Welcome to Py4Science

```
print"hello world"
```

Python is great

- ✔ 'scripting language'
- ✔ object orientated
- ✔ interpreted, dynamic typing
- ✔ minimal syntax
- ✔ flexible data structures

The Zen of Python

- ✔ Beautiful is better than ugly.
- ✔ Explicit is better than implicit.
- ✔ Simple is better than complex
- ✔ complex is better than complicated
- ✔ Flat is better than nested.
- ✔ Sparse is better than dense.
- ✔ Readability counts.

Here are some examples

```
import Bio.Seq as BioS
import Bio.Alphabet as BioA

genes = ["ACTCATATTTCTTAAATTTCTTAAACTCATGAATAACCTTATACTCATCAAAATTAA"
         "TCTGAAATGATTAATGCTCGT"]

for gene in genes:
    print "genetic code =\n", gene, "\namino acids = ",
    coding_dna = BioS.Seq(gene, BioA.IUPAC.unambiguous_dna)
    print coding_dna.translate(stop_symbol=' '), "\n"
```

```
#!/bin/bash

python -m SimpleHTTPServer 8080
```

```python
import os.path
import requests

def get_from_strawlab(name):
    """ Downloads and saves a file (if it doesnt exist) from the strawlab website """
    fname = name.split("/")[-1]
    if not os.path.exists(fname):
      data = requests.get("http://straw-static.imp.ac.at/py4science-vbc/%s" % name)
      with open(fname, "w") as f:
          f.write(data.content)
    return fname

if __name__ == "__main__":
    print open(get_from_strawlab("week1/README"),'r').read()
```

```
import scipy.misc
import matplotlib.pyplot as plt

import getfile

im = scipy.misc.imread(getfile.get_from_strawlab("week1/lena-gray.png"))
plt.imshow(im)
plt.colorbar()

plt.show()
```

```
#!/usr/bin/env python

import sys
import argparse
import Image

import getfile

parser = argparse.ArgumentParser()
parser.add_argument('images', metavar='N', type=str, nargs='*',
                    help='images (default:lena)',
                    default=[getfile.get_from_strawlab("week1/lena-color.png")])
parser.add_argument('-t', '--threshold', type=int, default=80)
args = parser.parse_args()

for fn in args.images:
    print "Thresholding %s at %d" % (fn, args.threshold)
    im = Image.open(fn)
    im = im.convert('L')
    thr = im.point(lambda x: (x > args.threshold)*255)
    thr.save("binary_"+fn)
```

```
import cv
import sys

import getfile

lena = "lena-gray.png"
img = cv.LoadImageM(getfile.get_from_strawlab("week1/"+lena))
cv.Smooth(img, img, smoothtype=cv.CV_GAUSSIAN, param1=11)
cv.SaveImage("smoothed-"+lena, img)
```

```
import sympy as sp
import numpy as np

M = sp.Matrix(sp.var('m0 m1 m2 m3 m4 m5 m6 m7 m8')).reshape(3, 3)
r = sp.Matrix(sp.var('x y z'))
t = sp.Matrix(sp.var('t0 t1 t2'))

eq = M * r - t
sp.pretty_print(eq)

solution = [sp.solve(surface, z) for surface in eq]

vm = np.array(((1.0, 0.1, -0.1), (0.1, 1.0, -0.1), (-0.1, -0.1, 1.0))).reshape(9)
vt = (np.random.random(3) - 0.5) * 0.1

p = sp.Plot()
p[1] = solution[0][0].subs(zip(M[:], vm)).subs(zip(t[:], vt))
p[2] = solution[1][0].subs(zip(M[:], vm)).subs(zip(t[:], vt))
p[3] = solution[2][0].subs(zip(M[:], vm)).subs(zip(t[:], vt))
```

```
import urllib2
import BeautifulSoup

soup = BeautifulSoup.BeautifulSoup(
        urllib2.urlopen('http://seminars.viennabiocenter.org/seminars.php').read(),
        convertEntities='html')
for seminar in soup('table', {'width': '700'}):
    title, body = seminar('td')
    title = title.text.replace("VBC Regular Seminar:", "")
    date = body('b')[0].text

    print "%s \n\t - %s" % (title, date)
```

```
import time
import ue9

jack = ue9.UE9()
outp = 0xAA

while True:
    jack.feedback(FIOMask=0xFF, FIODir=0xFF, FIOState=outp)
    time.sleep(0.5)
    jack.feedback(FIOMask=0xFF, FIODir=0xFF, FIOState=~outp)
    time.sleep(0.5)
```

```
import numpy as np
import matplotlib.pyplot as plt

import getfile

STATE_SKIP, _, STATE_SPEC_NAME, STATE_SPEC_DATE, _, STATE_DATA = range(6)
FREQ_FROM, FREQ_TO = 190, 841

spectrum_names = []
def func():
    state = STATE_SPEC_NAME
    for line in open(getfile.get_from_strawlab("week1/nanodrop-spectra.tsv")):
        if state == STATE_SPEC_NAME:
            spectrum_names.append(line.strip())
        elif state == STATE_DATA:
            wavelength, absorbance = line.strip().split()
            yield absorbance
            if int(wavelength) == (FREQ_TO - 1):
                state = STATE_SKIP
            continue
        state += 1
data = np.fromiter(func(), float)

wavelengths = np.arange(FREQ_FROM, FREQ_TO)
spectra = data.reshape(len(spectrum_names), FREQ_TO - FREQ_FROM).transpose()

plt.plot(wavelengths, spectra)
plt.legend(spectrum_names, loc=4)
plt.show()
```

```
import matplotlib.pyplot as plt
import matplotlib.animation as anim
import pylsm.lsmreader as lsm

import getfile

lsmfile = lsm.Lsmimage(getfile.get_from_strawlab("week1/DB331-brain.bin"))
lsmfile.open()

Z = lsmfile.header['CZ LSM info']['Dimension Z']

fig = plt.figure()
layer = lsmfile.get_image(stack=0, channel=0)
im = plt.imshow(layer, cmap=plt.cm.hot)

def updatefig(frame, i, Z, lsmfile):
    i[0] = i[0] + 1 if i[0] < Z - 1 else 0
    im.set_array(lsmfile.get_image(stack=i[0], channel=0))
    return im,

ani = anim.FuncAnimation(fig, updatefig,
                         fargs=([0], Z, lsmfile), interval=50, blit=True)
plt.show()
```

```
import cv
import time
import getfile

movie = cv.CaptureFromFile( getfile.get_from_strawlab("week1/flies.avi") )
nframes, rows, cols = map(lambda x: int(cv.GetCaptureProperty(movie, x)),
    [cv.CV_CAP_PROP_FRAME_COUNT, cv.CV_CAP_PROP_FRAME_HEIGHT, cv.CV_CAP_PROP_FRAME_WIDTH])

background = cv.CreateMat(rows, cols, cv.CV_8UC3)
for i in range(nframes):
    cv.Max(cv.QueryFrame(movie), background, background)
cv.ShowImage("background", background)

cv.SetCaptureProperty(movie, cv.CV_CAP_PROP_POS_FRAMES, 0)
for i in range(nframes):
    frame = cv.QueryFrame(movie)
    cv.ShowImage("before", frame)
    cv.Sub(background, frame, frame)
    cv.Smooth(frame, frame, param1=5)
    cv.Threshold(frame, frame, 70, 255, cv.CV_THRESH_BINARY)
    cv.ShowImage("after", frame)
    if cv.WaitKey(20) % 256 == ord('q'):
        break
```

```
import csv
import sqlite3 as sqlite
import numpy as np
import matplotlib.pyplot as plt

import getfile

csvfile = open(getfile.get_from_strawlab("week1/CTS.csv"),'rb')
con = sqlite.connect(':memory:')

with con and csvfile:
    csv = csv.reader(csvfile)
    cur = con.cursor()
    cur.execute("CREATE TABLE CTS(date INTEGER PRIMARY KEY, co2 FLOAT, temp FLOAT);")

    header = csv.next() #save the csv header row
    idx_date = header.index("yr_mn")
    idx_co2 = header.index("CO2")
    idx_temp = header.index("GISS")
    for row in csv:
        cur.execute("INSERT INTO CTS VALUES (?,?,?)", (row[idx_date],row[idx_co2],row[idx_temp]) )

    cur.execute("SELECT date, co2 FROM cts WHERE co2 != 'NA'")
    data = np.array(cur.fetchall())

    plt.plot(data[:,0],data[:,1])
    plt.xlabel("date"); plt.ylabel("CO2 (ppm)")
    plt.show()
```

```python
import numpy as np
import matplotlib.pyplot as plt

from scipy.io import wavfile
from getfile import get_from_strawlab

rate,data = wavfile.read(get_from_strawlab("week1/sirentone.wav"))
left = data[:,1]

nsamps = len(left)
t = np.arange(nsamps, dtype=float) / rate

plt.plot(t, left) #plot in time domain

xfreq = np.fft.fft(left)
fft_freqs = np.fft.fftfreq(nsamps, d=1./rate)

plt.figure()      #plot in freq domain
plt.loglog(fft_freqs[0:nsamps/2], np.abs(xfreq)[0:nsamps/2])

plt.figure()
plt.specgram(data[:,1])

plt.show()
```

```
import subprocess

from Bio import Entrez, SeqIO, AlignIO
from Bio.Emboss.Applications import NeedleCommandline

Entrez.email = "stowers@imp.ac.at"

handle = Entrez.efetch(db="nucleotide", id="AF182035",rettype="gb", retmode="text")
homo = SeqIO.read(handle, "genbank")

handle = Entrez.efetch(db="nucleotide", id="AY863830",rettype="gb", retmode="text")
drosophila = SeqIO.read(handle, "genbank")

open("homo.fasta","w").write(homo.format("fasta"))
open("drosophila.fasta","w").write(drosophila.format("fasta"))

cmdline = NeedleCommandline(
            asequence="homo.fasta",
            bsequence="drosophila.fasta",
            gapopen=10, gapextend=0.5, outfile="needle.txt")
subprocess.call(str(cmdline), shell=True)

align = AlignIO.read("needle.txt", "emboss")
print align
```

```python
#!/usr/bin/env python

from gi.repository import Gtk

import getfile
getfile.get_from_strawlab("week1/lena-color.png")
getfile.get_from_strawlab("week1/lena-gray.png")

class App(Gtk.Builder):
    def __init__(self):
        super(App, self).__init__()
        self.add_from_file("gui.ui")
        self.connect_signals(self)

        self._label = self.get_object("label1")
        self._image = self.get_object('image1')
        self._on_button_clicked()

        window = self.get_object('window1')
        window.connect('destroy', lambda x: Gtk.main_quit())
        window.show_all()

    def _on_button_clicked(self, *args):
        lena = "lena-color.png" if self._label.get_text() == "lena-gray.png" else "lena-gray.png"
        self._image.set_from_file(lena)
        self._label.set_text(lena)

app = App()
Gtk.main()
```

Sandwich!

What shall we talk about next week?

-- [command=_vbc python genslides.py] [slide-contents-exec=source-highlight -s py -i genslides.py