Python – Exercises

Part 1. Python shell and Python script

1.1 Python shell vs Bash shell

A "shell" is a program that takes commands from the keyboard and gives them to the operating system to perform. Bash is the default "shell" in most Linux systems.

```
#In a Bash shell, you can run bash command like "ls"
ls -al

#Switch from Bash to Python shell
python

#Run python commands
import os
os.listdir()

#Exit Python shell and back to Bash
quit()
```

1.2. Check versions of Python and Python packages in either bash shell or Python shell

```
#In a Bash shell, check versions for Python and Python package Numpy
python -V

pip show numpy

#Switch from Bash to Python shell
python

#In Python shell, check versions for Python and Python package Numpy
import sys
print(sys.version)

import numpy
print(numpy.__version__)
print(numpy.__file__)

#exit python shell
quit()
```

Python shell can be a powerful tool for debugging Python environment errors.

1.3 Working with Python script.

In the Linux workshop, you learned how to write a shell script ("Bash"). Just like a (Bash) shell script, which is a file of a set of "Bash" commands, a "Python" script is a file with a set of "Python" commands.

In this exercise, you will create a python script file named "myscript.py" with the following lines. You can use any text editor to create this file, for example, vi, nano, Notepad++ on Windows, BBEdit on Mac, et al.

```
#!/usr/bin/python3

import numpy

print("Hellow World")
print(numpy.__version__)
print(numpy.__file__)
```

If you have problem to use any of the Text editors, you can simply copy a pre-made script file to your current directory:

```
cp /shared_data/qisun/myscript.py ./
```

Run the script

```
#If you run script this way, you would get an error message "Permission denied". Try it.
./myscript.py

#However, the following command would work. Why? The answer is in the text right after this block of code.
python ./myscript.py

#Change the file to an executable file. Do you see the difference before and after you run "chmod a+x"?
ls -l myscript.py

chmod a+x myscript.py

ls -l myscript.py


#now try again
./myscript.py

#Can I run the script without the "./" part? Try it.
myscript.py

#if you include the current directory in the $PATH variable, now you can
export PATH=./:$PATH
myscript.py
```

The Linux shell command "ls -l" tells you whether a script file is readable or executable. If a script file is readable, you can run the script with the command "python myscript.py". In this case, the Python interpreter is the executable which read the content of the script file. However, if a script is executable and the file starts with a shebang line, you can run the script with the command "./myscript.py". The "./" part is needed here, because it specifies the path of the script file. You can skip "./" if you include the current directory in the Linux $PATH variable.

Part 2. Install Python software with PIP

2.1 Check the version of python and pip

On BioHPC computers, the "python" command points to "python3.6.7", with "pip" command linked to this default python. The "python2" command points to "python2.7.15", with "pip2" command linked to python2.

Before you install any python modules, it is always a good idea to ask these three questions.

1. Which copy of Python installation will you be running?
2. What is the version of the Python?
3. Whether the pip command you are running is associated with the Python you will be running?

Use the following commands to address these questions:

```
which python
ls -l /usr/local/bin/python
python -V         ##use  capital V
which pip
head -n1 /usr/local/bin/pip     # the "head -n1" command prints the first line of
a text file
```

2.2 Switching default python using "module" command.

To execute python 3.6, you can use the alias "python" as the command or full path command "/usr/bin/python3.6". Linux "module" function provides an easy way to switch default "python".

```
# list the available modules
module avail

# switch default python/pip to version 3.9.6
module load python/3.9.6
which python
which pip

# switch back to default python
module unload python/3.9.6
which python
which pip
```

2.3 Install and use python module python-dummy
Install a dummy package called "python-dummy" in the directory ~/.local/

```
pip install python-dummy --user
```

Check the directory ~/.local/ to find the newly installed package, you should see the installation directory "python_dummy-0.1.0.dist-info" and a file "dummy.py"

```
cd /home/qisun/.local/lib/python3.6/site-packages/
ls -lrt
```

2.4 Check the "pip-install-test" file in Python shell.

```
python

import dummy
dummy.__file__
```

After you are done, press “Ctrl-d” or type "quit()" to exit python shell.

- Make sure that you type **double-underline** for the part "__file__" .

Part 3. Install PYTHON software with Conda

3.1 If you do not already have Miniconda3 in your home directory, install it now. If you are not sure, run the command "ls -l $HOME". If you do not see a directory called "miniconda3", you need to install it.

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh

chmod u+x Miniconda3-latest-Linux-x86_64.sh

./Miniconda3-latest-Linux-x86_64.sh
```

- The “chmod u+x” command makes the file executable.
- During the installation, you will be asked for multiple questions:
  1) “Please, press ENTER to continue”: press “ENTER” key;
  2) “More”: keep pressing “SPACE” key until you reach next question;
  3) “Do you accept the license terms?”: enter “yes”
  4) “Miniconda3 will now be installed into this location … ”: press “ENTER” key and accept the default “/home/xxxxx/miniconda3”.
  5) **“Do you wish the installer to initialize Miniconda3”** : Press ENTER to accept the default **“no”;**

3.2 Install/run pysam in Conda base. In this exercise.

```
source $HOME/miniconda3/bin/activate

which python

python -V

conda install -c bioconda pysam

conda install -y pip

which pip

head -n1 ~/miniconda3/bin/pip
```

3.3 Install/run pysam in a Conda environment

Create a virtual environment and give it a name “pysam”, install pysam the virtual environment. This time, you will use "mamba", an alternative Conda package manager.

```
conda install mamba

mamba create -c bioconda -n pysam pysam
```

- The command “conda create -n pysam” would create an environment called “pysam” (you can use any names), and install the pysam package into it.
- This environment is independent from python base, you can have different versions of python, and different versions of dependencies (e.g. numpy) within the environment

Start the pysam environment, and run python

```
conda activate pysam
which python
python -V
which pip
```

Run "conda activate pysam" if you are already in Conda "base". Otherwise you can use a single command "source $HOME/miniconda3/bin/activate pysam" to get into the environment directly.

3.4 Create a conda environment with a different version of Python

Sometimes, you need to run an old script that requires python 2.7. You can create a conda environment with python =2.7

```
mamba create -n myNewPipeline python=2.7
mamba activate myNewPipeline
which python
which pip
python -V
```

Note in the first command there is no package name. This step would create an empty python2.7 environment, within which you can use pip to install other python modules.

When you run "conda" or "mamba" to install packages, you will be prompted to inspect the versions to be installed, and answer "y" to continue. If you wish to skip this step, use the "-y" option: "conda install -y".

3.5 Exit conda

You might need to run this command twice to exit Conda environment and Conda base.

```
conda deactivate
```

Part 4. Jupyter Notebook

Pick a number between 8009 and 8039 as the port you will use, for example 8029. Verify whether this port is taken or not. If you do not see "8019 ... LISTEN" with the following command, that means the port is available.

```
netstat -tulpn | grep 8029
```

Run following command to start Jupyter notebook daemon.

```
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=8029 --no-browser
```

Copy the URL, and open it in a web browser. You can detach the "screen" session by pressing "ctrl-d".