# AnYong Analytics 安勇

It's About Good Data Science

An AnYong Analytics, LLC
Technical Document

# Configuring Chromebook® Linux® Debian® 10 as an Open-Source Data Science Environment

Patrick Williams, President
February 24, 2021

## Contents:

------------------ For readability, trademarks are used only on the first two pages in their initial prominent uses. ----------------

**Introduction**

The purpose of this document is to describe how to configure a Chromebook® with an enabled Linux® Debian® 10 environment to engage in data science. This includes the use of the following open-source or free software:

- **Python®** – a powerful and popular language used in data science to access, transform, explore, model and visualize data, with many other applications outside of data science.
- **PyCharm® Community Version** – a robust project-based integrated development environment (IDE) from which the user can edit and directly submit Python® code, view results, etc. This free version includes a code editor, a Python® console and a Terminal console for the operating system on which it is installed. The non-free version of this software (PyCharm® Professional) has expanded capabilities to connect to other machines, databases, clouds, etc. It also offers the ability to write and submit R® code, having an R® console.
- **Git®** – a browser-based code version control program that interacts with GitHub®, in which code repositories can be placed on the internet. Git® also includes Git® Bash, which allows command line interaction with GitHub®.
- **R®** - a very powerful and popular language and environment for statistical computing and graphics/visualization.
- **Jupyter® Notebook** – a simple open-source browser-based application that comes with Python®. It allows the creation and sharing of Python® code, visualizations, narrative text and comments, etc.

Data science work often involves using Windows® (DOS) and Linux® operating system commands on those systems' terminal command lines. Here are some recommended free resources for looking up command syntax for both operating systems:

- Windows® - https://www.computerhope.com/msdos.htm#commands
- Linux® - https://www.computerhope.com/unix.htm#commands

This document focuses on the use of Chromebook®, which includes a Linux® Debian® operating system, offering new learners an opportunity for significant exposure to Linux®. Also, experienced Linux® users wishing to delve into data science will have a familiar environment in which to learn. Installing and configuring a data science system as described in this document should give even novice users the ability to jump right into the many educational services available for learning data science, including Udemy®, Cousera®, Data Camp®, etc.
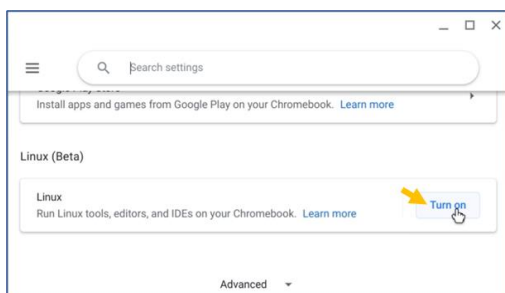
Good luck and good learning!
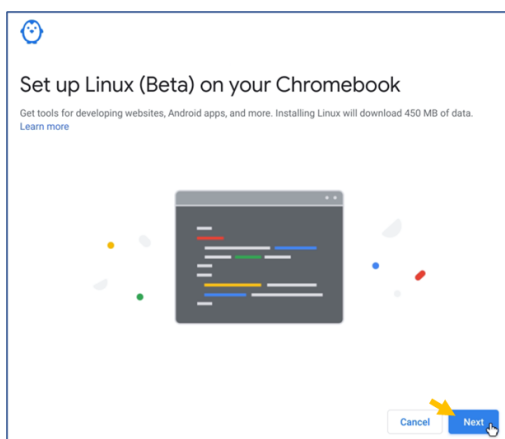
## 1. Enable Linux Operating System

Click in the **clock area** of the Chromebook Desktop. In the screen that appears, choose the **Settings** icon.
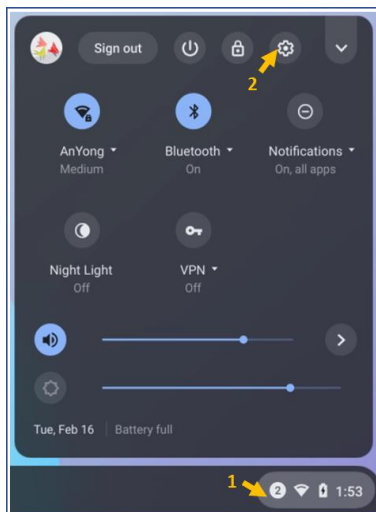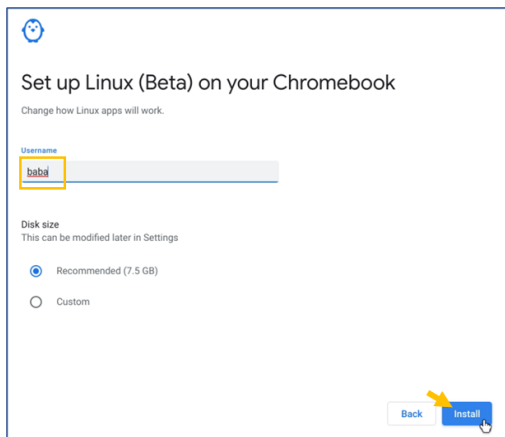


In the next screen, scroll down to the **Linux section** and choose **Turn On**.
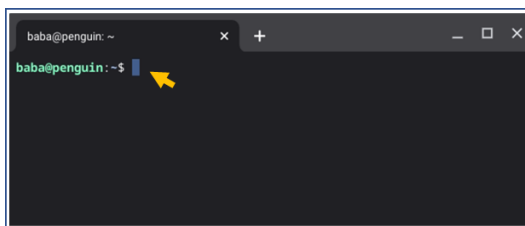


In the screen that appears, choose **next**.

In the **setup screen**, enter a **user name** of your choice and choose **install**.



After a few minutes, a **Linux command line** will appear, which is also called **Terminal**, meaning setup is complete. Leave this Terminal session open for use later.



## 2. Pin App Icons to Taskbar

Because we may be using the Terminal, Text and File apps frequently, we will pin them to the taskbar (a.k.a., Shelf) for convenience. To do this:

- Choose the **Search Icon** in the lower left of the Chromebook desktop. The Chromebook **search screen** will appear. Then scroll to the **Terminal** application, **right click** and choose **Pin to shelf**.
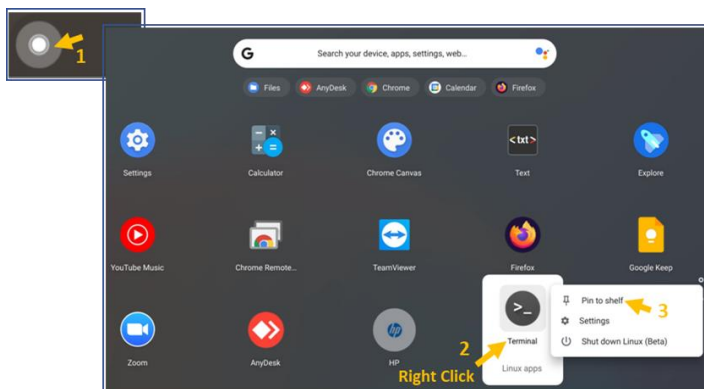
- Repeat these steps for the **Files** and **<txt>** applications, and the taskbar (shelf) should appear as below.
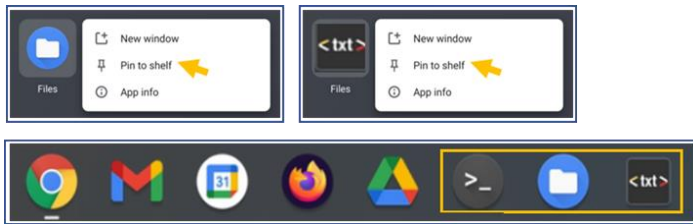


## 3. Create the /Python_Projects Directory

In this step we create a directory for all of your Python-related software and work.

- If you happened to close your Terminal session, choose the **Terminal ico**n you just added to the taskbar.



- At the Linux command prompt ($), type the following commands, using Enter after each (you can copy from commands from this document and paste into the command line):
    - **pwd**   (show the current directory, which is /home/baba, or ~)
    - **cd /**   (move to the top of the directory structure)
    - **ls -a**   (list directory contents alphabetically)
    - **mkdir Python_Projects**   (make a new directory called Pthyon_Projects)
    - **cd Py*_*** (change directory to Pthyon_Projects using wildcards)
    - **clear**   (clear the screen)



- At this point we will be working from the **/Python_Projects** directory, so don't close it.

## 4. Download Software to Python_Projects Directory

Now we use the Linux command line to download Python, PyCharm, Git, and R. Throughout the rest of this document, anytime a Linux command returns a confirmation dialog requiring a yes response, enter either **Y**, **Yes** or **y**, depending on what is being requested.

```
baba@penguin: /Python_Projects      ×      +

 r-cran-survival r-doc-html r-recommended
0 upgraded, 48 newly installed, 0 to remove and 0 not upgraded.
Need to get 76.9 MB of archives.
After this operation, 187 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

- **Python Download** – Here we install first dependencies needed for Python, and then download the Python software, which will be installed later.
    - If you are not already in the directory, enter the command **cd /Python_Projects**. Then clear the screen if needed by entering the command **clear**.

```
baba@penguin: /Python_Projects      ×      +        _  □  ×

baba@penguin:/Python_Projects$ cd /Python_Projects
baba@penguin:/Python_Projects$ clear
```

    - To update the software repository from which software is received, enter the command **sudo apt update**.

```
baba@penguin: /Python_Projects      ×      +        _  □  ×

baba@penguin:/Python_Projects$ sudo apt update
```

    - To install the dependencies required for Python, enter the command **sudo apt install build-essential zlib1g-dev libncurses5-dev libgdbm-dev libnss3-dev libssl-dev libsqlite3-dev libreadline-dev libffi-dev curl libbz2-dev**.

```
baba@penguin: /Python_Projects      ×      +        _  □  ×

baba@penguin:/Python_Projects$ sudo apt install build-ess
ential zlib1g-dev libncurses5-dev libgdbm-dev libnss3-dev
 libssl-dev libsqlite3-dev libreadline-dev libffi-dev cur
l libbz2-dev
```

Enter **Y** when asked if you wish to continue.

```
Need to get 48.3 MB of archives.
After this operation, 179 MB of additional disk space wil
l be used.
Do you want to continue? [Y/n] Y
```

Numerous messages will appear as the dependancies are verified and/or installed. When complete you are returned to the Linux command line, as below.

```
baba@penguin: /Python_Projects        ×    +         _  □  ×
Setting up g++ (4:8.3.0-1) ...
update-alternatives: using /usr/bin/g++ to provide /usr/b
in/c++ (c++) in auto mode
Setting up build-essential (12.6) ...
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for libc-bin (2.28-10) ...
baba@penguin:/Python_Projects$
```
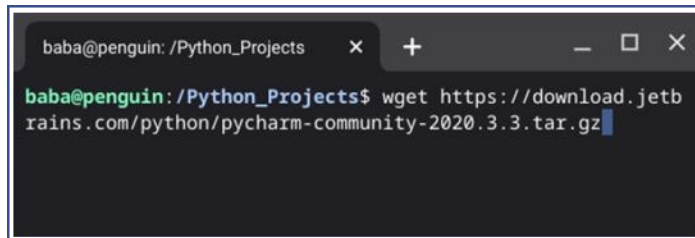
With dependancies installed, we now retrieve the Python compressed .tgz file, by entering the command **wget https://www.python.org/ftp/python/3.9.2/Python-3.9.2.tgz**.

```
baba@penguin: /Python_Projects        ×    +         _  □  ×
baba@penguin:/Python_Projects$ wget https://www.python.or
g/ftp/python/3.9.1/Python-3.9.2.tgz
```

When the download is complete the Linux window will appear as below. Now enter the **ls** command to see the file in the /Python_Projects directory.
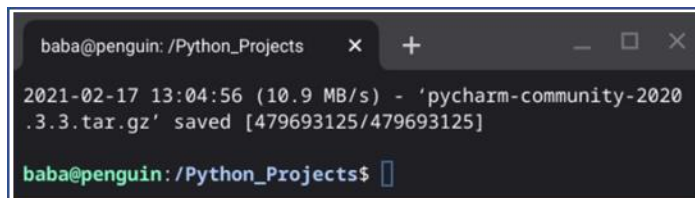
```
baba@penguin: /Python_Projects        ×    +         _  □  ×
Length: 25399571 (24M) [application/octet-stream]
Saving to: 'Python-3.9.2.tgz'

Python-3.9.2.tgz        100%[========================>]

2021-02-21 15:06:27 (7.68 MB/s) - 'Python-3.9.2.tgz' save
d [25399571/25399571]

baba@penguin:/Python_Projects$ ls
Python-3.9.2.tgz
baba@penguin:/Python_Projects$
```

- **PyCharm Download** – Here we have no dependencies, so we simply download the PyCharm software.
  - Enter the command **wget https://download.jetbrains.com/python/pycharm-community-2020.3.3.tar.gz** to get the PyCharm .tgz file.
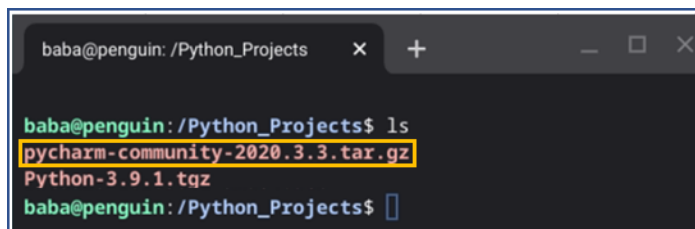
```
baba@penguin: /Python_Projects        ×   +          _  □  ×

baba@penguin:/Python_Projects$ wget https://download.jetb
rains.com/python/pycharm-community-2020.3.3.tar.gz
```

When the download completes, the Linux window appears similar to below.

```
baba@penguin: /Python_Projects        ×   +          _  □  ×

2021-02-17 13:04:56 (10.9 MB/s) - 'pycharm-community-2020
.3.3.tar.gz' saved [479693125/479693125]

baba@penguin:/Python_Projects$ ▯
```

Enter the command **ls** to verify that the PyCharm .tgz file is in the Python_Projects directory.

```
baba@penguin: /Python_Projects        ×   +          _  □  ×

baba@penguin:/Python_Projects$ ls
pycharm-community-2020.3.3.tar.gz
Python-3.9.1.tgz
baba@penguin:/Python_Projects$ ▯
```

## 5. Install Python-related Software in the Python_Projects Directory

In this section we install Python-related software in the /Python_Projects directory. We use this directory purely for convenience and to limit the directories we access as our Python work proceeds. For Python, we both un-compress the .tgz file and issue some additional commands to fully install the software. For PyCharm we simple un-compress the .tgz file.

- **Python Install**
  - In the /Python_Projects directory, un-compress the Python .tgz file using the command **tar -xf Python-3.9.1.tgz**. Note that this process has create a subdirectory **Python-3.9.2**, from which the configuration will be performed below.

- o Enter the following two commands:
  - **cd Python-3.9.1**
  - **./configure --enable-optimizations**



When the command line returns, the Terminal screen should appear similar to below.



- o Now enter the command **sudo make install**.



When the command line returns after a bit, something similar to the following should appear.
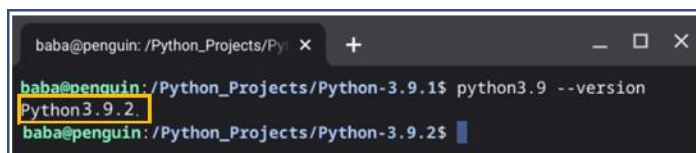


- o Now we can check the installed versions of both Python and PIP. Python is installed with a standard set of libraries (a.k.a., packages or modules) and functionality. However, there

are many additional libraries that offer functionality far beyond the standard packages. PIP is installed with Python, allows users to install and manage additional packages.

- **Verify Python and PIP Versions**
  - **Python Version** – Enter the command **python3.9 --version**.
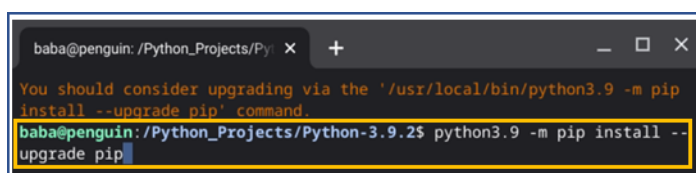


  - **PIP version** - Enter the command **pip3.9 --version.**
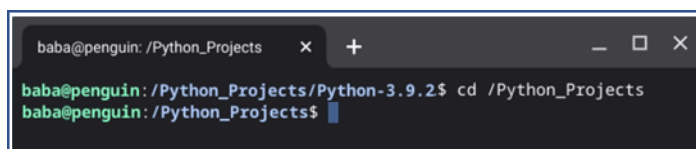


  - **Note: PIP Upgrade** – When a new version of PIP is available, the user is notified while using it to install software with a message similar to below. When this happens, simply enter the Linux command **python3.9 -m pip install --upgrade pip**. Given our aliases, the command **python -m pip –upgrade pip** would also work.
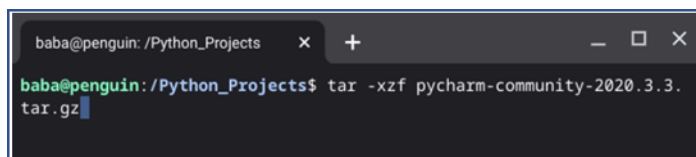


- **PyCharm Install**
  - **Change Directory** - Enter the command **cd /Python_Projects** (or **cd ..**).



  - **Un-compress the PyCharm .tgz file** – Enter the command tar **-xzf pycharm-2020.3.3.tar.gz**.

- o Note: the PyCharm software is placed in the **pycharm-community-2020.3.3** subdirectory.



At this point, the PyCharm software can be started with the command **/Python_Projects/pycharm-community-2020.3.3/bin/pycharm.sh**. However, we will create an alias below to simply the invocation of PyCharm, in addition to aliases for python3.9 and pip3.9 invocation.
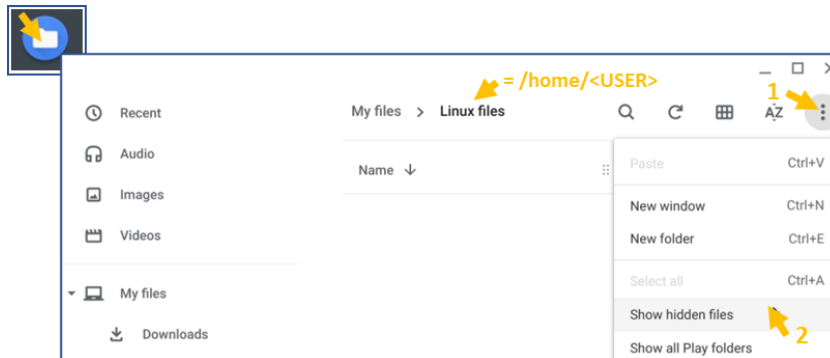
- **Edit the .bashrc File**

What is Bash and the .bashrc File? Bash is a Linux functionality wherein scripts are run in a shell session whenever the Linux environment is started interactively by a user. One of the scripts run in this session is the **/home/<USER>/.bashrc** script. Any command that you can run in the command line can be entered in this file. With .bashrc, a user can customize their environment, and it is common practice to specify aliases to simplify commands.

To edit the .bashrc file:

- o **Go to "home" directory -** Enter the following commands:
  - **cd ~**    (changes to your home directory, /home/<USER>)
  - **pwd**   (displays your home directory name)
  - **ls -a**   (lists the directory contents, including .bashrc file)



- o **Revise the .bashrc file** – From the taskbar, choose the **Files icon**, and navigate in the File application to **My Files > Linux Files**. This is simply an alias for the /home/<USER> directory. Then, choose the upper right **More** icon, and **Show hidden files**.
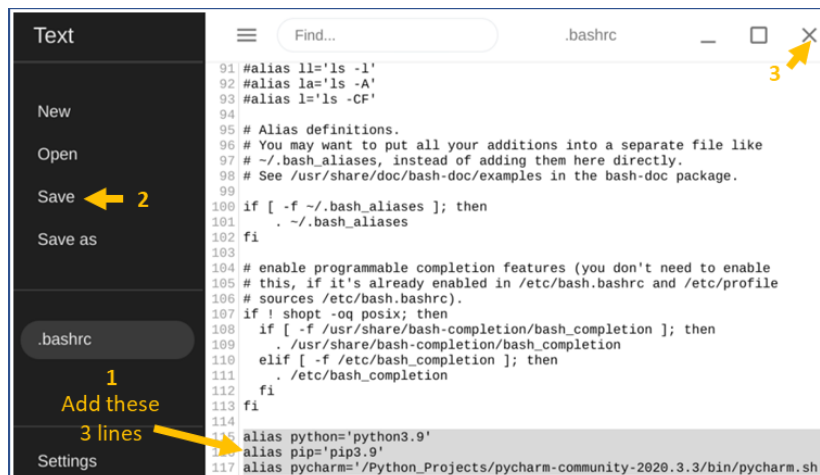
= /home/<USER>

Now choose the .bashrc file, and it opens in the **<txt>** application.
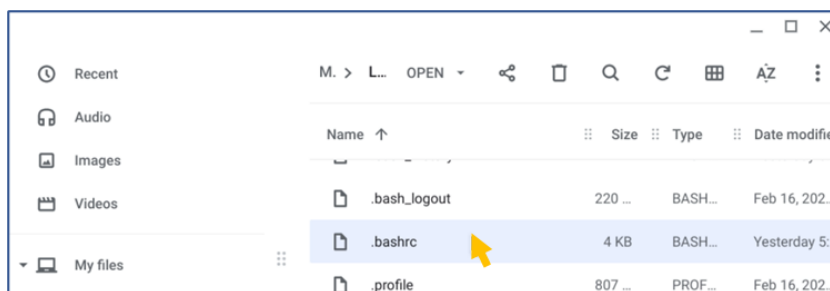


In the **<txt>** application, add the following three alias commands to the bottom of the file:

- **alias python='python3.9'**
- **alias pip='pip3.9'**
- **alias pycharm='/Python_Projects/pycharm-community-2020.3.3/bin/pycharm.sh'**



- Now choose **Save** to the left, and then **Close** the <txt> application.

- **Restart Linux** – Here we restart the Linux environment so the aliases we just added to the /home/<USER>/.bashrc file are available for our Linux session.

  To do this, simply **right click on the taskbar's Terminal icon** and choose **Shut Down Linux**. The **Terminal window closes**.
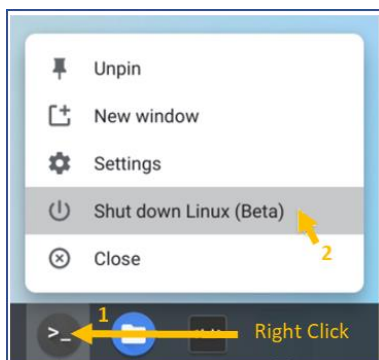


  Now do the following:

  - Restart Linux by choosing the **Terminal icon** on the taskbar.
  - To validate the pycharm alias, enter the alias **pycharm** on the command line, which invokes the  PyCharm software.
  - **Close** the PyCharm software.
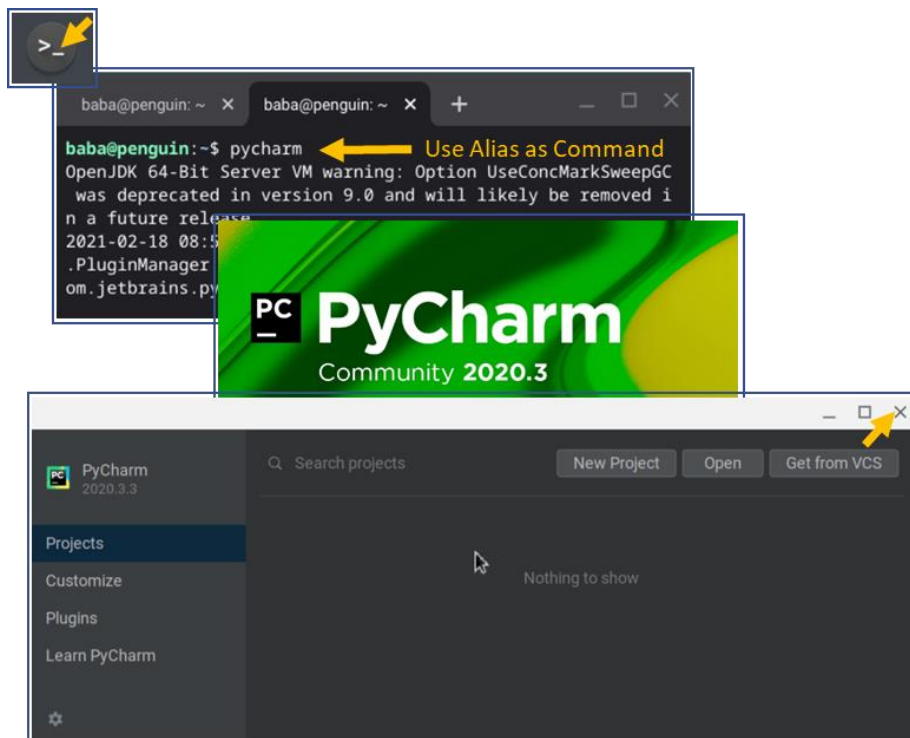
- To validate the python and pip aliases:
  - Enter the alias **python** on the **Linux command line**, which invokes Python 3.9.2, validating the alias. Then enter **import pandas** on the **Python command line**.



The expected Python error message indicates that we have not yet used PIP to install the Python Pandas library. Let's do that, but first enter the command **exit()** on the **Python command line** to return to the Linux command line.

  - Now enter the command **pip install pandas** on the **Linux command line**. The installation of this library into Python validates the pip alias.



  - Finally, invoke Python again and import Pandas. The Python error message no longer appears because the Pandas module was installed via the pip alias. Also, exit Python.

## 6. Install Non-Python Software in Default Directories

In this section we install software from the Linux repositories in their default directories. These include Git and R. These installations can be performed from any directory, such as /home/<USER>, as below.

- **Git Installation** – Execute the following three commands on the Linux command line, waiting after each for a return to the command line:
  - o **sudo apt get update**   (update the software repository)
  - o **susdo apt-get install git**   (install Git)
  - o **git –version**   (verify the Git version)

  That's it. Git works through the Linux command line in which commands that begin with git are used to interact with version-controlled repositories on GitHub.com.

- **R Installation** – The commands for this installation of the latest stable version of R are documented in https://cloud.r-project.org/bin/linux/debian/.
  - o **Pre-Installation Steps** – Execute the following three commands in this Linux command line, waiting after each for a return to the command line:
    - ▪ **sudo apt install dirmngr apt-transport-https ca-certificates software-properties-common gnupg2**
    - ▪ **sudo apt-key adv --keyserver keys.gnupg.net --recv-key 'E19F5F87128899B192B1A2C2AD5F960A256A04AF'**
    - ▪ **sudo add-apt-repository 'deb https://cloud.r-project.org/bin/linux/debian buster-cran40/'**

o **R Installation Steps** – Execute the following three commands in this Linux command line, waiting after each for a return to the command line:
  ▪ **sudo apt update**
  ▪ **sudo apt install r-base**
  ▪ **R --version**

- **Starting and Closing R** – To start R, enter the command **R** on the Linux command line. The R command line will appear. Enter quit() in the R command line to exit R and return to the Linux command line.



7. **Install and Configure Jupyter Notebook for R**
   - **Install Jupyter Notebook** – Enter the following four commands:
     - **cd /Python_Projects/Python-3.9.2**  (if not already there)
     - **pip install jupyter notebook**   (installs software)
     - **cd ..**   (moves up one level to /Python_Projects)
     - **jupyter notebook**  (starts Jupyter Notebook server and opens browser)

A browser opens showing Jupyter Notebook showing access to content from all of the directories below that from which we invoked it, which was /Python_Projects. Hence we know that **we can invoke Notebook from any directory below which our desired content (code) resides**.



Note that when attempting to create a new file (notebook), we are only given the option to create one for Python code. That is by design, since Jupyter Notebook is part of the Python software distribution.

What if we wish to use R code? We do that below.

- **Add R Kernel to Jupyter Notebook** – Do the following:
  - o Enter the command **R** on the **Linux command line**, causing the **R console** to appear.
  - o Enter the following four commands on the R command line:
    - ▪ **MyRepo='https://cran.case.edu'** (get R software from Case University)
    - ▪ **install.packages('IRkernel',MyRepo)** (install the R kernel package)
    - ▪ **IRkernel::installspec(user = TRUE)** (apply the R Kernel to the user installing it)
    - ▪ **quit()** (answer **n** to saving workspace; returns to the Linux command line, as below)

```
baba@penguin: ~                           ×   +              _  □  ✕

baba@penguin:~$ R

R version 4.0.4 (2021-02-15) -- "Lost Library Book"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

...

> MyRepo='https://cran.case.edu'
> install.packages('IRkernel',MyRepo)
Warning in install.packages("IRkernel", MyRepo) :
  'lib = "https://cran.case.edu"' is not writable
Would you like to use a personal library instead? (yes/No/cancel) yes
Would you like to create a personal library
'~/R/x86_64-pc-linux-gnu-library/4.0'
to install packages into? (yes/No/cancel) yes

...

> IRkernel::installspec(user = TRUE)
[InstallKernelSpec] Installed kernelspec ir in /home/baba/.local/share/j
>
> quit
function (save = "default", status = 0, runLast = TRUE)
.Internal(quit(save, status, runLast))
<bytecode: 0x5884623ec2e0>
<environment: namespace:base>
> quit()
Save workspace image? [y/n/c]: n
```

- o  Restart the Jupyter Notebook server by entering **Jupyter notebook** on the **Linux command line**.



```
baba@penguin: /Python_Projects           ×   +              _  □  ✕

baba@penguin:/Python_Projects$ jupyter notebook
```

A Jupyter Notebook opens as before, but with the ability to create a new notebook for R code, as below.

Upon quitting the notebook, the browser closes and the server stops, returning to the Linux command line.

At this point an excellent collection of Data Science tools has been installed on your machine. Congratulations!