AnYong Analytics 安勇
It's About Good Data Science

An AnYong Analytics, LLC
Technical Document

Configuring Windows® 10 as an Open-Source Data Science Environment

Patrick Williams, President
February 15, 2021

Contents:

Introduction
1. Pin the Command Icon to Taskbar
2. Create C:\Python_Projects Directory
3. Download and Move Software to Python_Projects Directory
    Python® PyCharm® Git® GitHub® Desktop® R® RStudio® Moving the Downloaded Files
4. Installing Python-related Software in the Python_Projects Directory
    Python® PyCharm®
5. Installing Non-Python Software in Default Directories
    Git® GitHub® Desktop® R® RStudio®
6. Reboot the Machine
7. Verify Python® and PIP® Versions
    Python® PIP®
8. Installing Jupyter® Notebook
    Opening Jupyter® Notebook Adding the R® Kernel to Jupyter® Notebook

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

Trademark Notices:

Software	Organization to which registered
Windows®	Microsoft, Inc.
Python®, PIP®	Python Software Foundation
GitHub®	GitHub, Inc.
Git®	Software Freedom Conservancy
R®	The R Foundation for Statistical Computing
RStudio®	RStudio, PBC
Jupyter®	NumFOCUS Foundation
Linux®	Linus Thorvalds

Introduction

The purpose of this document is to describe how to configure a Windows® 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®. It also uses Linux® commands, thus giving users with little or no Linux® exposure the opportunity to learn and use those commands on a Windows® system.
- **GitHub® Desktop®** – A more robust non-browser-based Windows® application that interacts with GitHub® for code version control.
- **R®** - a very powerful and popular language and environment for statistical computing and graphics/visualization.
- **RStudio®** – a desktop integrated development environment for R® to edit and directly submit R® code, etc.
- **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. It can also integrate with R®**.**

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

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.

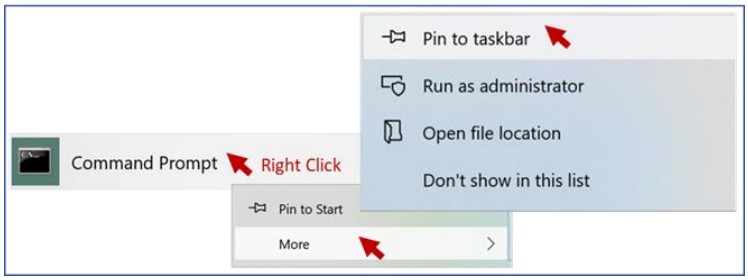
Good luck and good learning!

1. Pin the Command Icon to Taskbar

Because we may be using the Windows DOS Command Prompt frequently, we will pin it to the taskbar for convenience. To do this:

- Click the Windows Start icon on your taskbar and scroll to the **Command Prompt** program.

- *Right click* on the Command Prompt program, then choose **More > Pin to taskbar**.



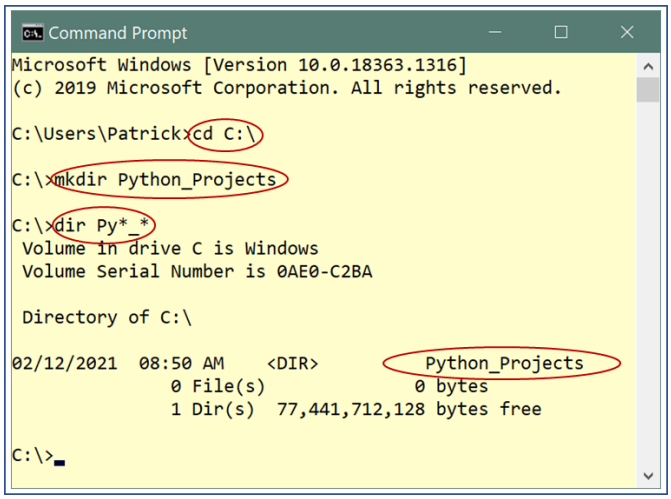
- The Command Prompt icon appears on your taskbar. You can reposition it to your liking.

2. Create C:\Python_Projects Directory

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

- Click on the **Command Prompt** you just added to the taskbar.

- At the command prompt (>), type the following four commands, using Enter after each (you can copy from commands from this document and paste into the command line):
  - **cd C:**
  - **mkdir Python_Projects**
  - **dir**
  - **cd Py*_*** (uses wildcards so we do not have to type out Pthyon_Projects)



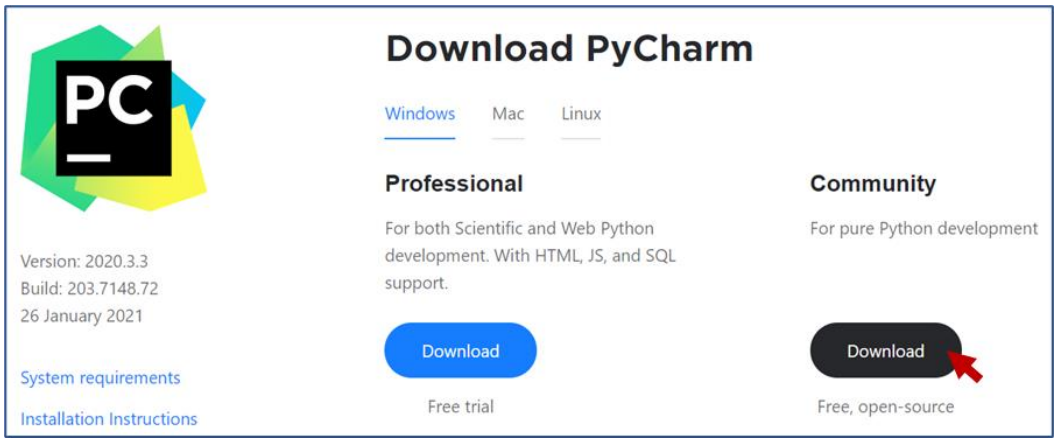
The "dir" command shows that the Python_Projects directory was created without showing all of the other directories on C, and the final command takes us to the just-created directory, where we will work. At this point we will be working from the **C:\Python_Projects** directory, so can minimize this window for later use , but don't close it.

3. Download and Move Software to Python_Projects Directory

Now we use your browser to download Python, PyCharm, Git, GitHub, R and RStudio software.

- **Python** – go to https://www.python.org/downloads/ and click on Download Python. The software will download to your Downloads directory, which is **C:\Users\<USER>\Downloads**, where <USER> is your Windows user name. The file name will be **python-3.9.1-amd64.exe**

- **PyCharm** – go to https://www.jetbrains.com/pycharm/download/#section=Windows and click on the **Community** (free) download button. This drops the file **pycharm-community-2020.3.3.exe** into your Download directory.

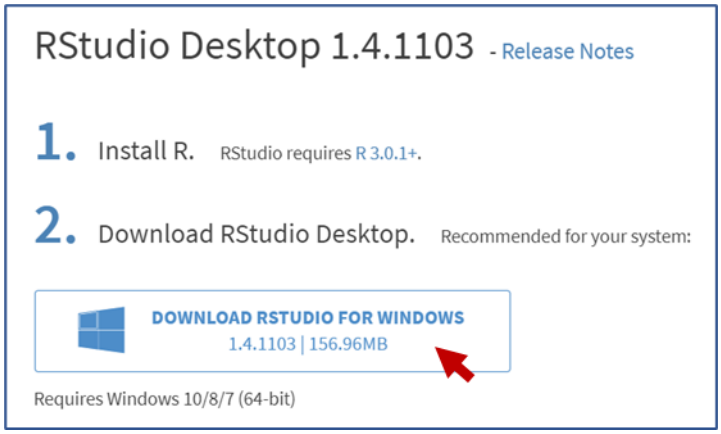


- **Git** – go to https://git-scm.com/download/win. In the **Downloading Git** section, click on **64-bit Git for Windows Setup**. This downloads the **Git-2.30.1-64-bit.exe** file.

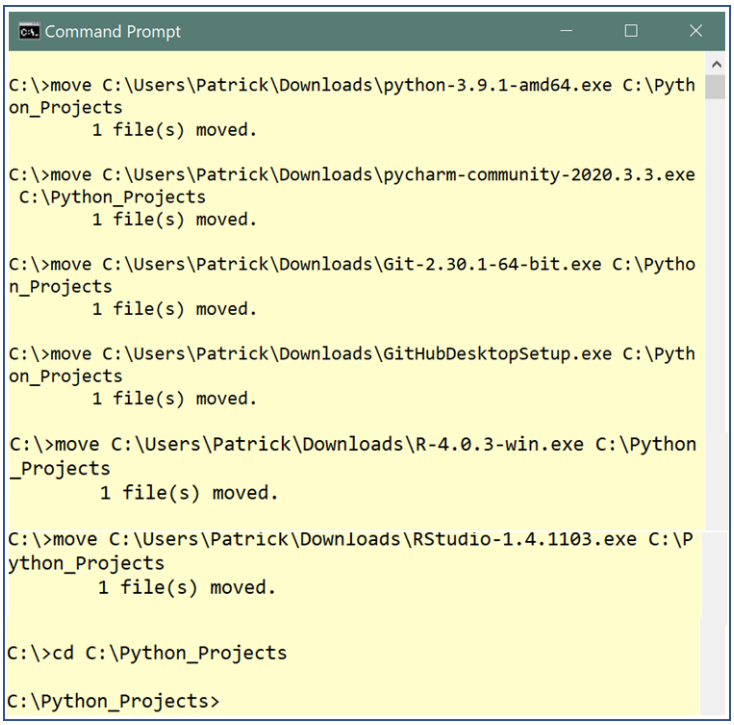
- **GitHub Desktop** – go to https://desktop.github.com/ and choose the **Download for Windows (64-bit)** button. The file **GitHubDesktopSetup.exe** is downloaded.

- **R** – go to https://cran.r-project.org/bin/windows/base/ and choose the **Download R 4.0.3 for Windows** hypertext. The **R-4.0.3-win.exe** file will be downloaded.

- **RStudio** – go to https://rstudio.com/products/rstudio/download/#download. Then choose the **DOWNLOAD RSTUDIO FOR WINDOWS** button. This downloads the **RStudio-1.4.1103.exe** file.



- **Moving the Downloaded Files** - Open the Windows **Command Prompt** and issue the following commands, each followed by **Enter**:
  - **cls** (This just clears the screen of previous text. Use it as needed.)
  - **move C:\Users\Patrick\Downloads\python-3.9.1-amd64.exe C:\Python_Projects**
  - **move C:\Users\Patrick\Downloads\pycharm-community-2020.3.3.exe C:\Python_Projects**
  - **move C:\Users\Patrick\Downloads\Git-2.30.1-64-bit.exe C:\Python_Projects**
  - **move C:\Users\Patrick\Downloads\GitHubDesktopSetup.exe C:\Python_Projects**
  - **move C:\Users\Patrick\Downloads\R-4.0.3-win.exe C:\Python_Projects**
  - **move C:\Users\Patrick\Downloads\RStudio-1.4.1103.exe C:\Python_Projects**
  - **cd C:\Python_Projects**

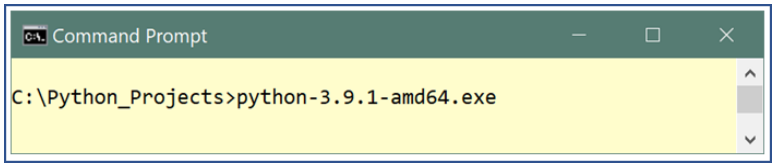


4. Installing Python-related Software in the Python_Projects Directory

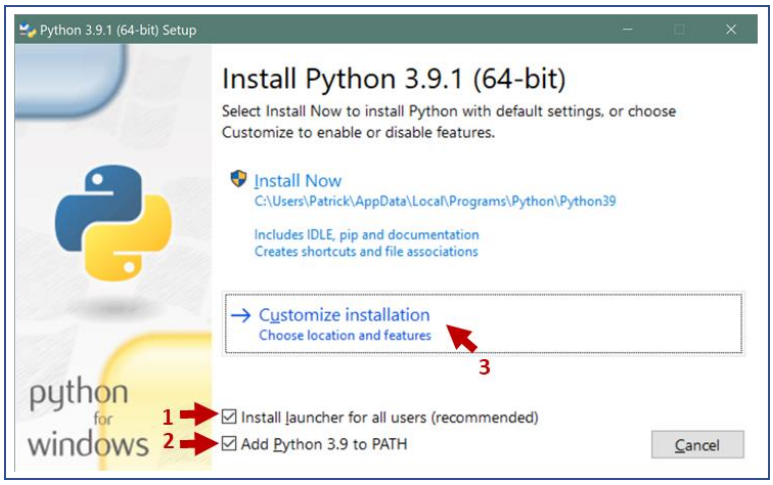
In this section we install Python-related software in the C:\Python_Prjects directory. We do this purely for convenience and to limit the directories in we access as our Python work proceeds.

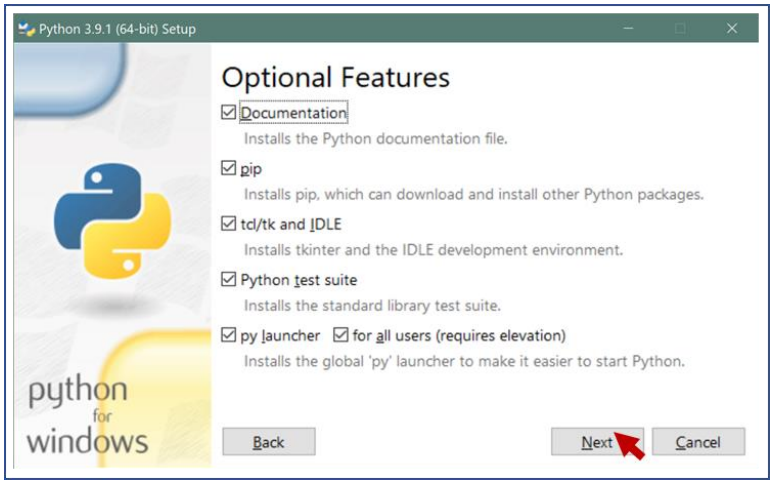
(NOTE: When installing any software, Windows most always asks in a screen "Do you want to allow this app to make changes to your machine?" Always choose **Yes**.)

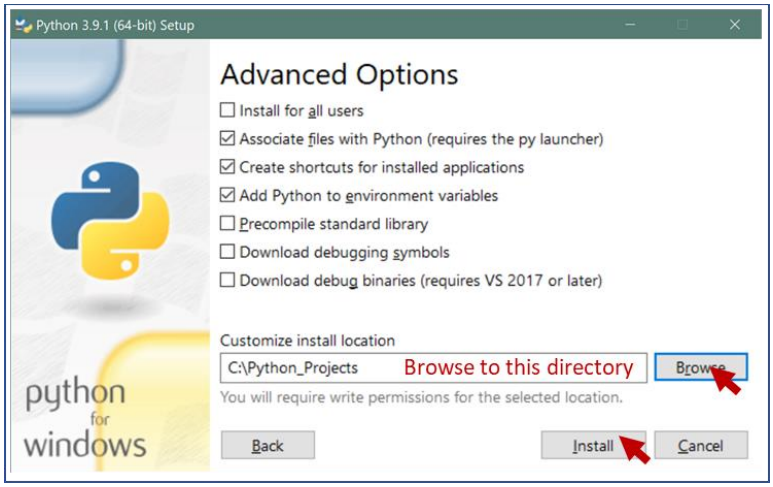
- **Python** – from the Command Prompt in the C:\Python_Projects directory, execute the following command: **python-3.9.1-amd64.exe**

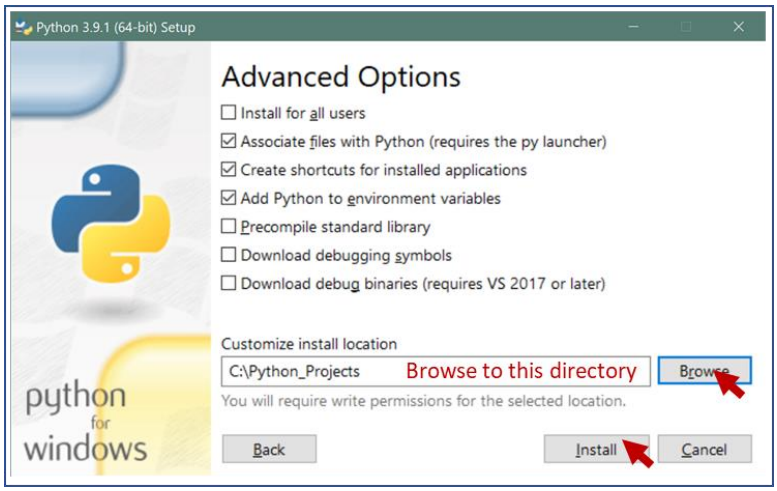


The following screens appear in the order of the graphics shown below. Please follow all instruction on these graphics. ***Check all boxes as shown in each graphic before choosing Next***. This will ensure that Python is installed in a consistent manner, and that the Windows PATH and Environment variable are properly revised during installation.

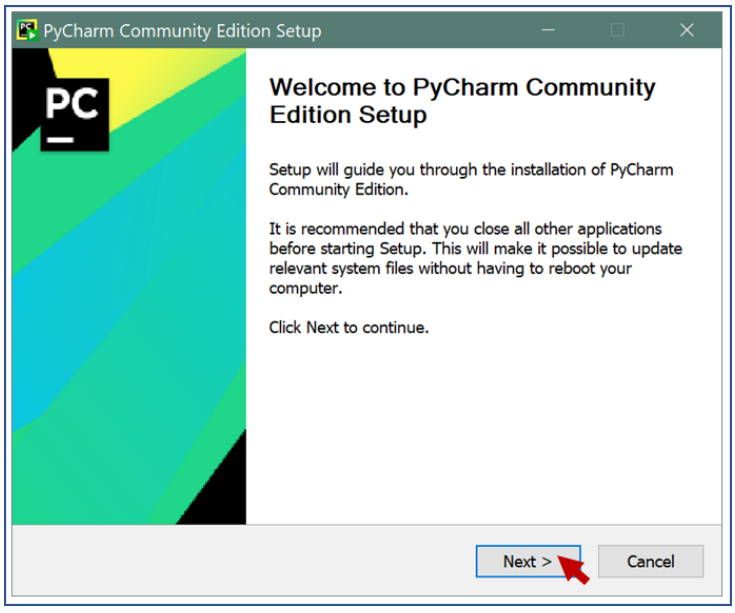

Python 3.9.1 (64-bit) Setup
Install Python 3.9.1 (64-bit)
Select Install Now to install Python with default settings, or choose
Customize to enable or disable features.
Install Now
C:\Users\Patrick\AppData\Local\Programs\Python\Python39
Includes IDLE, pip and documentation
Creates shortcuts and file associations
Customize installation
Choose location and features
3
python for windows
1
2
Install launcher for all users (recommended)
Add Python 3.9 to PATH
Cancel

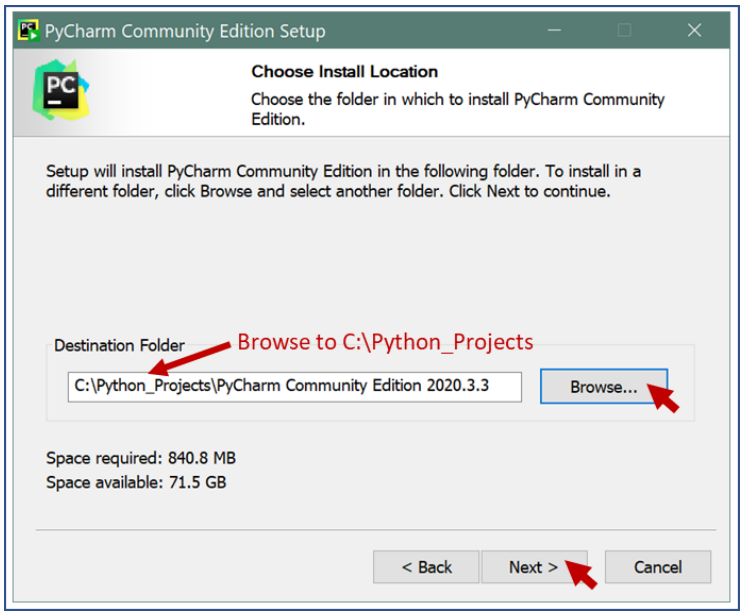

Python 3.9.1 (64-bit) Setup
Optional Features
Documentation
Installs the Python documentation file.
pip
Installs pip, which can download and install other Python packages.
tcl/tk and IDLE
Installs tkinter and the IDLE development environment.
Python test suite
Installs the standard library test suite.
py launcher
for all users (requires elevation)
Installs the global 'py' launcher to make it easier to start Python.
python for windows
Back
Next
Cancel

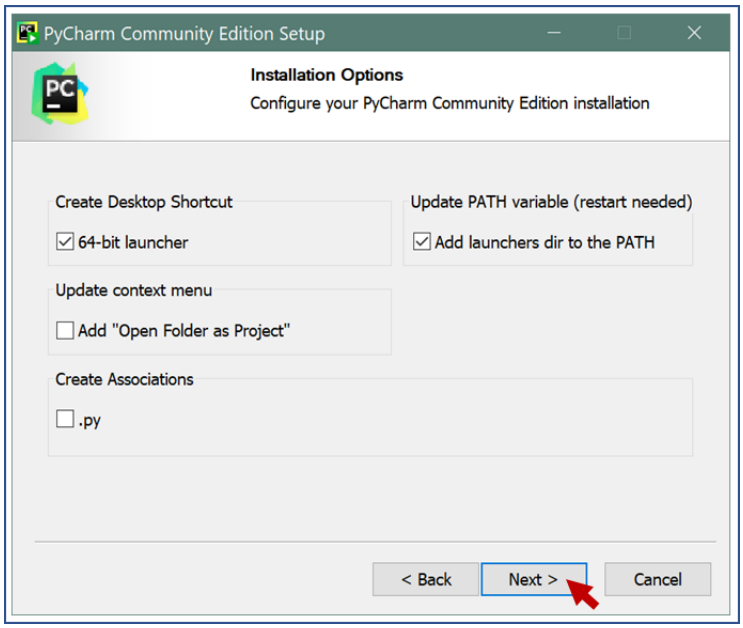

Python 3.9.1 (64-bit) Setup
Advanced Options
Install for all users
Associate files with Python (requires the py launcher)
Create shortcuts for installed applications
Add Python to environment variables
Precompile standard library
Download debugging symbols
Download debug binaries (requires VS 2017 or later)
Customize install location
C:\Python_Projects
Browse to this directory
Browse
You will require write permissions for the selected location.
python for windows
Back
Install
Cancel

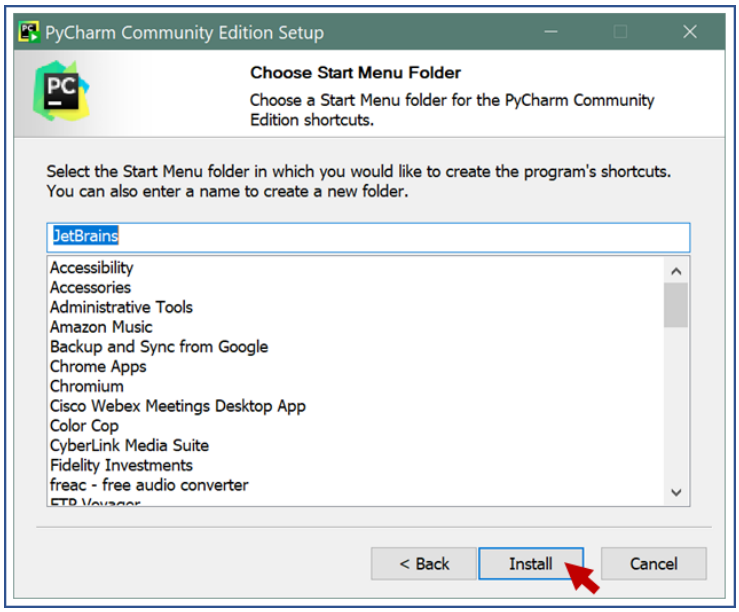


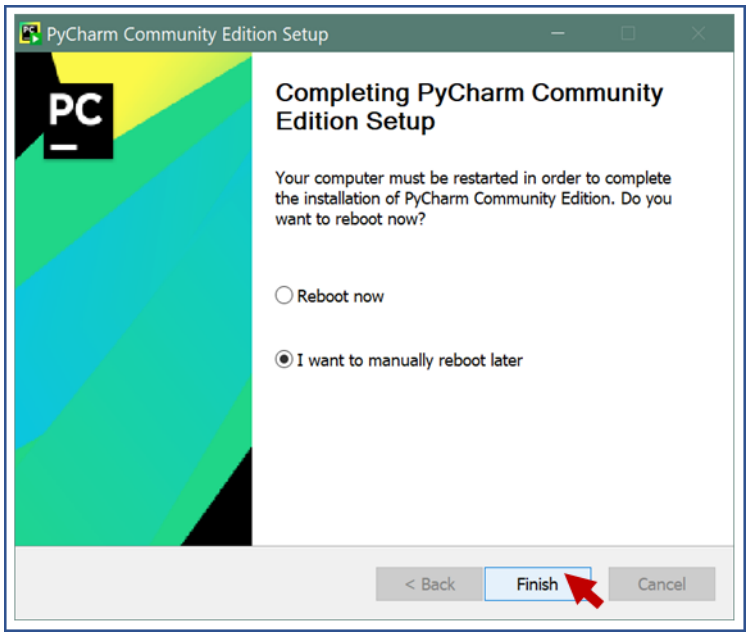
- **PyCharm** – from the Command Prompt in the C:\Python_Projects directory, execute the following command: **pycharm-community-2020.3.3.exe**, and follow the instructions in the graphics below.



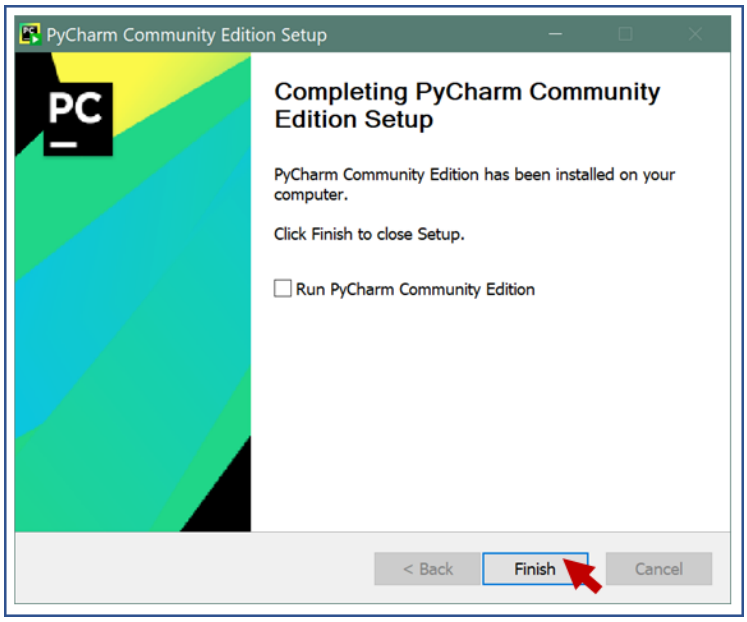

PyCharm Community Edition Setup
Choose Install Location
Choose the folder in which to install PyCharm Community Edition.
Setup will install PyCharm Community Edition in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
Browse to C:\Python_Projects
C:\Python_Projects\PyCharm Community Edition 2020.3.3
Browse...
Space required: 840.8 MB
Space available: 71.5 GB
< Back
Next >
Cancel


PyCharm Community Edition Setup
Installation Options
Configure your PyCharm Community Edition installation
Create Desktop Shortcut
64-bit launcher
Update PATH variable (restart needed)
Add launchers dir to the PATH
Update context menu
Add "Open Folder as Project"
Create Associations
.py
< Back
Next >
Cancel





(We will reboot after we install the additional software below.)

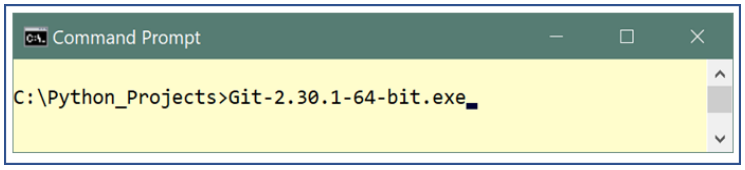


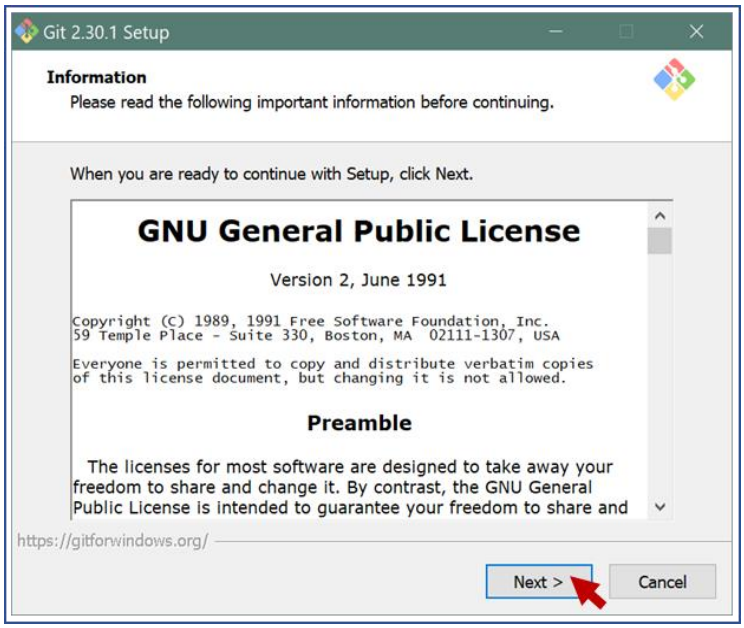
After PyCharm is installed, pin it to the taskbar, as you did for the Command Prompt.

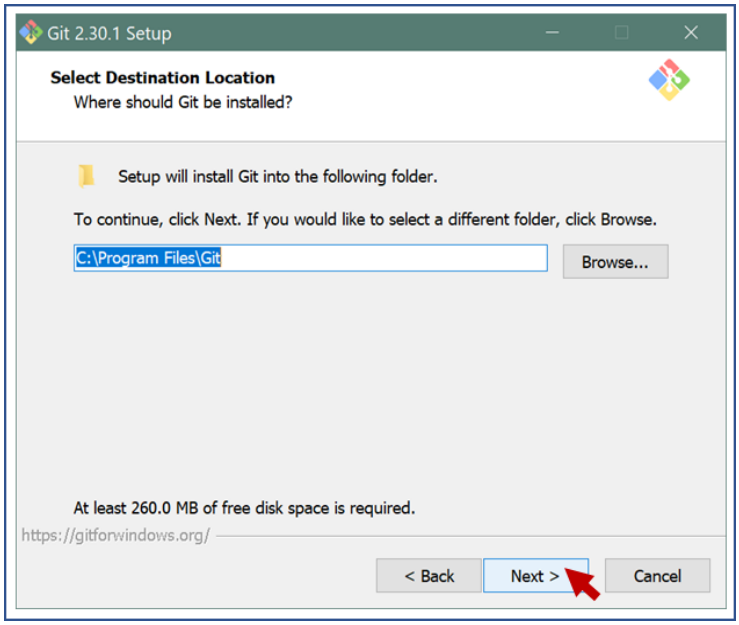
5. Installing Non-Python Software in Default Directories

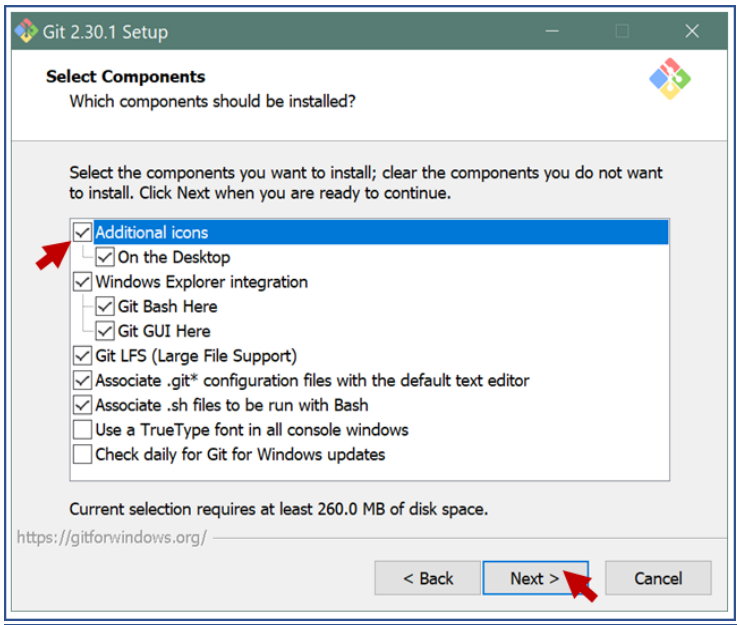
In this section we install non-Python related software in their default install directories. These include Git, GitHub, R and RStudio.

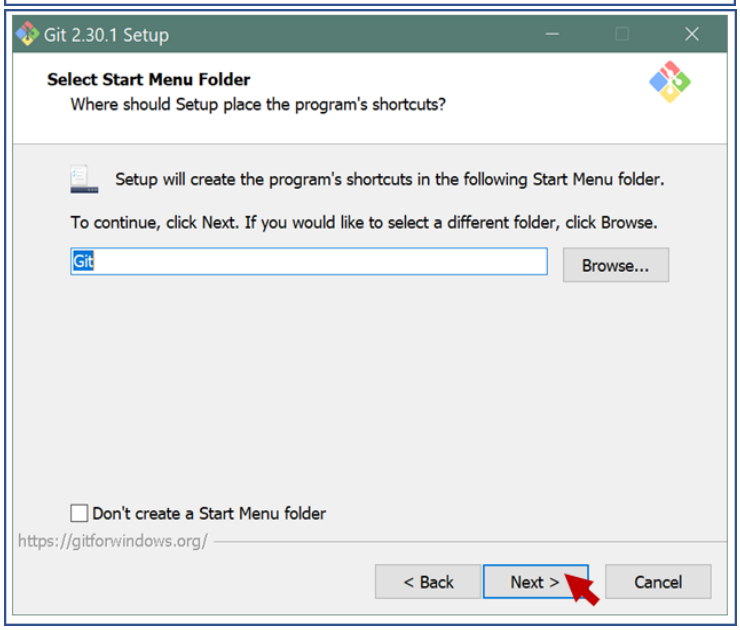
- **Git** – from the Command Prompt in the C:\Python_Projects directory, execute the following command: **Git-2.30.1-64-bit.exe**, and follow the instructions in the graphics below.

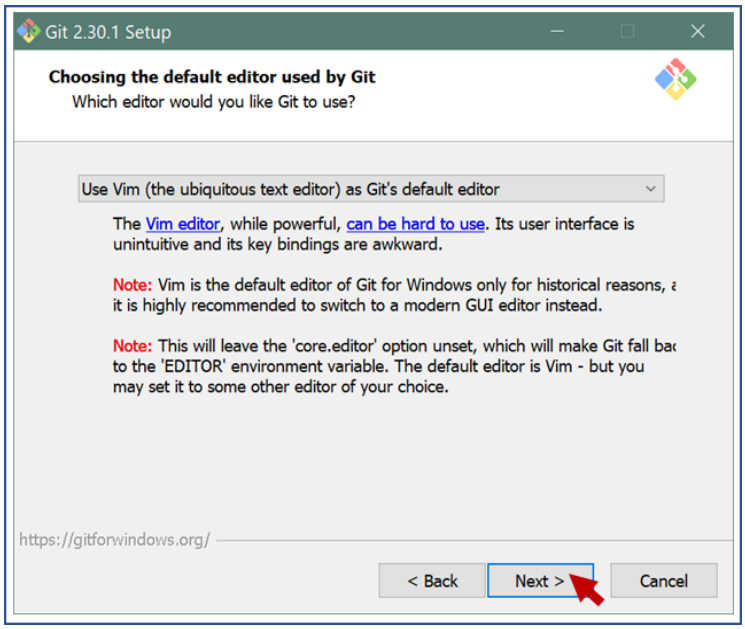


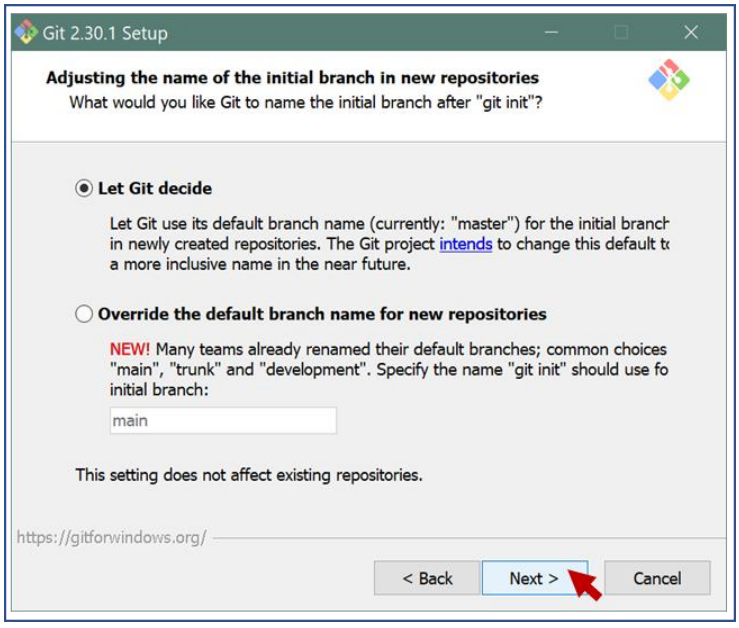

Git 2.30.1 Setup
Information
Please read the following important information before continuing.
When you are ready to continue with Setup, click Next.
GNU General Public License
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
Preamble
The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General
Public License is intended to guarantee your freedom to share and
https://gitforwindows.org/
Next >
Cancel

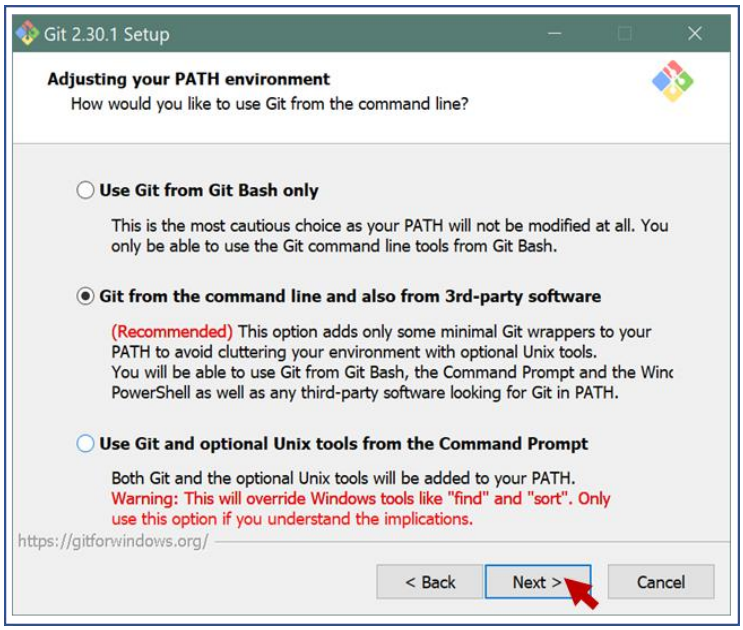

Git 2.30.1 Setup
Select Destination Location
Where should Git be installed?
Setup will install Git into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Program Files\Git
Browse...
At least 260.0 MB of free disk space is required.
https://gitforwindows.org/
< Back
Next >
Cancel

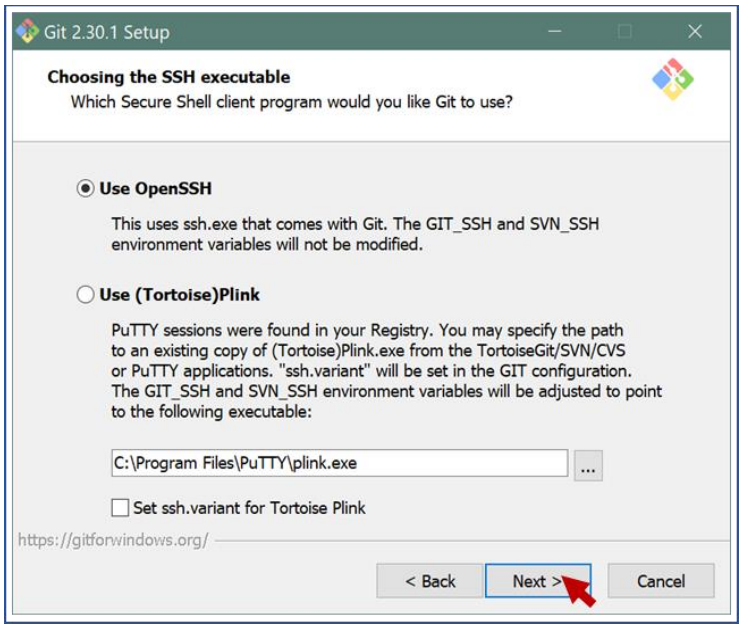

Git 2.30.1 Setup
Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Additional icons
On the Desktop
Windows Explorer integration
Git Bash Here
Git GUI Here
Git LFS (Large File Support)
Associate .git* configuration files with the default text editor
Associate .sh files to be run with Bash
Use a TrueType font in all console windows
Check daily for Git for Windows updates
Current selection requires at least 260.0 MB of disk space.
https://gitforwindows.org/
< Back
Next >
Cancel

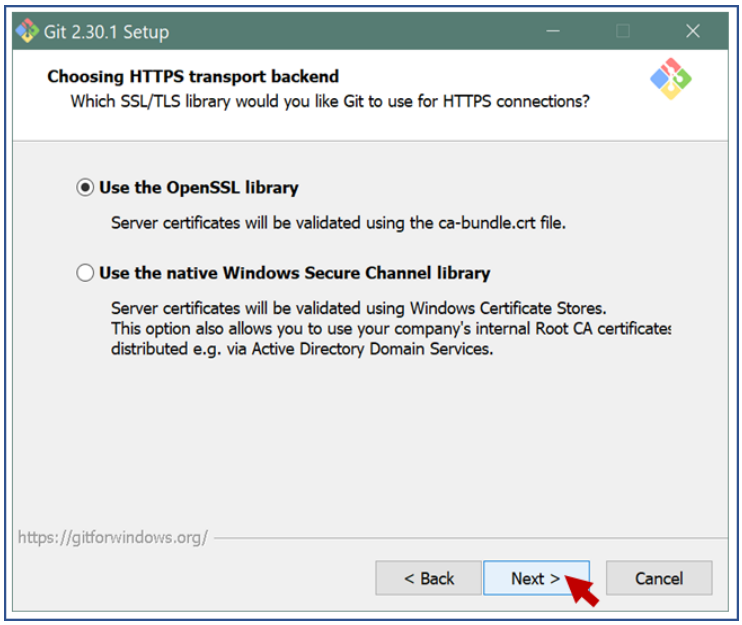

Git 2.30.1 Setup
Select Start Menu Folder
Where should Setup place the program's shortcuts?
Setup will create the program's shortcuts in the following Start Menu folder.
To continue, click Next. If you would like to select a different folder, click Browse.
Git
Browse...
Don't create a Start Menu folder
https://gitforwindows.org/
< Back
Next >
Cancel

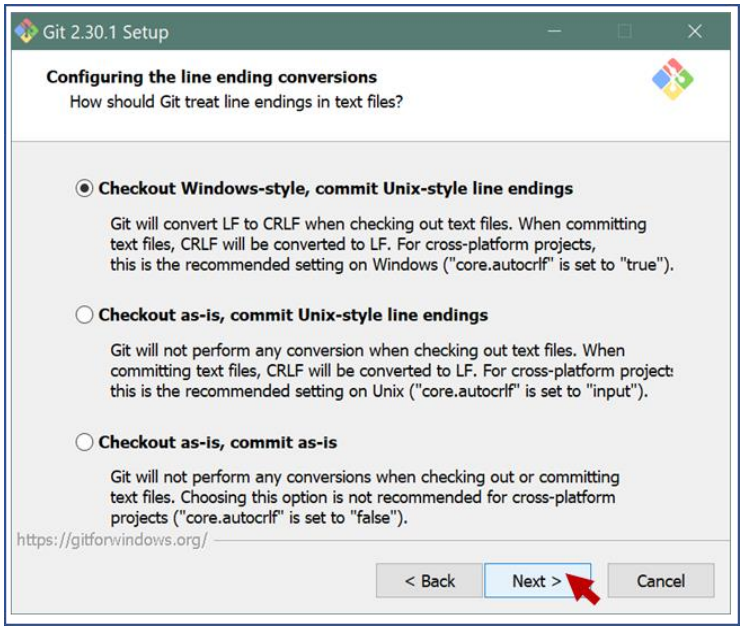

Git 2.30.1 Setup
Choosing the default editor used by Git
Which editor would you like Git to use?
Use Vim (the ubiquitous text editor) as Git's default editor
The Vim editor, while powerful, can be hard to use. Its user interface is unintuitive and its key bindings are awkward.
Note: Vim is the default editor of Git for Windows only for historical reasons, a
it is highly recommended to switch to a modern GUI editor instead.
Note: This will leave the 'core.editor' option unset, which will make Git fall bac
to the 'EDITOR' environment variable. The default editor is Vim - but you
may set it to some other editor of your choice.
https://gitforwindows.org/
< Back
Next >
Cancel

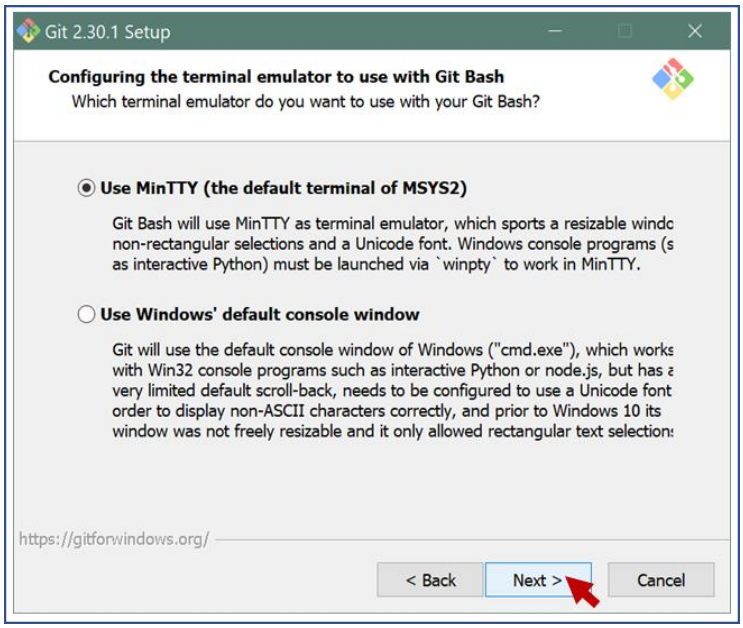

Git 2.30.1 Setup
Adjusting the name of the initial branch in new repositories
What would you like Git to name the initial branch after "git init"?
Let Git decide
Let Git use its default branch name (currently: "master") for the initial branch
in newly created repositories. The Git project intends to change this default to
a more inclusive name in the near future.
Override the default branch name for new repositories
NEW! Many teams already renamed their default branches; common choices
"main", "trunk" and "development". Specify the name "git init" should use fo
initial branch:
main
This setting does not affect existing repositories.
https://gitforwindows.org/
< Back
Next >
Cancel

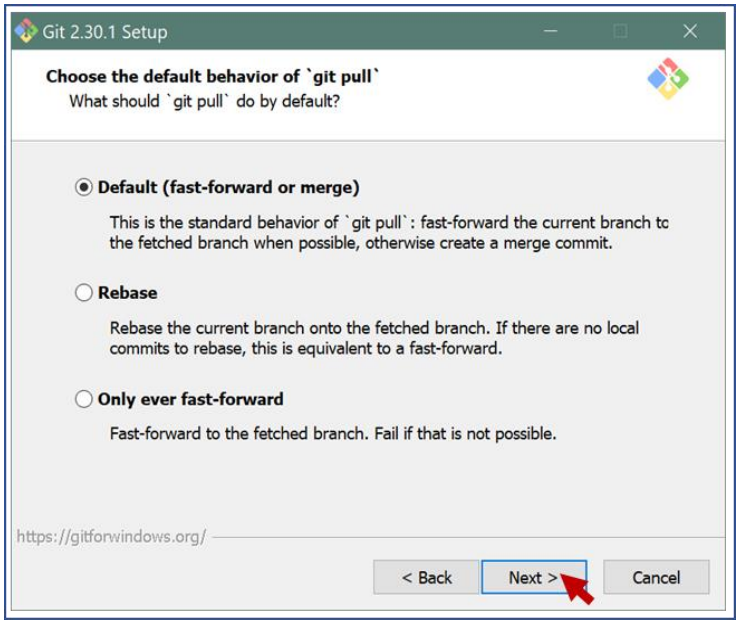

Git 2.30.1 Setup
Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git from Git Bash only
This is the most cautious choice as your PATH will not be modified at all. You only be able to use the Git command line tools from Git Bash.
Git from the command line and also from 3rd-party software
(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools.
You will be able to use Git from Git Bash, the Command Prompt and the Winc PowerShell as well as any third-party software looking for Git in PATH.
Use Git and optional Unix tools from the Command Prompt
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.
https://gitforwindows.org/
< Back
Next >
Cancel

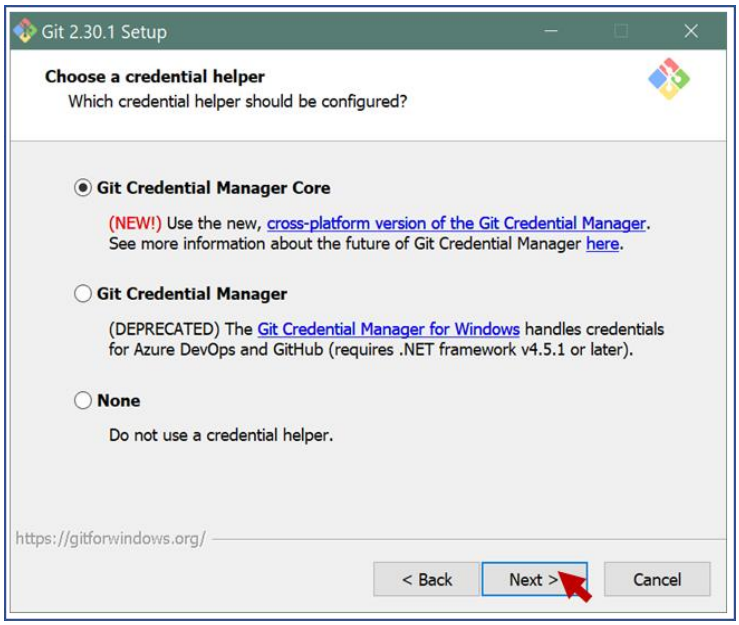

Git 2.30.1 Setup
Choosing the SSH executable
Which Secure Shell client program would you like Git to use?
Use OpenSSH
This uses ssh.exe that comes with Git. The GIT_SSH and SVN_SSH environment variables will not be modified.
Use (Tortoise)Plink
PuTTY sessions were found in your Registry. You may specify the path to an existing copy of (Tortoise)Plink.exe from the TortoiseGit/SVN/CVS or PuTTY applications. "ssh.variant" will be set in the GIT configuration. The GIT_SSH and SVN_SSH environment variables will be adjusted to point to the following executable:
C:\Program Files\PuTTY\plink.exe
...
Set ssh.variant for Tortoise Plink
https://gitforwindows.org/
< Back
Next >
Cancel

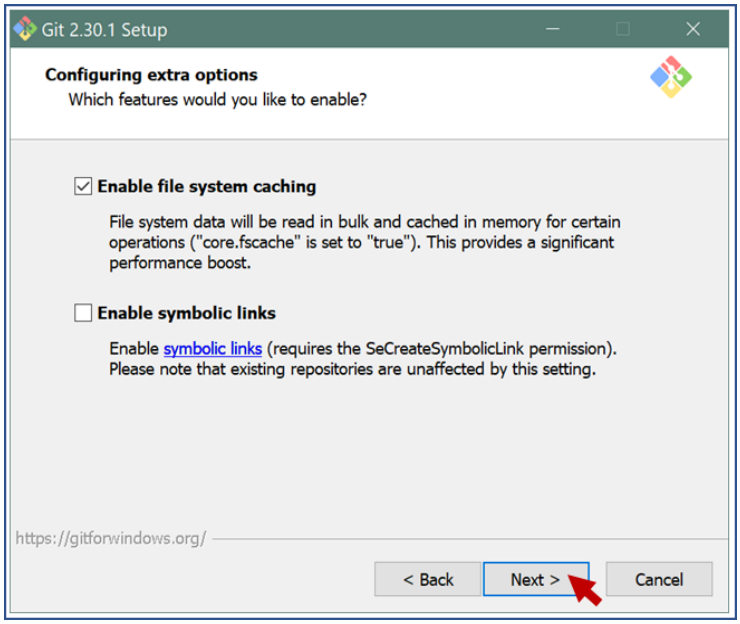

Git 2.30.1 Setup
Choosing HTTPS transport backend
Which SSL/TLS library would you like Git to use for HTTPS connections?
Use the OpenSSL library
Server certificates will be validated using the ca-bundle.crt file.
Use the native Windows Secure Channel library
Server certificates will be validated using Windows Certificate Stores.
This option also allows you to use your company's internal Root CA certificates
distributed e.g. via Active Directory Domain Services.
https://gitforwindows.org/
< Back
Next >
Cancel

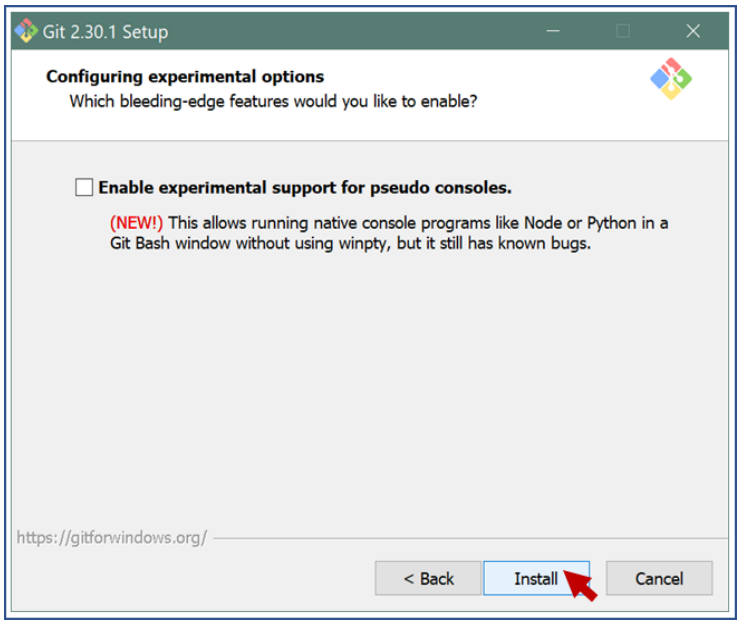

Git 2.30.1 Setup
Configuring the line ending conversions
How should Git treat line endings in text files?
Checkout Windows-style, commit Unix-style line endings
Git will convert LF to CRLF when checking out text files. When committing
text files, CRLF will be converted to LF. For cross-platform projects,
this is the recommended setting on Windows ("core.autocrlf" is set to "true").
Checkout as-is, commit Unix-style line endings
Git will not perform any conversion when checking out text files. When
committing text files, CRLF will be converted to LF. For cross-platform project
this is the recommended setting on Unix ("core.autocrlf" is set to "input").
Checkout as-is, commit as-is
Git will not perform any conversions when checking out or committing
text files. Choosing this option is not recommended for cross-platform
projects ("core.autocrlf" is set to "false").
https://gitforwindows.org/
< Back
Next >
Cancel

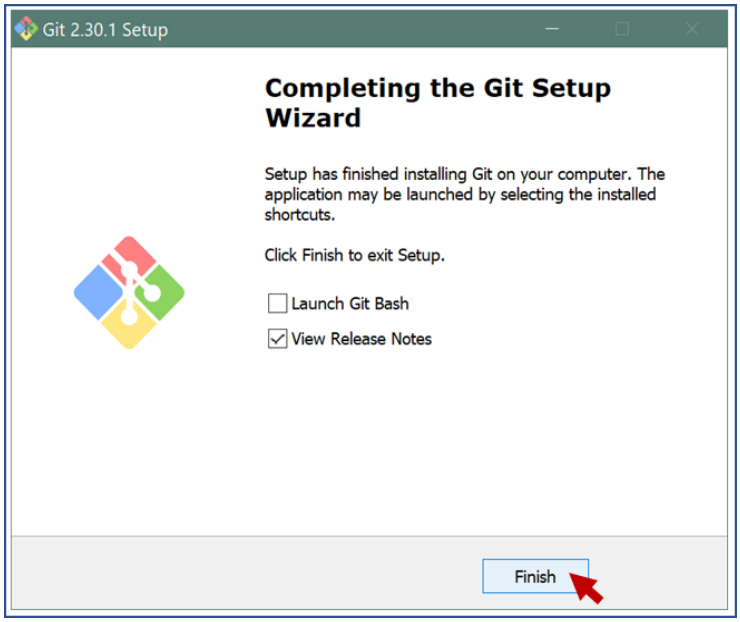

Git 2.30.1 Setup
Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?
Use MinTTY (the default terminal of MSYS2)
Git Bash will use MinTTY as terminal emulator, which sports a resizable windc
non-rectangular selections and a Unicode font. Windows console programs (s
as interactive Python) must be launched via `winpty` to work in MinTTY.
Use Windows' default console window
Git will use the default console window of Windows ("cmd.exe"), which works
with Win32 console programs such as interactive Python or node.js, but has a
very limited default scroll-back, needs to be configured to use a Unicode font
order to display non-ASCII characters correctly, and prior to Windows 10 its
window was not freely resizable and it only allowed rectangular text selection
https://gitforwindows.org/
< Back
Next >
Cancel


Git 2.30.1 Setup
Choose the default behavior of `git pull`
What should `git pull` do by default?
Default (fast-forward or merge)
This is the standard behavior of `git pull`: fast-forward the current branch tc
the fetched branch when possible, otherwise create a merge commit.
Rebase
Rebase the current branch onto the fetched branch. If there are no local
commits to rebase, this is equivalent to a fast-forward.
Only ever fast-forward
Fast-forward to the fetched branch. Fail if that is not possible.
https://gitforwindows.org/
< Back
Next >
Cancel

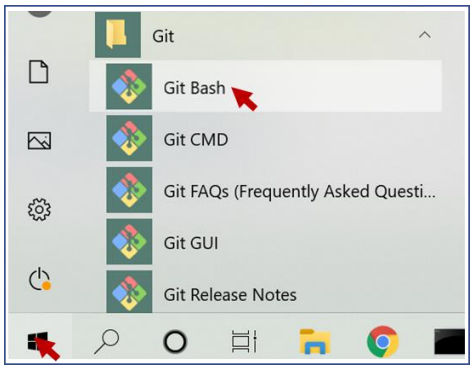

Git 2.30.1 Setup
Choose a credential helper
Which credential helper should be configured?
Git Credential Manager Core
(NEW!) Use the new, cross-platform version of the Git Credential Manager.
See more information about the future of Git Credential Manager here.
Git Credential Manager
(DEPRECATED) The Git Credential Manager for Windows handles credentials
for Azure DevOps and GitHub (requires .NET framework v4.5.1 or later).
None
Do not use a credential helper.
https://gitforwindows.org/
< Back
Next >
Cancel


Git 2.30.1 Setup
Configuring extra options
Which features would you like to enable?
Enable file system caching
File system data will be read in bulk and cached in memory for certain
operations ("core.fscache" is set to "true"). This provides a significant
performance boost.
Enable symbolic links
Enable symbolic links (requires the SeCreateSymbolicLink permission).
Please note that existing repositories are unaffected by this setting.
https://gitforwindows.org/
< Back
Next >
Cancel





Notice that when using the Start button to verify installation, Git has installed a number of applications.

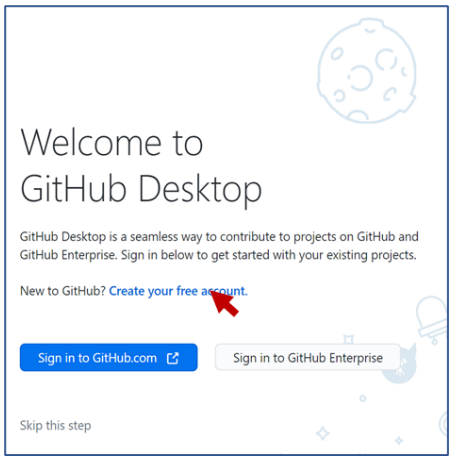


For now, pin only the **Git Bash** application to the taskbar, and move it next to PyCharm.

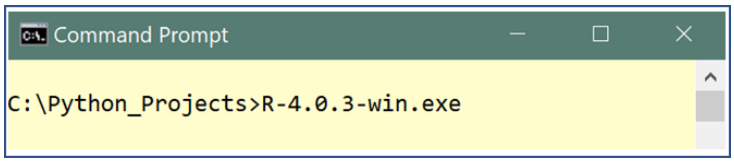
- **GitHub Desktop Studio** – from the Command Prompt in the C:\Python_Projects directory, execute the following command: **GitHubDesktopSetup.exe**.

The following screen will appear, and the software will be installed with no further input from you. (Nice, huh?)

After installation is complete, you will see the following screen which prompts you to sign in to GitHub if you have an account, or sign up for one. If you do not have an account, sign up for one by choosing the **Create your free account** hypertext. The account signup process is straightforward and not shown here.



- **R** – from the Command Prompt in the C:\Python_Projects directory, execute the following command: **R-4.0.3-win.exe**.

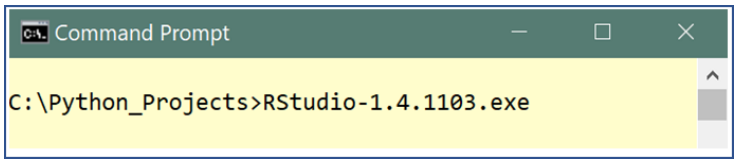


On the screens that appear next, do the following:

  - On the first **Select Setup Language** screen, choose the **OK** button to use the default language, English;
  - Choose the **Next** button on all of the screens that follow, except for the last screen, which confirms that setup is complete;
  - On the final screen, choose the **Finish** button.

Pin the 64-bit version of R to the taskbar, next to Git Bash.

- **RStudio** – from the Command Prompt in the C:\Python_Projects directory, execute the following command: **RStudio-1.4.1103.exe**.



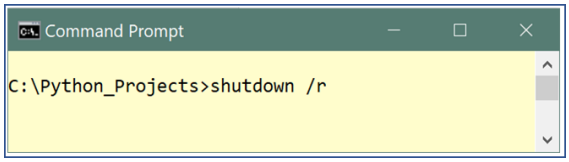
On the screens that appear next, choose the:

- **Next** button on the **first two** screens;
- **Install** on the **third** screen;
- Finish on the final screen, which indicates that setup is complete.

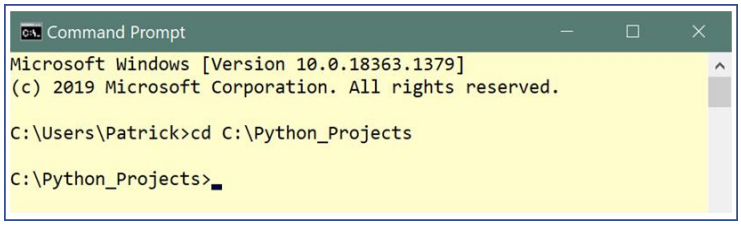
Pin RStudio to the taskbar, next to R.

6. Reboot the Machine

Now, restart your machine so that any environment variables and the PATH, which were revised in the installation process, are functional. To restart your machine, enter the following DOS command: **shutdown /r.**



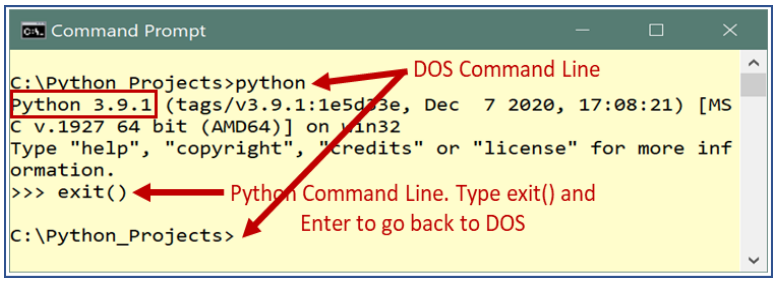
When the machine restarts, open another DOS command line, and go to the **Python_Projects** directory.



7. Verify Python and PIP Versions

Now we perform two verifications: the Python version and the version of software called PIP, which is installed with Python.

- **Python Version** - Now we verify the Python version by invoking it in the DOS Command Line by issuing the command **python**. This starts Python and returns the Python version information, and the Python command line appears, as below. Now issue the Python command **exit()** on the Python command line to return to DOS. We will be invoking Python in this manner.

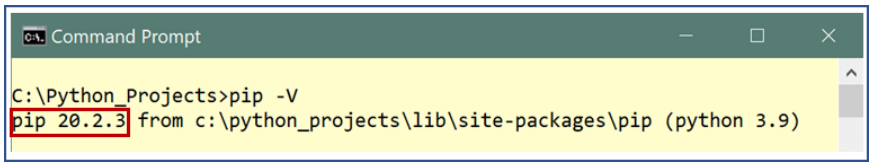


[Note: Python can be invoked in this manner from any directory. Also, we can just type command **python --version** at the DOS command line to see the Python version without going into Python.

You can also invoke Python by using the **Start** icon and choosing **Python 3.9 (64-bit),** as shown below, and the Python command line will appear in a DOS window. However, we will be using command lines frequently for other purposes, so this Start method will not be used start Python. This is because when Python is invoked via the Start icon, typing **exit()** on the Python command line will not take you back to a DOS command line, which we often want.]

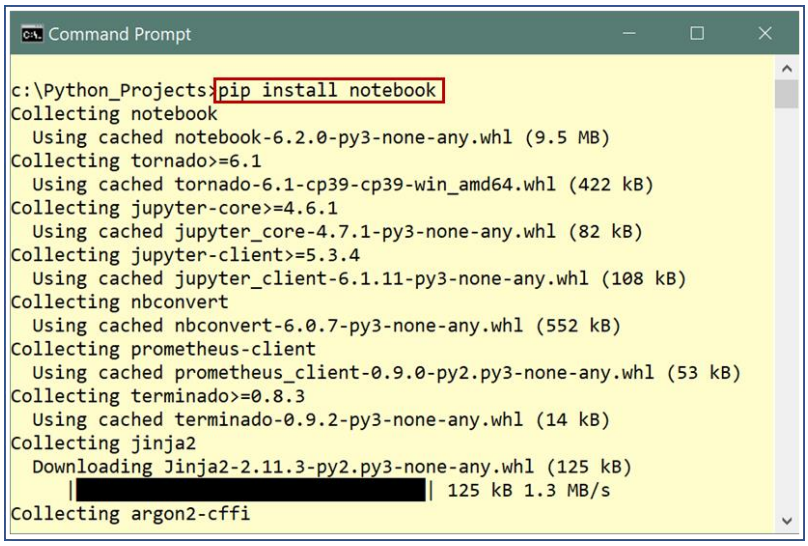
- **PIP Version** – PIP is installed with Python. Python is installed with a standard set of libraries (a.k.a., packages or modules) and functionality. However, there are many additional libraries that offer functionality far beyond the standard packages, and PIP allows users to install and manage those.

  To check the PIP version, enter the DOS command **pip -V**. The version information appears and you are returned to the DOS Command Prompt.



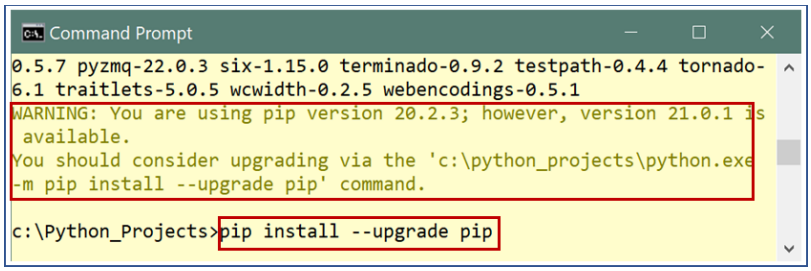
8. Installing Jupyter Notebook

Now let's use PIP to install Jupyter Notebook. Make sure you are at the **C:\Python_Proojects** directory, and enter the command **pip install notebook** on the DOS command line. The command will cause text related to the progress of the installation process, as seen in the graphic below.

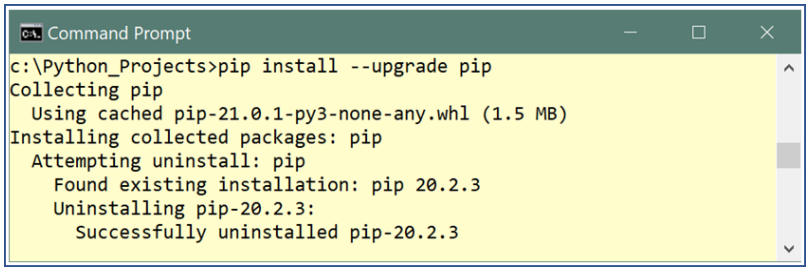


When the installation of Jupyter Notebook is complete, you will be returned to the DOS command line. We will also use pip frequently to install additional Python libraries as the need to use them arises.

[Note: When using PIP to install software, occasionally, as happened to me in writing this, your return to the DOS command line will be preceded by a warning that a newer version of PIP is available, as below.

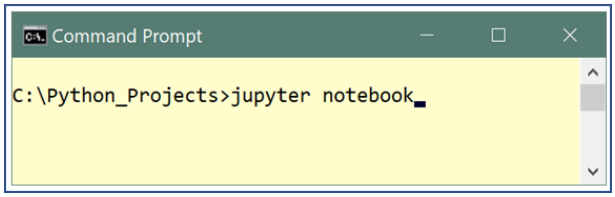


To upgrade pip, and since you are already in the Python install directory, simply enter the DOS command **pip install –upgrade pip**.



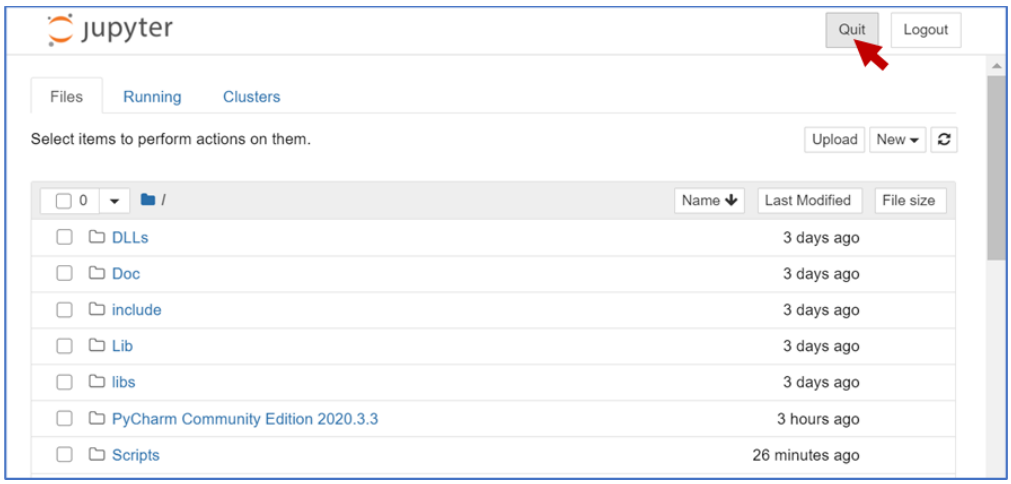
]

- **Opening Jupyter Notebook** - Now let's start Jupyter Notebook from the **C:\Python_Projects** directory by entering the DOS command **jupyter notebook**.



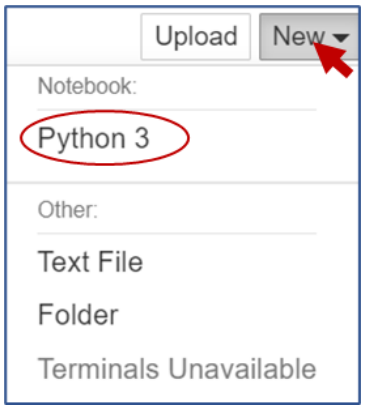
(Note: While you are working with Jupyter Notebook, do not close the Command Prompt window from which you started Jupyter, which will shut down the Jupyter server.)

This starts a Jupyter server and opens a browser with the address http://localhost:**8888/tree**. 8888 is the port through which this browser session operates, and **tree** refers to the **relative path of C:\Python_Projects, the directory from which the invocation was made**. Looking at the directory with the browser, we can see that the contents of the C:\Python_Projects directory are visible.



In fact, Jupyter Notebook can be invoked from **any** directory, and **the contents of that directory and everything below it will be accessible** in the browser. This allows us to navigate to whichever directory where our code resides, invoke Jupyter Notebook, and work from there.

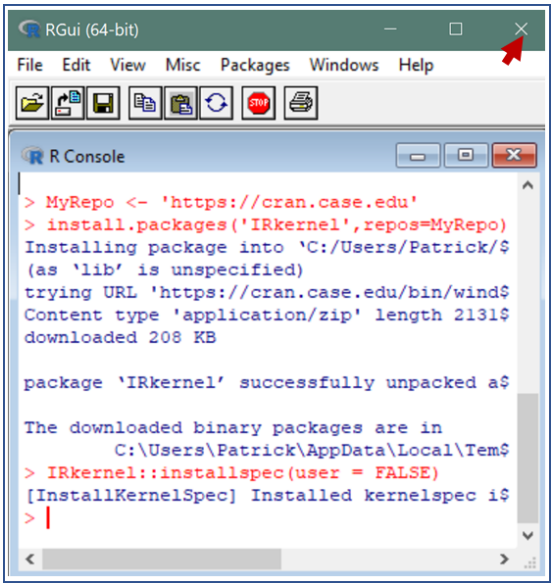
Note also that, currently, Jupyter Notebook can only support Python3 (or what is called the Python3 kernel). This is shown when trying to create a new notebook. When we choose **New**, our only choice is Pyhton3, as below.



In the next section we add the R kernel to Jupyter notebook, so we can also write and submit R code.

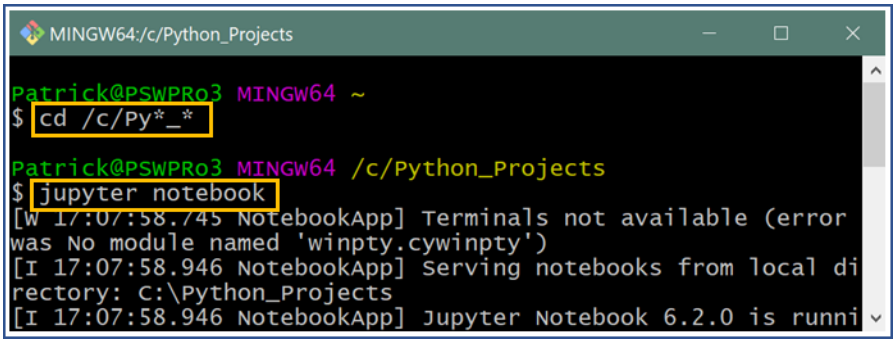
To quit Jupyter fully, choose the **Quit** button on the upper right, which stops the Jupyter server, and **close the browser** window.

- **Adding the R Kernel to Jupyter Notebook** – We accomplish this by installing a package in R that will allow Jupyter Notebook access the R kernel:
  - Start R using its **taskbar icon**.
  - In the command line in the R Console, enter the following R commands and press enter after each:
    - MyRepo <- 'https://cran.case.edu'
    - install.packages('IRkernel',repos=MyRepo)
    - IRkernel::installspec(user = FALSE)

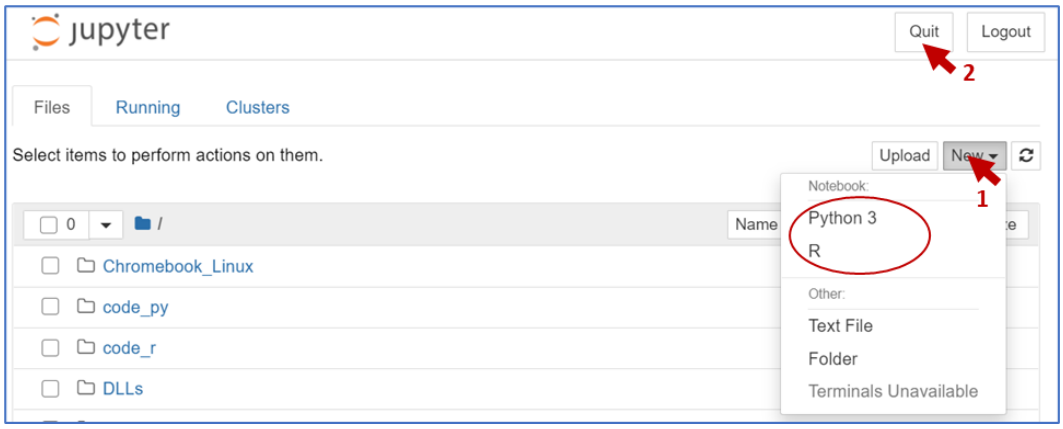


- The R Console can be closed.

- Just for fun, let's start up Jupyter Notebook using Git Bash, which uses Linux commands. Start Git Bash from its taskbar shortcut.
- In the Git Bash command line, enter the following two commands, pressing enter after each:
  - **cd /c/Py*_* (use wildcards)**
  - **jupyter notebook**



- A browser opens up with Jupyter Notebook open. However, now we can create R notebooks also, because that kernel can now be accessed by Notebook. Choose Quit to stop the server, and close the browser.



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