Practical Python on Odyssey

Aaron Kitzmiller, Meghan Porter-Mahoney, and Adam Freedman

Overview

- Learn python by debugging existing code
- See common errors and their solutions
- Learn how to search for programming solutions
- Odyssey-specific lessons, including Anaconda clones

Covered subjects

Not necessarily in this order

- Structure (if/then, for, tuples, arrays, dicts, functions)
- Regular expressions, dates
- Interacting with your environment (os, environment variables, files, executing other tools)
- Packages and virtual environments (pip, python setup.py, virtualenv, Anaconda, clones)
- Parallel programming (multiprocess)

Setup

- Login to Odyssey
- Get the course materials

```
[akitzmiller@holy2a ~]$ tar xvf /n/regal/informatics/workshops/python-workshop.tar.gz
```

- Check python

```
[akitzmiller@holy2a ~]$ python --version
Python 2.7.5
[akitzmiller@holy2a ~]$ which python
/usr/bin/python
```

- Hop in to the interpreter

```
[akitzmiller@holy2a ~]$ python
Python 2.7.5 (default, Aug  4 2017, 00:39:18)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-16)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Use `Ctl-d` to get out

Course materials

- `bin/hisnhers.py` - The broken script
- `bin/megaAssembler` - The high memory assembler
- `bin/hyperAssembler` - The fast, efficient assembler
- `ha/annotate.py` - The annotation module
- `https://github.com/harvardinformatics/lookkool.git` - The palindrome finder

hisnhers.py

Broken script that attempts to

1. read in a FASTQ file

````
@HWUSI-EAS300R_0005_FC62TL2AAXX:8:30:1207:12132#0/1
AGGGCGACCGGCAAGCAGGGGTTCGAACGGCAGGAGCCCC
+HWUSI-EAS300R_0005_FC62TL2AAXX:8:30:1207:12132#0/1
b```acdffff[fffgffgggggggddRdbfffc]]bddd
@HWUSI-EAS300R_0005_FC62TL2AAXX:8:30:1293:12132#0/1
CGGGCCACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
+HWUSI-EAS300R_0005_FC62TL2AAXX:8:30:1293:12132#0/1
L_BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
````

2. write to FASTA and feed to an assembler to create contigs

```
@HWUSI-EAS300R_0005_FC62TL2AAXX:8:30:1207:12132#0/1
AGGGCGACCGGCAAGCAGGGGTTCGAACGGCAGGAGCCCC
@HWUSI-EAS300R_0005_FC62TL2AAXX:8:30:1293:12132#0/1
CGGGCCACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

3. annotate the contigs using the ha.annotate module

`ha/annotate.py`

Annotation module that will be called by `hisnhers.py` serially and, then, in parallel

```
AGGGCGACCGAATTCTACGATGCATC
         CTTAAG
         ------  palindrome 10..15
```

The Python Language

- General purpose, interpreted scripting language in which everything (numbers, lists, strings, functions, classes, types) is an object.
- Code blocks (functions, loops, if/else, etc.) defined by colon and indent level
- Significant changes to the language from Python 2.x to Python 3.x
- Massive PyPI package repository (`pip install <something from PyPI>`)
- A file is a module, a directory can be a package

Run the script

```
[akitzmiller@holy2a python-workshop]$ bin/hisnhers.py
```

Bad interpreter error

```
[akitzmiller@holy2a python-workshop]$ bin/hisnhers.py
-bash: bin/hisnhers.py: /usr/local/bin/python: bad interpreter: No such file or directory
```

Bad interpreter fix

Flexible interpreter path in the shebang line that picks up the default python in your environment

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

Run the script

```
[akitzmiller@holy2a python-workshop]$ bin/hisnhers.py
```

Indentation error

```
[akitzmiller@holy2a python-workshop]$ bin/hisnhers.py
  File "bin/hisnhers.py", line 20
    seqs = []
    ^
IndentationError: unexpected indent
```

Indentation fix

Move lines 20 and 21 back two spaces.

Do not use tabs

```
15 def fastqToSequenceList(fileh):
16     '''
17     Takes a fastq file handle, returns a list tuples including
18     seqid, sequence bases, and quality scores
19     '''
20     seqs = []
21     seqid = bases = qscores = None
22 
23     if fileh.closed:
```

Run the script

```
[akitzmiller@holy2a python-workshop]$ bin/hisnhers.py
```

Import error

```
[akitzmiller@builds python-workshop]$ bin/hisnhers.py
Traceback (most recent call last):
  File "bin/hisnhers.py", line 118, in <module>
    sys.exit(main())
NameError: name 'sys' is not defined
```

Import error fix

Add `import sys` to the import section of the script

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

'''
hisnhers.py
Harvard Informatics Script for Nextgen HiSeq Extraction and Reporting of Sequences

'''
import sys
import os, traceback, re
import json
import subprocess
import time
```

`import`

- A name (function, class, variable, module) cannot be used unless it is imported, defined, or a *built-in*
- You can import a module (which is a file) and use it's named things

```
[akitzmiller@holy2a ~]$ ls /usr/lib64/python2.7/os.py
/usr/lib64/python2.7/os.py
```

```
>>> import os
>>> os.makedirs('/tmp/a/j/k')
```

- or you can import something from a module

```
[akitzmiller@holy2a ~]$ grep "def makedirs" /usr/lib64/python2.7/os.py
def makedirs(name, mode=0777):
```

```
>>> from os import makedirs
>>> makedirs('/tmp/a/j/k')
```

`import`

- Imports are based on paths, where path separators, `/` , are converted to periods

```
[akitzmiller@holy2a python-workshop]$ find ha -name "annotate.py"
ha/annotate.py
```

```
from ha.annotate import annotateStartStopCodon
```

- Valid paths depends on `sys.path` , including `PYTHONPATH`

```
[akitzmiller@holy2a ~]$ echo $PYTHONPATH
/odyssey/rc_admin/sw/admin/rcpy:

[akitzmiller@holy2a ~]$ python
Python 2.7.5 (default, Aug  4 2017, 00:39:18)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-16)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> print sys.path
['', '/odyssey/rc_admin/sw/admin/rcpy', '/n/home_rc/akitzmiller/workshop',
'/usr/lib64/python27.zip', '/usr/lib64/python2.7', '/usr/lib64/python2.7/plat-linux2',
'/usr/lib64/python2.7/lib-tk', '/usr/lib64/python2.7/lib-old', '/usr/lib64/python2.7/lib-dynload',
'/usr/lib64/python2.7/site-packages', '/usr/lib64/python2.7/site-packages/gtk-2.0',
'/usr/lib/python2.7/site-packages']
```

- Watch out for `~/.local`

`os` and `sys` modules

- `os` includes functions that vary between operating systems

```
# On Linux
>>> os.path.join(['usr','local','bin'])
usr/local/bin

# On Windows
>>> os.path.join(['usr','local','bin'])
usr\local\bin

# Interact with environment variables
>>> os.environ['PATH']
'/usr/local/bin:/usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin'
>>> os.system('which module-query')
/usr/local/bin/module-query
0
>>> os.environ['PATH'] = '/n/sw/www/apps/apps/bin:%s' % os.environ['PATH']
>>> os.system('which module-query')
/n/sw/www/apps/apps/bin/module-query
0
```

- `sys` includes functions and data about the Python interpreter
  ** `sys.exit()` exits the Python interpreter
  ** `sys.argv` contains the arguments passed to the script

Run the script

```
[akitzmiller@holy2a python-workshop]$ bin/hisnhers.py
```

Attribute error

```
[akitzmiller@holy2a ~]$ bin/hisnhers.py
 Traceback (most recent call last):
   File "./bin/hisnhers.py", line 263, in <module>
     sys.exit(main())
   File "./bin/hisnhers.py", line 131, in main
     fastqToSequenceList(fqfilename)
   File "./bin/hisnhers.py", line 98, in fastqToSequenceList
     if fileh.closed:
 AttributeError: 'str' object has no attribute 'closed'
```

- Stack trace shows you where to look

Google "python open file handle"

The top hit may not be the best one

Attribute error fix

Open a file handle

```
def main():

    # Read fastq file and report length, base counts
    seqs = []
    fqfilename = '/n/regal/informatics/aaron/testfile.fq'
    fqfileh = open(fqfilename, 'r')
    seqs = fastqToSequenceList(fqfileh)

```

Run the script

```
[akitzmiller@holy2a python-workshop]$ bin/hisnhers.py
```

No such file or directory error

```
[akitzmiller@holy2a python-workshop]$ bin/hisnhers.py
Traceback (most recent call last):
  File "bin/hisnhers.py", line 119, in <module>
    sys.exit(main())
  File "bin/hisnhers.py", line 54, in main
    fqfileh = open(fqfilename, 'r')
IOError: [Errno 2] No such file or directory: '/n/regal/informatics/aaron/testfile.fq'
```

No such file or directory error fix

```python
def main():

    # Read fastq file and report length, base counts
    seqs = []
    fqfilename = 'data/example.fq'
    fqfileh = open(fqfilename, 'r')
    seqs = fastqToSequenceList(fqfileh)
```

Convert the hardcoded file name into a command argument

`sys.argv` is the command line argument list for the python interpreter

```
# > /usr/bin/python hisnhers.py data/example.fq
# sys.argv == ['hisnhers.py','data/example.fq']
if len(sys.argv) < 2:
    print 'Must supply a file name'
    return 1
fqfilename = sys.argv[1]
```

No such file or directory error fix (better)

```python
def main():

    # Read fastq file and report length, base counts
    seqs = []
    # sys.argv == ['hisnhers.py','data/example.fq']
    if len(sys.argv) < 2:
        print 'Must supply a file name'
        return 1

    fqfilename = sys.argv[1]
    if not os.path.exists(fqfilename):
        raise Exception('File %s does not exist' % fqfilename)

    fqfileh = open(fqfilename, 'r')
    seqs = fastqToSequenceList(fqfileh)
```

File processing with context managers

- `f = open()` returns a file handle
- `with` block is a context manager that closes the file handle on exit

```
# Code block defined by colon and indent
with open(fqfilename,'r') as f:
    seqs = fastqToSequenceList(f)
```

No such file or directory error fix (even better)

```
def main():

    # Read fastq file and report length, base counts
    seqs = []
    # sys.argv == ['hisnhers.py','data/example.fq']
    if len(sys.argv) < 2:
        print 'Must supply a file name'
        return 1

    fqfilename = sys.argv[1]
    if not os.path.exists(fqfilename):
        raise Exception('File %s does not exist' % fqfilename)

    with open(fqfilename,'r') as f:
        seqs = fastqToSequenceList(f)
```

Contigs file error

```
[akitzmiller@holy2a python-workshop]$ ./bin/hisnhers.py data/example.fq
Writing to data/example.fa
Traceback (most recent call last):
  File "./hisnhers.py", line 127, in <module>
    sys.exit(main())
  File "./hisnhers.py", line 90, in main
    with open(contigfilename,'r') as c:
IOError: [Errno 2] No such file or directory: 'data/example.fa.contigs'
[akitzmiller@holy2a python-workshop]$
```

Running commands with `os.system()`

- There are about a dozen Python functions for running a command line tool, but only two of them are worth using.
- `os.system()` runs a command using the shell and returns only the return code. stdout and stderr are sent to the console. If you need to capture the contents, they must be redirected.

```
>>> os.system("echo 'hello' > hello.out")
0
>>> f = open('hello.out','r')
>>> print f.readlines()
['hello\n']
```

Running commands with `Popen()`

- `subprocess.Popen` supports all available options for synchronous execution

```
>>> import subprocess
>>> proc = subprocess.Popen(
     "echo 'hello'",
     shell=True,
     stdout=subprocess.PIPE,
     stderr=subprocess.PIPE
)
>>> stdoutstr,stderrstr = proc.communicate()
>>> print proc.returncode
0
>>> print stdoutstr
hello
```

Running commands

- Avoid bash shell processing if you need to

```
>>> args = ['/usr/bin/sed','-i','-e','s/$PATH/${PATH}/','/home/path with some spaces in it']
>>> proc = subprocess.Popen(args,shell=False)
```

- Write to stdin

```
>>> lyrics = '''
... Sundown, you better take care
... If I find you been creepin
... Down my back stair
... '''
>>> args = ['/bin/grep','been creepin']
>>> from subprocess import PIPE,Popen
>>> proc = Popen(args,shell=False,stdin=PIPE,stdout=PIPE,stderr=PIPE)
>>> stdout,stderr = proc.communicate(input=lyrics)
>>> stdout
'If I find you been creepin\n'
>>>
```

Running commands

- You may need to alter the environment of the subprocess
- Loading modules can work with `&&`

```
proc = Popen('module load bowtie2 && bowtie2 -1 m1.in.bz2 -2 m2.in.bz2',shell=True)
```

- You can set environment values in the parent

```
>>> path = os.environ.get('PATH','')
>>> os.environ['PATH'] = '/n/sw/fasrcsw/apps/Core/bowtie2/2.3.1-fasrc01/bin:%s' % path
>>> proc = Popen('bowtie2 -1 m1.in.bz2 -2 m2.in.bz2',shell=True)
```

- or in the subprocess itself

```
>>> path = os.environ.get('PATH','')
>>> env = {'PATH' : '/n/sw/fasrcsw/apps/Core/bowtie2/2.3.1-fasrc01/bin:%s' % path}
>>> proc = Popen('bowtie2 -1 m1.in.bz2 -2 m2.in.bz2',shell=True,env=env)
```

Remove the stderr and stdout redirection from the os.system call.

Set your PATH to include project bin directory

```
> export PATH=`pwd`/bin:$PATH
```

Replace the call to megaAssembler with a Popen-based call to hyperAssembler.

Capture return code, stdout, and stderr

Call to hyperAssembler

```
import subprocess

def runcmd(cmd):
    '''
    Execute a command and return stdout, stderr, and the return code
    '''
    proc = subprocess.Popen(cmd,shell=True,stdout=subprocess.PIPE,stderr=subprocess.PIPE)
    stdoutstr, stderrstr = proc.communicate()
    return (proc.returncode,stdoutstr,stderrstr)

# Run hyperAssembler with fastq file input and read the output contig
contigfilename = '%s.contigs' % fafilename
cmd = 'hyperAssembler {inputfilename} {outputfilename}'.format(inputfilename=fafilename,outputfilename=c
returncode, stdoutstr, stderrstr = runcmd(cmd)

if returncode != 0:
    raise Exception('Error running assembler with cmd %s\nstdout: %s\nstderr: %s' % (cmd,stdoutstr,stder
```

Missing lookkool module

```
Traceback (most recent call last):
File "./bin/hisnhers.py", line 179, in <module>
  sys.exit(main())
File "./bin/hisnhers.py", line 160, in main
  annotations += annotatePalindromes(seqid, contig)
File "./ha/annotate.py", line 66, in annotatePalindromes
  from lookkool import findPalindromes
ImportError: No module named lookkool
```

Python packages

- A package is a set of Python modules and scripts (and possibly C, Fortran, etc. supporting code) that can be installed in a Python environment
- Python library called setuptools (son of distutils) allows packages of Python code to be installed in a standard fashion

```
[akitzmiller@holy2a ~]$ tar xvf mpi4py-2.0.0.tar.gz
[akitzmiller@holy2a ~]$ cd mpi4py-2.0.0
[akitzmiller@holy2a mpi4py-2.0.0]$ python setup.py install
```

- Install directly from PyPI with `pip` , including dependencies

```
[akitzmiller@holy2a /tmp]$ pip install Flask-Script
Collecting Flask-Script
  Downloading Flask-Script-2.0.5.tar.gz (42kB)
    100% |████████████████████████████████| 51kB 710kB/s
Collecting Flask (from Flask-Script)
  Downloading Flask-0.12-py2.py3-none-any.whl (82kB)
    100% |████████████████████████████████| 92kB 2.0MB/s
```

- Install from a git repository (including branch or tag)

```
[akitzmiller@holy2a ~]$ pip install git+https://github.com/harvardinformatics/MISO.git@slurm
```

Anaconda

- Python distribution that includes the most popular scientific and utility packages (numpy, scipy, matplotlib, etc.)
- Package management system (`conda install/remove/update`)
  ** pip-like dependency recursion
  ** maintains compatible versions among dependencies
  ** may include compiled C / Fortran libraries
  ** supports multiple "channels"
  ** update Python itself
- Odyssey python modules are Anaconda modules

```
[akitzmiller@holy2a ~]$ module load python/2.7.11-fasrc01

[akitzmiller@holy2a ~]$ module list

Currently Loaded Modules:
  1) Anaconda/2.5.0-fasrc01   2) python/2.7.11-fasrc01
```

Anaconda

- Get the latest

```
[akitzmiller@holy2a ~]$ conda install netcdf4
```

- or a specific version

```
[akitzmiller@holy2a ~]$ conda install netcdf4==1.2.1
```

Virtual environments - virtualenv

- You don't have root so you can't install to system library paths.
- You can use `install --prefix` and `PYTHONPATH`, but it is a pain and some packages are poorly behaved
- Some packages depend on mutually exclusive versions of other packages
- virtualenv allows you to create one or more Python environments over which you have control

```
[akitzmiller@holy2a envs]$ virtualenv workshop
New python executable in /n/home_rc/akitzmiller/envs/workshop/bin/python
Installing setuptools, pip, wheel...done.
[akitzmiller@holy2a envs]$ source workshop/bin/activate
(workshop) [akitzmiller@holy2a envs]$ which python
```

Anaconda virtual environments

- Make a local conda environment

```
[akitzmiller@holy2a ~] module load python/2.7.14-fasrc02
[akitzmiller@holy2a ~] conda create -n workshop pip -y
Fetching package metadata .................
Solving package specifications: .

Package plan for installation in environment /n/home_rc/akitzmiller/.conda/envs/workshop:

The following NEW packages will be INSTALLED:

    ca-certificates: 2018.8.24-ha4d7672_0 conda-forge
    ...
    pip:             18.1-py27_1000       conda-forge
    python:          2.7.15-h33da82c_4    conda-forge
    ...
    zlib:            1.2.11-h470a237_3    conda-forge

#
# To activate this environment, use:
# $ source activate workshop
#
# To deactivate this environment, use:
# $ source deactivate
#
```

- Clone names can be a full path

```
[akitzmiller@holy2a ~] conda create -p /n/my_lab/shared/software/pyenv --clone $PYTHON_HOME
```

- Install package from Continuum

```
(workshop)[akitzmiller@holy2a ~] conda install Django --yes
```

- Or from a particular conda channel

```
 (workshop)[akitzmiller@holy2a ~] conda install --channel conda-forge tensorflow
```

- Or do a pip install

```
 (workshop)[akitzmiller@holy2a ~] pip install BioPython
 Collecting BioPython
   Downloading biopython-1.68.tar.gz (14.4MB)
     100% |████████████████████████████████| 14.4MB 24kB/s
```

- Compiled code in conda packages can be a problem

```
(workshop)[akitzmiller@holy2a ~] conda install -c conda-forge tensorflow

(workshop)[akitzmiller@holy2a ~] python
>>> import tensorflow as tf
Traceback (most recent call last):
...
  File "/n/home_rc/akitzmiller/.conda/envs/clone/lib/python2.7/site-packages/tensorflow/python/pywr
    _mod = imp.load_module('_pywrap_tensorflow', fp, pathname, description)
ImportError: /usr/lib64/libstdc++.so.6: version `GLIBCXX_3.4.19' not found (required by /n/home_rc/
>>>
```

- Installing with pip instead of conda generally compiles source code, which may or may not be better

```
(workshop)[akitzmiller@holy2a ~] pip install gattlib
Collecting gattlib
  Downloading gattlib-0.20150805.tar.gz (1.7MB)
    100% |████████████████████████████████| 1.7MB 170kB/s
Building wheels for collected packages: gattlib
  Running setup.py bdist_wheel for gattlib ... error
  Complete output from command /n/home_rc/akitzmiller/.conda/envs/clone/bin/python -u -c "import setu
  running bdist_wheel
  running build
  running build_ext
  building 'gattlib' extension
  creating build
  ...
  creating build/temp.linux-x86_64-2.7/src/bluez/btio
  gcc -pthread -fno-strict-aliasing -g -O2 -DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I
  cc1plus: warning: command line option "-Wstrict-prototypes" is valid for Ada/C/ObjC but not for C+
  src/gattservices.cpp:6:33: error: bluetooth/bluetooth.h: No such file or directory
  src/gattservices.cpp:7:27: error: bluetooth/hci.h: No such file or directory
  src/gattservices.cpp:8:31: error: bluetooth/hci_lib.h: No such file or directory
  In file included from src/gattlib.h:22,
                   from src/gattservices.cpp:12:
  src/bluez/attrib/gatt.h:25:27: error: bluetooth/sdp.h: No such file or directory
  In file included from src/gattlib.h:19,
                   from src/gattservices.cpp:12:
  src/bluez/lib/uuid.h:153: error: ‘uint128_t’ does not name a type
```

Fix the missing lookkool module by installing from the Harvard Informatics github repository into an Anaconda clone

```
pip install git+https://github.com/harvardinformatics/lookkool.git
```

or

```
> cd ..
> tar xvf /n/regal/informatics/workshops/lookkool.tar.gz
> cd lookkool
> pip install .
```

Source code installation error

```
creating build/temp.linux-x86_64-2.7
creating build/temp.linux-x86_64-2.7/src
gcc -pthread -fno-strict-aliasing -g -O2 -DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/n
src/lookkool.c:3:23: error: stdatomic.h: No such file or directory
error: command 'gcc' failed with exit status 1
```

Google "error: stdatomic.h: No such file or directory"

Source code installation fix

Load a newer gcc module

```
> module load gcc/4.9.3-fasrc01
> pip install .
```

Parallel Python - Multiprocessing

- The Python interpreter does not support real parallel threading
- The `multiprocessing` module simulates a typical threading library using forked processes
- Do something else, while a tool runs in the background

```
from multiprocessing import Process

def runAnalysis(parametersfile){
   cmd = 'OMA %s' % parametersfile
   os.system(cmd)
}
p = Process(target=runAnalysis,args=(parametersfile))
p.start()
# Do some other stuff
...
p.join()
```

Parallel Python - Multiprocessing Pool

- If you're doing a variable number of simultaneous processes, you may want to use a Pool

```
>>> from multiprocessing import Pool
>>> import os
>>> def echo(echoable):
...     os.system('echo %s && sleep 10' % echoable)
...
>>> echoables = [
...     'ajk',
...     '123',
...     'qwerty',
...     'uiop',
...     'lkjdsa',
... ]
>>> numprocs = os.environ.get('NUMPROCS',3)
>>> pool = Pool(numprocs)
>>> result = pool.map(echo,echoables)
123
ajk
qwerty
lkjdsa
uiop
```

Parallel Python - Multiprocessing Pool

- `Pool.map` does not work if you have more than one argument, so iterate through and use `apply_async`
- You'll need to "get" the return value from the result object(s)

```
>>> from multiprocessing import Pool
>>> import os
>>> def greet(name,message):
...     os.system('echo "Hi %s, %s" && sleep 10' % (name,message))
...     return '%s was greeted' % name
...
>>> greetings = [
...     ('Aaron',"What's up?"),
...     ('Bert',"Where's Ernie?"),
...     ('Donald',"What're you thinking?"),
...     ('folks','Chill!'),
... ]
>>> numprocs = os.environ.get('NUMPROCS',3)
>>> pool = Pool(numprocs)
>>> results = []
>>> for greeting in greetings:
...    result = pool.apply_async(greet,greeting)
...    results.append(result)
Hi Bert, Where's Ernie?
Hi Aaron, What's up?
Hi Donald, What're you thinking?
Hi folks, Chill!
>>> for result in results:
...    print result.get()
Aaron was greeted
Bert was greeted
Donald was greeted
folks was greeted
```

Analyze the contigs using a multiprocessing pool. Compare the elapsed time with the for loop version.

Analyze contigs with a multiprocessing pool.

```
starttime = time.time()
from multiprocessing import Pool
numprocs = os.environ.get('ANNOTATION_PROC_NUM',2)
pool = Pool(numprocs)

annotations = []
results = []
for contig in contigs:
    result = pool.apply_async(annotateStartStopCodons,contig)
    results.append(result)
    result = pool.apply_async(annotatePalindromes,contig)
    results.append(result)

for result in results:
    annotations += result.get()

endtime = time.time()
```

Python dictionaries

- A dictionary is like a list, but can be indexed by non-integers (AKA hash map)
- Elements are not necessarily in the order you think

```
>>> basecounts = { 'A' : 230, 'T' : 120, 'C' : 999, 'G' : 100 }
>>> for base, count in basecounts.iteritems():
...     print '%s: %d' % (base,count)
...
A: 230
C: 999
T: 120
G: 100
```

- OrderedDict is available in Python 2.7 and you can order output by sorting keys

```
>>> for base in sorted(basecounts.keys()):
...     print '%s: %d' % (base,basecounts[base])
...
A: 230
C: 999
G: 100
T: 120
```

- Dictionary of lists

```
# Make a dictionary keyed by contig name
annotatedcontigs = {}
for annotation in annotations:
    annotatedcontigs.setdefault(annotation['seqid'],[]).append(annotation)
```

Python can be used to submit Slurm jobs

- Use a "heredoc" and format method to write a Slurm script

```
>>> script = '''#!/bin/bash
... #SBATCH -p {partition}
... #SBATCH -t {time}
... #SBATCH --mem {mem}
... #SBATCH -n {cores}
... #SBATCH -N {nodes}
...
... {cmd}
... '''.format(partition='gpu',time='100',mem='500',cores='1',nodes='1',cmd='hostname')
>>> print script
#!/bin/bash
#SBATCH -p serial_requeue
#SBATCH -t 1-0:00
#SBATCH --mem 1000
#SBATCH -n 1
#SBATCH -N 1

hostname

>>>
```

- Use a subprocess to submit and monitor your job
- Catch the job id from sbatch output

```
>>> from subprocess import Popen,PIPE
>>> def submit(filename):
...     proc = Popen('sbatch %s' % filename,shell=True,stdout=PIPE,stderr=PIPE)
...     stdout,stderr = proc.communicate()
...     return stdout.strip('Submitted batch job ')
...
>>>
```

- and use it to check sacct

```
from subprocess import Popen,PIPE
def isDone(jobid):
    dones = ['COMPLETED','CANCELLED','FAILED','TIMEOUT','PREEMPTED','NODE_FAIL']
    proc = Popen('sacct --format state --noheader -j %d' % int(jobid),shell=True,stdout=PIPE,stderr=
    stdout,stderr = proc.communicate()
    if proc.returncode != 0:
        raise Exception('Error running sacct: %s' % stderr)
    if stdout.strip() == '':
        return False
    lines = stdout.split()
    if lines[0].strip() in dones:
        return True
    return False
```

Add sequence length and base counts

- Print out base frequencies and sequence length for each sequence

```
>>> print seqs[0]
('HWUSI-EAS300R_0005_FC62TL2AAXX:8:30:18447:12115#0/1',
 'CGTAGCTGTGTGTACAAGGCCCGGGAACGTATTCACCGTG',
 'acdd^aa_Z^d^ddc`^_Q_aaa`_ddc\\dfdffff\\fff')
```

```
Sequence 1 Length: 106 A: 4, T: 4, C: 4, G: 4
```

Lists and tuples

- 0 indexed list of data items that is either modifiable (lists) or unmodifiable (tuples)

```
>>> bases = ['A','T','C','G']
>>> bases[1]
'T'
>>> bases.append('U')
>>> bases[4]
'U'
>>> bases = ('A','T','C','G')
>>> bases[1]
'T'
>>> bases.append('U')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
```

Lists and tuples

- Iteration

```
for base in bases:
    print base

for i, base in enumerate(bases):
    print base
```

- Indexing

```
>>> bases = ['A','T','C','G']
>>> print bases[1:2]
['T']
>>> print bases[1:3]
['T', 'C']
>>> print bases[-1:]
['G']
```

Lists and tuples

- Concatenating

```
allbases = dnabases + rnabases
```

- Counting

```
 >>> bases
 ['A', 'T', 'C', 'G']
 >>> len(bases)
 4
 >>> bases.count('A')
 1
```

- Short hand list initialization by another iterable (list comprehension)

```
baselengths = [len(base) for base in bases]

complements = [dna.complement(base) for base in bases]
```

Strings

- Strings are lists of characters ...

```
>>> contig = 'ATCACTAGTCGTCG'
>>> contig[1:3]
'TC'
```

- ... that can be constructed with Python formatting tools

```
>>> reagent = 'SDS'
>>> 'You will need %.2f mg of %s in %d mL' % (.565,reagent,100)
'You will need 0.56 mg of SDS in 100 mL'
>>> 'You will need {reagentmass:.2f} of {reagent} in {volume} mL'.format(
    reagentmass=0.565,
    reagent='SDS',
    volume=100
 )
'You will need 0.56 of SDS in 100 mL'
```

Add sequence length and base counts

Sequence length and base count

```
# >>> seqs[0]
# ('HWUSI-EAS300R_0005_F2AAXX:8:30:18447:12115#0/1\n', 'CGTAGCTAACGTATTCACCGTG', '')
for i,seqdata in enumerate(seqs):
    seqstr = seqdata[1]
    seqlen = len(seqstr)

    basecountline = 'Sequence %d Length: %d ' % (i,seqlen)
    for base in ['A','T','C','G']:
        basecountline += '%s: %d ' % (base,seqstr.count(base))
    print basecountline
```

or

```
    basecountstrs = ['Sequence %d Length: %d' % (i,seqlen)]
    for base in ['A','T','C','G']:
        basecountstrs.append('%s: %d' % (base,seqstr.count(base)))
    print ' '.join(basecountstrs)
```

Capture stdout and parse date information

Regular expressions

- Google: python regular expressions
- Python regular expressions are a full set of processing options (character classes, capture groups, quantifiers, etc)
- Match the beginning of your string. Use a "raw" string to avoid backslash proliferation

```
>>> teststr = 'w00t!'
>>> import re
>>> re.match(r'[a-z]\d+.*',teststr)
<_sre.SRE_Match object at 0x7f0e518c3098>
```

- Use `re.search` if your pattern is later in the string

```
>>> re.match(r'\d+.*',teststr)
>>> re.search(r'\d+.*',teststr)
<_sre.SRE_Match object at 0x7f0e518c3098>
```

Regular expressions

- Use parens to "capture" text

```
>>> segment = 'TATGCGGCAAGTTACAAAAAAAAAAAAAAAATAAAGTTAAAAAAAAAAAAATGCTA'
>>> re.findall(r'(A{3,})T',segment)
['AAAAAAAAAAAAAAAA', 'AAAAAAAAAAAAA']
```

- Split with a regex (with or without capture group)

```
>>> re.split(r'(A{3,})T',segment)
['TATGCGGCAAGTTAC', 'AAAAAAAAAAAAAAAA', 'AAAGTT', 'AAAAAAAAAAAAA', 'GCTA']
>>> re.split(r'A{3,}',segment)
['TATGCGGCAAGTTAC', 'T', 'GTT', 'TGCTA']
```

- Process multiline text

```
>>> fasta = '''
... > transcript_1
... ATCGATCGATTACGTACAAAAAAAAATACGTAGCTAAAAAAAAATCAGCTACG
... AAAAAAAAAAAAAAAAAAACTAGTCGATGCTAGCTATCGATCGTATATATGAC
... '''
>>> re.findall(r'A{3,}',fasta)
['AAAAAAAAA', 'AAAAAAAAA', 'AAAAAAAAAAAAAAAAAAA']
```

Date handling

- Google python datetime
- The `datetime` and `timedelta` modules come with Python

```
>>> from datetime import datetime,timedelta
>>> datetime.now()
datetime.datetime(2017, 3, 16, 16, 52, 33, 639252)
>>> feb = datetime(2017,2,1)
>>> nextmonth = feb + timedelta(days=30)
>>> nextmonth
datetime.datetime(2017, 3, 3, 0, 0)
```

- `strftime` formats date objects

```
>>> nextmonth.strftime('%d/%m/%Y')
'03/03/2017'
```

- `strptime` parses dates according to a strict specification

```
>>> datetime.strptime('03/03/2017','%d/%m/%Y')
datetime.datetime(2017, 3, 3, 0, 0)
```

- `python-dateutil` package parses whatever you throw at it

```
>>> from dateutil import parser
>>> parser.parse('03/03/2017')
datetime.datetime(2017, 3, 3, 0, 0)

>>> parser.parse('March 3, 2017')
datetime.datetime(2017, 3, 3, 0, 0)
```

Get the start and end dates from the hyperAssembler output and calculate the time

```
Assembling genome in data/example.fa
Start time: 04:01:00 PM
280
140
Finished assembling data/example.fa.  Writing contigs into data/example.fa.contigs.
End time: 04:01:05 PM
```

Get the start and end dates

```
# Get the start and end time from stdout
from dateutil import parser
match = re.search(r'Start time: (.*)\n', stdoutstr, re.MULTILINE)
if match:
    starttime = parser.parse(match.group(1))
match = re.search(r'End time: (.*)\n', stdoutstr, re.MULTILINE)
if match:
    endtime = parser.parse(match.group(1))
if starttime and endtime:
    delta = endtime - starttime
    print 'Elapsed assembly time %d seconds' % delta.total_seconds()
```