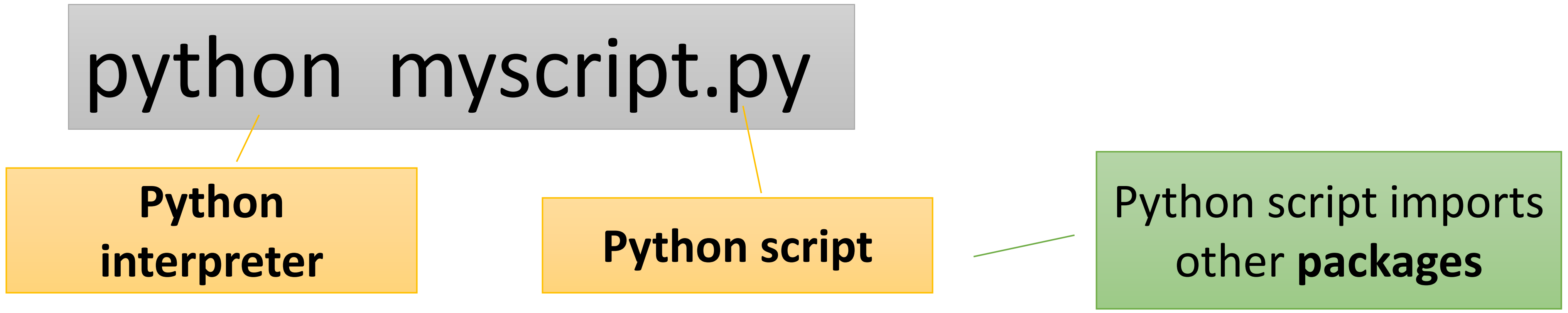
Three players when running a python script



```
python  myscript.py
```

- **Python interpreter:** software to compile/execute the script.
- **Python script:** script you wrote
- **Python packages:** python libraries called by script

Two different ways to run a python script

```
python myscript.py
```

If the script has a shebang* line, you can also run the script like this:

```
myscript.py
```

* Shebang line is the first line of a script to specify path of the interpreter, e.g. “#!/usr/bin/python3.9.6”;

** In Linux, file extension like “.py” is ignored. It is the “Shebang” line that defines the type of a script.

In Linux, it is the Shebang line that defines the script type.

Python script: bamCoverage.py

```
#!/usr/bin/python3.6

import deeptools.misc

if __name__ == "__main__":
    args = None
    if len(sys.argv) == 1:
        args = ["--help"]
    main(args)
```

In Windows, the file name extension define the script type.

In Linux, the Shebang line define the script type, whether it is a Python, R, Perl, or shell script.

Two different formats of Shebang line

#!/usr/bin/python3.6 ← Full path of the Python interpreter

#!/usr/bin/env python3 ← Default python3 on the system, as defined in $PATH.

Python interpreter

&

Python packages (libraries)

Which Python?

Multiple Python installations co-exist on the same computer. On BioHPC, we have v2.7.5, v2.7.15, v3.6.7, v3.9.6. There are more versions of Python in Conda.

How to verify which Python is being used?

```
which python

python -V
```

Alternative ways to use a different version of Python.

- Shebang line   `#!/usr/bin/python3.9.6`
- Add to PATH   `export PATH=/programs/python-3.9.6/bin:$PATH`
- Linux Module   `module load python/3.9.6`

Each Python has its own library directories, and a companion “pip” for library installation

For example:

Python	**/usr/bin/python3.6** Alias (symbolic link): /usr/local/bin/python
Pip	**/usr/bin/pip3.6** Alias (symbolic link): /usr/local/bin/pip
Packages	**/usr/lib/python3.6/ & /usr/lib64/python3.6/**

If you run “pip install”, you will get an error message “permission denied”. You need to run “pip install --user” which would install python packages under your home directory.

When running a script, Python looks for packages from three different places, and following this order. The first found is used.

Directories defined in $PYTHONPATH

- Custom location, e.g. export PYTHONPATH=/workdir/lib:$PYTHONPATH. This is independent of which “python” or which version of “python” you use.

$HOME/.local

- If you run “pip install --user packageName”, the package are installed under $HOME/.local. This is independent of which “python” you use, but different for each python version.

sys.path
e.g. /usr/lib/python3.6

- Each python installation has its own unique sys.path.

Install python software with Pip

```
pip install deepTools
```

→ sys.path
e.g. /usr/lib/python3.6

# you need write permission to the sys.path.

```
pip install deepTools --user
```

→ $HOME/.local

# packages are only accessible by the user

```
pip install deepTools  --prefix=/workdir/$USER
```

→ /workdir/$USER

#when using this library, you need to specified it in $PYTHONPATH

(Pip download software from PyPI)

Some other features of pip

1. Install a specific version of a python package

```
pip install --user deepTools==3.5.1
```

2. Upgrade a package including its dependencies to latest

```
pip install --upgrade deepTools
```

Conda

What is Conda?

- Online software repository (independent from PyPI);
- A package manager for software installation;
- **An environment manager for running software**;

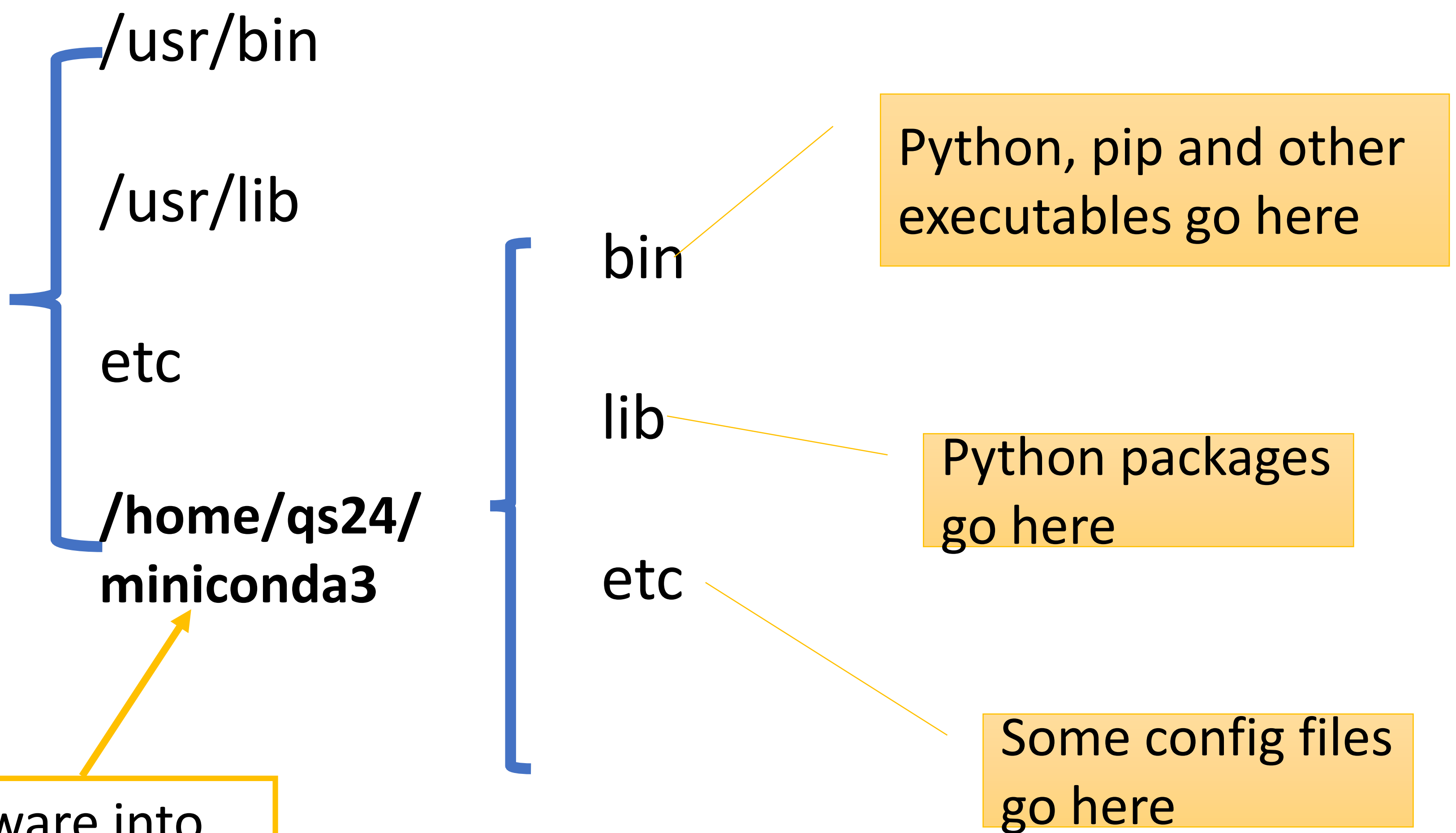
Why Conda?

Traditionally, Linux software are installed into these three directories

/usr/bin	For executables
/usr/lib	For libraries
/etc	For config files

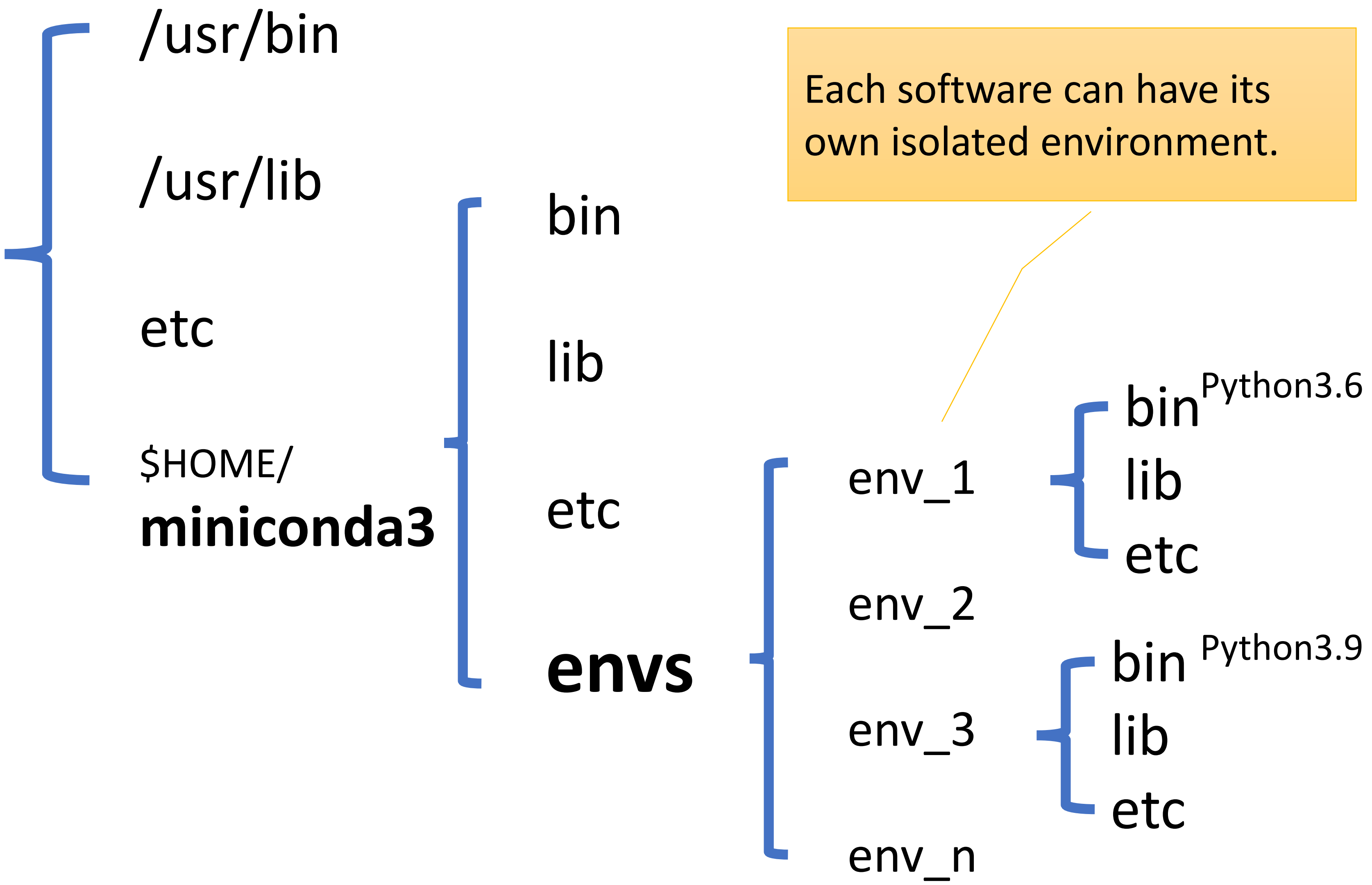
Only a system admin can install software into these directories.

Conda adds a directory where user can install software

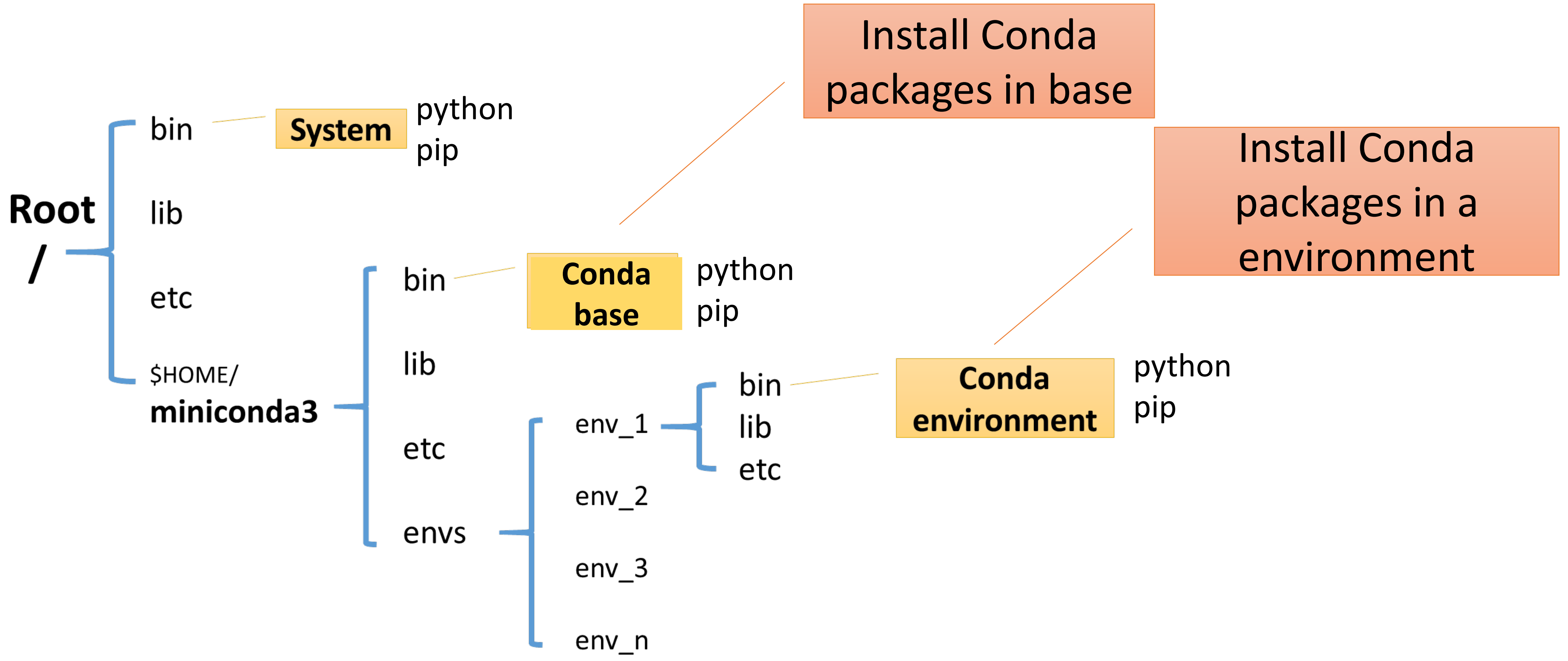


A regular user can install software into these directories.

Conda **envs** directory is a collection of multiple environments



Each Conda environment has its own python, libraries and companion pip



Install softwere in Conda base vs Conda environment

Install under Conda base:

```
conda install -c bioconda deeptools
```

Create a Conda environment and install software:

```
conda create -c bioconda -n deeptools deeptools
```

Name of Conda channel. It is the place where conda find the package

Name of the environment you will create. It can be any name.

Name of the Conda package. This name must exists in the channel.

Activate/deactivate a Conda environment

Activate

```
#activate conda base
source ~/miniconda/bin/activate
#activate an environment
conda activate busco
```

or

```
#activate conda base
source ~/miniconda/bin/activate busco
```

De-activate

```
conda deactivate
```

During Conda installation, it tries to trick you to make conda activated by default. Don’t do that!!!   If you have already done that, disable it by modifying .bashrc file.

Within a conda environment, you can run either "conda install" or "pip install".

# create and activate an environment, which only has python in it

```
conda create -n myEnv python=3.9

conda activate myEnv
```

# install deeptools in the environment

```
conda install deeptools
```

#installation through Anaconda repository

or

```
pip install deeptools
```

#installation through Pypi repository

Compatibility of software versions within a Conda environment

When depositing a software, the developer provides an installation recipe

For example, the recipe for Deeptools:

```
run:
      - deeptoolsintervals >=0.1.8
      - matplotlib-base >=3.1.0
      - numpy >=1.9.0
      - plotly >=2.0.0
      - py2bit >=0.2.0
      - pybigwig >=0.2.3
      - pysam >=0.14.0
      - python >=3
      - scipy >=0.17.0
```

When installing a software, Conda package manager reads the recipe to determine which version to download.

- Check whether a package exists in the current environment;
- Find a package available in the repository and compatible with all software within the same environment.

Conda as a package manager

conda create -n deeptools deeptools

```
The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    _openmp_mutex-4.5          |            1_gnu          22 KB
    brotli-1.0.9               |       he6710b0_2         375 KB
    deeptools-3.5.1            |             py_0         143 KB
bioconda
    deeptoolsintervals-0.1.9   |   py37h516909a_0          72 KB
bioconda
    fonttools-4.25.0           |     pyhd3eb1b0_0         632 KB
    intel-openmp-2021.3.0      |   h06a4308_3350          1.4 MB
    jpeg-9d                    |       h7f8727e_0         232 KB
    libgfortran-ng-7.5.0       |      ha8ba4b0_17          22 KB
    libgfortran4-7.5.0         |      ha8ba4b0_17         995 KB
    lz4-c-1.9.3                |       h295c915_1         185 KB
    matplotlib-base-3.4.2      |   py37hab158f2_0         5.6 MB
    mkl-2021.3.0               |     h06a4308_520       141.2 MB
    mkl-service-2.4.0          |   py37h7f8727e_0          56 KB
    mkl_fft-1.3.0              |   py37h42c9631_2         170 KB
    mkl_random-1.2.2           |   py37h51133e4_0         287 KB
    munkres-1.0.7              |             py_1          10 KB
bioconda
    numpy-1.20.3               |   py37hf144106_0          23 KB
    numpy-base-1.20.3          |   py37h74d4b33_0         4.5 MB
    openjpeg-2.4.0             |       h3ad879b_0         331 KB
    pillow-8.3.1               |   py37h2c7a002_0         635 KB
    pip-21.2.2                 |   py37h06a4308_0         1.8 MB
    py2bit-0.3.0               |   py37h14c3975_2          22 KB
bioconda
    pybigwig-0.3.17            |   py37hc013797_0          77 KB
bioconda
    scipy-1.7.1                |   py37h292c36d_2        16.4 MB
    setuptools-58.0.4          |   py37h06a4308_0         775 KB
    six-1.16.0                 |     pyhd3eb1b0_0          18 KB
    ------------------------------------------------------------
                                           Total:       175.8 MB
```

Mamba, an alternative to Conda package manager

Install mamba:

```
conda install mamba
```

Use mamba:

```
mamba install …

mamba create …
```

* Mamba is often much faster than conda and more robust.

A few tips of using Conda

Sometimes, a little intervention is needed.

For example, when “biopython” was upgraded to 1.77, it was not compatible with “hicexplorer”. In this case, you need to explicitly specify a lower biopython version.

```
conda install -c bioconda hicexplorer biopython=1.76
```

* Afterwards, hicexplorer developers noticed this problem and updated its recipe to “<1.77”

You might need to update Conda software once in a while

```
conda update conda
```

Conda channels

```
conda install -c bioconda -c conda-forge deeptools
```

* conda-forge is more comprehensive, but less strictly managed. Including conda-forge could take much longer to “solve packages”.

Troubleshooting Python

Step 1. verify which Python you are using

```
which python
```

Common errors:

1. You are using a wrong version of python;

   For example, running Python2 script with Python3. You would see this error message: SyntaxError: Missing parentheses in call to 'print'.

   To fix:
   ```
   module load python/2.7.15
   ```

2. A python module is missing, and you need to install it.

   If you are using system Python
   ```
   pip install --user theModuleName
   ```

   If you are using Python in Conda
   ```
   pip install theModuleName
   ```

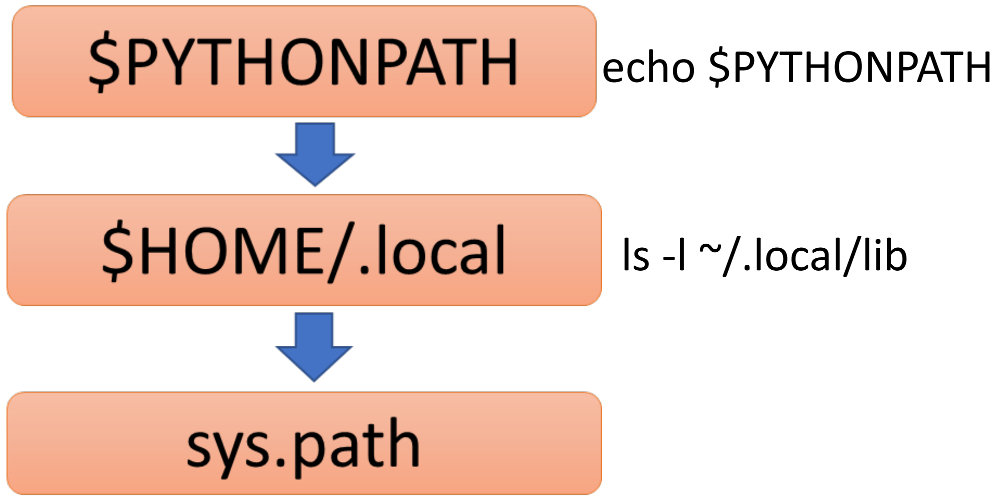
3. You are using a wrong version of Python modules. You need to re-install the right version.

   ```
   pip install theModuleName==3.12
   ```

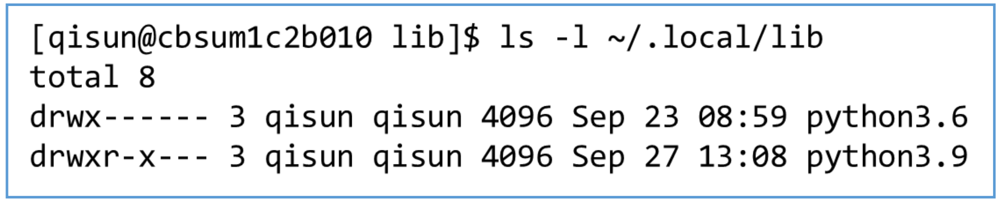
   * When running into version issue, it is better to do it within a Conda environment, to avoid interference with other software.

If you installed the right version, but still got error message. You need to verify which python module is actually being used

Python follows this order to find a library



Under $HOME/.local, libraries for different python versions are separated



```
[qisun@cbsum1c2b010 lib]$ ls -l ~/.local/lib
total 8
drwx------ 3 qisun qisun 4096 Sep 23 08:59 python3.6
drwxr-x--- 3 qisun qisun 4096 Sep 27 13:08 python3.9
```

>>> import numpy

>>> print numpy.__file__
/usr/lib64/python2.7/site-packages/numpy/__init__.pyc

>>> print numpy.__version__
1.14.3

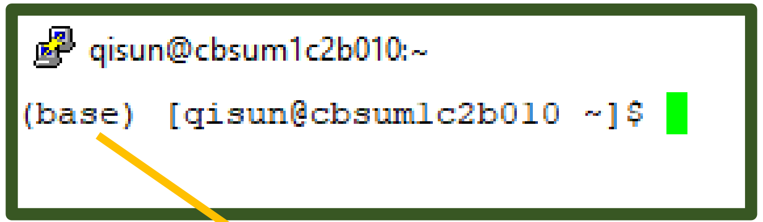
* run these commands in “python” prompt

The most common error: you are in Conda base, but try to run a software not installed through Conda

- System default
- Conda base
- Conda environment

You are in Conda base, but try to run a Python script installed by BioHPC admin.

How to tell that you are in Conda?

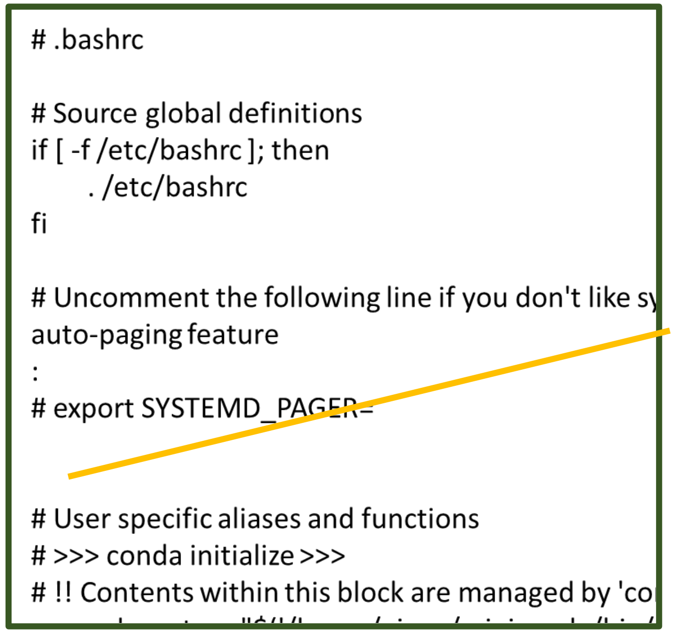


```
qisun@cbsum1c2b010:~
(base) [qisun@cbsum1c2b010 ~]$
```

"(base)"

How to correct?

Edit the .bashrc file in your home directory.



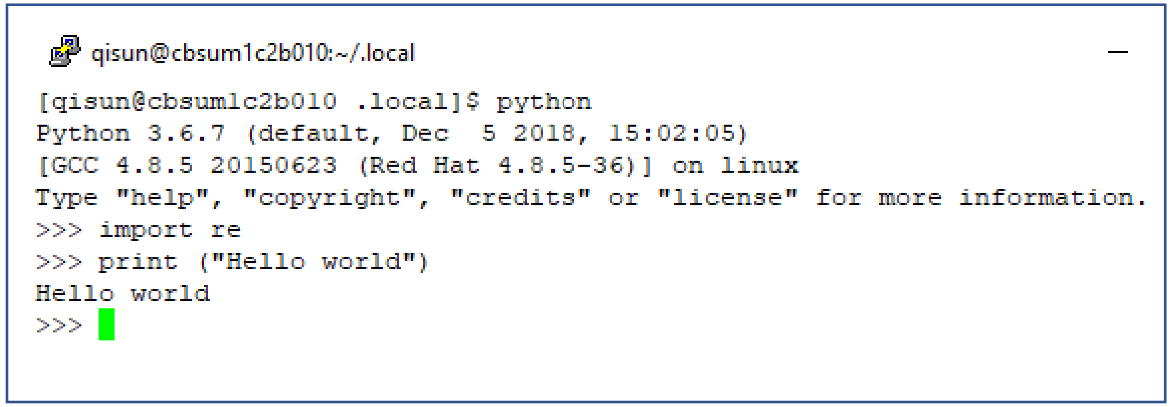
```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# Uncomment the following line if you don't like sy
auto-paging feature
:
# export SYSTEMD_PAGER=


# User specific aliases and functions
# >>> conda initialize >>>
# !! Contents within this block are managed by 'co
```

Insert a line with the word "return" before "conda initialize". Then logout and login again.

Jupyter Notebook

Three ways to run Python:
Python shell, Python script and Jupyter Notebook (Jupyter Lab)

Python shell


qisun@cbsum1c2b010:~/.local

```
[qisun@cbsum1c2b010 .local]$ python
Python 3.6.7 (default, Dec  5 2018, 15:02:05)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-36)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import re
>>> print ("Hello world")
Hello world
>>> 
```

Python script (run in Linux shell)

```
python myscript.py
```

or

```
./myscript.py
```

(#!shebang line ignored)

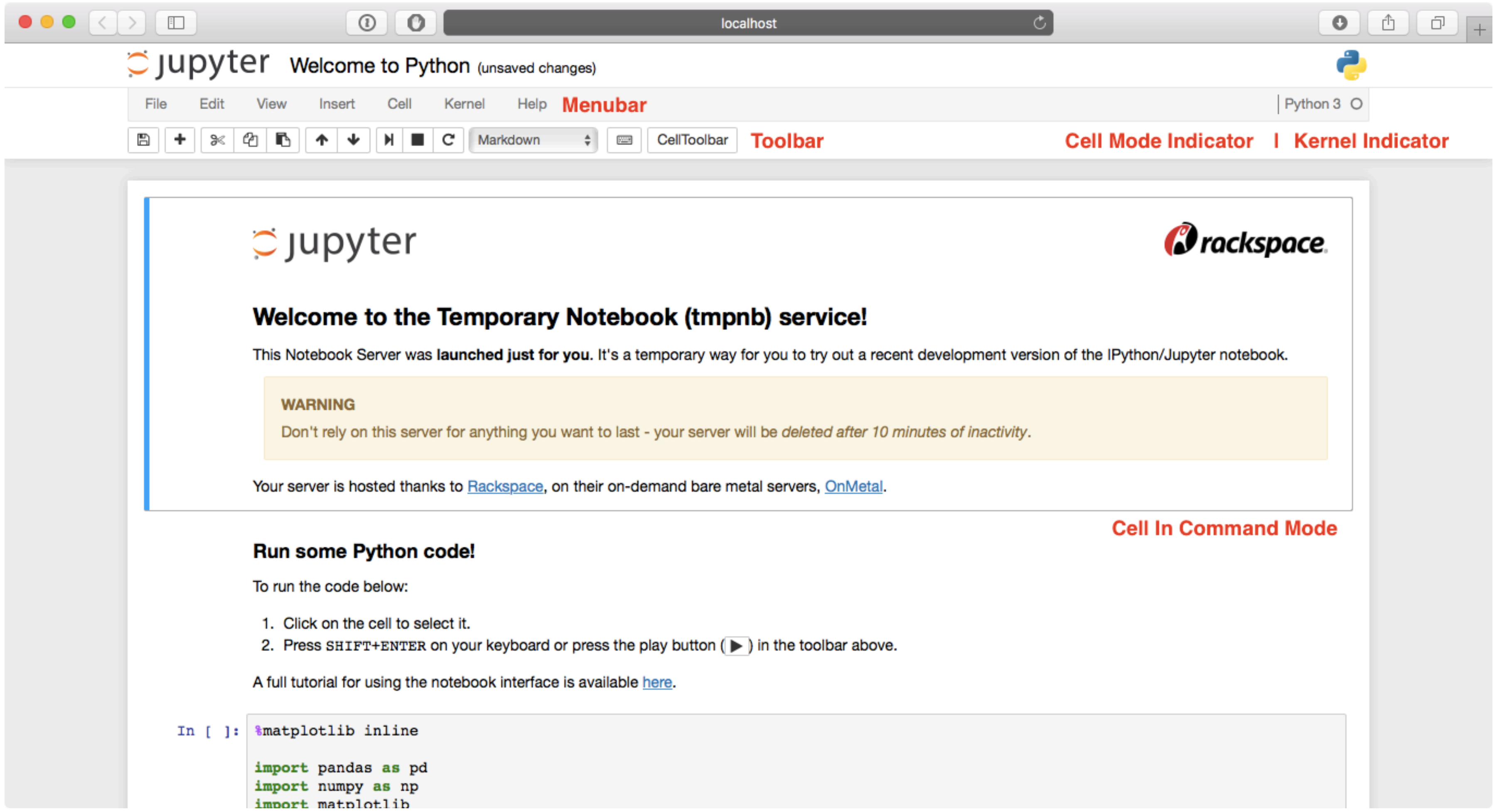
(#!shebang line define which python interpreter to use)

Jupyter notebook (Jupyter Lab)

(https://biohpc.cornell.edu/lab/userguide.aspx?a=software&i=263#c)

Jupyter notebook runs Python through a web browser

http://cbsum1c2b010.biohpc.cornell.edu:8016/?token=72cc017561bd59ba4dab4a5604d7857c93dd8f68a45d520b


localhost
jupyter Welcome to Python (unsaved changes)
File Edit View Insert Cell Kernel Help
Menubar
Python 3
Markdown
CellToolbar
Toolbar
Cell Mode Indicator | Kernel Indicator
jupyter
rackspace
Cell In Command Mode

Welcome to the Temporary Notebook (tmpnb) service!

This Notebook Server was **launched just for you**. It's a temporary way for you to try out a recent development version of the IPython/Jupyter notebook.

WARNING

Don't rely on this server for anything you want to last - your server will be *deleted after 10 minutes of inactivity*.

Your server is hosted thanks to Rackspace, on their on-demand bare metal servers, OnMetal.

Run some Python code!

To run the code below:

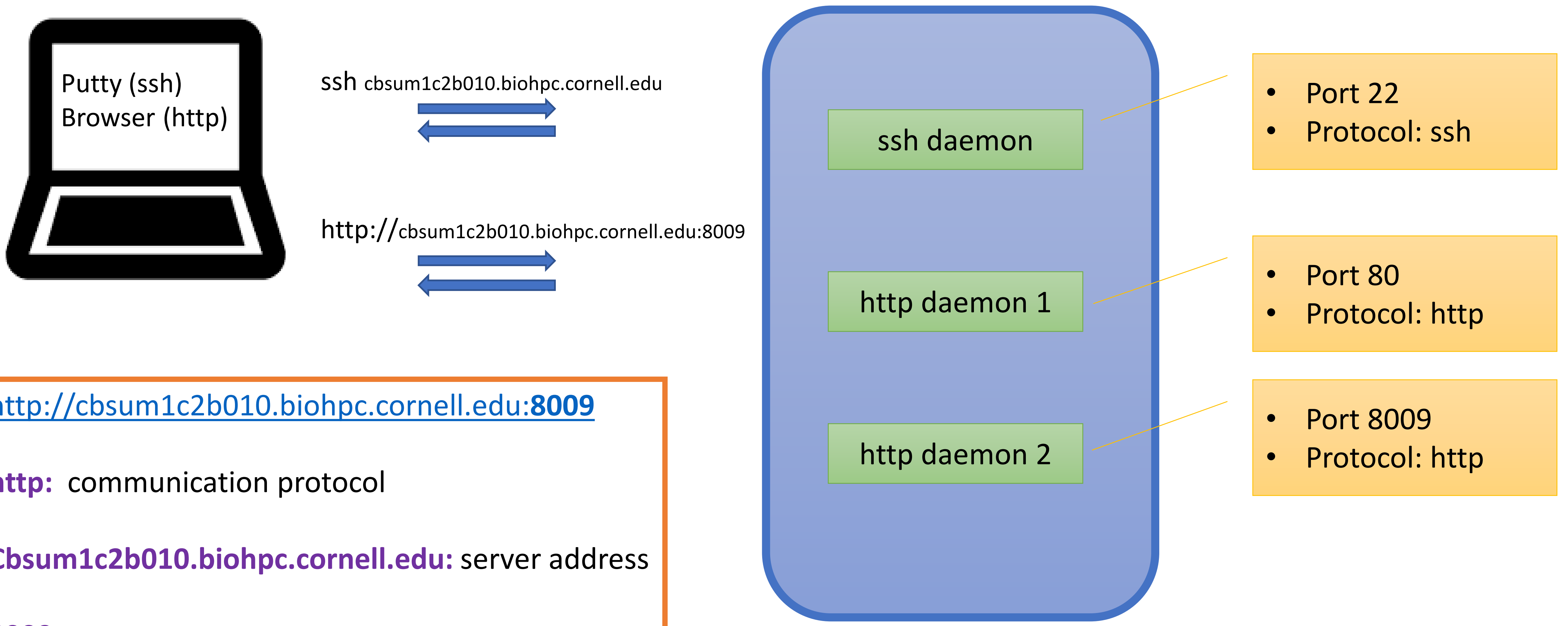
1. Click on the cell to select it.
2. Press `SHIFT+ENTER` on your keyboard or press the play button (▶) in the toolbar above.

A full tutorial for using the notebook interface is available here.

```
In [ ]: %matplotlib inline

        import pandas as pd
        import numpy as np
        import matplotlib
```

Client: your laptop

Server: cbsum1c2b010.biohpc.cornell.edu



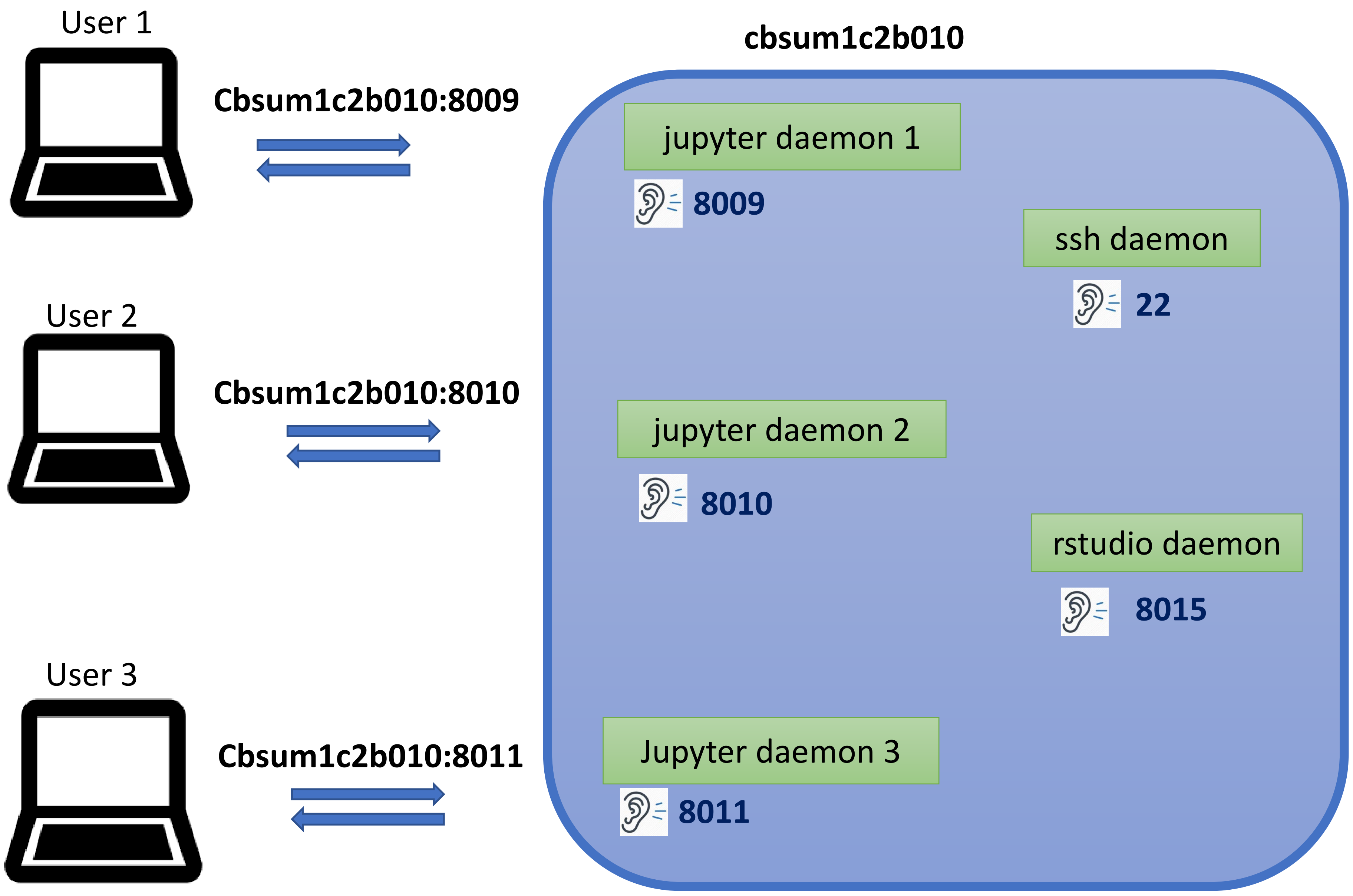
http://cbsum1c2b010.biohpc.cornell.edu:**8009**

http: communication protocol

Cbsum1c2b010.biohpc.cornell.edu: server address

8009: port

- A 'daemon' is a software process that is continuously running in a background, often listening to a port;



With ssh and rstudio, one daemon can serve multiple users.

To start a Jupyter notebook daemon with default Python (v3.6)

It is important keep the server daemon running in a persistent "screen" session

```
screen

export PYTHONPATH=/programs/jupyter3/lib/python3.6/site-packages:/programs/jupyter3/lib64/python3.6/site-packages

export PATH=/programs/jupyter3/bin:$PATH

jupyter notebook --ip=0.0.0.0 --port=8017 --no-browser
```

You will be provided with a URL which you can open in a web browser:

http://cbsum1c2b010.biohpc.cornell.edu:8017/?token=dfe3b002ca2d7721c4a2c0c641de91645e74f59d6519e31b

How to use "screen": https://biohpc.cornell.edu/lab/doc/Linux_exercise_part2.pdf

If you need a different version of Python, install and run Jupyter with Conda or Docker

```
source ~/miniconda3/bin/activate

conda create -n mypython3 python=3.8

conda activate mypython3

mamba install -c conda-forge notebook
```

#activate Conda

#create a Conda environment “mypython3” with python v3.8

#activate mypython3 environment

#install Jupyter Notebook. I use “mamba” here as it is a lot faster than Conda.

To run Jupyter installed in a Conda environment:

```
screen

source ~/miniconda3/bin/activate mypython3

jupyter notebook --ip=0.0.0.0 --port=8019 --no-browser
```

- On BioHPC, only ports between 8009-8039 are open to users;
- Check if a port is already being used: **netstat -tulpn | grep 8019**

In summary

Installing Python software

Python software repository:

Pypi

Anaconda

Installation package manager:

Pip

Conda or Mamba

Installation directory:

Pip: sys.path or ~/.local (--user option)

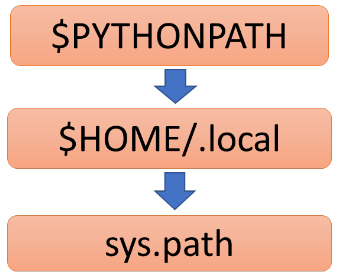
Conda: Conda base and Conda environment

Running Python software

Which python interpreter?

```
which python
python -V
#check shebang line of the script
```

Which python package?



Some afterthoughts

Why is it so complicated?

Because a server is shared by many people and many applications. To work peacefully together, we have to follow certain rules.

Maybe someday a computer is cheap enough, I can have a dedicated computer for each job.

Not likely in the near future.
… But wait, we have something that is close enough, “Docker” and “Singularity”.