Python is an interpreted programming language that has become increasingly popular in high-performance computing environments because it's available with an assortment of numerical and scientific computing libraries (NumPy, SciPy, Pandas, etc.), relatively easy to learn, open source, and free.

1 Versions of Python on the ACCRE Cluster

When a user initially logs into the cluster, the system version (Python that comes pre-installed on the operating system) is 2.7.6 and is located in `/usr/local/bin`:

```
[bob@vmps14 ~]$ which python
/usr/local/bin/python
[bob@vmps14 ~]$ python --version
Python 2.7.6
```

This version of Python may be sufficient for simple tasks, but it does not include many of the essential numerical libraries cluster users need. ACCRE administrators have hand-compiled multiple other versions of Python that are linked against highly optimized linear algebras like ATLAS and Intel's MKL, and therefore will in general yield better performance (faster execution time). To see a list of installed versions of Python on the cluster, use `pkginfo`

```
[bob@vmps14 ~]$ pkginfo | grep "^\s*python"
python              Python (2.7.2)
python2             Python (2.7.2)
python2.7.4         Python (2.7.4)
python2.7.8         Python (2.7.8)
python2.7.8_intel14 Python (2.7.8, Intel-compiled + Intel MKL)
python3             Python (3.2.2)
python3.4.2         Python (3.4.2)
python_xnat         python_xnat for NI research (2.7.3)
```

Multiple versions of Python 2 and Python 3 are available on the cluster. We encourage users to use the most recent versions of Python 2 (2.7.8) and Python 3 (3.4.2) installed. We also have a version of Python 2.7.8 compiled with Intel compilers and linked against Intel's MKL library, which should yield better performance on our Intel processors. To load a particular version of Python, use `setpkgs`:

```
[bob@vmps14 ~]$ setpkgs -a python3.4.2
[bob@vmps14 ~]$ python --version
Python 3.4.2
[bob@vmps65 ~]$ which python
/usr/local/python3/3.4.2/x86_64/gcc46/nonet/bin/python
```

Two distributions of Anaconda, are also available on the cluster:

```
[bob@vmps14 ~]$ pkginfo | grep "conda"
                     anaconda2   Anaconda2 Python (2.7.8)
                     anaconda3   Anaconda3 Python (3.5.1)
```

Anaconda simplifies managing dependencies and will be discussed in Section 4.

2 Checking Installed Packages

Each different version of Python has its own packages installed into it, so as you are switching between different versions it is prudent to check the packages that are available. One way to do this is using `pip`:

```
[jill@vmps14 ~]$ setpkgs -a python2.7.8
[jill@vmps14 ~]$ pip freeze | sort
alignlib-lite==0.2.3
appdirs==1.4.0
backports.ssl-match-hostname==3.4.0.2
bcbio-gff==0.6.2
BeautifulSoup==3.2.1
beautifulsoup4==4.3.2
biopython==1.65
bokeh==0.7.1
brewer2mpl==1.4.1
bx-python==0.7.1
certifi==14.5.14
CGAT==0.2.3
CGATReport==0.2
click==3.3
colorama==0.3.3
Cython==0.21.2
decorator==3.4.0
deepTools==1.5.9.1
docutils==0.12
drmaa==0.7.6
enum34==1.0.4
ete2==2.2.1072
Flask==0.10.1
freetype-py==1.0
```

```
future==0.14.3
futures==2.2.0
geojson==1.0.6
geopy==1.8.1
gevent==1.0.1
gevent-websocket==0.9.3
ggplot==0.6.5
greenlet==0.4.5
hgapi==1.7.2
html5lib==0.999
ipython==3.0.0
isodate==0.5.1
itsdangerous==0.24
jdcal==1.0
Jinja2==2.7.3
lxml==3.4.2
Mako==1.0.1
Markdown==2.5.2
MarkupSafe==0.23
matplotlib==1.5.dev1
matplotlib-venn==0.10
mock==1.0.1
more-itertools==2.2
mpi4py==1.3.1
mpld3==0.2
MySQL-python==1.2.5
natsort==3.5.6
networkx==1.9.1
nose==1.3.6
numpy==1.9.1
oncodriveclust==0.5.0
Oncotator==1.5.1.0
openpyxl==2.1.4
pandas==0.15.2
parse==1.4.1
patsy==0.3.0
pdb==0.1
pep8==1.5.7
psycopg2==2.5.4
py==1.4.26
pybedtools==0.6.9
pycuda==2014.1
Pygments==2.0.1
pyparsing==2.0.3
pysam==0.7.5
pysqlite==2.6.3
```

```
pystache==0.5.4
pytest==2.7.0
python-dateutil==2.4.2
python-gnupg==0.3.7
python-memcached==1.54
pytools==2014.3.5
pytz==2015.2
PyVCF==0.6.7
PyYAML==3.11
pyzmq==14.4.1
rdflib==4.1.2
requests==2.5.1
rpy2==2.4.4
ruffus==2.5
scikit-bio==0.2.3
scikit-learn==0.15.2
scimath==4.1.2
scipy==0.14.1
seaborn==0.4.dev0
Shapely==1.5.6
shove==0.5.6
six==1.9.0
SPARQLWrapper==1.6.4
Sphinx==1.2.3
sphinxcontrib-programoutput==0.8
SQLAlchemy==0.9.8
statsmodels==0.6.1
stuf==0.9.4
threadpool==1.2.7
tornado==4.0.2
traits==4.5.0
weblogo==3.4
web.py==0.37
websocket==0.2.1
Werkzeug==0.9.6
xlwt==0.7.5
```

Using `pip` is convenient because it gives you package version information and also lists any packages you have installed locally (in your home directory). Occasionally you may also want to see a list of Python modules, which you can see by using the `python\_pkginfo.py` command:

```
[jill@vmps14 ~]$ python_pkginfo.py --type both

For output options type 'python_pkginfo.py -h'

Installed Packages and Modules for Python version 2.7.8
```

```
[GCC 4.6.1]:

To see package versions try: pip freeze | sort
after issuing the appropriate "setpkgs -a python" command

abc (Module)                _multiprocessing (Module)
_abcoll (Module)            multiprocessing (Package)
aifc (Module)               mutex (Module)
alignlib_lite (Module)      _mysql (Module)
antigravity (Module)        _mysql_exceptions (Module)
anydbm (Module)             MySQLdb (Package)
appdirs (Module)            natsort (Package)
argparse (Module)           netrc (Module)
array (Module)              networkx (Package)
ast (Module)                new (Module)
asynchat (Module)           nis (Module)
asyncore (Module)           nntplib (Module)
atexit (Module)             nose (Package)
audiodev (Module)           ntpath (Module)
audioop (Module)            nturl2path (Module)
backports (Package)         numbers (Module)
base64 (Module)             numpy (Package)
BaseHTTPServer (Module)     oncodriveclust (Package)
Bastion (Module)            oncotator (Package)
BCBio (Package)             opcode (Module)
bdb (Module)                openpyxl (Package)
BeautifulSoup (Module)      operator (Module)
BeautifulSoupTests (Module) optparse (Module)
binascii (Module)           os (Module)
binhex (Module)             os2emxpath (Module)
Bio (Package)               ossaudiodev (Module)
BioSQL (Package)            _osx_support (Module)
bisect (Module)             pandas (Package)
_bisect (Module)            parse (Module)
bokeh (Package)             parser (Module)
brewer2mpl (Package)        past (Package)
bs4 (Package)               patsy (Package)
_bsddb (Module)             pdb (Module)
bsddb (Package)             pep8 (Module)
builtins (Package)          pickle (Module)
bx (Package)                pickletools (Module)
bx_extras (Package)         pip (Package)
bz2 (Module)                pipes (Module)
calendar (Module)           pkg_resources (Package)
Canvas (Module)             pkgutil (Module)
CDROM (Module)              platform (Module)
```

```
certifi (Package)           plistlib (Module)
CGAT (Package)              popen2 (Module)
CGATReport (Package)        poplib (Module)
CGATReportPlugins (Package) posixfile (Module)
CGATScripts (Package)       posixpath (Module)
cgi (Module)                pprint (Module)
CGIHTTPServer (Module)      profile (Module)
cgitb (Module)              pstats (Module)
chunk (Module)              psyco_full (Module)
click (Package)             psycopg2 (Package)
cmath (Module)              pty (Module)
cmd (Module)                py (Package)
code (Module)               py_compile (Module)
codecs (Module)             pybedtools (Package)
_codecs_cn (Module)         pyclbr (Module)
_codecs_hk (Module)         pycuda (Package)
_codecs_iso2022 (Module)    pydoc (Module)
_codecs_jp (Module)         pydoc_data (Package)
_codecs_kr (Module)         pyexpat (Module)
_codecs_tw (Module)         pygments (Package)
codeop (Module)             _pyio (Module)
collections (Module)        pylab (Module)
_collections (Module)       pyparsing (Module)
colorama (Package)          pysam (Package)
colorsys (Module)           pysqlite2 (Package)
commands (Module)           pystache (Package)
compileall (Module)         pytest (Module)
compiler (Package)          _pytest (Package)
concurrent (Package)        python_pkginfo (Module)
ConfigParser (Module)       pytools (Package)
configparser (Package)      pytz (Package)
contextlib (Module)         pyximport (Package)
Cookie (Module)             Queue (Module)
cookielib (Module)          queue (Package)
copy (Module)               quopri (Module)
copy_reg (Module)           random (Module)
copyreg (Package)           _random (Module)
corebio (Package)           rdflib (Package)
cPickle (Module)            re (Module)
cProfile (Module)           readline (Module)
crypt (Module)              repr (Module)
cStringIO (Module)          reprlib (Package)
csv (Module)                requests (Package)
_csv (Module)               resource (Module)
_ctypes (Module)            rexec (Module)
ctypes (Package)            rfc822 (Module)
```

```
_ctypes_test (Module)       rlcompleter (Module)
_curses (Module)            robotparser (Module)
curses (Package)            rpy2 (Package)
_curses_panel (Module)      ruffus (Package)
cython (Module)             runpy (Module)
Cython (Package)            sched (Module)
datetime (Module)           scimath (Package)
dateutil (Package)          scipy (Package)
dbhash (Module)             ScrolledText (Module)
dbm (Module)                seaborn (Package)
decimal (Module)            select (Module)
decorator (Module)          sets (Module)
deeptools (Package)         setuptools (Package)
Dialog (Module)             sgmllib (Module)
difflib (Module)            sha (Module)
dircache (Module)           shapely (Package)
dis (Module)                shelve (Module)
distutils (Package)         shlex (Module)
DLFCN (Module)              shove (Package)
doctest (Module)            shutil (Module)
docutils (Package)          SimpleDialog (Module)
DocXMLRPCServer (Module)    SimpleHTTPServer (Module)
drmaa (Package)             SimpleXMLRPCServer (Module)
dumbdbm (Module)            site (Module)
dummy_thread (Module)       six (Module)
_dummy_thread (Package)     skbio (Package)
dummy_threading (Module)    sklearn (Package)
easy_install (Module)       smtpd (Module)
_elementtree (Module)       smtplib (Module)
email (Package)             sndhdr (Module)
encodings (Package)         socket (Module)
enum (Package)              _socket (Module)
ete2 (Package)              SocketServer (Module)
fcntl (Module)              socketserver (Package)
filecmp (Module)            SPARQLWrapper (Package)
FileDialog (Module)         sphinx (Package)
fileinput (Module)          spwd (Module)
FixTk (Module)              sqlalchemy (Package)
flask (Package)             _sqlite3 (Module)
fnget (Module)              sqlite3 (Package)
fnmatch (Module)            sre (Module)
formatter (Module)          sre_compile (Module)
fpformat (Module)           sre_constants (Module)
fractions (Module)          sre_parse (Module)
freetype (Package)          ssl (Module)
ftplib (Module)             _ssl (Module)
```

```
functools (Module)          stat (Module)
_functools (Module)         statsmodels (Package)
future (Package)            statvfs (Module)
__future__ (Module)         string (Module)
future_builtins (Module)    StringIO (Module)
futures (Package)           stringold (Module)
gdbm (Module)               stringprep (Module)
genericpath (Module)        strop (Module)
geojson (Package)           _strptime (Module)
geopy (Package)             struct (Module)
getopt (Module)             _struct (Module)
getpass (Module)            stuf (Package)
gettext (Module)            subprocess (Module)
gevent (Package)            sunau (Module)
geventwebsocket (Package)   sunaudio (Module)
ggplot (Package)            symbol (Module)
glob (Module)               symtable (Module)
gnupg (Module)              sysconfig (Module)
greenlet (Module)           _sysconfigdata (Module)
grp (Module)                syslog (Module)
gzip (Module)               tabnanny (Module)
hashlib (Module)            tarfile (Module)
_hashlib (Module)           telnetlib (Module)
heapq (Module)              tempfile (Module)
_heapq (Module)             termios (Module)
hgapi (Package)             test (Package)
hmac (Module)               _testcapi (Module)
_hotshot (Module)           tests (Package)
hotshot (Package)           textwrap (Module)
html (Package)              this (Module)
html5lib (Package)          _thread (Package)
htmlentitydefs (Module)     threading (Module)
htmllib (Module)            _threading_local (Module)
HTMLParser (Module)         threadpool (Module)
http (Package)              time (Module)
httplib (Module)            timeit (Module)
idlelib (Package)           Tix (Module)
ihooks (Module)             tkColorChooser (Module)
imaplib (Module)            tkCommonDialog (Module)
imghdr (Module)             Tkconstants (Module)
importlib (Package)         Tkdnd (Module)
imputil (Module)            tkFileDialog (Module)
IN (Module)                 tkFont (Module)
inspect (Module)            Tkinter (Module)
io (Module)                 _tkinter (Module)
_io (Module)                tkinter (Package)
```

```
IPython (Package)           tkMessageBox (Module)
isodate (Package)           tkSimpleDialog (Module)
itertools (Module)          toaiff (Module)
itsdangerous (Module)       token (Module)
jdcal (Module)              tokenize (Module)
jinja2 (Package)            tornado (Package)
_json (Module)              trace (Module)
json (Package)              traceback (Module)
keyword (Module)            traits (Package)
lib2to3 (Package)           ttk (Module)
libfuturize (Package)       tty (Module)
libpasteurize (Package)     turtle (Module)
linecache (Module)          types (Module)
linuxaudiodev (Module)      TYPES (Module)
locale (Module)             unicodedata (Module)
_locale (Module)            unittest (Package)
logging (Package)           urllib (Module)
lsblock (Module)            urllib2 (Module)
_lsprof (Module)            urlparse (Module)
_LWPCookieJar (Module)      user (Module)
lxml (Package)              UserDict (Module)
macpath (Module)            UserList (Module)
macurl2path (Module)        UserString (Module)
mailbox (Module)            uu (Module)
mailcap (Module)            uuid (Module)
mako (Package)              vcf (Package)
markdown (Package)          warnings (Module)
_markerlib (Package)        wave (Module)
markupbase (Module)         weakref (Module)
_markupbase (Package)       _weakrefset (Module)
markupsafe (Package)        web (Package)
math (Module)               webbrowser (Module)
matplotlib (Package)        weblogolib (Package)
matplotlib_venn (Package)   websocket (Package)
md5 (Module)                werkzeug (Package)
memcache (Module)           whichdb (Module)
mhlib (Module)              winreg (Package)
mimetools (Module)          wsgiref (Package)
mimetypes (Module)          xdrlib (Module)
MimeWriter (Module)         xlwt (Package)
mimify (Module)             xml (Package)
mmap (Module)               xmllib (Module)
mock (Module)               xmlrpc (Package)
modulefinder (Module)       xmlrpclib (Module)
more_itertools (Package)    _yaml (Module)
_MozillaCookieJar (Module)  yaml (Package)
```

```
mpi4py (Package)            zipfile (Module)
mpl_toolkits (Package)      zlib (Module)
mpld3 (Package)             zmq (Package)
_multibytecodec (Module)
multifile (Module)
```

python_pkginfo.py will also list any packages/modules you have installed locally in your home directory.

3 Installing New Packages

If you find that a particular package you need is missing from one of the Python versions, you have a few options. If you believe the package will be widely used by other cluster users, you can open a helpdesk ticket and request that we install the package for cluster-wise access. The other option is to install the package yourself into your home directory. There are multiple ways to install Python packages (pip, easy_install, from source). Just make sure you have the appropriate version of Python (via ***setpkgs -a***) in your environment when perform the installation. To install a package (e.g. a packaged called **word-count**) into your home directory with *pip*:

```
[jill@vmps14 ~]$ pip install word-count --user
Collecting word-count
  Downloading word_count-0.1.0-py2.py3-none-any.whl
Installing collected packages: word-count
Successfully installed word-count-0.1.0
[jill@vmps14 ~]$ ls ~/.local/lib/python2.7/site-packages/ | grep word_count
word_count-0.1.0.dist-info
word_count.py
word_count.pyc
[jill@vmps14 ~]$ pip freeze | grep word-count
word-count==0.1.0
```

Alternatively, you can build and install a package from source code into your home directory. See our FAQ page for instructions on doing this. Note that it is generally better practice to use Anaconda virtual environments and install new packages into the virtual environment (see instructions below). This helps isolate packages and to specific projects and prevents complex dependency problems when, for example, you need a different version of a package for two different projects.

4 Example Scripts

Running a Python script within a SLURM job is generally straightforward. Unless you are attempting to run one of Python's multi-processing packages, you will want to request a single task, load the appropriate version of Python from your SLURM script, and then redirect your Python file to the Python interpreter. The following example runs Python 2.7.8 on a simple Python script demonstrating the utility of writing vectorized Python code:

```
[bob@vmps14 run1]$ ls
python.slurm  vectorization.py
```

```
[bob@vmps14 run1]$ cat python.slurm
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --time=00:10:00
#SBATCH --mem=500M
#SBATCH --output=python_job_slurm.out

setpkgs -a python2.7.8

python < vectorization.py
```

```
[bob@vmps14 run1]$ cat vectorization.py
#!/usr/bin/env python
#
# Python 2.7 script demonstrating vectorized execution
#
import numpy as np
import time

# 10 million entries
t = np.linspace(-10,10,10000000)
x1 = np.zeros(len(t))
x2 = np.zeros(len(t))

time1 = time.clock()
# naive, non-vectorized implementation
for i,ti in enumerate(t):
    x1[i] = np.sin(ti)
time2 = time.clock()
print '%s: %0.2f seconds elapsed' % ("naive implementation", time2-time1)

# vectorized implementation
time1 = time.clock()
x2 = np.sin(t)
time2 = time.clock()
print '%s: %0.2f seconds elapsed' % ("vectorized implementation", time2-time1)

if ( np.array_equal(x1,x2) ):
    print "arrays equal!"
```

```
[bob@vmps14 run1]$ sbatch python.slurm
Submitted batch job 1832675
```

After waiting a few minutes:

```
[bob@vmps14 run1]$ ls
python_job_slurm.out  python.slurm  vectorization.py
```

```
[bob@vmps14 run1]$ cat python_job_slurm.out
naive implementation: 23.35 seconds elapsed
vectorized implementation: 0.68 seconds elapsed
arrays equal!
```

5 Managing Packages with Anaconda

Anaconda and its package manager "conda" aim to simplify package and dependency management. To load Anaconda, simply type

```
[bob@vmps14 ~] setpkgs -a anaconda3
```

or

```
[bob@vmps14 ~] setpkgs -a anaconda3
```

(Note that both versions of Anaconda support installing any available Python version.) Anaconda encapsulates packages into virtual environments; in this way, packages between different projects do not conflict with one another. To create a virtual environment running Python version 3.4:

```
$ conda create --name myenvironment python=3.4
Fetching package metadata .......
Solving package specifications: ..........

Package plan for installation
in environment /home/bob/.conda/envs/myenvironment:

The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    openssl-1.0.2h             |                1         3.2 MB
    xz-5.2.2                   |                0         644 KB
    python-3.4.5               |                0        15.2 MB
    ------------------------------------------------------------
                                           Total:        19.0 MB

The following NEW packages will be INSTALLED:
    openssl:    1.0.2h-1
    pip:        8.1.2-py34_0
    python:     3.4.5-0
    readline:   6.2-2
    setuptools: 26.1.1-py34_0
    sqlite:     3.13.0-0
    tk:         8.5.18-0
    wheel:      0.29.0-py34_0
    xz:         5.2.2-0
    zlib:       1.2.8-3

Proceed ([y]/n)? y

Fetching packages ...
openssl-1.0.2h 100% |######################| Time: 0:00:03   1.06 MB/s
xz-5.2.2-0.tar 100% |######################| Time: 0:00:01 561.46 kB/s
python-3.4.5-0 100% |######################| Time: 0:00:03   4.00 MB/s
Extracting packages ...
[      COMPLETE      ]|##########################| 100%
Linking packages ...
[      COMPLETE      ]|##########################| 100%
#
# To activate this environment, use:
# $ source activate myenvironment
#
# To deactivate this environment, use:
# $ source deactivate
#
```

As mentioned in the conda message, to activate the environment, type:

```
[bob@vmps14 ~] source activate myenvironment
```

Then, to view the installed packages:

```
(myenvironment) [bob@vmps14 ~]$ conda list
# packages in environment at /home/bob/.conda/envs/myenvironment:
#
openssl                   1.0.2h                        1
pip                       8.1.2                    py34_0
python                    3.4.5                         0
readline                  6.2                           2
setuptools                26.1.1                   py34_0
sqlite                    3.13.0                        0
tk                        8.5.18                        0
wheel                     0.29.0                   py34_0
xz                        5.2.2                         0
```

```
zlib                      1.2.8                        3
```

Notice that "(myenvironment)" is prepended to the bash command prompt and that all the required packages for the setting up the environment are shown. New packages can be found using the *conda search* command, and they can be added to the current environment using *conda install*, for example:

```
(myenvironment) [bob@vmps14 ~]$ conda install numpy
Fetching package metadata .......
Solving package specifications: ..........

Package plan for installation
in environment /home/bob/.conda/envs/myenvironment:

The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    numpy-1.11.1               |           py34_0         6.1 MB

The following NEW packages will be INSTALLED:

    mkl:   11.3.3-0
    numpy: 1.11.1-py34_0

Proceed ([y]/n)? y

Fetching packages ...
numpy-1.11.1-p 100% |######################| Time: 0:00:00  10.65 MB/s
Extracting packages ...
[      COMPLETE      ]|################################| 100%
Linking packages ...
[      COMPLETE      ]|################################| 100%
```

Executing the code above installed numpy version 1.11.1 for Python 3.4 as well the package it depends on, mkl version 11.3.3. To use *myenvironment* in a batch script, it is necessary to first set the Anaconda package and then activate the environment in the *.slurm* file:

```
setpkgs -a anaconda3
source activate myenvironment
```

Other useful conda commands can be found in the Conda Cheatsheet.

6 Jupyter (iPython) Notebooks

Jupyter notebooks (formerly iPython notebooks) enable a user to interactively code in Python from a web browser with support for inline plotting, equation editing, among many other things. In general, a cluster environment is used for batch processing rather than interactive processing, however we do allow users to launch notebooks from compute nodes and then do editing on a local machine via a web browser. In order to do this, first submit a batch job with a SLURM script similar to the following:

```
#!/bin/bash
#SBATCH --ntasks=1
#SBATCH --mem=2G
#SBATCH --time=0-04:00:00      # 4 hours
#SBATCH --output=notebook.out
setpkgs -a anaconda3
jupyter notebook --no-browser --ip='*' --port=7777
```

Here, a user is requesting four hours of walltime and is using Anaconda3. The important line is the last one, where a few important options must be passed to the jupyter interpreter. Make sure you include these same options in your SLURM script. If you want to launch multiple notebooks from the same compute node, you would need to use a different port (try incrementing by one). Use *squeue* to monitor when your job has started. Once it has started, write down the name of the node where it is running and open `notebook.out` to see the password for this notebook; it should look something like

```
   Copy/paste this URL into your browser when you connect for the first time,
              to login with a token:
                http://localhost:9999/?token=0gt123481a68b53a92490c64a1712f0e2af696ebe2db
```

Open a Linux terminal/shell on your local machine (lab desktop, laptop, etc.), and enter the following command:

```
ssh -L 9999:vmp506:7777 vunetid@login.accre.vanderbilt.edu
```

Be sure to replace vmp506 with the name of the node where your job is running, and *vunetid* with your VUNetID. This command binds port 9999 on localhost (i.e. your local machine) to port 7777 on vmp506. Finally, point a web browser on your local machine to `localhost:9999` (enter this where you would normally put a URL address), and you should get a Jupyter notebook login screen. Enter the token from above, and you'll launch a session in your web browser, where processing within the notebook will take place on the compute node. This ssh session must remain active in order for you to continue accessing the notebook from your local machine. You might consider adding these lines to your ~/.ssh/config file to ensure your ssh session does not time out:

```
ServerAliveInterval 60
TCPKeepAlive yes
KeepAlive yes
```

Note that your account will be charged based on the length of your job, and not the amount of CPU time used by the job.

For more tips on using Jupyter notebooks, check out ACCRE's Python repository on GitHub.

7 Contributing New Examples

In order to foster collaboration and develop local Python expertise at Vanderbilt, we encourage users to submit examples of their own to ACCRE's Python Github repository. Instructions for doing this can be found on this page. You can also find the examples from Sections 4 and 5 in the ACCRE repository.