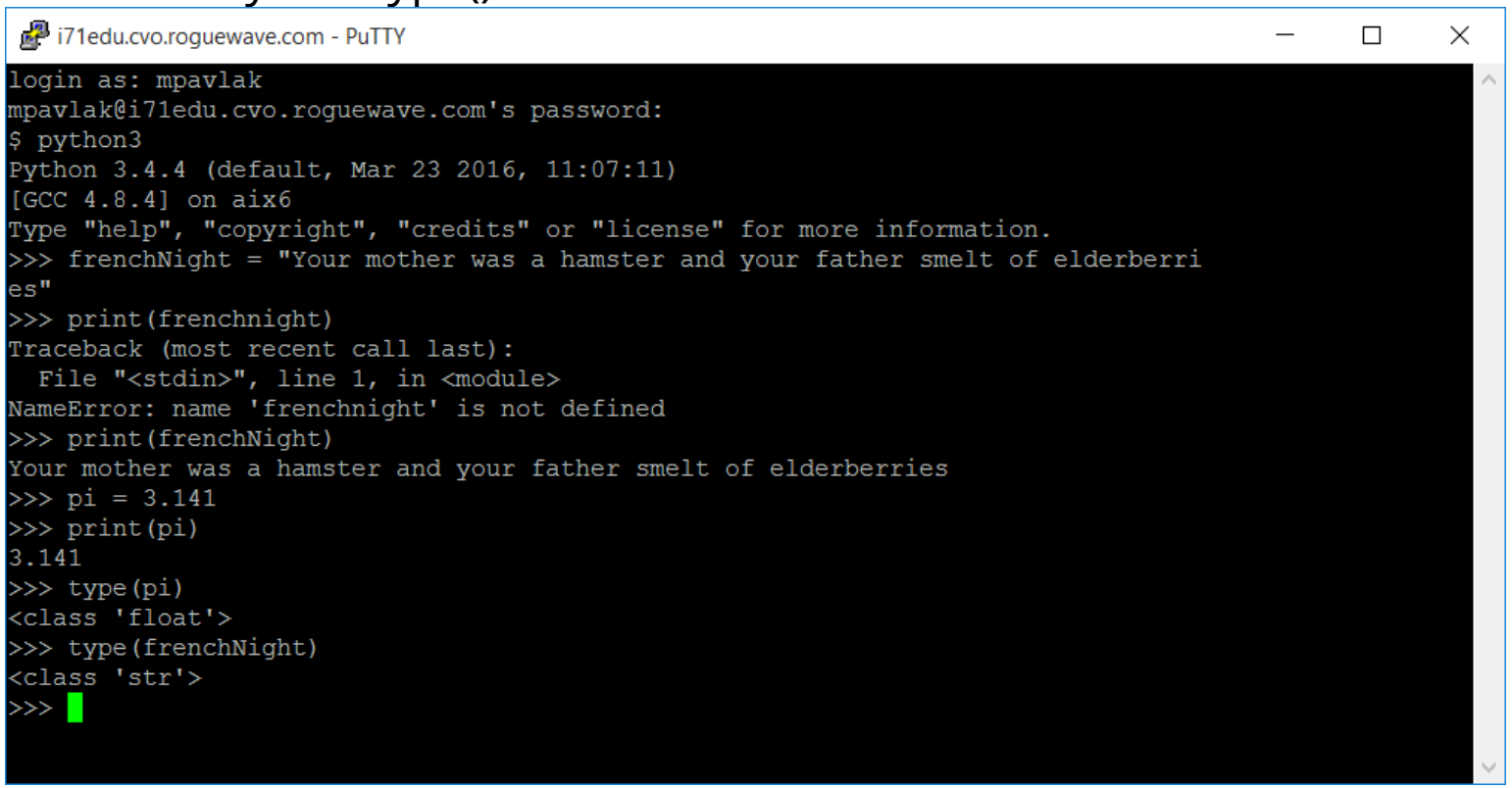
Built in types

- 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)

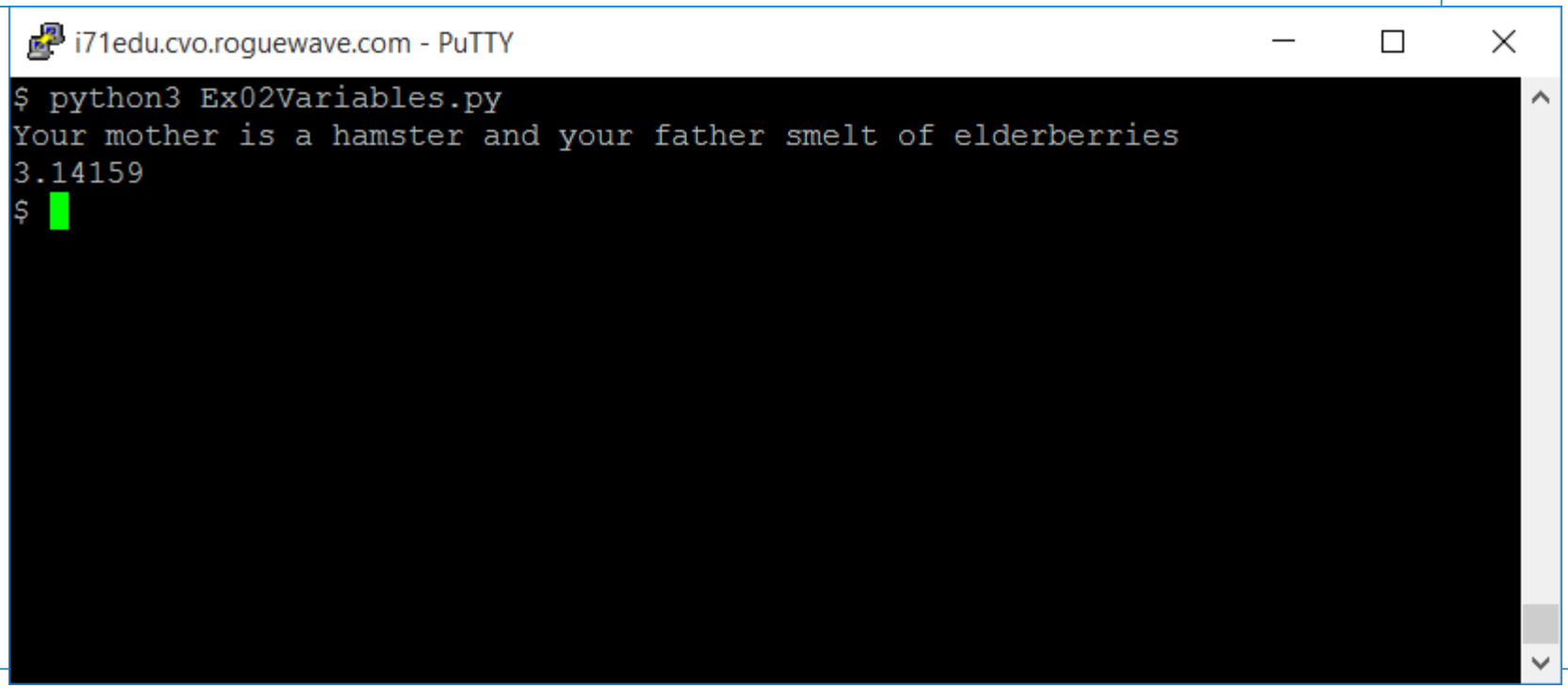
Variables on the fly

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



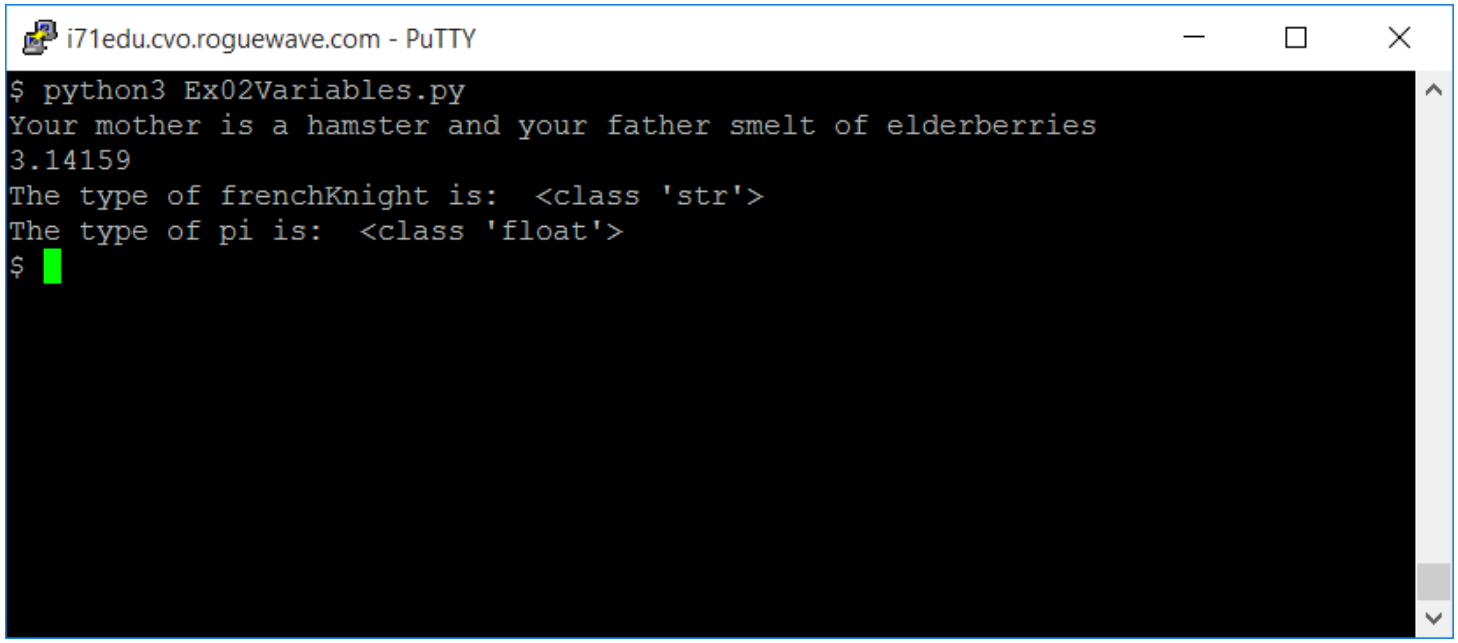
Variables in a file

```
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)
```



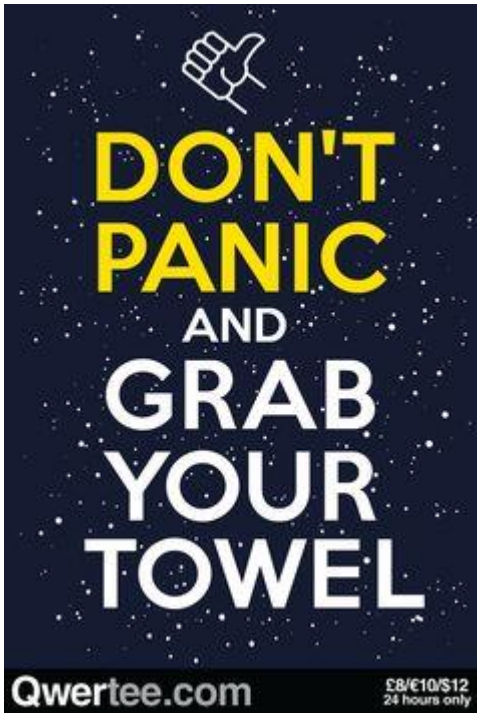
Data Type?

```
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))
```



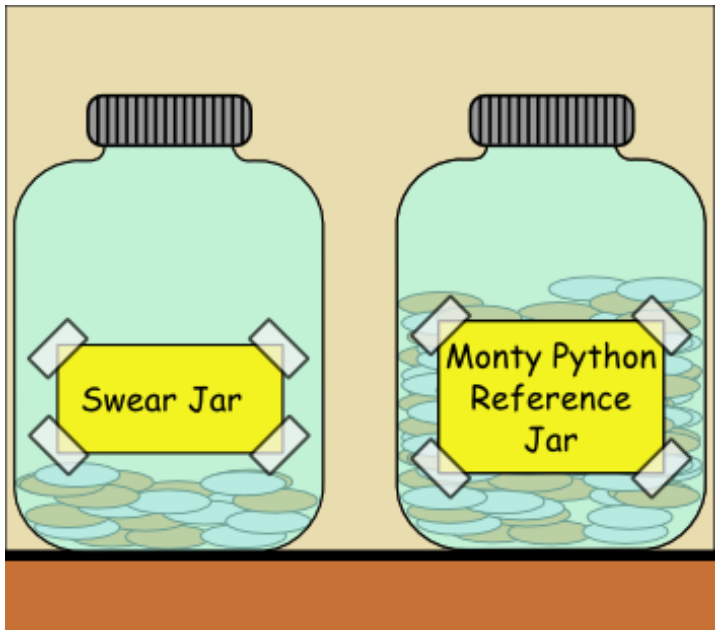
Every variable is implemented as a class!

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

- Numbrs - 3 Data Types
  - Int 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

String formatting

- Interpolation, of sorts

```
1 #
2 # String example
3 #
4
5 count = 0
6 while count < 6:
7     string1 = "I have {} dead parrots!".format(count)
8     print(string1)
9     count = count+1
10 print("\nThank you for shopping!")
```

Lists

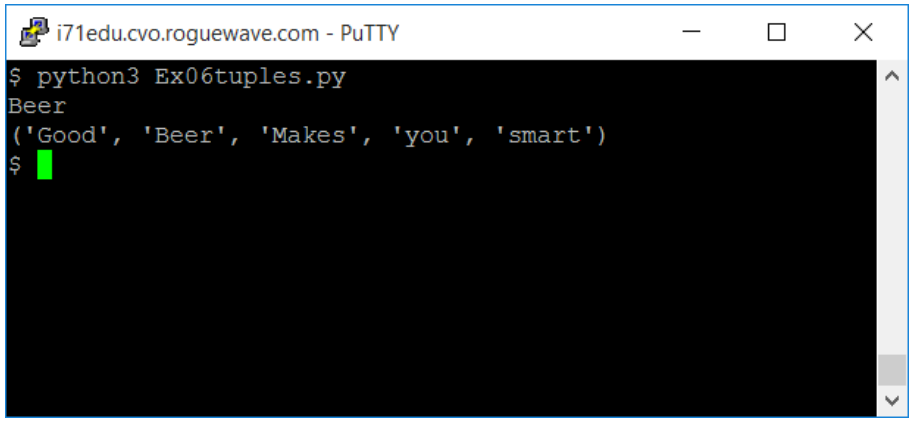
- Ordered group, similar to array
- Different data types, ok
- Multi-dimensional (sub lists)
- Mutable (changeable)

```
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)
```

Tuples

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

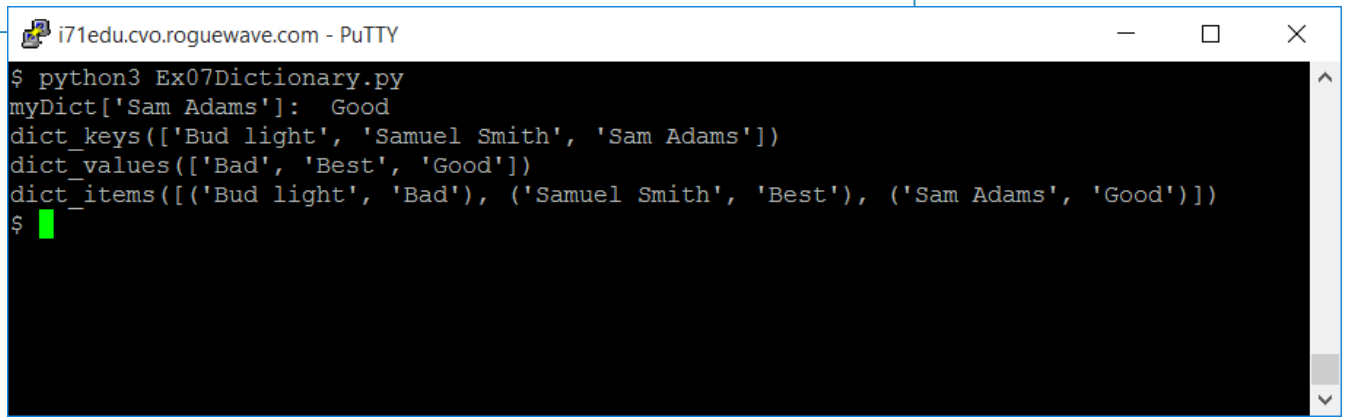
```
1 #
2 # Tuples Examples
3 #
4
5 mytuple = ("Good", "Beer", "Makes", "you", "smart")
6 print(mytuple[1])
7 print(mytuple)
```



Dictionary

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

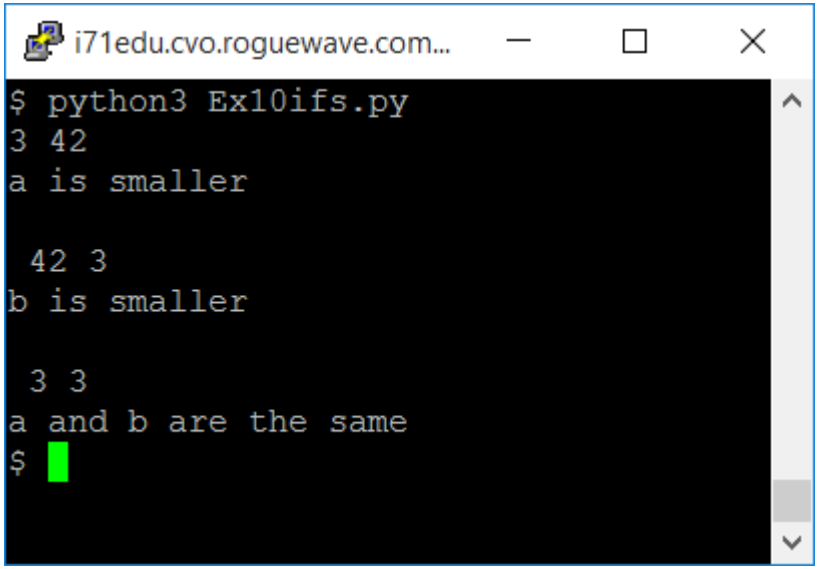
```
1#
2 # Dictionary Examples
3 #
4
5 myDict = {"Sam Adams":"Good", "Samuel Smith":"Best", "Bud light": "Bad"}
6
7 print("myDict['Sam Adams']: ", myDict["Sam Adams"])
8
9 print(myDict.keys())
10 print(myDict.values())
11 print(myDict.items())
```



Control Structures

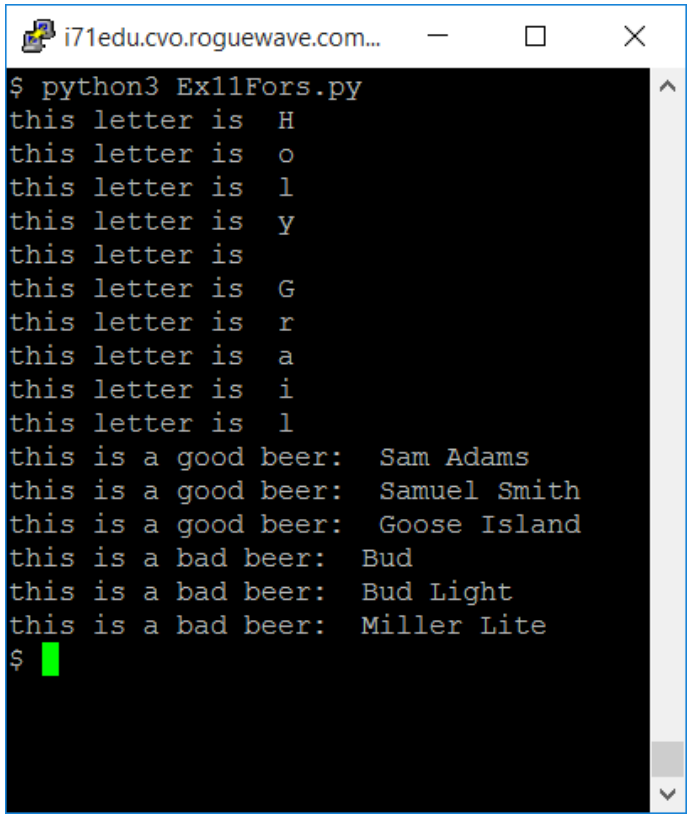
Ifs

```
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")
```



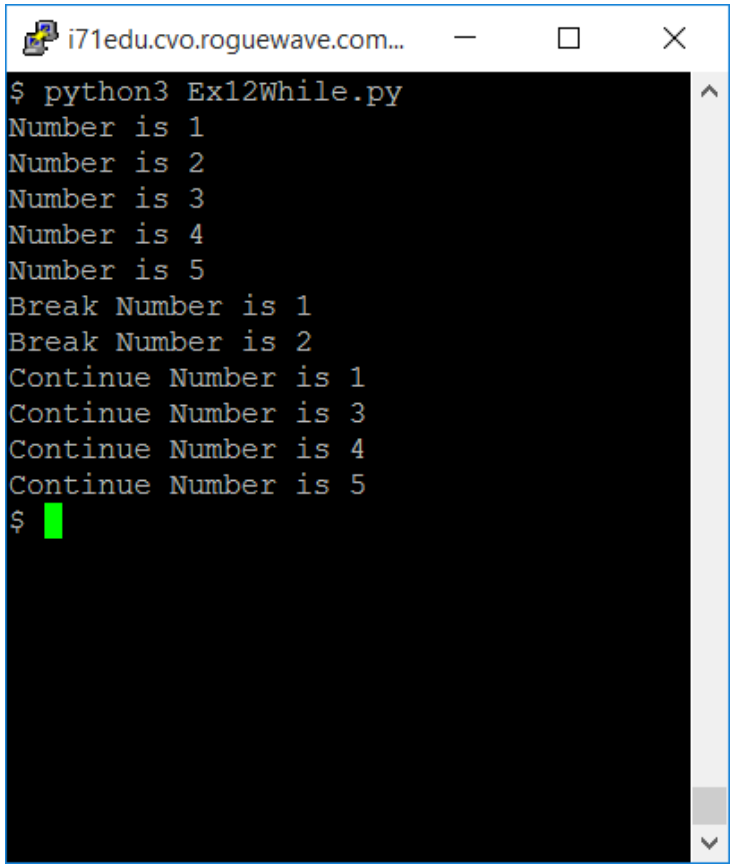
For loop

```
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

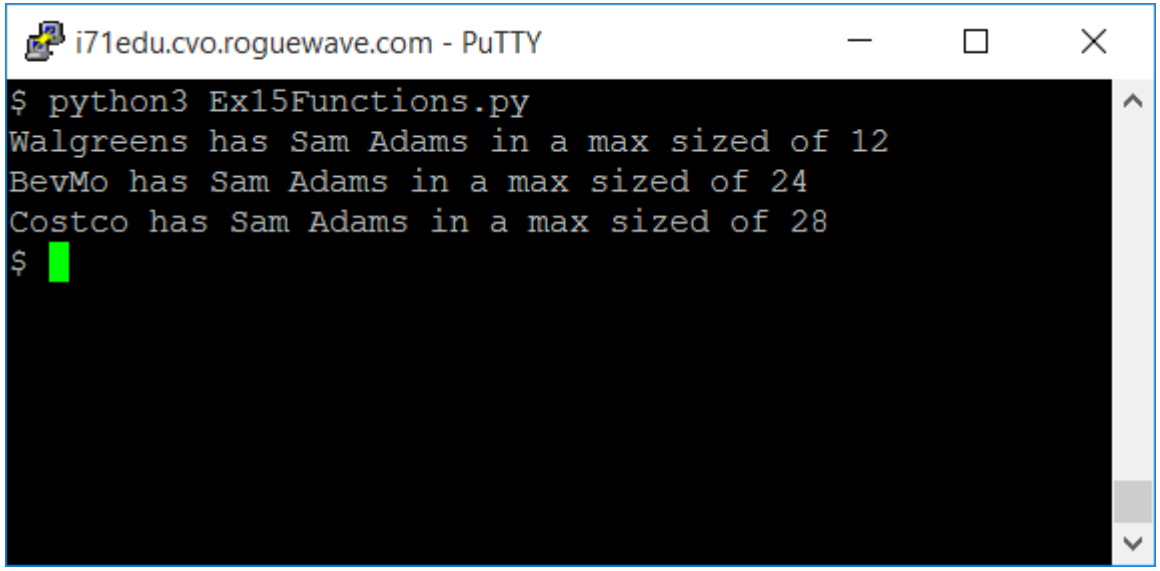
```
1  #
2  # While Loop Examples
3  #
4
5  count, limit = 0,5
6  while count < limit:
7      count = count+1
8      print("Number is", count)
9
10 count = 0
11 while count < limit:
12     count = count+1
13     if count==3:
14         break
15     print("Break Number is", count)
16
17
18 count = 0
19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



Functions

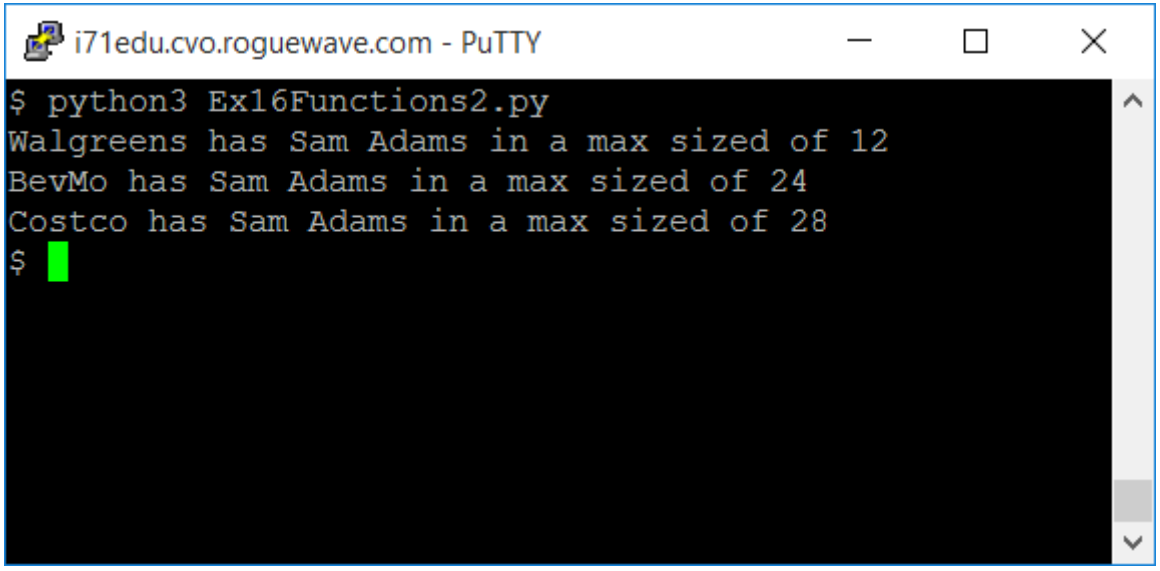
Basic functions

```
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)
```



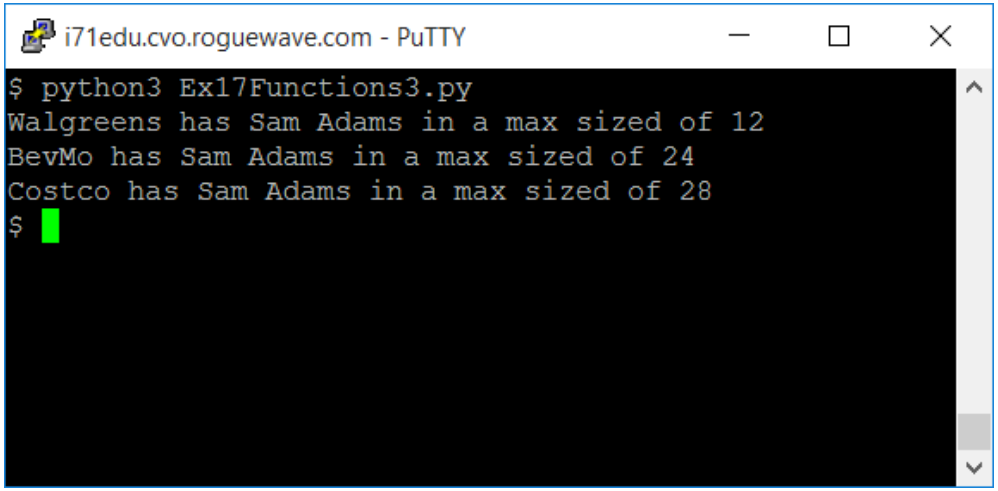
Functions with defaults

```
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)
```

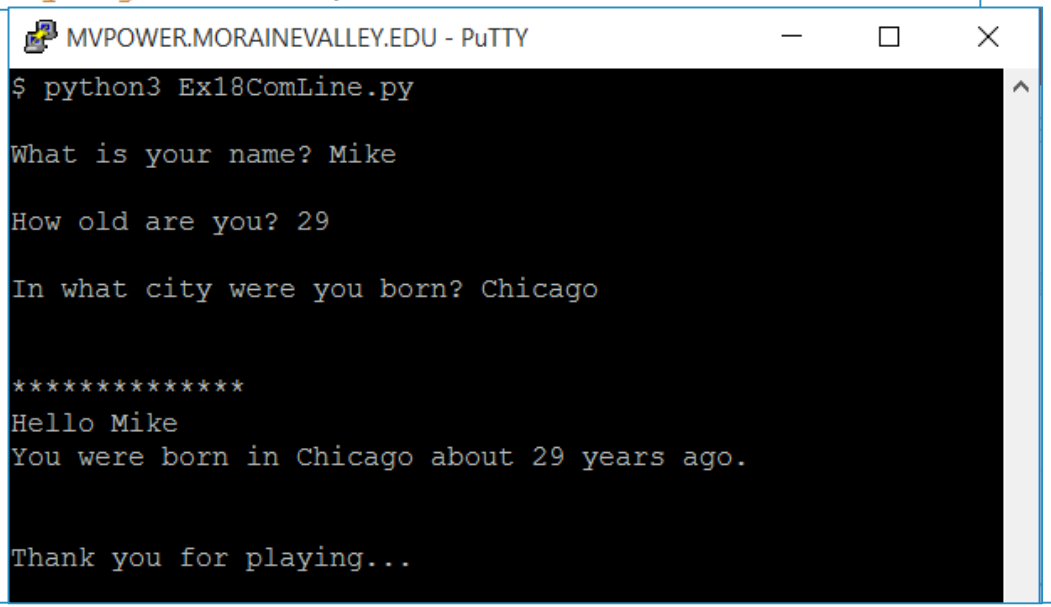


Command Line

Input from command line

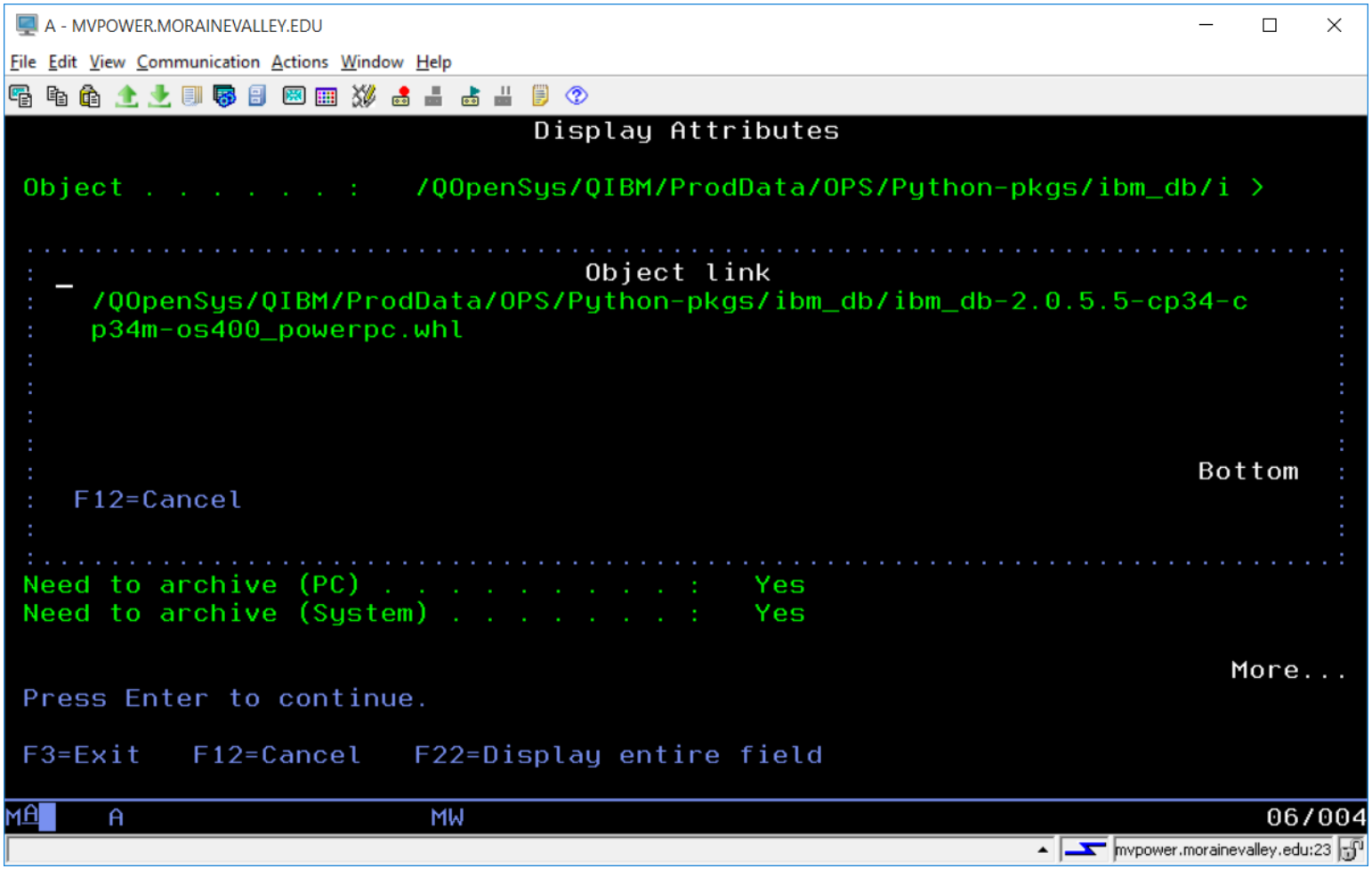
- Talk with the script

```
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:

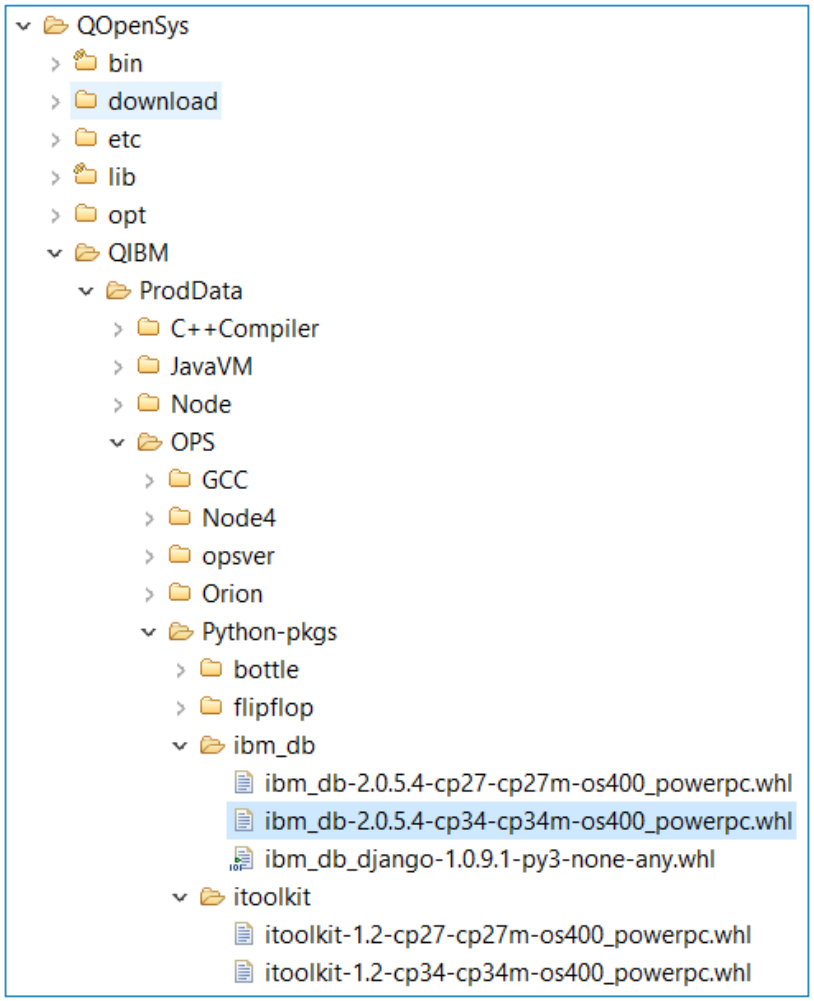
```
pip3 install /QOpenSys/QIBM/ProdData/OPS/Python-pkgs/itoolkit/itoolkit-*-cp34m-*.whl
```

To install FastCGI gateway support:

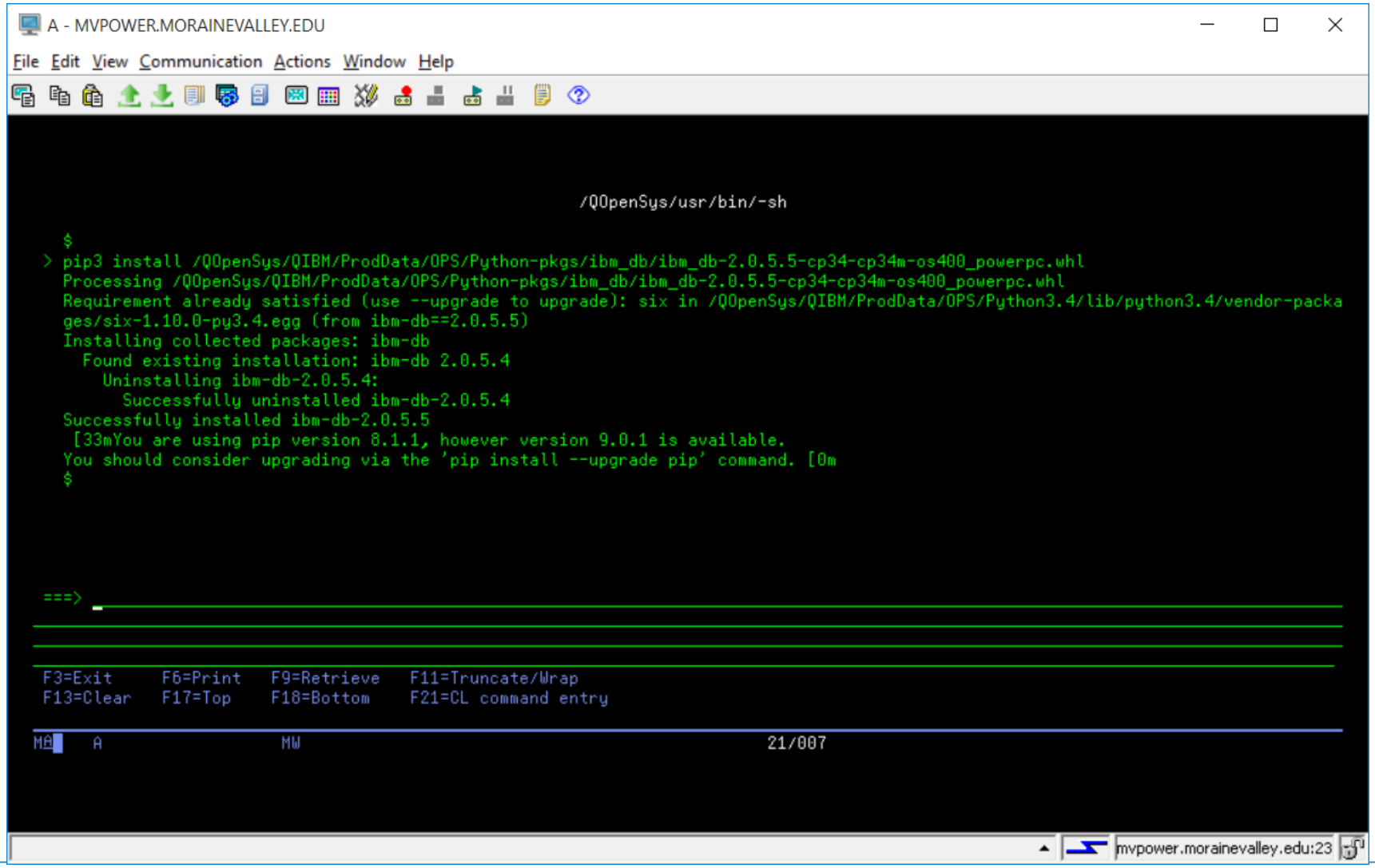
```
pip3 install /QOpenSys/QIBM/ProdData/OPS/Python-pkgs/flipflop/flipflop-*-py34-*.whl
```

Find the connector

- YMMV
- With wheels



Run the pip install…



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
$
```

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")
```

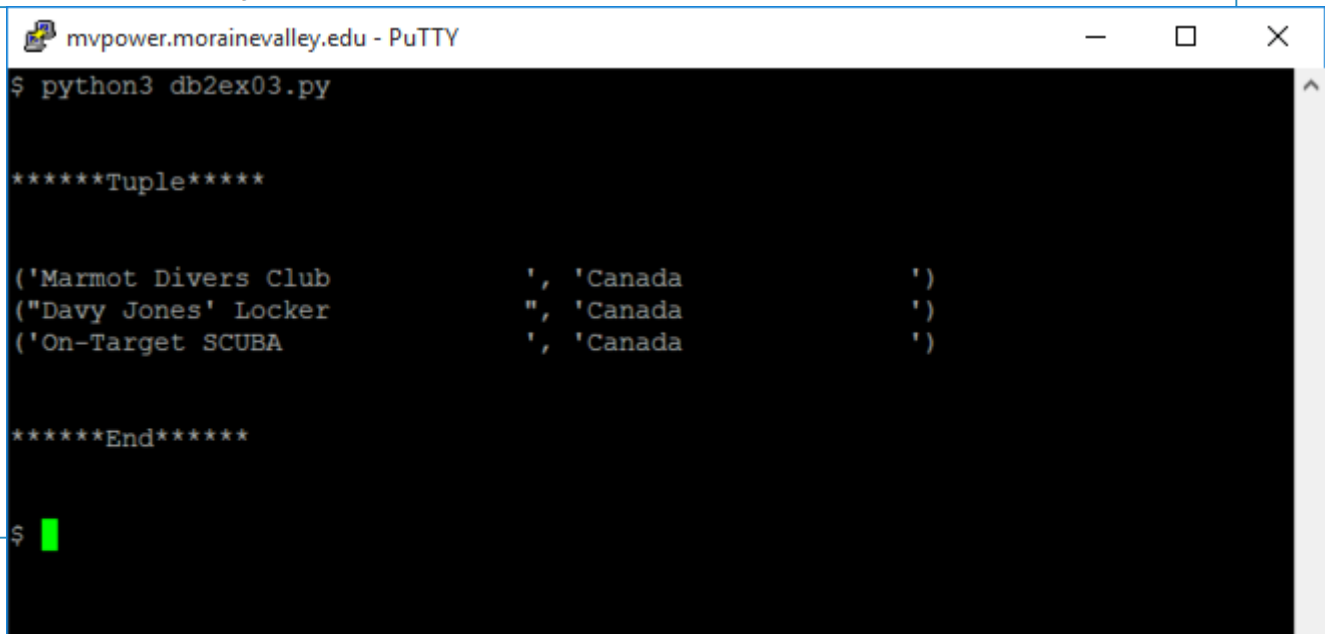
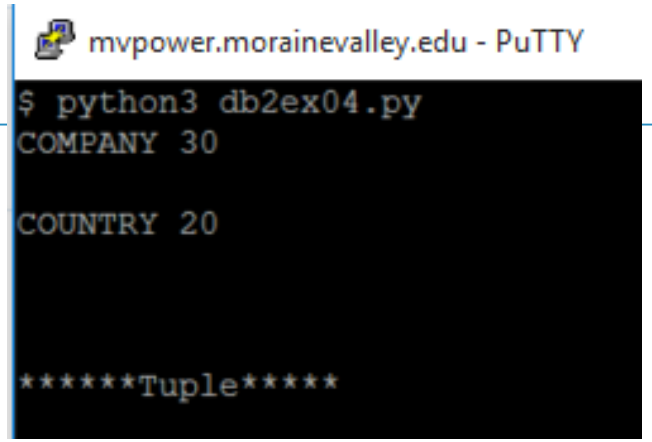


Table information

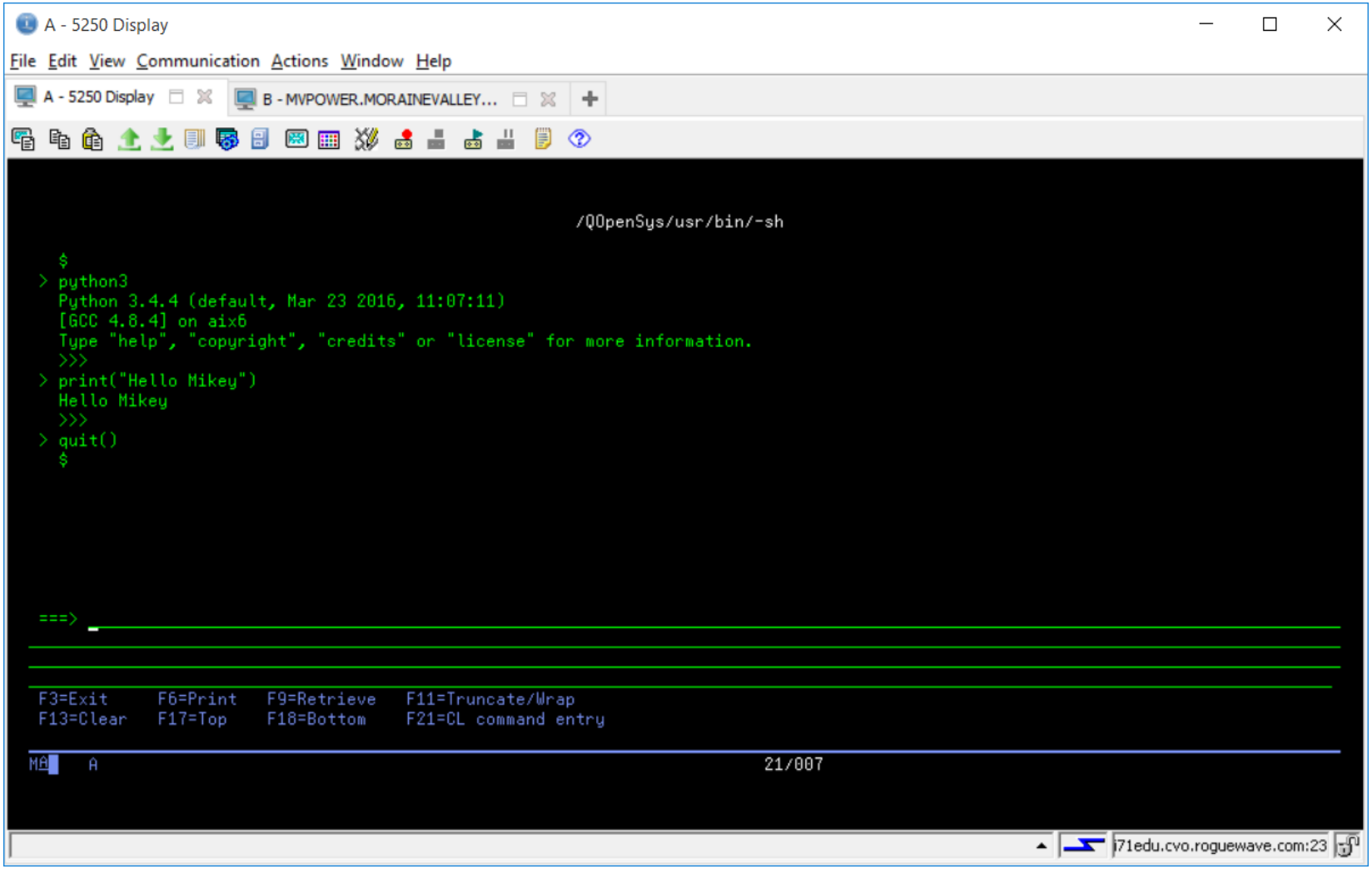
```
1  import ibm_db_dbi as dbi
2  conn = dbi.connect()
3  sql = "SELECT COMPANY, COUNTRY FROM ZENDSVR6.SP_CUST where country = 'Canada'"
4  c01 = conn.cursor()
5  c01.execute(sql)
6  desc = c01.description
7  print(desc[0][0], desc[0][4], "\n")
8  print(desc[1][0], desc[1][4], "\n")
9
10 #Bring it in 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



Who was that masked man?

Mike Pavlak – Solution Consultant, Rogue Wave Software

Mike.Pavlak@roguewave.com

Thank You!


RogueWave
SOFTWARE
Accelerating Great Code