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, objects)
- 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

```
$ tar xvf /n/informatics/coursestuff/practical-python/stuff.tar.gz
```

- Check python

```
$ python --version
Python 2.6.6
$ which python
/usr/bin/python
```
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
2. report some information about the sequences
3. write to FASTA and feed to an assembler to create contigs
4. annotate the contigs

**ha/annotate.py**

Annotation module that will be called by `hisnhers.py` serially and, then, in parallel
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/hisnkers.py
• Script permissions should be executable

    [akitzmiller@holya2a python-workshop] $ chmod +x bin/hisnhers.py

• Flexible interpreter path in the shebang

    #!/usr/bin/env python

• Indents must match - 4 spaces, do not use tabs

• Use a proper return value for modules named __main__

    if __name__ == "__main__":
        sys.exit(main())

• import sys
Google Interlude: Magic Variables, Magic Functions

- Meta data about a python module or package
- Double underbar (dunder) designation, e.g. \_\_name\_, \_\_ispkg\_
- **Google: python magic variables**
- Python objects also have magic functions that allow you to override basic behaviors (e.g. \_\_str\_\_)
**imports**

- A name (function, variable, module, etc) can't 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

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

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

- or you can import something from a module

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

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

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

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

```python
from ha.annotate import annotateStartStopCodon
```
**imports**

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

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

[akitzmiller@holy2a ~]$ python
Python 2.6.6 (r266:84292, Jan 22 2014, 09:42:36)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-4)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> print sys.path
... "/usr/lib64/python2.6/site-packages/webkit-1.0", "/usr/lib/python2.6/site-packages", "/usr/lib/python2.6/site-packages/setuptools-0.6c11-py2.6.egg-info"]
```

- Watch out for `~/.local`
**os and sys modules**

- **os** includes functions that vary between operating systems

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

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

- **sys** includes functions and data about the Python interpreter
  - **sys.exit()** exits the Python interpreter
  - **sys.argv** contains the arguments passed to the script
Fix the file reading 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
File reading error solution

```python
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 %f does not exist' % fqfilename)

with open(fqfilename, 'r') as f:
    seqs = fastqToSequenceList(f)
```
File processing with context managers

- \( f = \text{open()} \) returns a file handle
- \textbf{with} block is a context manager that closes the file handle on exit

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

- \textbf{for} loop on a handle iterates over file lines

```python
# Code block defined by colon and indent
for line in fileh:
    line = line.strip()
    if line == '\n':
        continue
```
Convert the hardcoded file name into a command argument

```python
# if block defined by colon, indent
if len(sys.argv) < 2:
    print 'Must supply a file name'
    return 1
fqfilename = sys.argv[1]
```

or use argparse to handle real arguments

```python
from argparse import ArgumentParser, RawDescriptionHelpFormatter

parser = ArgumentParser(description='Python workshop tool', formatter_class=RawDescriptionHelpFormatter)
parser.add_argument('FASTQ_FILE', help='Fastq file')
args = parser.parse_args()

fqfilename = args.FASTQ_FILE
```
Add sequence length and base counts

- Print out base frequencies and sequence length for each sequence:

  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)

```python
>>> 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**

```python
for base in bases:
    print(base)

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

- **Indexing**

```python
>>> bases = ['A', 'T', 'C', 'G']
>>> print(bases[1:2])
['T']
>>> print(bases[1:3])
['T', 'C']
>>> print(bases[-1:])
['G']
```
Lists and tuples

- Concatenating

\[
\text{allbases} = \text{dnabases} + \text{rnabases}
\]

- Counting

\[
\text{bases} = [\text{\textquotesingle A\textquotesingle}, \text{\textquotesingle T\textquotesingle}, \text{\textquotesingle C\textquotesingle}, \text{\textquotesingle G\textquotesingle}]
\]

\[
\text{len(bases)}
\]

4

\[
\text{bases.count(\text{\textquotesingle A\textquotesingle})}
\]

1

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

\[
\text{baselengths} = [\text{len(base)} \text{ for base in bases}]
\]

\[
\text{complements} = [\text{dna.complement(base)} \text{ for base in bases}]
\]
Strings

- Strings are lists of characters ...

```python
>>> contig = 'ATCAGTCGTCG'
>>> contig[1:3]
'TC'
```

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

```python
>>> 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
)
'0.56 of SDS in 100 mL'
```
Add sequence length and base counts
Sequence length and base count

```python
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
```
Fix contigs file error

```
[akitzmiller@holy2a python-workshop]$ ./bin/hisnhers.py data/example.fq
Writing to data/example.fa
sh: line 0: fg: no job control
Traceback (most recent call last):
  File "./hisnhers.py", line 210, in <module>
    sys.exit(main())
  File "./hisnhers.py", line 135, in main
    with open(contigfilename,'r') as c:
IOError: [Errno 2] No such file or directory: 'data/example.fq.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.

```python
>>> os.system("echo 'hello' > hello.out")
0
>>> f = open('hello.out','r')
>>> print f.readlines()
['hello
']
```
Running commands with Popen()

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

```python
>>> 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

```python
>>> 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

```python
>>> 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 & &

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

- You can set environment values in the parent

```python
>>> 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

```python
>>> 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)
```
Replace the call to megaAssembler with a Popen-based call to hyperAssembler.

Capture return code, stdout, and stderr
Call to hyperAssembler

```python
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
assemblerargs = [  
    'hyperAssembler',
    fafilename,
]

cmd = ' '.join(assemblerargs)
returncode, stdoutstr, stderrstr = runcmd(cmd)

if returncode != 0:
    raise Exception('Error running assembler with cmd %s
stdout: %s
stderr: %s' % (cmd, stdoutstr, stderrstr))
```

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

```python
>>> 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

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

- Use parens to "capture" text

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

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

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

- Process multiline text

```python
>>> fasta = '... > transcript_1
... ATCGATCGATTACGTAACATAACCACATACGCTAATACGCTAATCAGCTACG
... AAAAAAAAAAAAAAAAAACTAGCTAGCTAGCTATCGATCGATCGATATAGAC
... ...
>>> re.findall(r'A{3,}', fasta)
['AAAAAAA', 'AAAAAAA', 'AAAAAAAAAAAAAAAAAAAA']
```
Date handling

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

```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

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

- `strptime` parses dates according to a strict specification

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

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

```python
>>> 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 hyperAssemble 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

```python
# 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()
```
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.

```bash
[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
```

- Avoid doing this

```bash
[akitzmiller@holy2a mpi4py-2.0.0]$ python setup.py install --user
```
Python packages - pip

- pip installs directly from the huge PyPI repository and recurses 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
Collecting click>=2.0 (from Flask->Flask-Script)
  Downloading click-6.7-py2.py3-none-any.whl (71kB)
    100% |████████████████████████████████| 71kB 4.9MB/s

... 

Building wheels for collected packages: Flask-Script, itsdangerous
  Running setup.py bdist_wheel for Flask-Script ... done
  Running setup.py bdist_wheel for itsdangerous ... done
Successfully built Flask-Script itsdangerous
Installing collected packages: click, Jinja2, Werkzeug, itsdangerous, Flask, Flask-Script
Successfully installed Flask-0.12 Flask-Script-2.0.5 Jinja2-2.9.5 Werkzeug-0.12.1 click-6.7 itsdangerous
```
• Copy an entire python setup with pip

```
pip freeze > requirements.txt
pip install -r requirements.txt
```

• 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
[akitzmiller@holy2a ~]$ conda install netcdf4
Fetching package metadata .........
Solving package specifications: ...

The following NEW packages will be INSTALLED:

- h5py: 2.6.0-np111py27_2
- hdf4: 4.2.12-0
- hdf5: 1.8.17-1
- libnetcdf: 4.4.1-0
- netcdf4: 1.2.4-np111py27_0

The following packages will be UPDATED:

- astropy: 1.1.2-np110py27_0 --> 1.3-np111py27_0
- bottleneck: 1.0.0-np110py27_0 --> 1.2.0-np111py27_0
- curl: 7.45.0-0 --> 7.49.0-1
- llvmmlite: 0.9.0-py27_0 --> 0.16.0-py27_0
- matplotlib: 1.5.1-np110py27_0 --> 1.5.1-np111py27_0
- numba: 0.24.0-np110py27_0 --> 0.31.0-np111py27_0
- numexpr: 2.5-np110py27_0 --> 2.5.2-np111py27_0
- numpy: 1.10.4-py27_1 --> 1.11.0-py27_0
- pandas: 0.18.0-np110py27_0 --> 0.19.2-np111py27_1
- patsy: 0.4.0-np110py27_0 --> 0.4.1-py27_0
- pycurl: 7.19.5.3-py27_0 --> 7.43.0-py27_0
- pytables: 3.2.2-np110py27_1 --> 3.3.0-np111py27_0
- scikit-image: 0.12.3-np110py27_0 --> 0.12.3-np111py27_1
- scikit-learn: 0.17.1-np110py27_0 --> 0.17.1-np111py27_0
- scipy: 0.17.0-np110py27_2 --> 0.17.0-np111py27_2
- statsmodels: 0.6.1-np110py27_0 --> 0.8.0-np111py27_0

Proceed ([y]/n)?
Anaconda

- or a specific version

```
[akitzmiller@holy2a ~]$ conda install netcdf4==1.2.1
Fetching package metadata ..........  
Solving package specifications: .

Package plan for installation in environment /home/akitzmiller/anaconda2/envs/workshop:

The following NEW packages will be INSTALLED:

  h5py:      2.6.0-np110py27_0
  hdf5:      1.8.15.1-3
  libnetcdf: 4.3.3.1-3
  netcdf4:   1.2.1-np110py27_0

Proceed ([y]/n)?
```
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 ~]$ mkdir envs
[akitzmiller@holy2a ~]$ cd envs
[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
~/envs/workshop/bin/python
(workshop) [akitzmiller@holy2a envs]$ pip install -r workshop-requirements.txt
...
(workshop) [akitzmiller@holy2a envs]$ deactivate
[akitzmiller@holy2a envs]$ which python
/usr/bin/python
```
Anaconda virtual environments

- Make an environment (make sure pip is installed)

```bash
[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

[akitzmiller@holy2a ~]$ conda create -n new pip
The following NEW packages will be INSTALLED:

openssl: 1.0.2k-1
pip: 9.0.1-py27_1
python: 2.7.13-0
...
wheel: 0.29.0-py27_0
zlib: 1.2.8-3

Proceed ([y]/n)? y
...
#
# To activate this environment, use:
# > source activate new
#
# To deactivate this environment, use:
# > source deactivate new
#
[akitzmiller@holy2a ~]$ source activate new
(new) [akitzmiller@holy2a ~]$ which python
~/.conda/envs/new/bin/python
(new) [akitzmiller@holy2a ~]$ source deactivate
[akitzmiller@holy2a ~]$
```
Anaconda virtual environments

- Make a clone of the parent environment (may take a while) so that all base packages are included

```
[akitzmiller@holy2a ~] module load python/2.7.13-fasrc01
[akitzmiller@holy2a ~] conda create -n clone --clone $PYTHON_HOME
Using Anaconda Cloud api site https://api.anaconda.org
Fetching package metadata: ........
src_prefix: '/n/sw/fasrcsw/apps/Core/Aconda/2.5.0-fasrc01/x'
dst_prefix: '/n/home_rc/akitzmiller/.conda/envs/clone'
Packages: 163
Files: 2254
Linking packages ...
[ COMPLETE ]|############################| 100%
#
# To activate this environment, use:
# $ source activate clone
#
# 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

```
(clone)[akitzmiller@holy2a ~] conda install Django --yes
Using Anaconda Cloud api site https://api.anaconda.org

The following packages will be downloaded:

```
<table>
<thead>
<tr>
<th>package</th>
<th>build</th>
</tr>
</thead>
<tbody>
<tr>
<td>ca-certificates-2017.1.23</td>
<td>0</td>
</tr>
<tr>
<td>django-1.10.6</td>
<td>py27_0</td>
</tr>
</tbody>
</table>

Total: 4.7 MB
```

With some of our Anacondas, you may need to do this:

```
(clone)[akitzmiller@holy2a ~] conda remove conda-env conda-build --yes
```
• Or from a particular conda channel

```
(clone)[akitzmiller@holy2a ~] conda install --channel conda-forge tensorflow
Using Anaconda Cloud api site https://api.anaconda.org

The following NEW packages will be INSTALLED:

- ca-certificates: 2017.1.23-0
- mkl: 11.3.3-0
- mock: 2.0.0-py27_0
- numpy: 1.11.2-py27_0
- pbr: 1.10.0-py27_0
- protobuf: 3.2.0-py27_0
- scipy: 0.18.1-np111py27_0
- tensorflow: 1.0.0-py27_0

The following packages will be UPDATED:

- numexpr: 2.5.2-np111py27_nomkl_1 [nomkl] --> 2.5.2-np111py27_1
- python: 2.7.11-0 --> 2.7.13-0
- scikit-learn: 0.17.1-np111py27_nomkl_1 [nomkl] --> 0.17.1-np111py27_1
- sqlite: 3.9.2-0 --> 3.13.0-0

Proceed ([y]/n)? n
```
• Or do a pip install

```
(clone)[akitzmiller@holy2a ~] pip install BioPython
Collecting BioPython
  Downloading biopython-1.68.tar.gz (14.4MB)
     100% |████████████████████████████████| 14.4MB 24kB/s
Building wheels for collected packages: BioPython
  Running setup.py bdist_wheel for BioPython ... done
  Stored in directory: /n/home_rc/akitzmiller/.cache/pip/wheels/a7/40/7e/cf0e1056601c97bbf42acac9cb
Successfully built BioPython
Installing collected packages: BioPython
Successfully installed BioPython-1.68
(clone)[akitzmiller@holy2a ~]
```
Compiled code in conda packages can be a problem

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

Using Anaconda Cloud api site https://api.anaconda.org

The following NEW packages will be INSTALLED:

...  
numpy:  1.11.2-py27_0  
pbr:  1.10.0-py27_0  
protobuf:  3.2.0-py27_0  
scipy:  0.18.1-np111py27_0  
tensorflow:  1.0.0-py27_0

The following packages will be UPDATED:

...

Unlinking packages ...

[ COMPLETE ]|###########################| 100%
Linking packages ...

[ COMPLETE ]|###########################| 100%

(clone)[akitzmiller@holy2a ~] python
Python 2.7.13 |Anaconda 2.5.0 (64-bit)| (default, Dec 20 2016, 23:09:15)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2

Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io thanks and https://anaconda.org

>>> import tensorflow as tf

Traceback (most recent call last):
...

File "/n/home_rc/akitzmiller/.conda/envs/clone/lib/python2.7/site-packages/tensorflow/python/pywrap_tensorflow.py", line 14, in _mod
    _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 compiles source code, which may not be better.

```
(clone)[akitzmiller@holy2a ~] pip install gattlib
Collecting gattlib
  Downloading gattlib-0.20150805.tar.gz (1.7MB)
    100% 1.7/1.7 (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 setup tools; tokenize; _file_ = '/scratch/tmp/pip-bld...')(_file_).read().replace(' \rn', ' \n'), _file_, 'exec')"
    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 -Icc1plus: 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
```
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

```python
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

```python
>>> from multiprocessing import Pool
>>> import os
>>> def echo(echoable):
...     os.system('echo %s && sleep 10' % echoable)
...
>>> echoables = ['ajk', '123', 'qwerty', 'uiop', 'lkjdsa', 'uiop

>>> numprocs = os.environ.get('NUMPROCS',3)
>>> pool = Pool(numprocs)
>>> result = pool.map(echo,echoables)
123
ajk
qwerty
lkjdsa
uiop
Analyze the contigs using a multiprocessing pool. Compare the elapsed time with the for loop version.
Python dictionaries are your friend

- A dictionary is like a list, but can be indexed by non-integers (AKA hash map)
- Element order is random
Let Python write JSON for you
Python can be used to submit Slurm jobs

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

```python
>>> 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