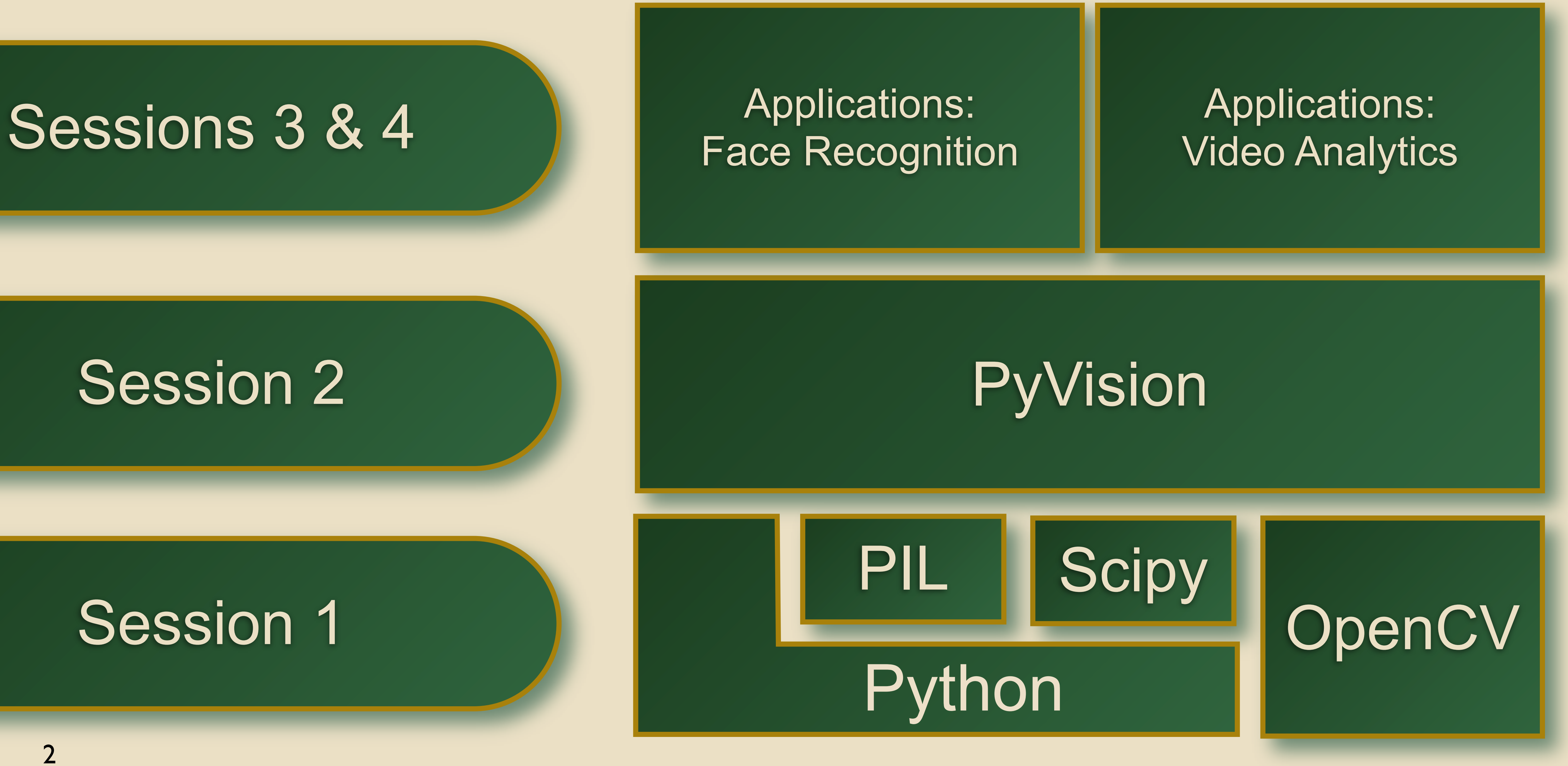
Introduction to PyVision for Computer Vision Applications

David Bolme and Stephen O'Hara

Colorado State University

System Architecture



Sessions 3 & 4

Session 2

Session 1

Applications: Face Recognition

Applications: Video Analytics

PyVision

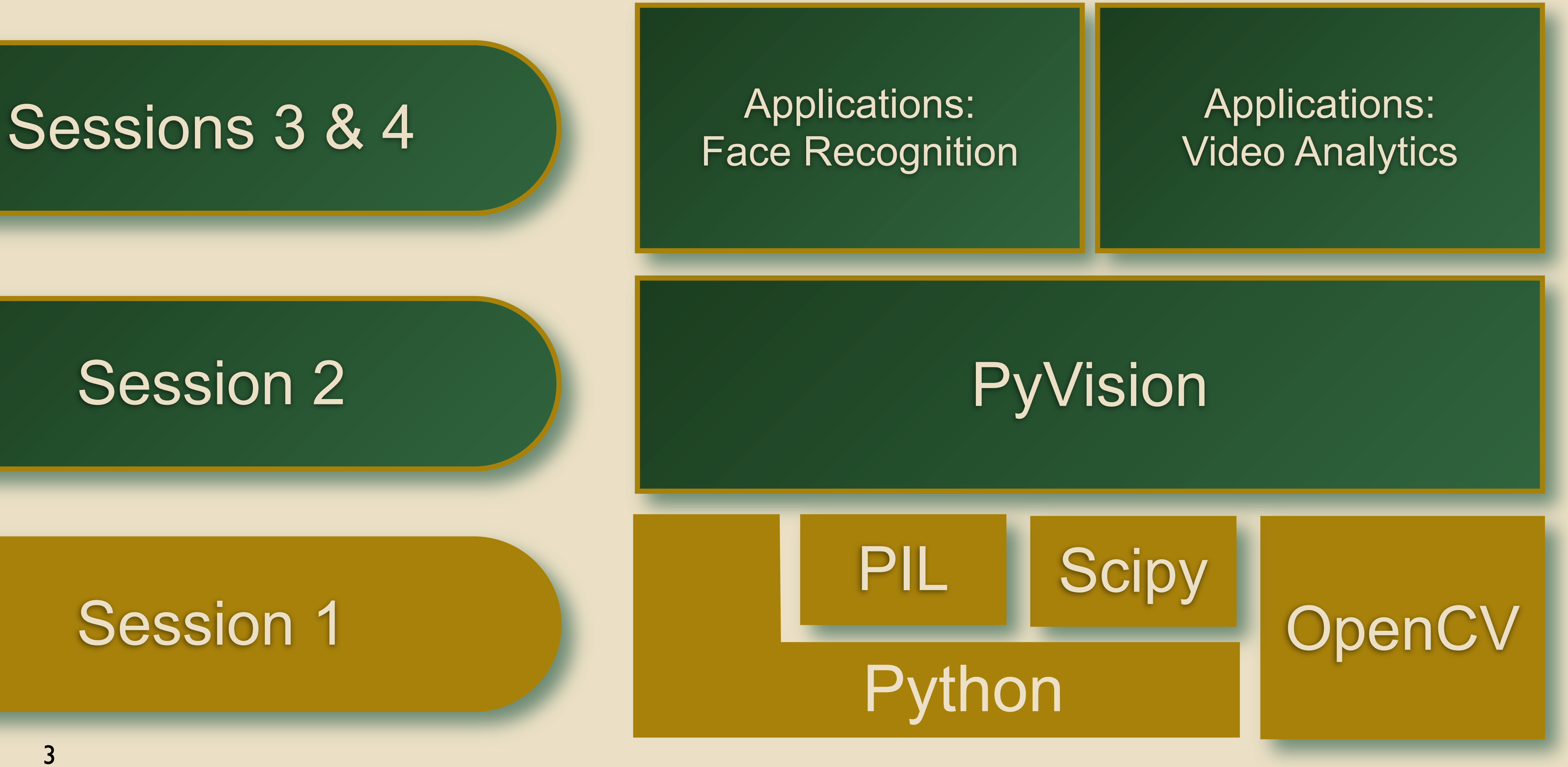
PIL

Scipy

Python

OpenCV

System Architecture



Sessions 3 & 4

Session 2

Session 1

Applications:
Face Recognition

Applications:
Video Analytics

PyVision

PIL

Scipy

Python

OpenCV

Session 1 Goals

- Install Virtual Box and Appliance.
- Hands on help.
- Introduction to Python for computer vision.
    - PIL, NumPy, SciPy, OpenCV

Setup For Department Machines

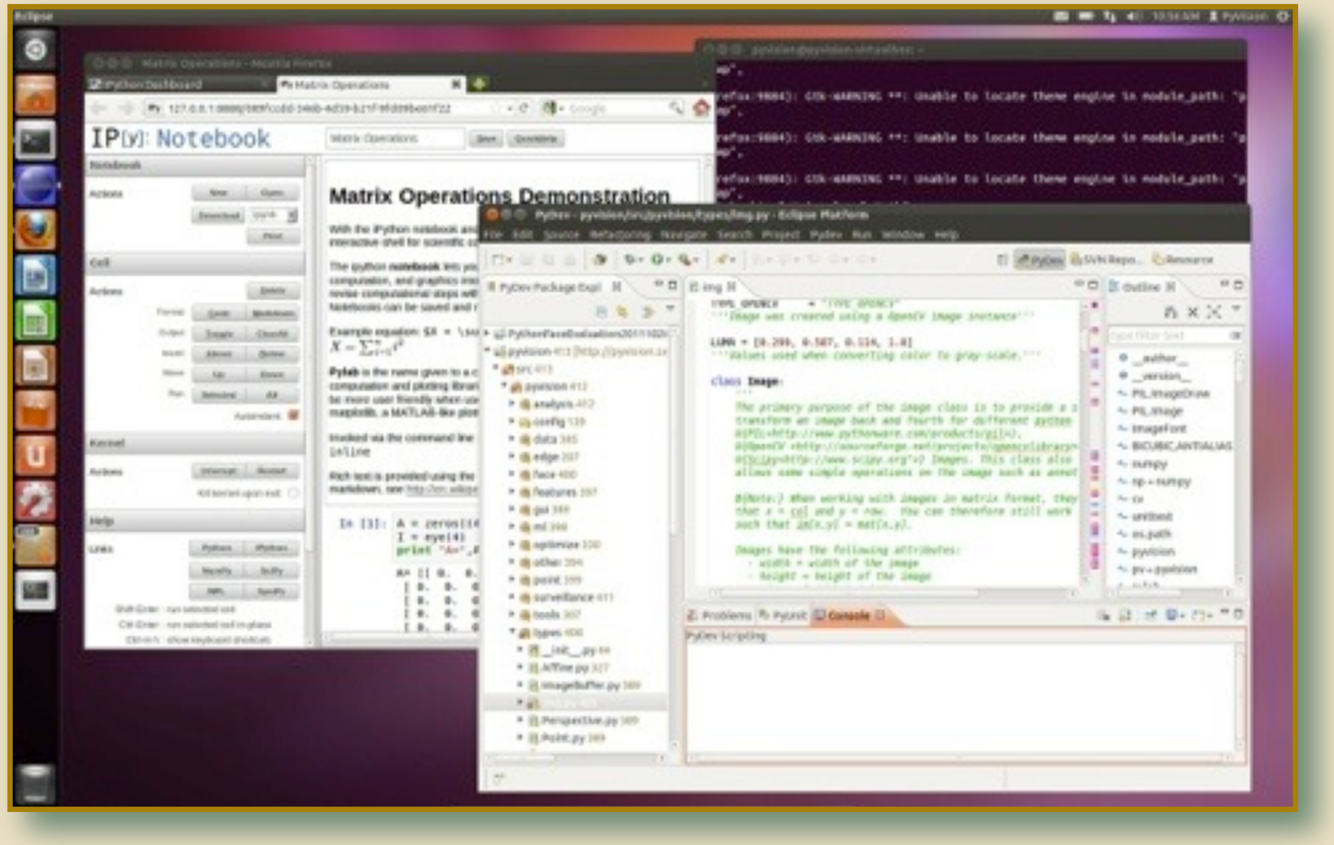
```
# config file for bash

#for opencv+python
export PYTHONPATH=/usr/local/OpenCV-2011-11-09/lib/python2.7/site-packages

#for scikits.learn python library
export PYTHONPATH=${PYTHONPATH}:/usr/local/scikits.learn/lib64/python2.7/site-packages

#for pyvision
export PYTHONPATH=${PYTHONPATH}:~vision/pyvision/src/
```

Installation Flash Drive

- Virtual Box for Windows, Mac OS, and Linux
- PyVision Virtual Appliance
- Tutorial Slides

Virtual Box Appliance

https://www.virtualbox.org/wiki/Downloads



Eclipse
Matrix Operations - Mozilla Firefox
IP[y]: Notebook
PyDev - pyvision/src/pyvision/types/img.py - Eclipse Platform
File Edit Source Refactoring Navigate Search Project Pydev Run Window Help
PyDev Package Expl
img
Outline
Problems
PyUnit
Console

Matrix Operations Demonstration

```
refox:9884): Gtk-WARNING **: Unable to locate theme engine in module_path: "p
refox:9884): Gtk-WARNING **: Unable to locate theme engine in module_path: "p
refox:9884): Gtk-WARNING **: Unable to locate theme engine in module_path: "p
```

```
class Image:
```

- Ubuntu Linux
- python, scipy, numpy, pil,
- Eclipse with pydev&subclipse
- FireFox

- PyVision
- CSU Face Baseline
- iPython + Html Notebook
- R (Statistics)

Things to know...

- **Username: pyvision**
- **Password: pyvision**
- Ubuntu 11.10
- 32bit Single Processor
- 1 GB Ram
- 16 GB Hard Drive

Installation Requirements

python

SciPy.org

OpenCV

PyVision

- Python (2.7 recommended)
- Python Imaging Library (PIL)
- NumPy and SciPy
- OpenCV (ver 2.2 or 2.3)

- PyVision
- Optional:
  - IPython
  - Matplotlib

Hands On Installation

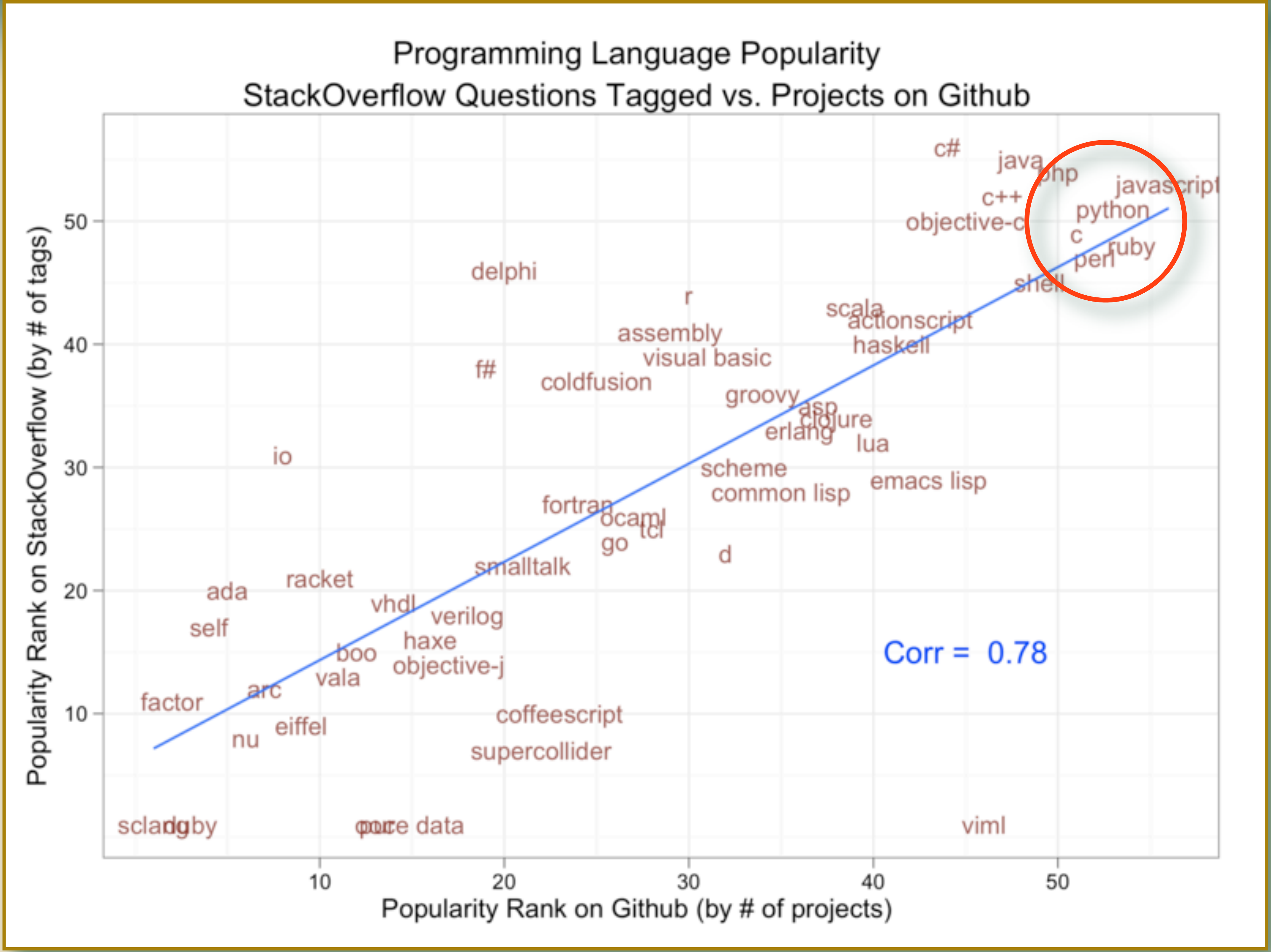
Quick Introduction Python

Benefits of Python

- Similar syntax/functionality to MATLAB through Scipy
- Supports modern programming language constructs
- Interfaces to OpenCV, LibSVM, and many other open source libraries
- Quick and easy prototyping
- Free

Popularity

- Python is one of the most popular and fastest-growing dynamic languages
- Graphic is from: http://blog.revolutionanalytics.com/2010/12/programming-languages-ranked-by-popularity.html



Indentation / Control

- Indentation determines block structure.
- Use colon “:” instead of braces
- Set your text editor to use spaces instead of tabs.
- Reference Counting / GC

```python
def foo(a,b):
    ''' A function that adds two numbers '''
    return a + b

# Count from 0 to 9
i = 0
while i < 10:

    print "Number:",i,

    if i % 2 == 0:
        print "even"
    else:
        print "odd"

    i = foo(i, 1)
```

File: TutorialControl.py

Results

```
Number: 0 even
Number: 1 odd
Number: 2 even
Number: 3 odd
Number: 4 even
Number: 5 odd
Number: 6 even
Number: 7 odd
Number: 8 even
Number: 9 odd
```

The “main” script

- Basic script structure.
- Executes from top to bottom.
- “__main__” if statement
- Arguments: sys.argv
- Functions “def”
- Classes “class”
    - “self” parameter

```python
def add(a,b):
    return a + b

class Add:
    def __init__(self,a,b):
        self.val = a+b

    def getValue(self):
        return self.val

if __name__ == '__main__':
    # execute this code if this
    # is the main script
    print "Hello World!!!"
    print "2 + 3 =",add(2,3)
    my_obj = Add(2,3)
    print "Add(2,3)=", my_obj.getValue()
```

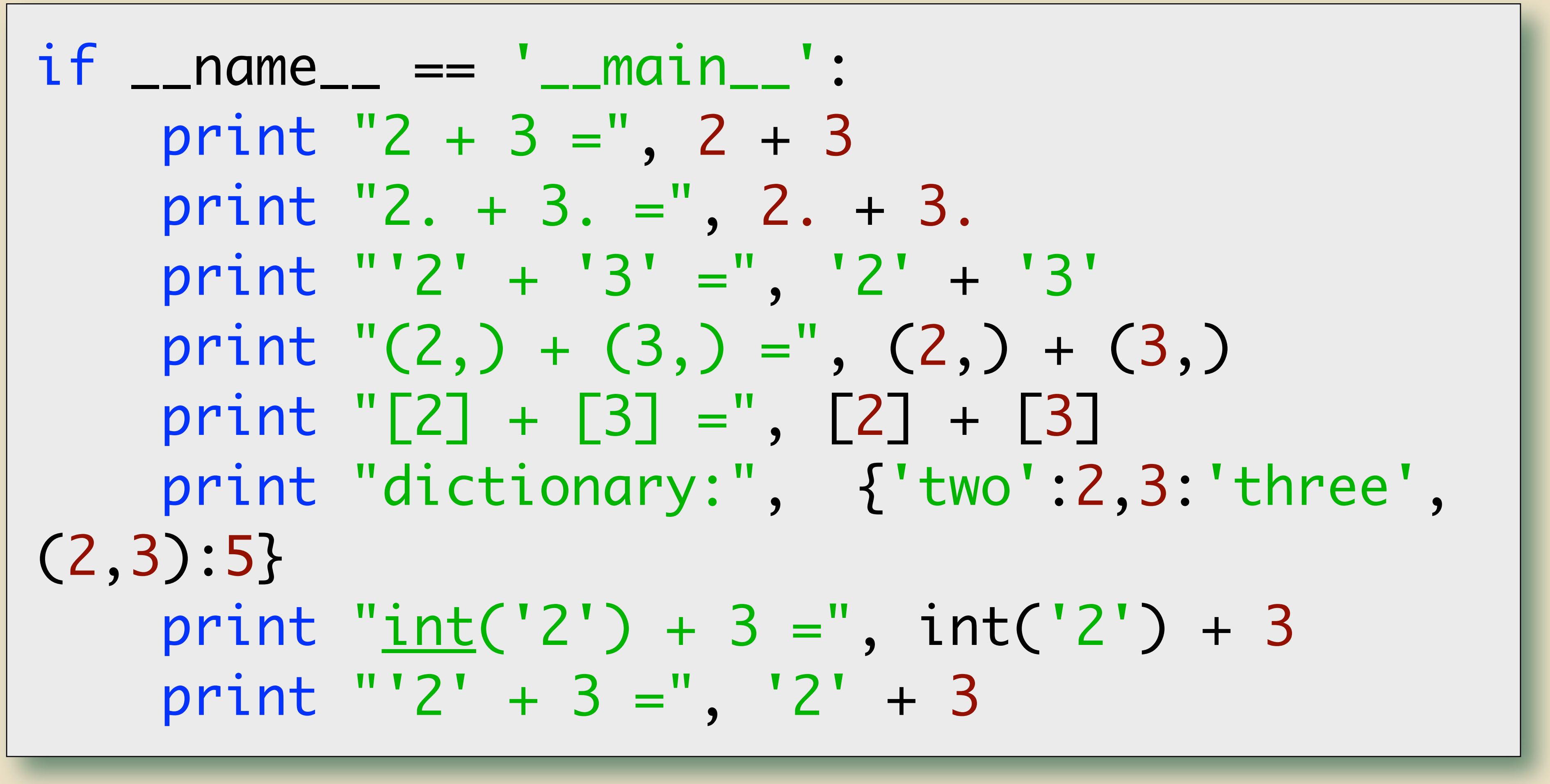
File: TutorialMain.py

Results

```
Hello World!!!
2 + 3 = 5
Add(2,3)= 5
```

Data and Types

- object - myobj = MyClass()
- int - 1
- float - 1.0
- str / buffer - “Hello World”
- list - [1,2.0,”three”,myobj]
- dict - {“key”:”val”, 203:myobj}



```
if __name__ == '__main__':
    print "2 + 3 =", 2 + 3
    print "2. + 3. =", 2. + 3.
    print "'2' + '3' =", '2' + '3'
    print "(2,) + (3,) =", (2,) + (3,)
    print "[2] + [3] =", [2] + [3]
    print "dictionary:",  {'two':2,3:'three', (2,3):5}
    print "int('2') + 3 =", int('2') + 3
    print "'2' + 3 =", '2' + 3
```

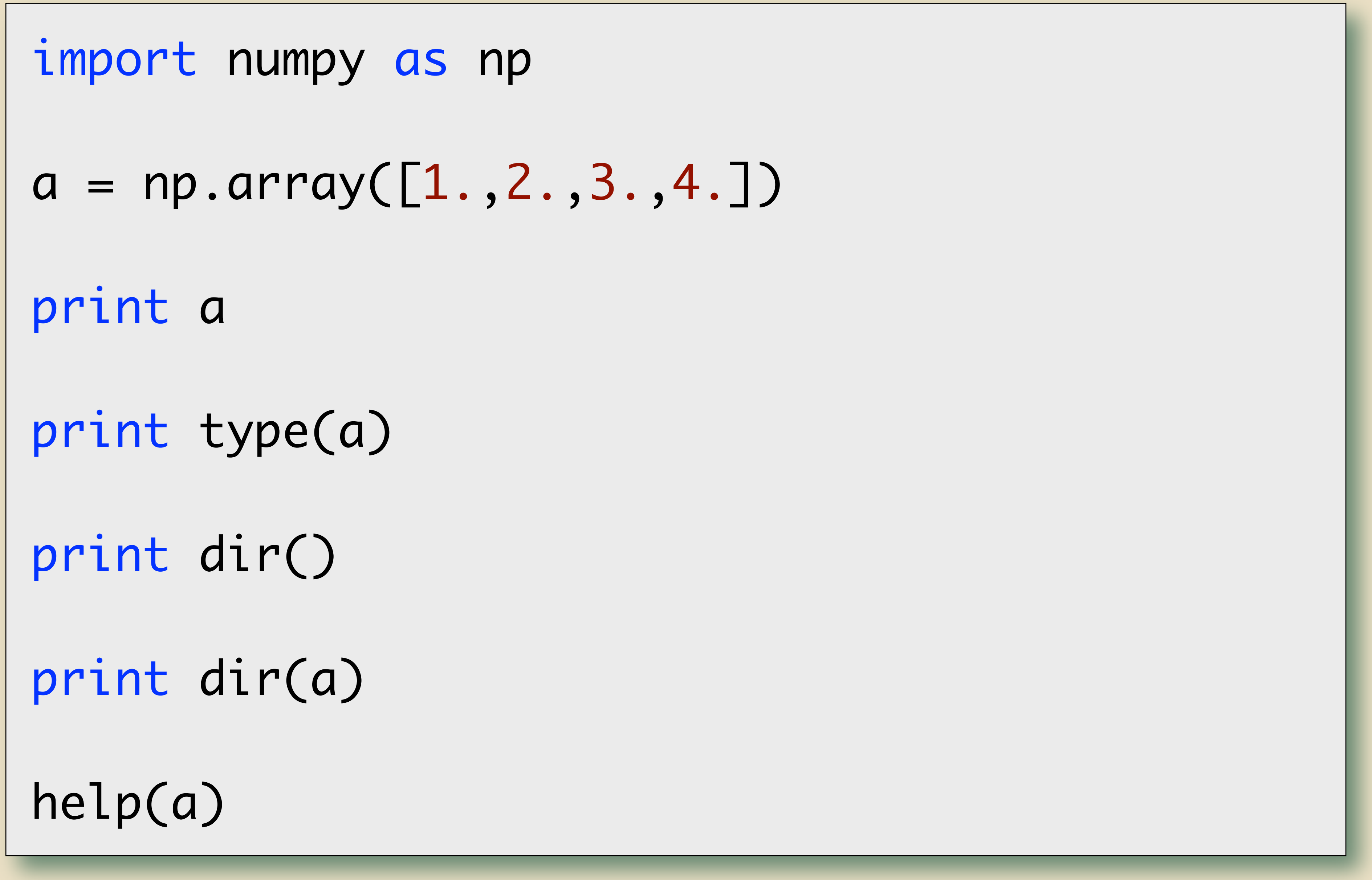
File: TutorialTypes.py

Results

```
2 + 3 = 5
2. + 3. = 5.0
'2' + '3' = 23
(2,) + (3,) = (2, 3)
[2] + [3] = [2, 3]
dictionary: {3: 'three', (2, 3): 5, 'two': 2}
int('2') + 3 = 5
'2' + 3 =
Traceback (most recent call last):
  File "/Users/bolme/Documents/workspace/FaceRec/src/experiments/tutorials/TutorialTypes.py", line 9, in <module>
    print "'2' + 3 =", '2' + 3
TypeError: cannot concatenate 'str' and 'int' objects
```

Introspection and Help

- print - print object info
- type(object) - get the object type.
- dir() - list variables, functions, etc in current scope.
- dir(object) - list members/ methods of object.
- help(object/module) - get help on an object, function, or module.



```
import numpy as np

a = np.array([1.,2.,3.,4.])

print a

print type(a)

print dir()

print dir(a)

help(a)
```

File: TutorialHelp.py

Results

```
[ 1.  2.  3.  4.]
<type 'numpy.ndarray'>
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'a', 'np']
['T', '__abs__', '__add__', '__and__', ..., 'all', 'any', 'argmax', 'argmin',...]
Help on ndarray object:

class ndarray(__builtin__.object)
 |  ndarray(shape, dtype=float, buffer=None, offset=0,
 |          strides=None, order=None)
 |
 |  An array object represents a multidimensional, homogeneous array
 |  of fixed-size items.  An associated data-type object describes the
 |  format of each element in the array (its byte-order, how many bytes it
 |  occupies in memory, whether it is an integer, a floating point number,
 |  or something else, etc.)
```

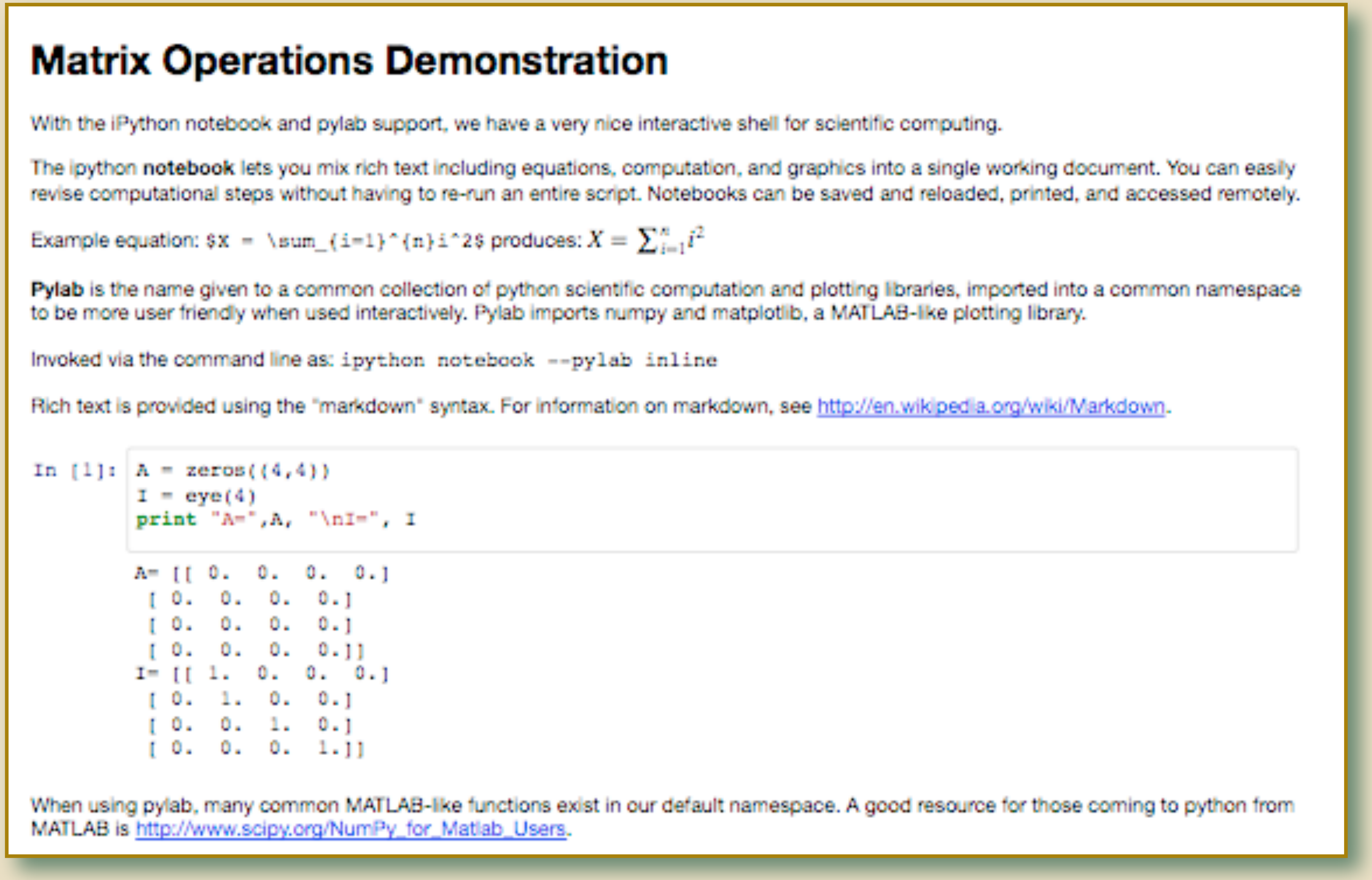
Matrix Manipulation

- Numpy is the numeric python library
- Scipy has additional scientific programming packages, is superset of Numpy
- “ndarray” type, optional “matrix” type
- Scipy linalg package
- http://www.scipy.org/NumPy_for_Matlab_Users

iPython and PyLab

- iPython is an enhanced interactive python interpreter
- iPython Notebook
- PyLab is built on iPython and aims to be an interactive workspace for scientific programming
- Matplotlib is a MATLAB-syntax plotting facility for python

Interactive Demonstration



Matrix Operations Demonstration

With the iPython notebook and pylab support, we have a very nice interactive shell for scientific computing.

The ipython **notebook** lets you mix rich text including equations, computation, and graphics into a single working document. You can easily revise computational steps without having to re-run an entire script. Notebooks can be saved and reloaded, printed, and accessed remotely.

Example equation: `$X = \sum_{i=1}^{n}i^2$` produces: $X = \sum_{i=1}^{n} i^2$

Pylab is the name given to a common collection of python scientific computation and plotting libraries, imported into a common namespace to be more user friendly when used interactively. Pylab imports numpy and matplotlib, a MATLAB-like plotting library.

Invoked via the command line as: `ipython notebook --pylab inline`

Rich text is provided using the "markdown" syntax. For information on markdown, see http://en.wikipedia.org/wiki/Markdown.

```
In [1]: A = zeros((4,4))
        I = eye(4)
        print "A=",A, "\nI=", I
```

```
A= [[ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]]
I= [[ 1.  0.  0.  0.]
 [ 0.  1.  0.  0.]
 [ 0.  0.  1.  0.]
 [ 0.  0.  0.  1.]]
```

When using pylab, many common MATLAB-like functions exist in our default namespace. A good resource for those coming to python from MATLAB is http://www.scipy.org/NumPy_for_Matlab_Users.

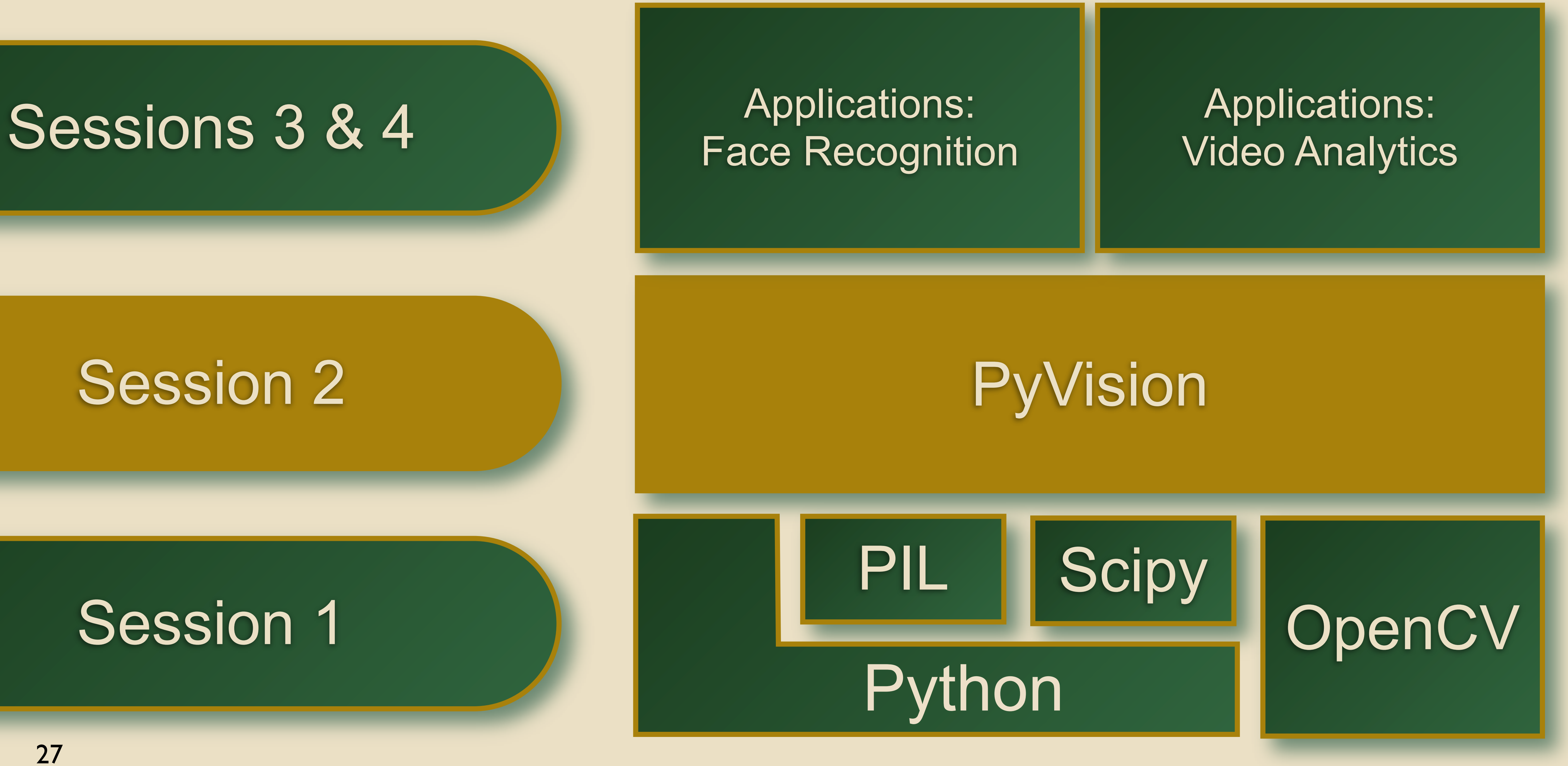
Standard Library Highlights

- Operating System - os, os.path
- Shell - shutil
- Binary Data - struct
- Math - math, cmath, random
- Object Serialization - pickle
- XML - ElementTree,dom,sax
- DB/Tables - csv, bsddb,sqlite3
- Compression: zlib,bz2,zipfile,tarfile
- Security: md5,sha,ssl
- Time: time, calendar, ...
- Multiple CPU: multiprocessing
- Networking: socket
- Web: urllib, email, htmllib, ftplib
- Other: unittest, string, copy, sys

Third Party Libraries

- Interfaces to C, Java, Matlab, R ...
- Web services, Databases, Networking, XML ...
- Scientific Computing, Machine Learning, cuda ...
- GUI: wxPython, Qt, Gnome, Cocoa, Windows...
- Bindings to most popular open source resources.

System Architecture



Sessions 3 & 4

Applications: Face Recognition

Applications: Video Analytics

Session 2

PyVision

Session 1

PIL

Scipy

Python

OpenCV

Session 2 Goals

- Introduction to PyVision
- Basic datatypes
- Tools for understanding algorithms
- Using NumPy / SciPy
- Using OpenCV

Face Detection in OpenCV

```python
#!/usr/bin/python
""""
This program is demonstration for face and object
detection using haar-like features.
The program finds faces in a camera image or video
stream and displays a red box around them.

Original C implementation by:  ?
Python implementation by: Roman Stanchak, James
Bowman
""""
import sys
import cv
from optparse import OptionParser

# Parameters for haar detection
# From the API:
# The default parameters (scale_factor=2,
min_neighbors=3, flags=0) are tuned
# for accurate yet slow object detection. For a
faster operation on real video
# images the settings are:
# scale_factor=1.2, min_neighbors=2,
flags=CV_HAAR_DO_CANNY_PRUNING,
# min_size=<minimum possible face size

min_size = (20, 20)
```

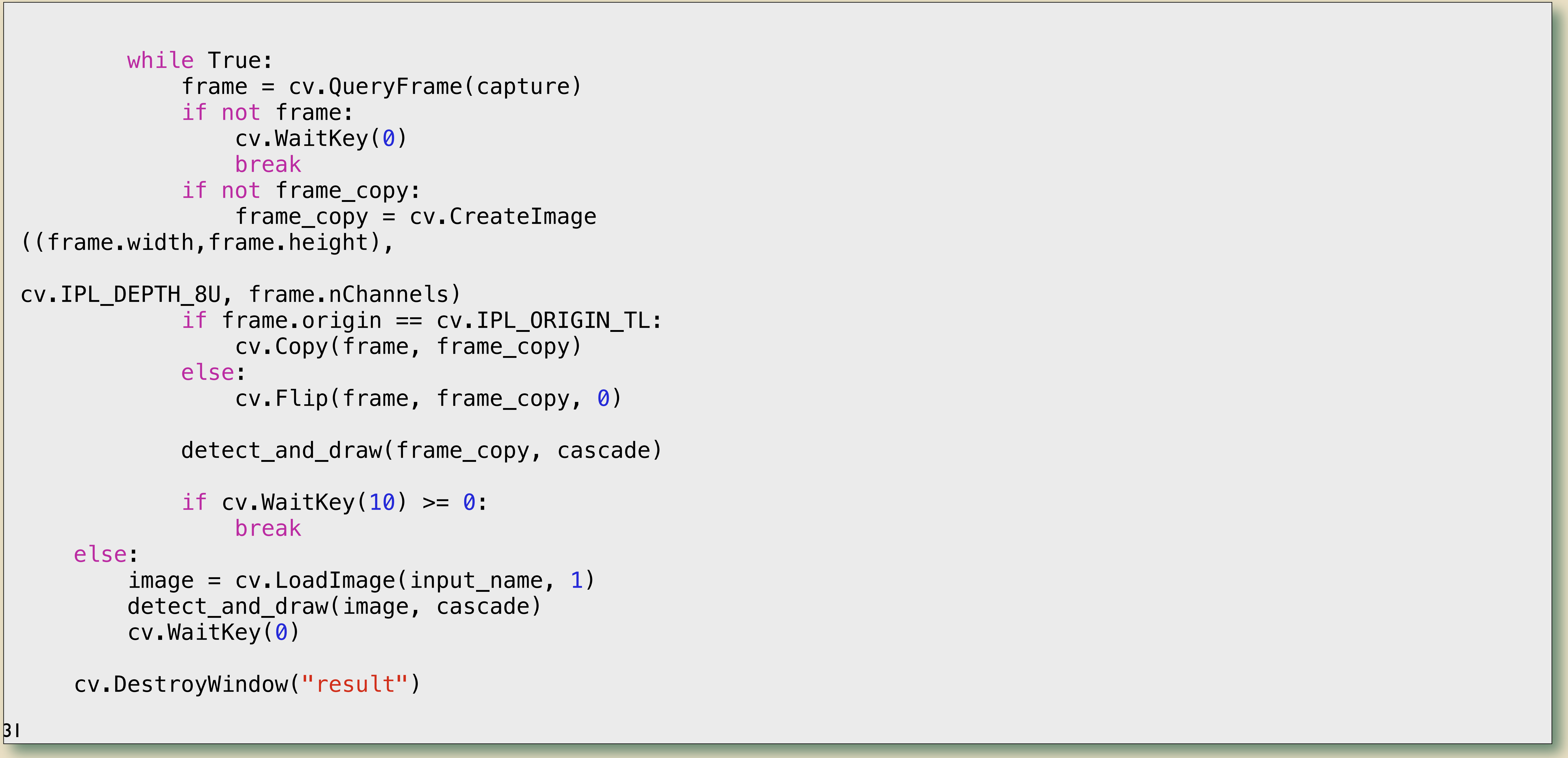
```python
image_scale = 2
haar_scale = 1.2
min_neighbors = 2
haar_flags = 0

def detect_and_draw(img, cascade):
    # allocate temporary images
    gray = cv.CreateImage((img.width,img.height), 8,
1)
    small_img = cv.CreateImage((cv.Round(img.width /
image_scale),
                 cv.Round (img.height /
image_scale)), 8, 1)

    # convert color input image to grayscale
    cv.CvtColor(img, gray, cv.CV_BGR2GRAY)

    # scale input image for faster processing
    cv.Resize(gray, small_img, cv.CV_INTER_LINEAR)

    cv.EqualizeHist(small_img, small_img)

    if(cascade):
        t = cv.GetTickCount()
        faces = cv.HaarDetectObjects(small_img,
cascade, cv.CreateMemStorage(0),
                                     haar_scale,
```

Face Detection in OpenCV

```
 cascade, cv.CreateMemStorage(0),
                                        haar_scale,
min_neighbors, haar_flags, min_size)
        t = cv.GetTickCount() - t
        print "detection time = %gms" % (t/
(cv.GetTickFrequency()*1000.))
        if faces:
            for ((x, y, w, h), n) in faces:
                # the input to cv.HaarDetectObjects
was resized, so scale the
                # bounding box of each face and
convert it to two CvPoints
                pt1 = (int(x * image_scale), int(y *
image_scale))
                pt2 = (int((x + w) * image_scale),
int((y + h) * image_scale))
                cv.Rectangle(img, pt1, pt2, cv.RGB
(255, 0, 0), 3, 8, 0)

    cv.ShowImage("result", img)

if __name__ == '__main__':

    print "hello world"

    parser = OptionParser(usage = "usage: %prog
[options] [filename|camera_index]")
```

```
    parser.add_option("-c", "--cascade",
action="store", dest="cascade", type="str",
help="Haar cascade file, default %default", default =
"../data/haarcascades/
haarcascade_frontalface_alt.xml")
    (options, args) = parser.parse_args()

    print "load cascade"
    cascade = cv.Load(options.cascade)

    print "Print help"
    if len(args) != 1:
        parser.print_help()
        sys.exit(1)

    input_name = args[0]
    print input_name
    if input_name.isdigit():
        capture = cv.CreateCameraCapture(int
(input_name))
    else:
        capture = None

    cv.NamedWindow("result", 1)

    if capture:
        frame_copy = None
```

Face Detection in OpenCV



```
        while True:
            frame = cv.QueryFrame(capture)
            if not frame:
                cv.WaitKey(0)
                break
            if not frame_copy:
                frame_copy = cv.CreateImage
((frame.width,frame.height),

cv.IPL_DEPTH_8U, frame.nChannels)
            if frame.origin == cv.IPL_ORIGIN_TL:
                cv.Copy(frame, frame_copy)
            else:
                cv.Flip(frame, frame_copy, 0)

            detect_and_draw(frame_copy, cascade)

            if cv.WaitKey(10) >= 0:
                break
    else:
        image = cv.LoadImage(input_name, 1)
        detect_and_draw(image, cascade)
        cv.WaitKey(0)

    cv.DestroyWindow("result")
```

Face Detection Demo

- PyVision Philosophy:
    - PyVision is designed for researchers.
    - Algorithms should have simple interfaces and intelligent defaults.
    - Support standard datatypes.
    - Using OpenCV, SciPy, and PIL together should be easy.
    - Results should be easy to understand and debug.

```
import pyvision as pv
import pyvision.face.CascadeDetector as cd

if __name__ == '__main__':

    detector = cd.CascadeDetector()

    cam = pv.Webcam()
    while True:
        frame = cam.query()
        rects = detector(frame)
        for rect in rects:
            frame.annotateRect(rect)
        frame.show()
```

File: TutorialFaceDetect.py

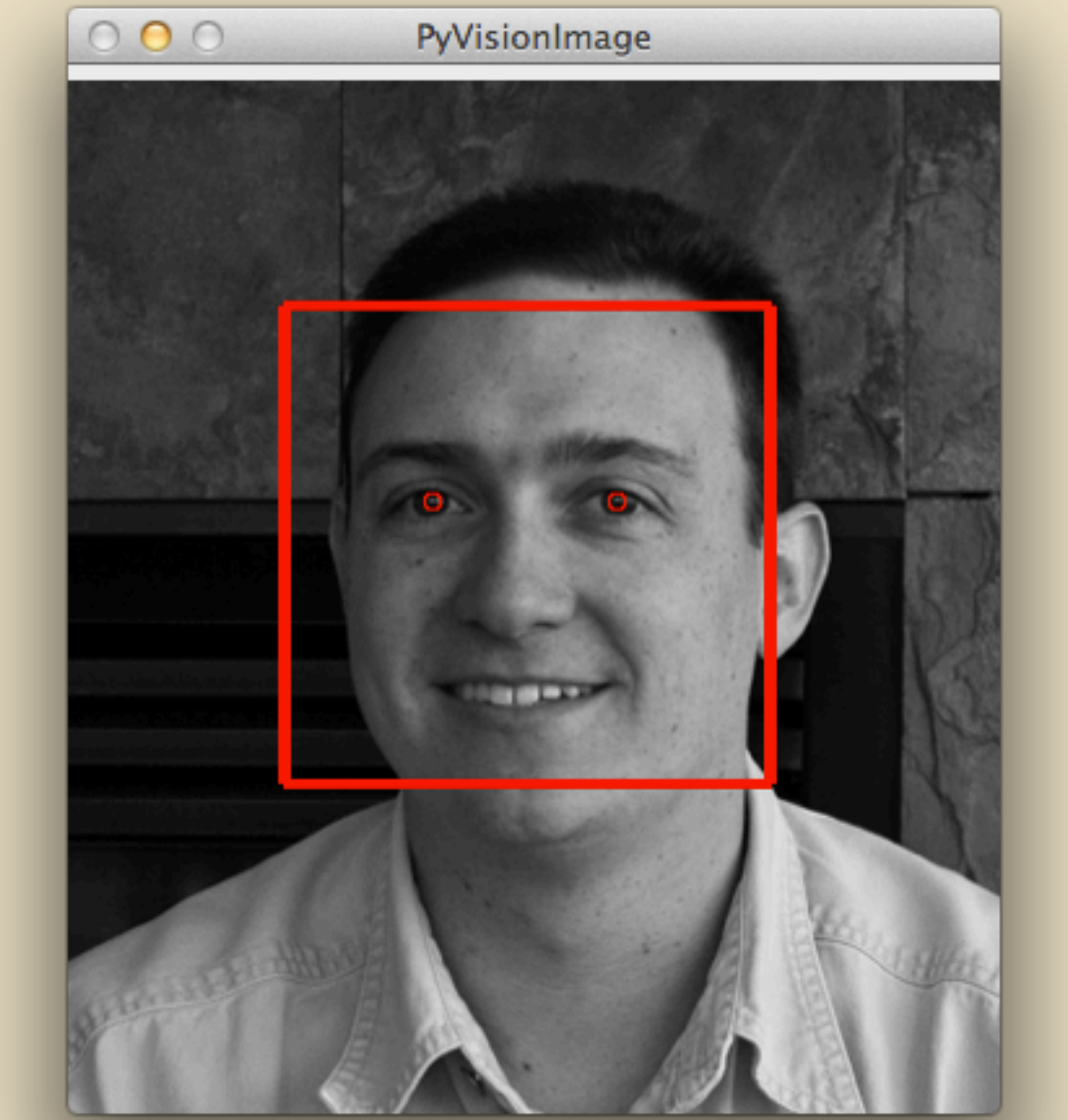
Eye Detection

- Read image from disk: pv.Image().
- bw image to make annotations stand out.
- Thicker detection rectangle using Polygon and width=4.
- Also detect eyes.

```python
import pyvision as pv
import pyvision.face.CascadeDetector as cd
import pyvision.face.FilterEyeLocator as ed

face_detect = cd.CascadeDetector()
eye_detect = ed.FilterEyeLocator()

im = pv.Image("face.png",bw_annotate=True)

faces = face_detect(im)
eyes = eye_detect(im,faces)

for face,eye1,eye2 in eyes:
    im.annotatePolygon(face.asPolygon(),
        width=4)
    im.annotatePoints([eye1,eye2])

im.show(delay=0)
```

File: TutorialEyeDetect.py

Result



What PyVision Provides

- Read and convert common data types: video, image, matrix, rects, points, ...
- Common computer vision functions: preprocessing, transforms, detectors, interest points, motion detection, surf, lda, pca, svm, ...
- Analysis and Visualization: Annotation, Plots, Logs, Montage...
- Integration with OpenCV

PyVision Directory Structure

Points, Rects, Images, Videos, ...

- pv.Point - A point (x,y,[z,[w]])
- pv.Rect - A rect (x,y,w,h)
- pv.Image - JPG, PNG, TIF, ...
- pv.ImageBuffer - Set of images
- pv.Video - AVI, MOV, M4V, IP network cameras
- pv.Webcam - USB Webcams
- and other classes that implement a video interface...

PyVision Image Class

- Easily convert to common formats
- Maintain annotations separate from source image
- Convenience methods for loading, saving, displaying, resizing, cropping, and other common operations

PIL, SciPy, and OpenCV

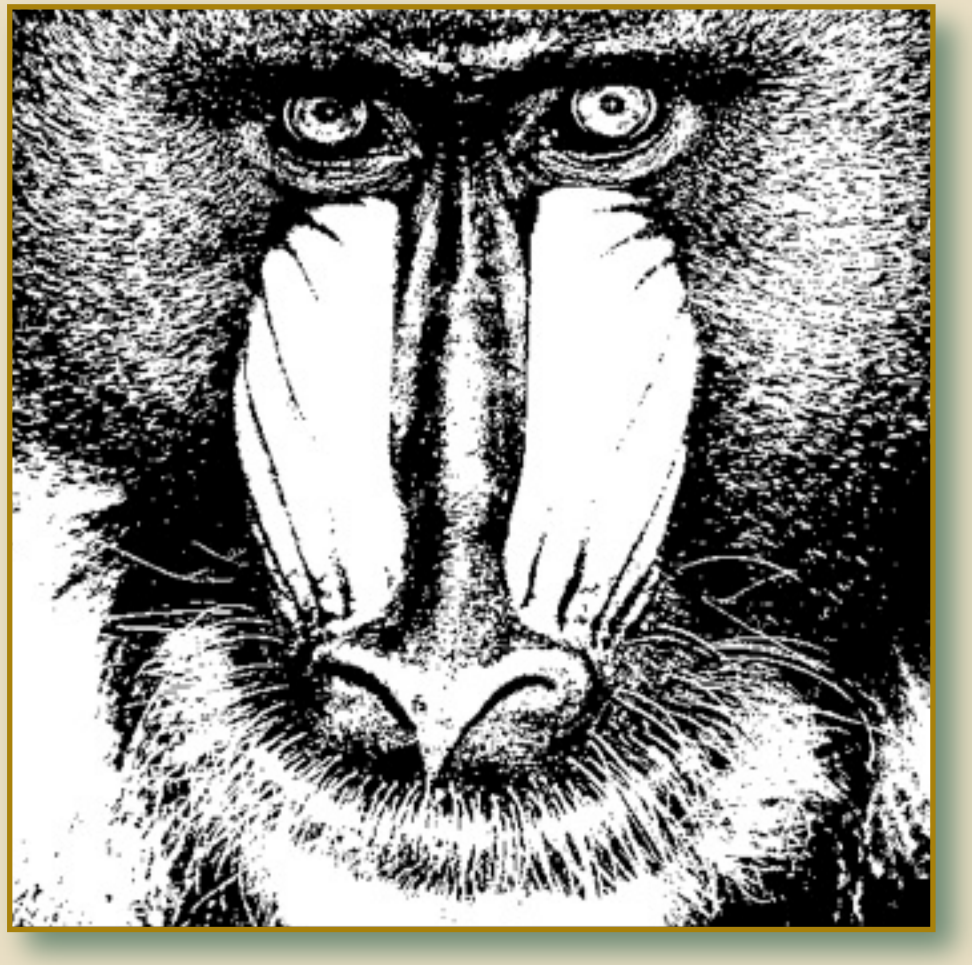
- Use im.as<Format> to get PIL, Scipy, and OpenCV images.
- Perform operations using preferred library.
- Convert back using pv.Image()
- Note: Scipy format matrices are transposed so that mat[x,y] correspond to the x and y image axis.

File: TutorialThresh.py

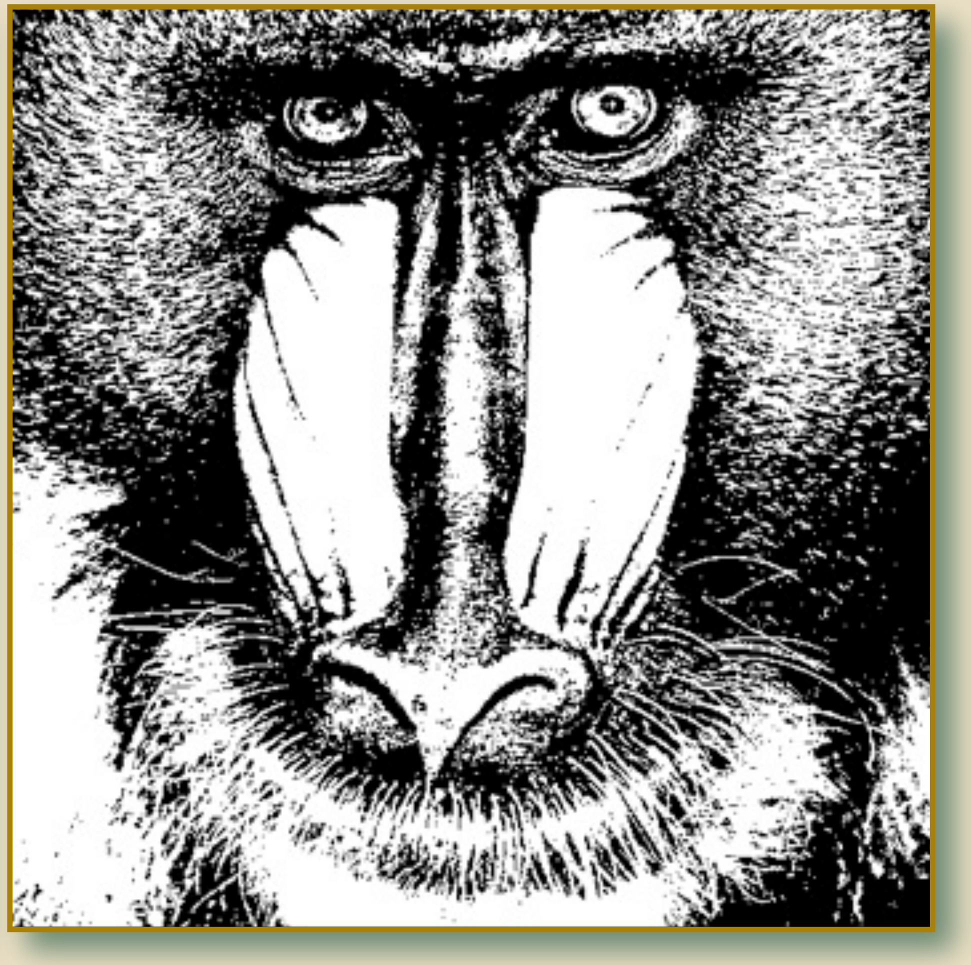
```
import pyvision as pv
import PIL, cv
ilog = pv.ImageLog()

im = pv.Image("baboon.jpg")

pil = im.asPIL()
gray = pil.convert('L')
thresh = PIL.Image.eval(gray, lambda x: 255*
(x>127.5) )
ilog(pv.Image(thresh),"PILThresh")

mat = im.asMatrix2D()
thresh = mat > 127.5
ilog(pv.Image(1.0*thresh),"ScipyThresh")

cvim = im.asOpenCVBW()
dest=cv.CreateImage(im.size,cv.IPL_DEPTH_8U,1)
cv.CmpS(cvim,127.5,dest,cv.CV_CMP_GT)
ilog(pv.Image(dest),"OpenCVThresh")

ilog.show()
```

Results

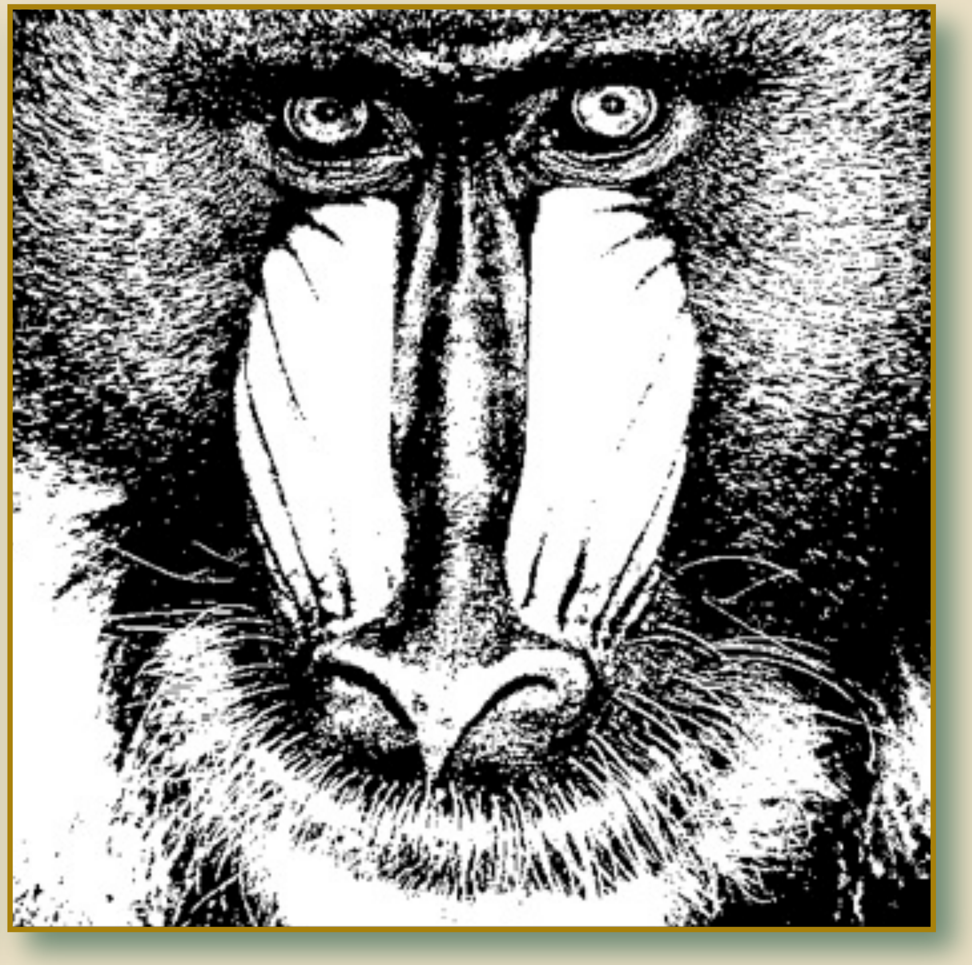
PIL



SciPy



OpenCV



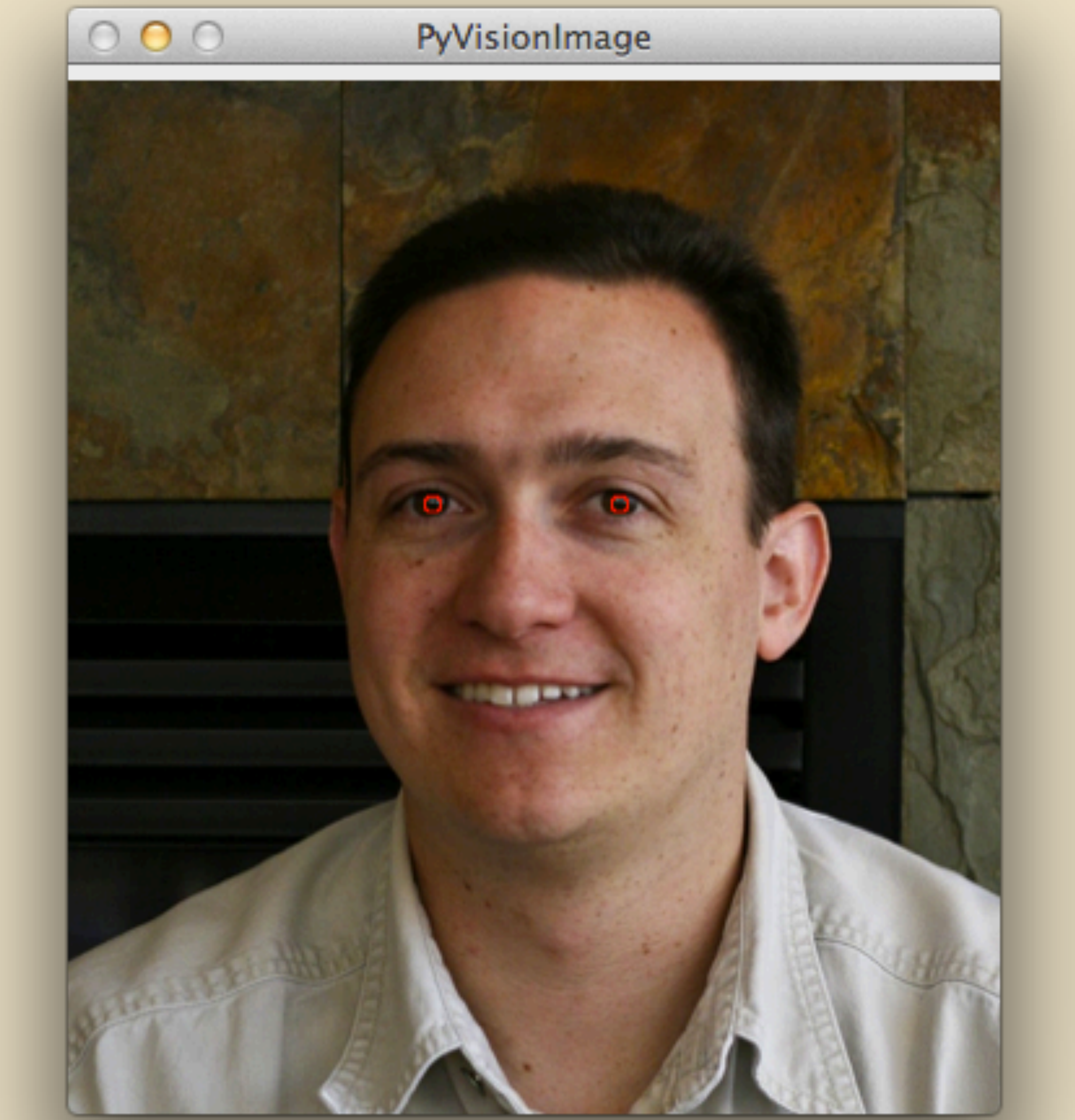
Affine Transformations

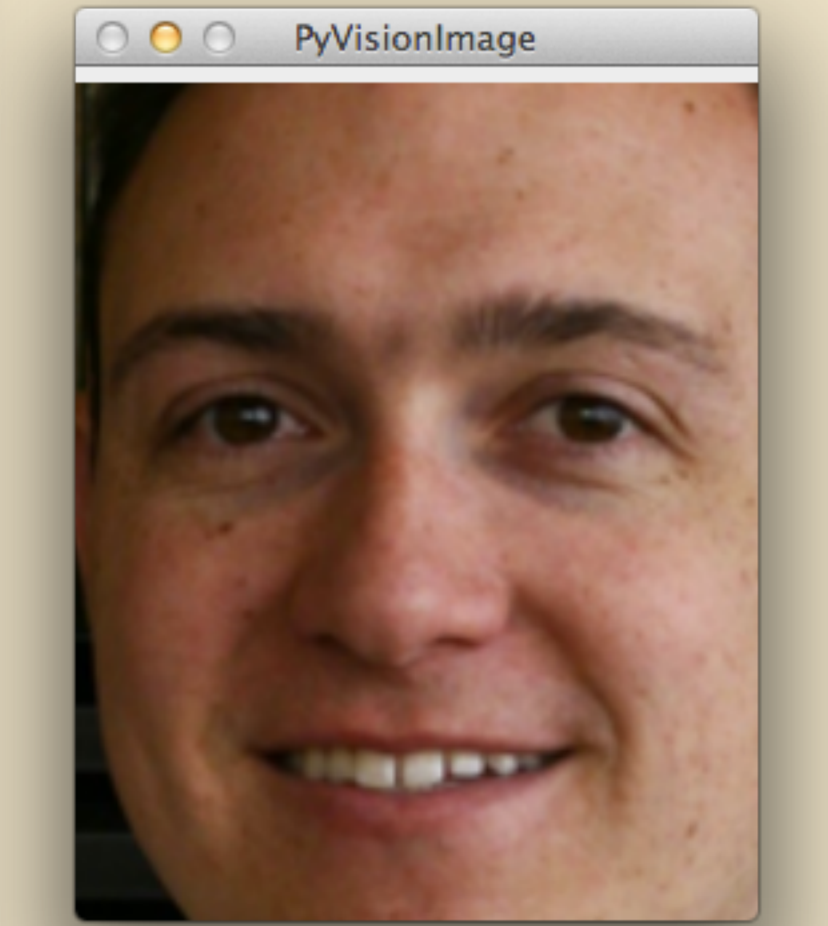
- affine = pv.AffineTransform (matrix,size)
- new_im = affine(old_im)
- new_pts = affine(old_pts)
- both = affine1*affine2
- affine.invert(pts)
- Helper Functions: pv.AffineFromPoints, pv.AffineFromRect, pv.Affine [Scale,Rotate,Trans...]
- pv.PerspectiveTransform

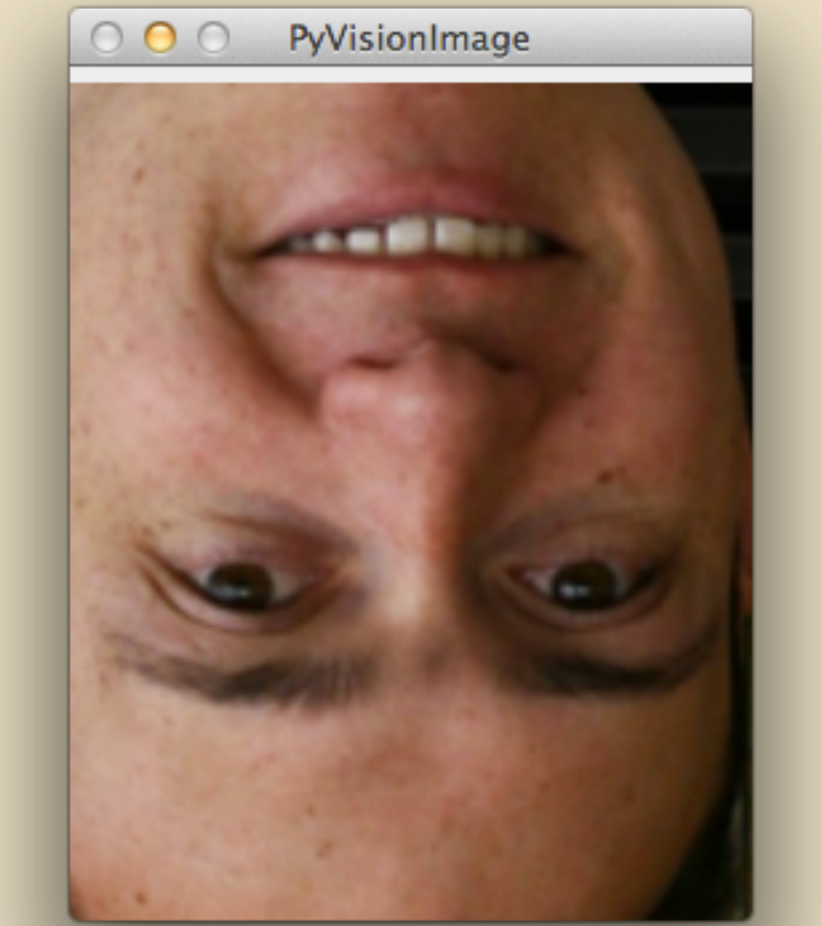
```
import pyvision as pv

if __name__ == '__main__':
    im = pv.Image("face.png")
    eye1,eye2 = pv.Point(140,165),...
    out1,out2 = pv.Point(64,128),...

    im.annotatePoints([eye1,eye2])
    im.show(delay=0)

    affine = pv.AffineFromPoints(eye1,eye2,
                   out1,out2,(256,320))
    tile = affine(im)
    tile.show(delay=0)

    affine = pv.AffineRotate(3.1415,(256,320),
            center=pv.Point(128,160))*affine;
    tile = affine(im)
    tile.show(delay=0)
```

File: TutorialAffine.py

Results







Annotation and Logging Results

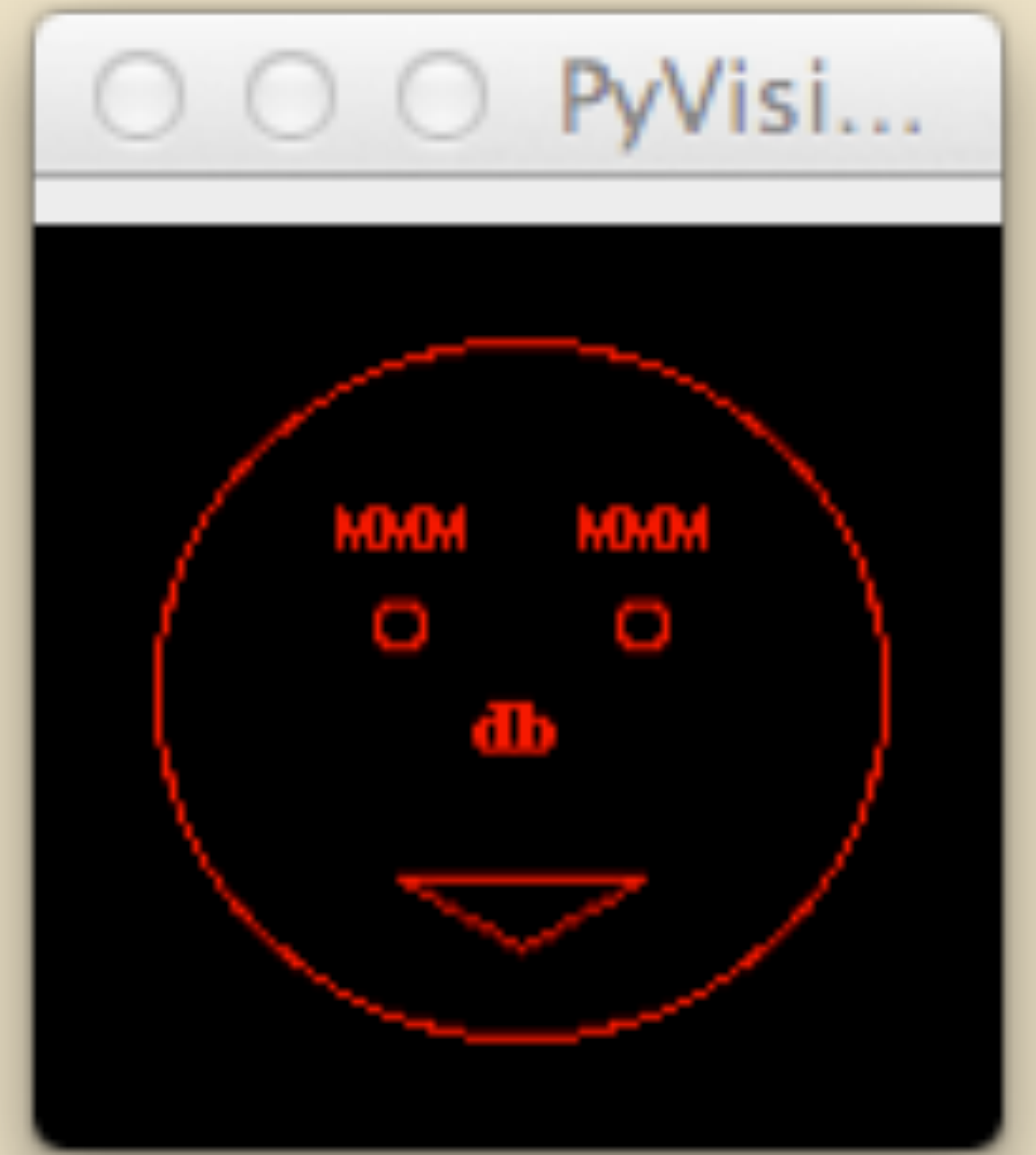
Image Annotation

- Implemented in PIL.
- Annotate images with points, rects, circles, ellipses, polygons, lines, and labels.
- A separate copy of the image is created within the object just for annotations.
- Supports colors and other drawing options: color = “red” or “#FF0000”

```
import pyvision as pv
import scipy as sp
if __name__ == '__main__':
    im = pv.Image(sp.zeros((128,128)))

    pts = [pv.Point(48,55),pv.Point(80,55)]
    im.annotatePoints(pts)

    elipse = pv.CenteredRect(64,64,96,96)
    im.annotateEllipse(elipse)

    im.annotatePolygon([pv.Point(48,90),
        pv.Point(80,90),pv.Point(64,100)])

    im.annotateLabel(pv.Point(40,36),"MMM")
    im.annotateLabel(pv.Point(72,36),"MMM")
    im.annotateLabel(pv.Point(58,64),"db")

    im.show(delay=0)
```

File: --Example.py--

Result



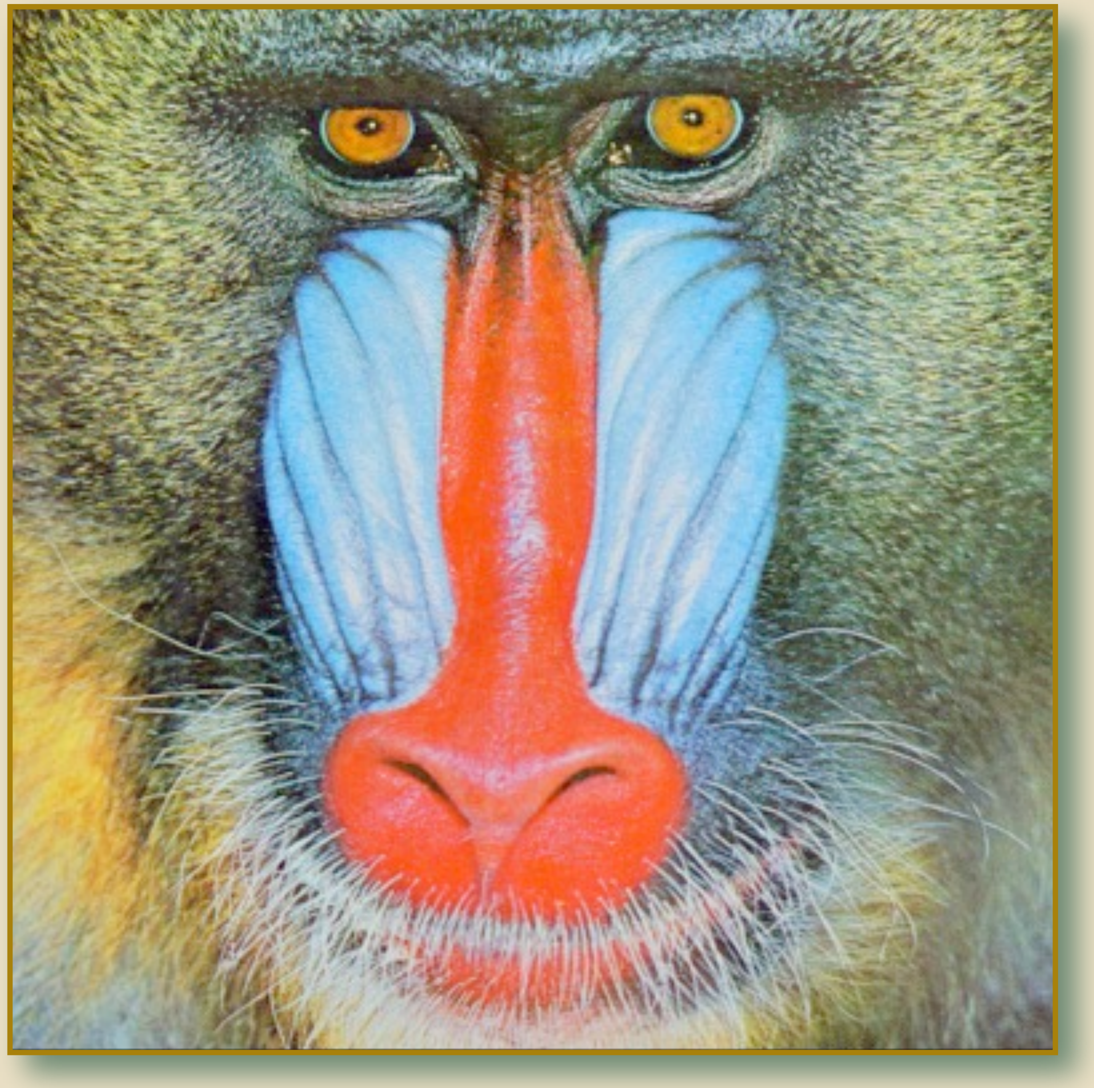
Logs, Tables, Timers

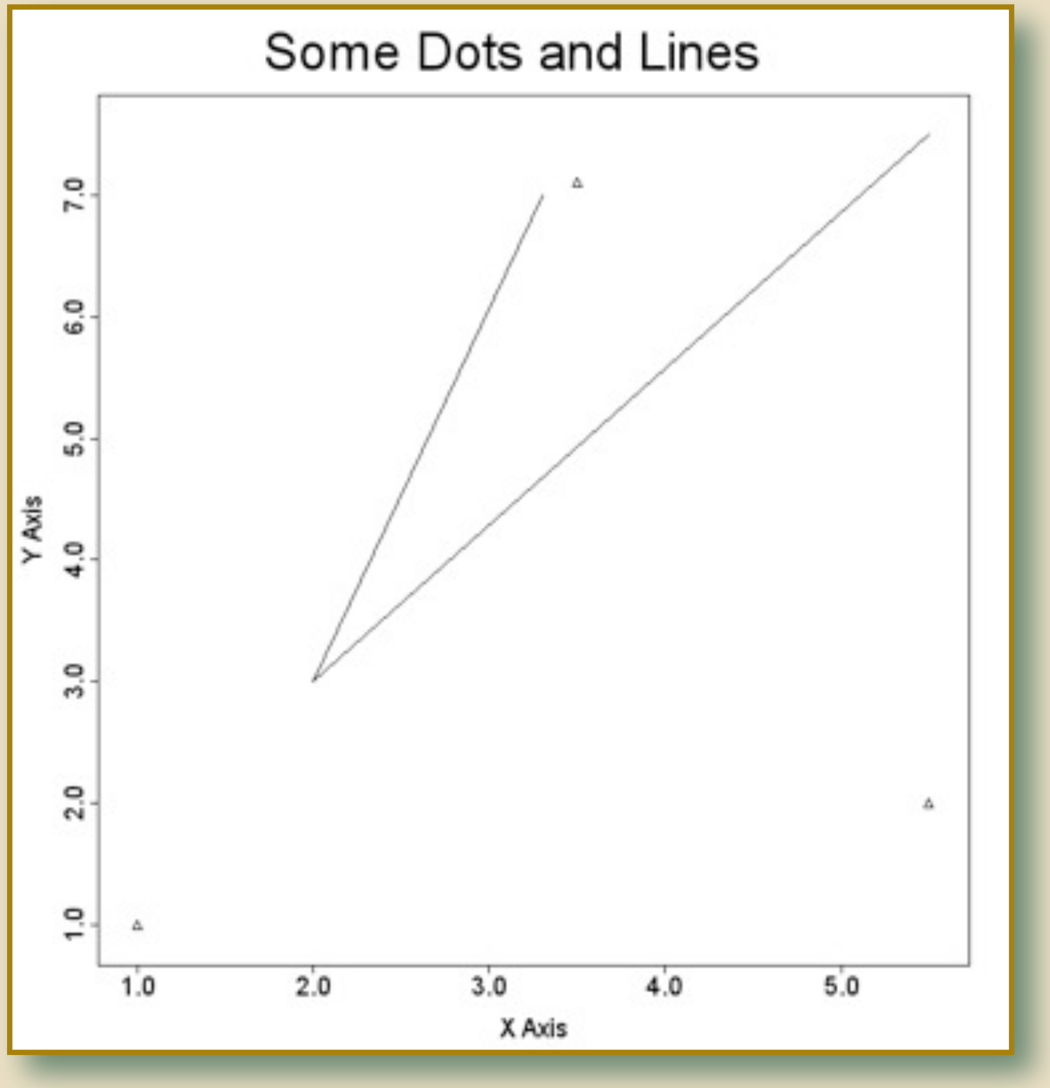
- pv.ImageLog - A collection of images, tables, plots, etc that is saved to disk for later analysis.
- pv.Table - Tabular data that support pretty printing and csv.
- pv.Timer - Time functions and processes.
- pv.Plot - line and scatter plots.

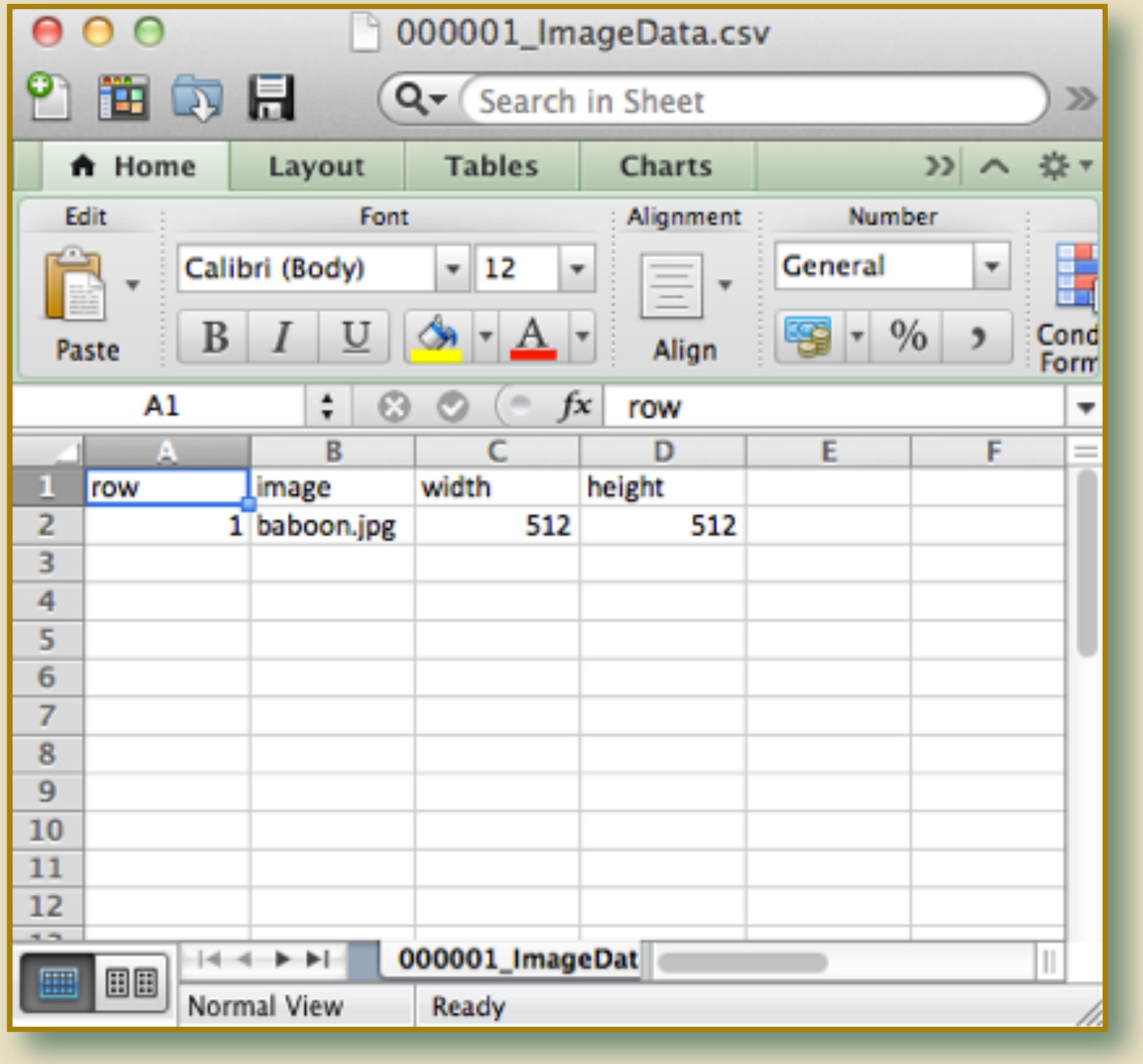
```
import pyvision as pv

ilog = pv.ImageLog()
im = pv.Image("baboon.jpg")
ilog(im,"Baboon")

table = pv.Table()
table[1,"image"] = im.filename
table[1,"width"] = im.size[0]
table[1,"height"] = im.size[1]
ilog(table,"ImageData")
print table

plot = pv.Plot(title="Some Dots and Lines");
plot.points([[3.5,7.1],[1,1],[5.5,2]],shape=2)
plot.lines([[5.5,7.5],[2,3],[3.3,7]])
ilog(plot,"MyPlot")

ilog.show()
```

File: TutorialLogsPlots.py

Results



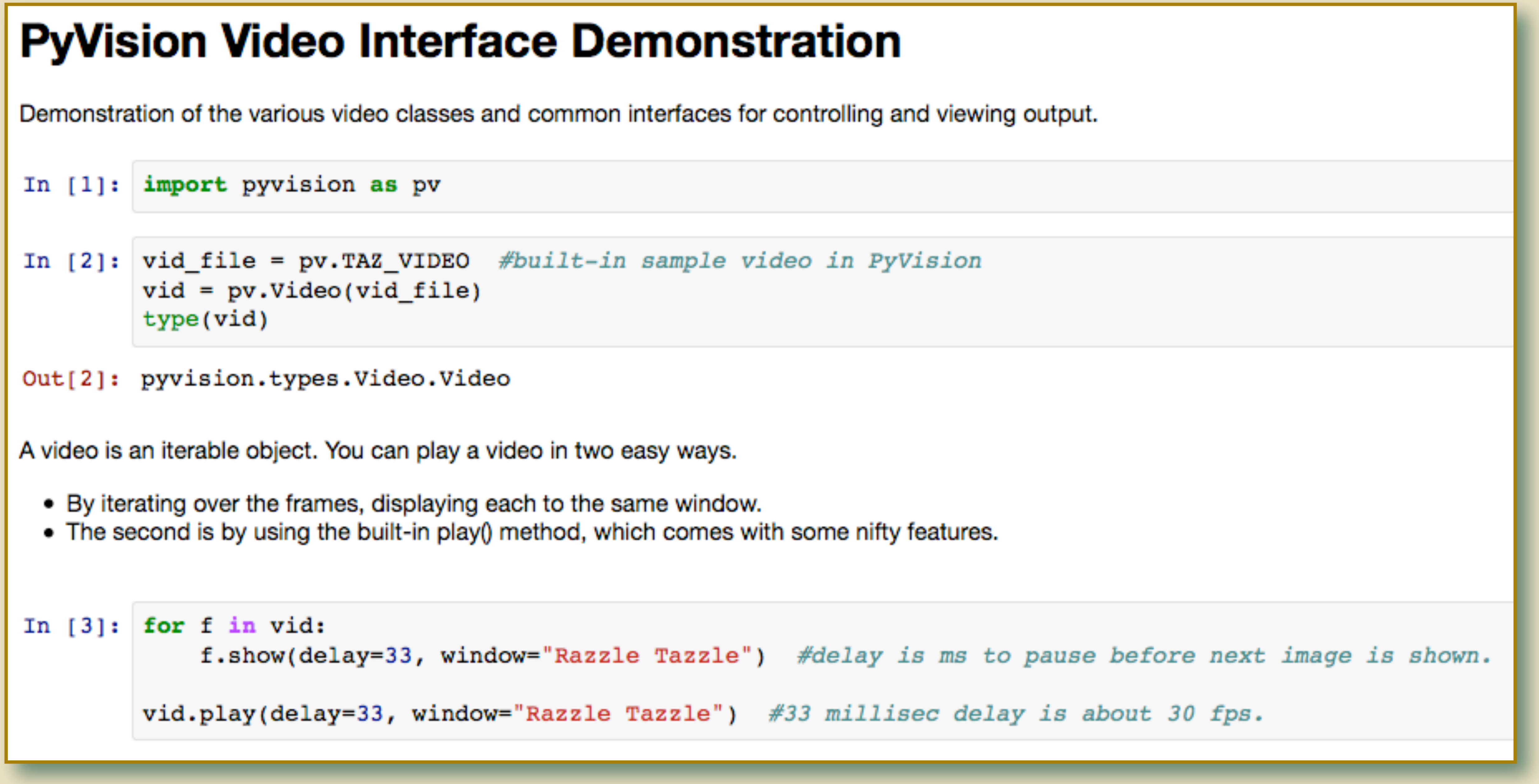




A1: row

	A	B	C	D	E	F
1	row	image	width	height		
2	1	baboon.jpg	512	512		
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						

PyVision Video Interface Classes



PyVision Video Interface Demonstration

Demonstration of the various video classes and common interfaces for controlling and viewing output.

```
In [1]: import pyvision as pv
```

```
In [2]: vid_file = pv.TAZ_VIDEO  #built-in sample video in PyVision
        vid = pv.Video(vid_file)
        type(vid)

Out[2]: pyvision.types.Video.Video
```

A video is an iterable object. You can play a video in two easy ways.

- By iterating over the frames, displaying each to the same window.
- The second is by using the built-in play() method, which comes with some nifty features.

```
In [3]: for f in vid:
            f.show(delay=33, window="Razzle Tazzle")  #delay is ms to pause before next image is shown.

        vid.play(delay=33, window="Razzle Tazzle")  #33 millisec delay is about 30 fps.
```

Question / Answer