LECTURE 7:
STUDENT REQUESTED TOPICS

Introduction to Scientific Python, CME 193
Feb. 20, 2014
Please download today's exercises from:
`web.stanford.edu/~ermartin/Teaching/CME193-Winter15`
Eileen Martin

Overview Today

- Writing scripts that interact with the operating system
- Multithreading
- PCA & applications
- Discussion of this week's assignment

Scripts to interact with the computer

- Three very useful modules for this are:
  - os
    - A variety of operating system interfaces
    - http://www.pythonforbeginners.com/os/pythons-os-module
  - shutil
    - File and directory handling
    - https://docs.python.org/2/library/shutil.html
  - sys
    - System specific parameters/functions
    - https://docs.python.org/2/library/sys.html

- Examples:
  - Create a directory, copy a few files from another directory (os, shutil)
    - copyFile.py
  - Rename a directory (os)
    - renameFolder.py
  - Add paths to look for modules, check the type of computer, avoid asking recursive programs to go too far, or get user input at the beginning of a program (sys)
    - get_sys_info.py

Example scripts

- Create a directory, copy a few files from another directory (os,shutil)
  - copyFile.py
- Rename a directory (os)
  - renameFolder.py
- Add paths to look for modules, check the type of computer, avoid asking recursive programs to go too far, or get user input at the beginning of a program (sys)
  - get_sys_info.py

Exercise:

Work in small groups to modify the code in copyFile.py so that if you typed this in the command line:

```
python copyFile.py 4
```

your code would create 4 copies of `startTestDir` named:

```
copyDir1
copyDir2
copyDir3
copyDir4
```

that each contain `test_file1.txt` and `test_file2.txt`

Shell scripts, unix commands

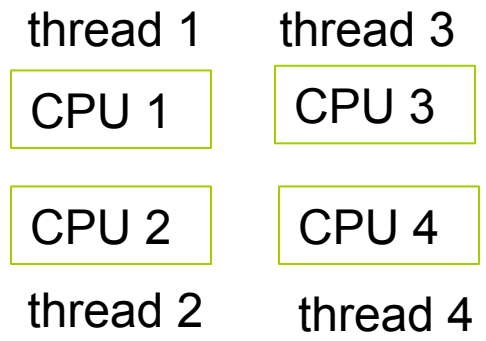
- You can also make shell scripts (like bash)
  - Start with `#!/usr/bin/env python`
- Or use the `subprocess` module
  - https://docs.python.org/2/library/subprocess.html
  - Example: openVI.py (Hit i to start typing, then `esc : wq` to save+quit)
- You can submit jobs to clusters using Python scripts
  - https://wiki.anl.gov/cnm/HPC/Submitting_and_Managing_Jobs/Example_Job_Script

Overview Today

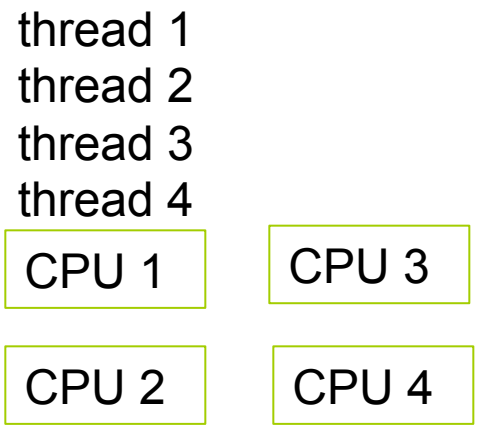
- Writing scripts that interact with the operating system
- Multithreading
- PCA & applications
- Discussion of this week's assignment

Multiprocessing vs multithreading

- Multiprocessing allows us to split up a procedure amongst multiple processes
  - Each process uses different memory space
  - Don't need to worry about synchronization or race conditions as much so code is more straightforward than multithreading
  - Can take advantage of multiple CPUs



- As opposed to the multithreading module, which runs multiple threads
  - Threads share memory space
  - Easier to share variables between threads
  - Quicker to start up threads (lower memory overhead too) than processes



The multiprocessing module's pools

- Create a Pool object with some number of processes

```
p = multiprocessing.Pool(processes = 4)
```

- Use the Pool's methods to apply a function

- The method may block computation from continuing until the function application is completed, or be asynchronous

- The method may subdivide the input (map) or not (apply)

Ways to apply a function

method to apply	description
`p.apply(f[,args[,keywords]])`	• Applies function f in separate process • Only one process in the pool runs f(args,keywords) • Blocks until result is read

`result = p.apply_async(f[,args [,keywords[,callback]]])`
 	
`answer = result.get()`
	• Like apply, but returns a result
	• Can apply callback to it as soon as the result is ready
	• apply_async() better suited to parallel than apply()
	• Must call get() method on ApplyResult object (blocks)

`p.map(f,iterable[, chunksize])`	• Breaks iterable into chunks, each thread gets chunk • Blocks program from continuing until this is complete • Applies f to each element of the iterable

`result = p.map_async(f, iterable[,chunksize[, callback]])`
 	
`answer = result.get()`
	• Breaks iterable in chunks & each thread gets a chunk
	• Applies callback or moves on through the code
	• Must call get() method on MapResult object(blocks)

Example of using a Pool

- The is_prime() function from assignment 2 was pretty slow, so we want to speed up checking each int in a long list.
- Because each prime check is independent, we can split up the list into chunks and each chunk gets all its entries checked.
- Open up and run prime.py and primeMultiproc.py
  - Checks if each entry of [0, 1, …, 99999] is prime, then checks if each entry of [100000,100001,…,199999] is prime
  - This is done in three ways and the times are compared:
    - Serial (one process)
    - Uses map, which blocks (as many processes as CPUs)
    - Uses map_async, which doesn't block (as many processes as CPUs)
- More examples and details on using multiprocessing: http://pymotw.com/2/multiprocessing/basics.html

Overview Today

- Writing scripts that interact with the operating system
- Multithreading
- PCA & applications
- Discussion of this week's assignment

Principal component analysis (PCA)

- Finds most important directions that explain a dataset
- 1st vector is less dependent on your choice of coordinate system than least-squares
- Steps, D data matrix:
  - get covariance matrix D*D
  - get eigenvalues & eigenvectors of D*D
  - keep e-vecs from largest e-vals in rows of matrix E
  - transform data ED*

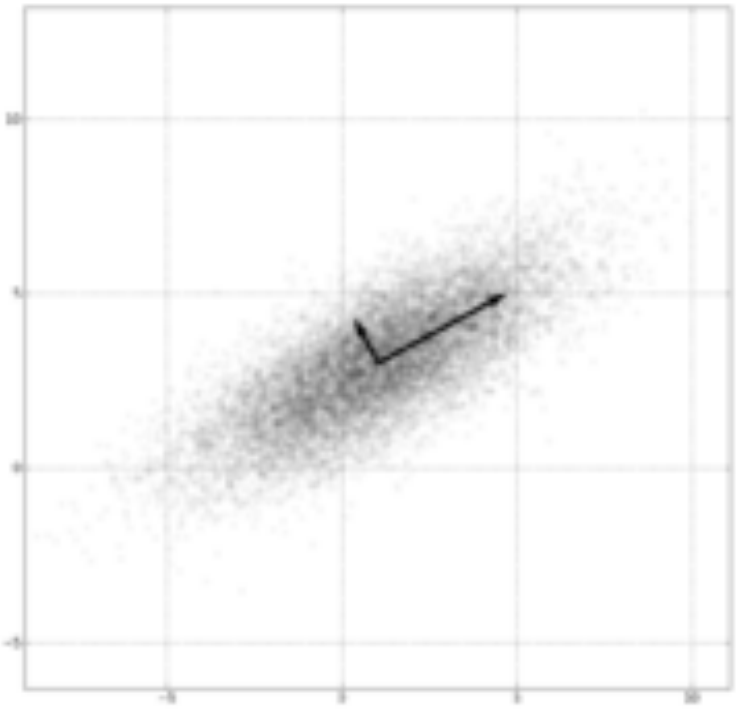


Image from wikimedia commons

Fisher's Iris data set

- Can we distinguish between different types of iris flowers based on a few measurements?
- Example: irisPCA.txt
- Tab delimited text file irisData.txt
- 5 columns in this order:
  - Sepal length (float)
  - Sepal width (float)
  - Petal length (float)
  - Pedal width (float)
  - Species (string)

Image from wikimedia commons

Want to do machine learning in Python? Check out scikit-learn

- Can download from scikit-learn.org
- Built on top of NumPy, SciPy, matplotlib
- Simple syntax and pre-tested functions
  - PCA can be called via `sklearn.decomposition.PCA()`
- Has many preloaded datasets to test algorithms on
- You can see an example of PCA with the Iris data in their tutorials using 3 dimensions:
  - http://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_iris.html

Overview Today

- Writing scripts that interact with the operating system
- Multithreading
- PCA & applications
- Discussion of this week's assignment

Assignment 7

- This lecture covered a few topics requested by students, but there is a lot more functionality available in Python

- For this week, try something you think is interesting to do with Python and make a < 5 minute video about it. Possibilities:
  - Use some of the tools we've already discussed this quarter to solve a problem you're interested in
  - Explore a new module, show a few of its functions or data structures
  - Test whether multithreading speeds up some particular problem
  - Analyze and/or visualize some data set
  - Compare performance & code in Python to another language you know
  - Try out the map-reduce framework
  - Download a Python package that isn't in the standard library and show a few simple things you can do with it
  - Create a simple web-crawler
  - Learn how to do some exception handling so code doesn't stop running due to errors

Assignment 7: Details

- You will submit a text file with a link to your video and your sharing preference

- On Mac it's easiest to do screenshot videos with Quicktime (make sure the sound is on)
- screenr.com lets you record up to 5 minute videos without downloading any software

- Some options for **posting** your video:
  - Youtube (public, unlisted, or private)
  - Vimeo (anyone, only people with a password- you provide instructors with the password)

- If you are okay with these being shared, choose your **privacy** level:
  - public – video link on course website can be viewed by anyone
  - class only- video link posted on a password protected part of the course website
  - private (default)- only instructors can view video, no link will be posted anywhere on the course website