Introduction to Python

- Objectives
  - Learn the history of Python
  - See the differences between Python 2 and Python 3
  - Install and configure Python on your OS
  - Become proficient with the interactive shell
  - Learn about different implementations of Python
  - Choose an IDE

What is Python?

- High-level programming language
- Interpreted
- Object-oriented (especially Python 3)
- Strongly-typed with dynamic semantics
- Syntax emphasizes readability
- Supports modules and packages
- Batteries included (large standard library [1])

A brief history of Python

Guido van Rossum begins development on Python

Python 2.0 released

Python 3.0 released

1989

2000

2008

Getting started

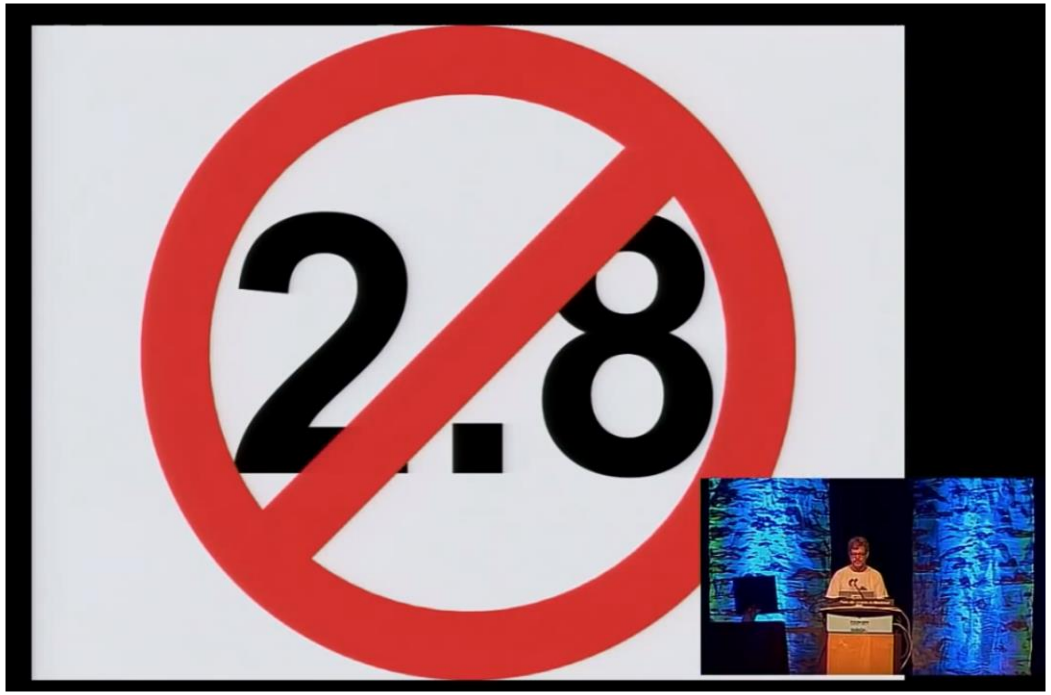
- Installing and configuring Python
  - http://www.python.org/download/
  - Some versions Python come pre-configured on some OS's

Python 2	Python 3
OS X Linux (Ubuntu)	Linux (Ubuntu)

Python 2 vs. Python 3

This course focuses on Python 3 because it literally is the future:

- Guido has announced that there will NOT be new features in Python 2



Python 2 vs. Python 3

print(“”) vs
print “”

Python 3.x is the present
and future of the language

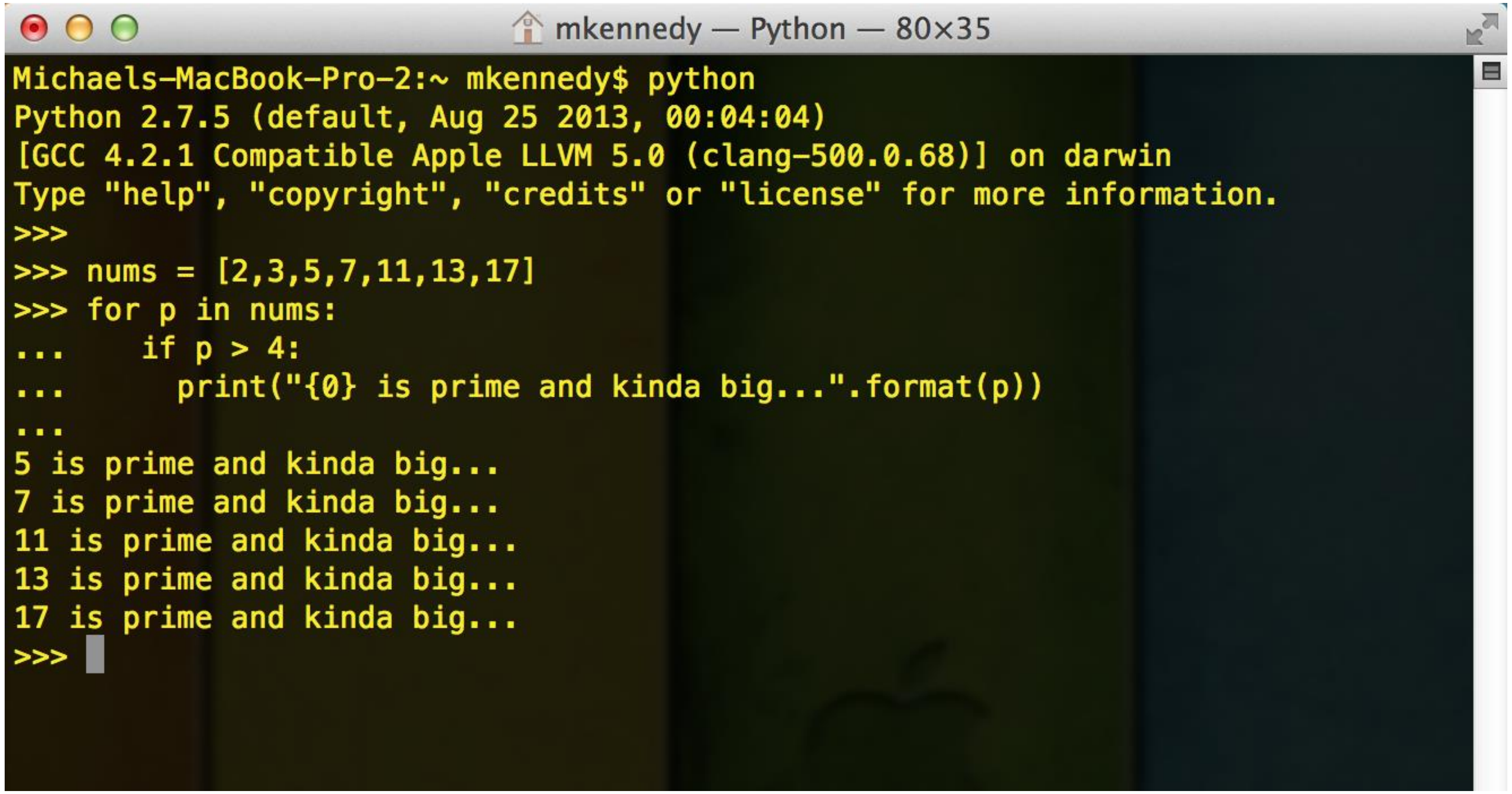
2.7 is the
end-of-life release

intentionally
incompatible

Full OO
support

Using the interactive shell

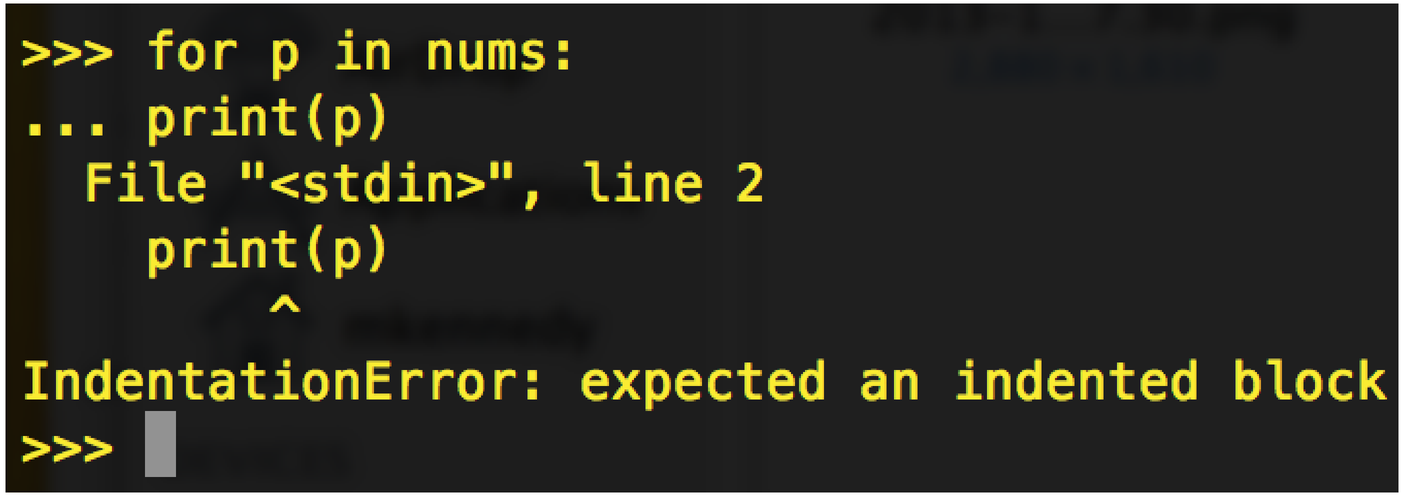
- Python comes with REPL (read–eval–print loop) interactive language shell.



```
Michaels-MacBook-Pro-2:~ mkennedy$ python
Python 2.7.5 (default, Aug 25 2013, 00:04:04)
[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.0.68)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> nums = [2,3,5,7,11,13,17]
>>> for p in nums:
...     if p > 4:
...        print("{0} is prime and kinda big...".format(p))
...
5 is prime and kinda big...
7 is prime and kinda big...
11 is prime and kinda big...
13 is prime and kinda big...
17 is prime and kinda big...
>>>
```

Using the interpreter [tips]

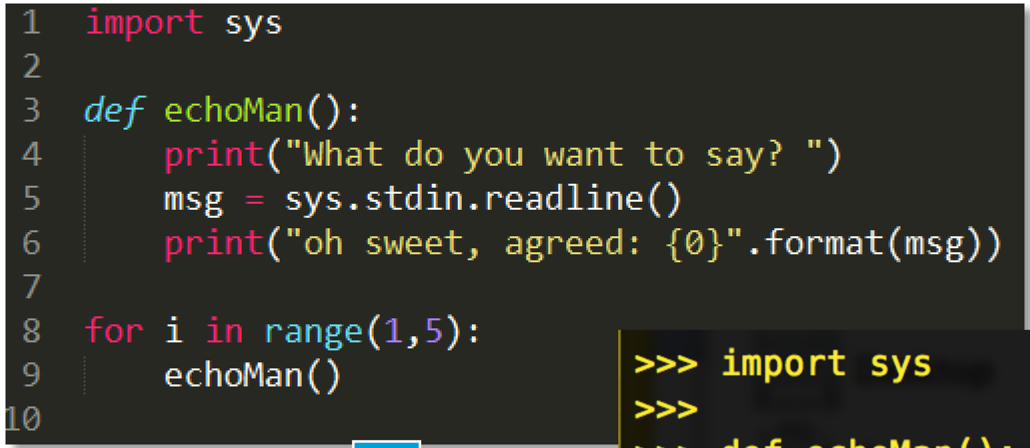
- Language shell is good for experimenting
  - For real programs, we use scripts and maybe an IDE
- Tips:
  - Modules and scripts can be imported (e.g. `import  pymongo`)
  - Single line expressions and methods can be run
  - Multi line expressions can be entered (... implies more input)
  - Don’t forget the spaces for multi lines.



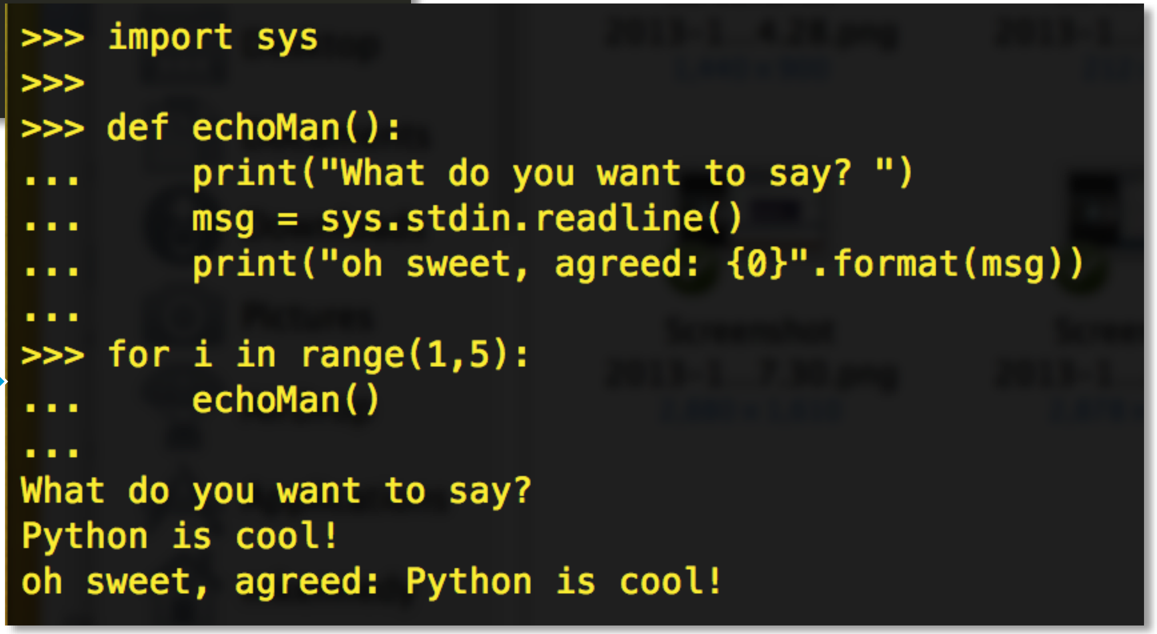
```
>>> for p in nums:
... print(p)
  File "<stdin>", line 2
    print(p)
        ^
IndentationError: expected an indented block
>>> 
```

Using the interpreter [text editor]

- For more complex code, you can use a text editor and then paste multiple lines (or use ‘real’ scripts of course)



```python
import sys

def echoMan():
    print("What do you want to say? ")
    msg = sys.stdin.readline()
    print("oh sweet, agreed: {0}".format(msg))

for i in range(1,5):
    echoMan()

```

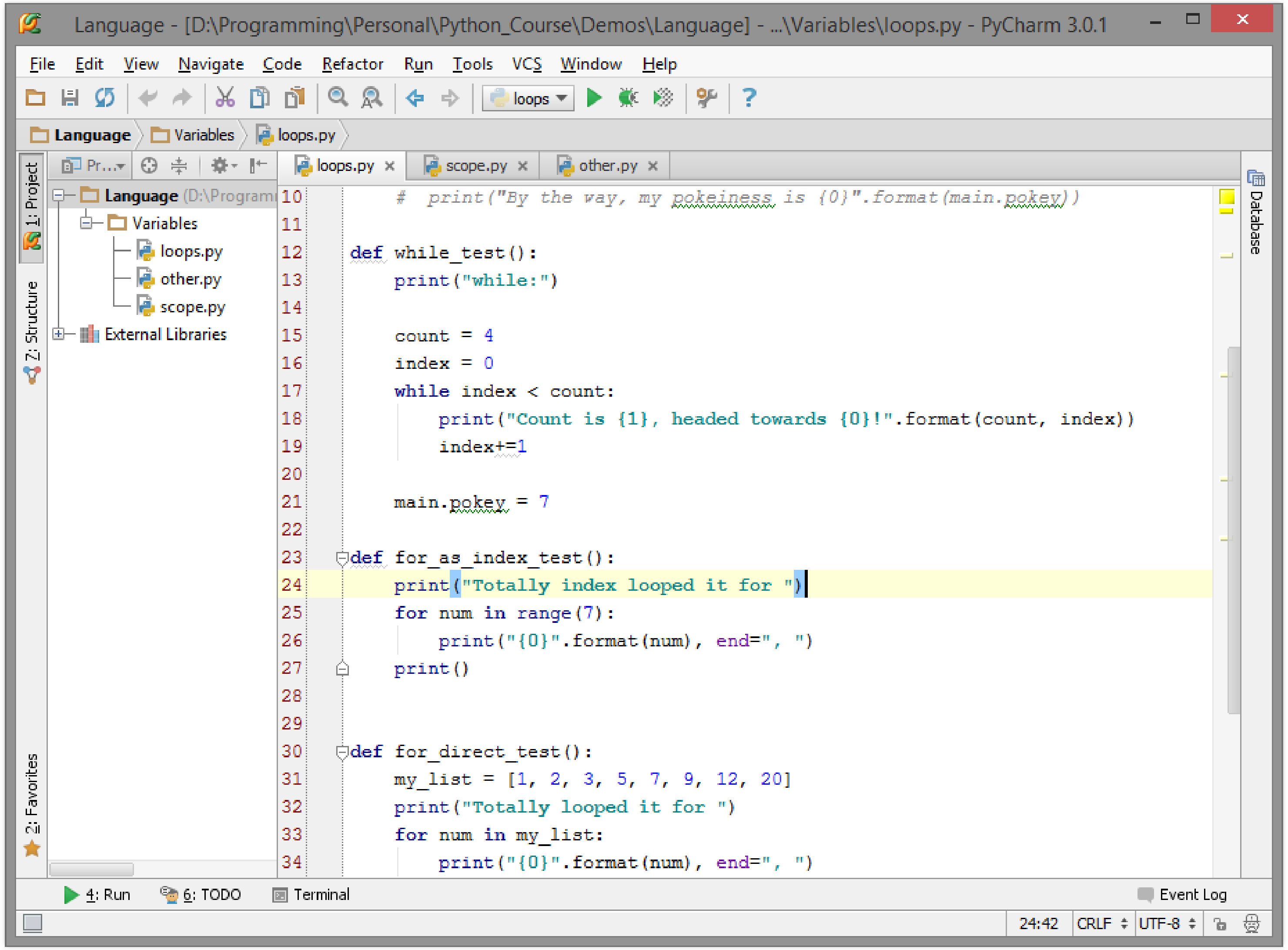


```
>>> import sys
>>>
>>> def echoMan():
...     print("What do you want to say? ")
...     msg = sys.stdin.readline()
...     print("oh sweet, agreed: {0}".format(msg))
...
>>> for i in range(1,5):
...     echoMan()
...
What do you want to say?
Python is cool!
oh sweet, agreed: Python is cool!
```

Choosing an IDE

- IDEs have many advantages
  - Quick access to multiple files within a project
  - Debugging via breakpoints and code stepping
  - Creation and management of virtual environments
  - Unit testing
  - Refactoring
  - Code completions (intellisense)
  - Go to definition
  - Framework support (Django, Pyramid, etc.)
  - Code inspection
  - Code navigation
- IDEs are not required
  - Can use a basic text editor
  - Can use full featured editors (e.g. PyCharm, Visual Studio)

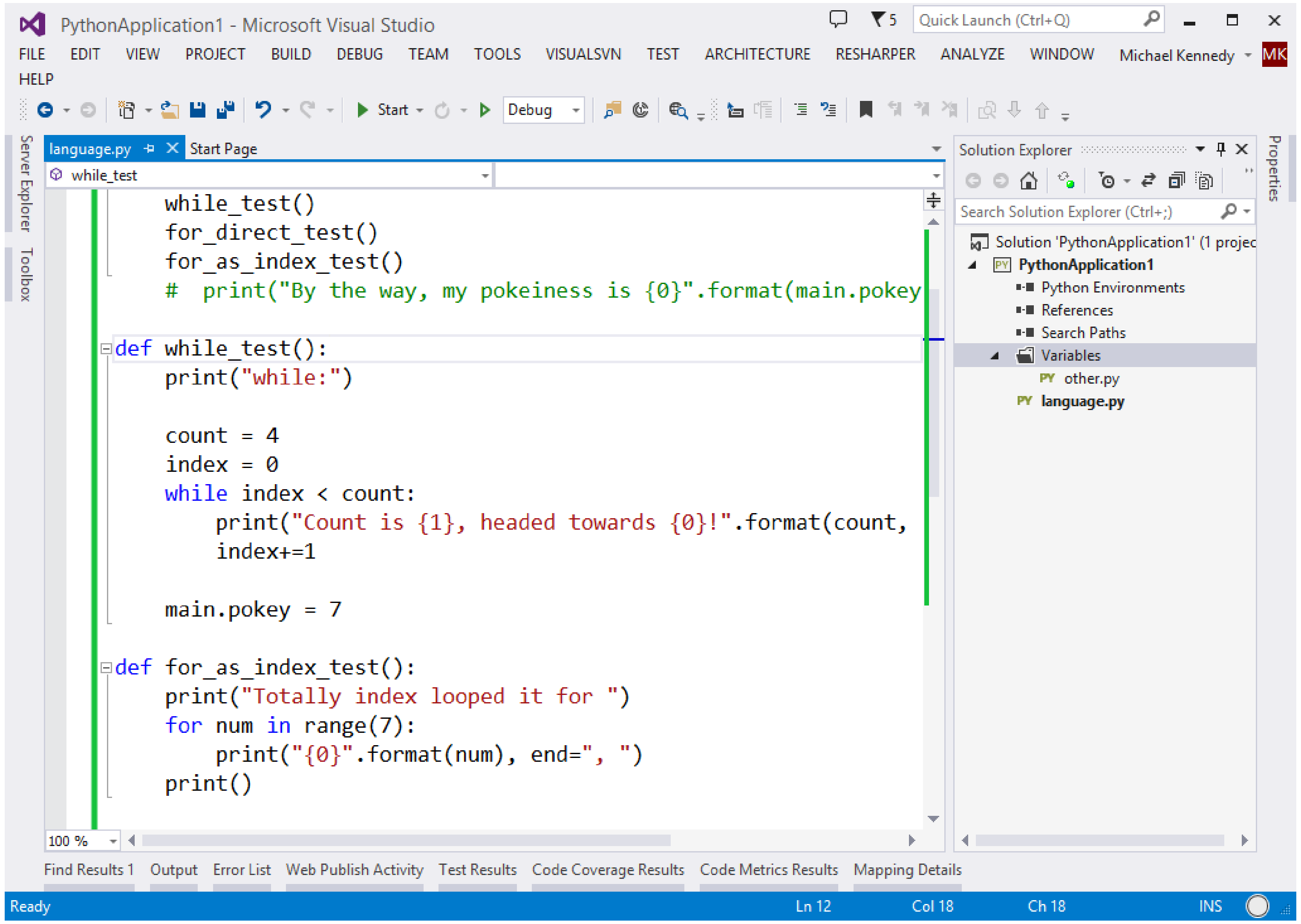
Choosing an IDE [PyCharm]


Language - [D:\Programming\Personal\Python_Course\Demos\Language] - ...\Variables\loops.py - PyCharm 3.0.1
File Edit View Navigate Code Refactor Run Tools VCS Window Help
loops
Language Variables loops.py
1: Project
7: Structure
2: Favorites
Pr...
Language (D:\Program
Variables
loops.py
other.py
scope.py
External Libraries
loops.py scope.py other.py
Database
4: Run 6: TODO Terminal
Event Log
24:42 CRLF UTF-8

```
    #  print("By the way, my pokeiness is {0}".format(main.pokey))

def while_test():
    print("while:")

    count = 4
    index = 0
    while index < count:
        print("Count is {1}, headed towards {0}!".format(count, index))
        index+=1

    main.pokey = 7

def for_as_index_test():
    print("Totally index looped it for ")
    for num in range(7):
        print("{0}".format(num), end=", ")
    print()


def for_direct_test():
    my_list = [1, 2, 3, 5, 7, 9, 12, 20]
    print("Totally looped it for ")
    for num in my_list:
        print("{0}".format(num), end=", ")
```

Java-based IDE from JetBrains: http://www.jetbrains.com/pycharm/

Choosing an IDE [Visual Studio + Python Tools]



```
    while_test()
    for_direct_test()
    for_as_index_test()
    #  print("By the way, my pokeiness is {0}".format(main.pokey

def while_test():
    print("while:")

    count = 4
    index = 0
    while index < count:
        print("Count is {1}, headed towards {0}!".format(count,
        index+=1

    main.pokey = 7

def for_as_index_test():
    print("Totally index looped it for ")
    for num in range(7):
        print("{0}".format(num), end=", ")
    print()
```

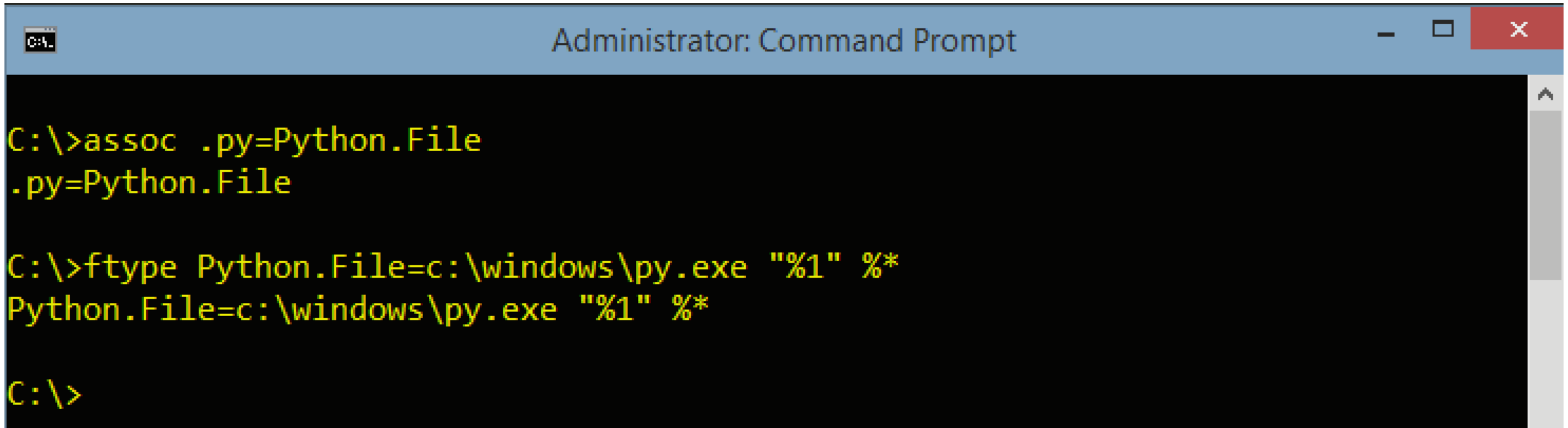
Runs within Visual Studio: http://pytools.codeplex.com/

Running scripts from command-line [Windows]

- Associating Python scripts with the Python runner
  1. **assoc** .py=Python.File
  2. **ftype** Python.File=C:\windows\py.exe "%1" %*

py.exe requires at least Python 3.3 or higher.

Must be run with **elevated privileges**



```
C:\>assoc .py=Python.File
.py=Python.File

C:\>ftype Python.File=c:\windows\py.exe "%1" %*
Python.File=c:\windows\py.exe "%1" %*

C:\>
```

Exploring the standard library

- There are many modules in the standard library. Here are the major functionality areas from
  - Built-in Functions
  - Built-in Types
  - Text Processing Services
  - Data Types
  - Mathematical Modules
  - Functional Programming Modules
  - File and Directory Access
  - Data Persistence
  - Compression and Archiving
  - Common File Formats
  - Cryptographic Services
  - Operating System Services
  - Concurrent Execution
  - Networking
  - Internet Data Handling
  - Structured Markup Processing Tools
  - Internet Protocols and Support
  - Multimedia Services
  - Internationalization
  - Program Frameworks
  - Graphical User Interfaces with Tk
  - Unit testing and mocking
  - Debugging and Profiling
  - Python Runtime Services
  - Custom Python Interpreters
  - Importing Modules
  - Python Language Services
  - Windows Specific Services
  - Unix Specific Services

Summary

- Python was created in 1991 by Guido van Rossum
- Python 3 is cleaner than Python 2
  - Many of the features of Python 3 have been back-ported
- Python 3 has to be installed on OS X and Windows
- Python’s interactive shell lets you try ideas quickly
- PyCharmj and Visual Studio are all good IDEs