



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

Windows®

Python®, PIP®

GitHub®

Git®

R®

RStudio®

Jupyter®

Linux®

##### Organization to which registered

Microsoft, Inc.

Python Software Foundation

GitHub, Inc.

Software Freedom Conservancy

The R Foundation for Statistical Computing

RStudio, PBC

NumFOCUS Foundation

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 the user to execute Linux commands on a Windows environment, thus giving users with little or no Linux exposure the opportunity to learn and use Linux commands on this 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.

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®, Coursera®, Data Camp®, etc.

Good luck and good learning!

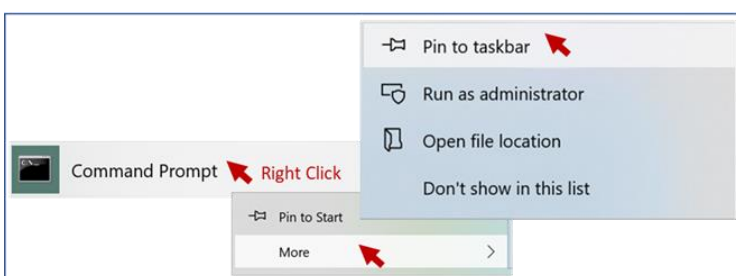
## 1. Pin the Command Icon to Taskbar

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)

```

C:\Users\Patrick>cd C:\
C:\>mkdir Python_Projects
C:\>dir Py*_*
Volume in drive C is Windows
Volume Serial Number is 0AE0-C2BA

Directory of C:\

02/12/2021  08:50 AM    <DIR>          Python_Projects
               0 File(s)                0 bytes
               1 Dir(s)  77,441,712,128 bytes free

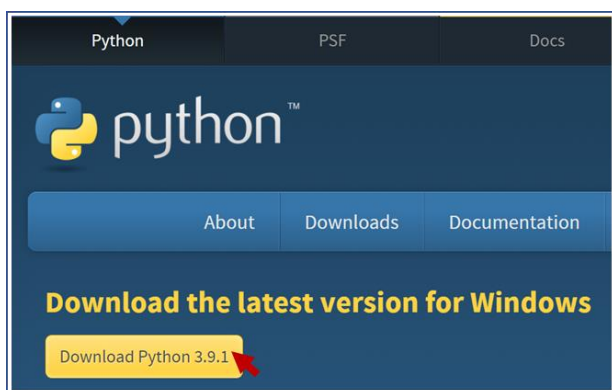
C:\>_
  
```

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.

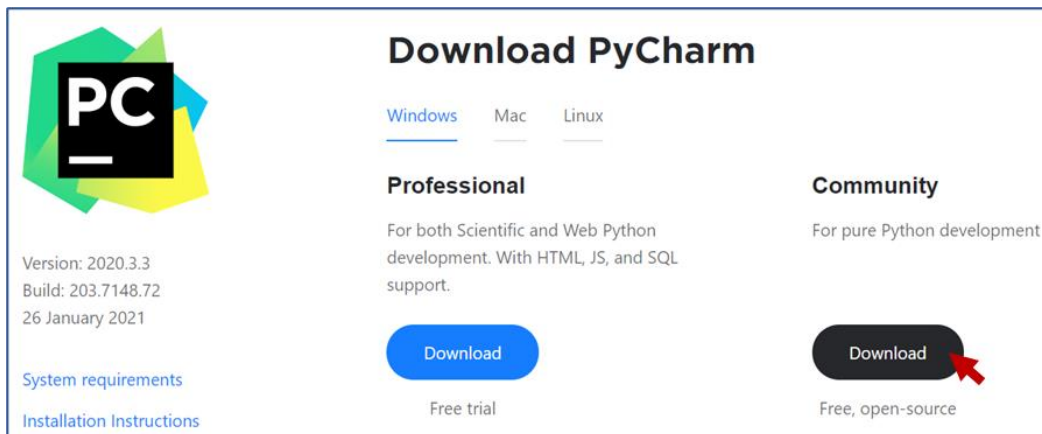
### 3. Download and Move Software to Python\_Projects Directory

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.



The image shows the PyCharm download page. On the left is the PyCharm logo (a green and yellow hexagon with 'PC' and a minus sign). Below it, version and build information are listed: Version: 2020.3.3, Build: 203.7148.72, 26 January 2021. There are links for 'System requirements' and 'Installation Instructions'. In the center, under the 'Windows' tab, the 'Professional' edition is highlighted. It is described as 'For both Scientific and Web Python development. With HTML, JS, and SQL support.' and has a blue 'Download' button labeled 'Free trial'. On the right, the 'Community' edition is shown, described as 'For pure Python development', with a dark blue 'Download' button labeled 'Free, open-source'. A red arrow points to the Community download button.

- **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.



The image shows the 'Downloading Git' page. It features a large downward arrow icon. Text states: 'You are downloading the latest (2.30.1) 64-bit version of Git for Windows. This is the most recent maintained build. It was released 3 days ago, on 2021-02-09.' Below this is a link 'Click here to download manually'. A section titled 'Other Git for Windows downloads' lists 'Git for Windows Setup', '32-bit Git for Windows Setup.', and '64-bit Git for Windows Setup.'. A red arrow points to the '64-bit Git for Windows Setup.' link.

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

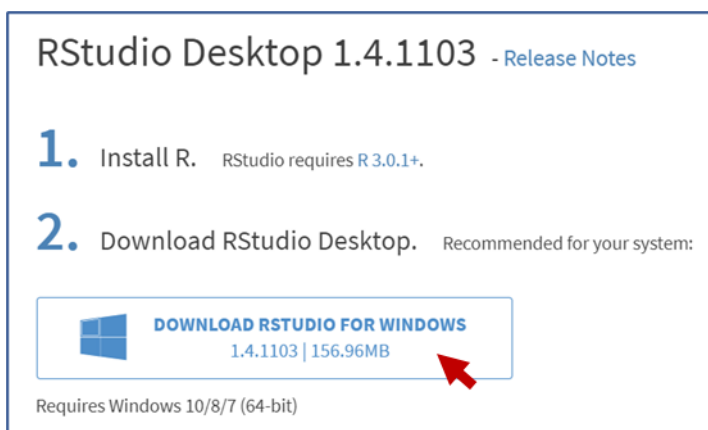


The image shows the GitHub Desktop download page. It has a dark background with the GitHub logo (a white cat face in a purple circle) at the top center. Navigation links 'Overview', 'Release Notes', and 'Help' are visible. The title 'GitHub Desktop' is prominently displayed. Below it, a tagline reads: 'Focus on what matters instead of fighting with Git. Whether you're new to Git or a seasoned user, GitHub Desktop simplifies your development workflow.' At the bottom, there is a purple button labeled 'Download for Windows (64bit)'. A red arrow points to this button.

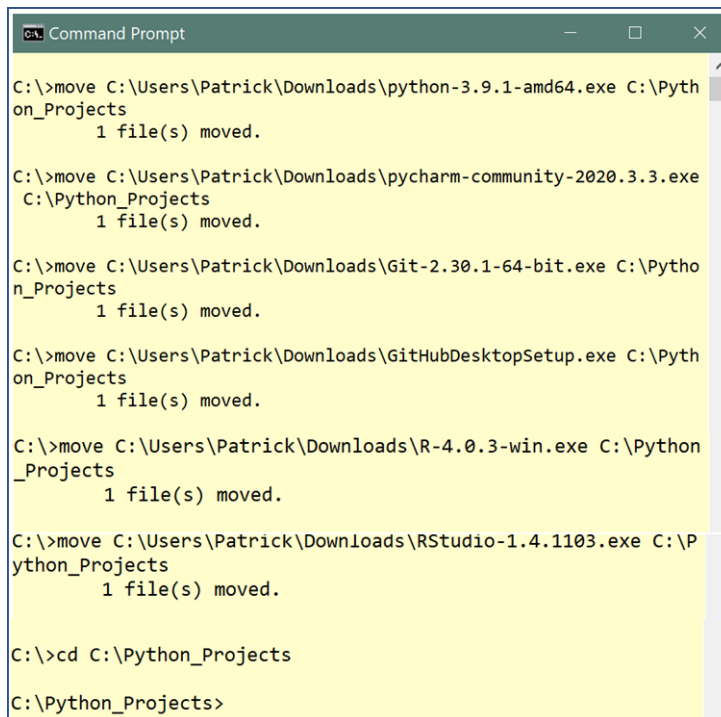
- **R** – go to <https://cran.r-project.org/bin/windows/base/> and choose the **Download R 4.0.3 for Windows** hypertext. The **R-4.0.3-win.exe** file will be downloaded.



- **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**



```
Command Prompt

C:\>move C:\Users\Patrick\Downloads\python-3.9.1-amd64.exe C:\Python_Projects
1 file(s) moved.

C:\>move C:\Users\Patrick\Downloads\pycharm-community-2020.3.3.exe C:\Python_Projects
1 file(s) moved.

C:\>move C:\Users\Patrick\Downloads\Git-2.30.1-64-bit.exe C:\Python_Projects
1 file(s) moved.

C:\>move C:\Users\Patrick\Downloads\GitHubDesktopSetup.exe C:\Python_Projects
1 file(s) moved.

C:\>move C:\Users\Patrick\Downloads\R-4.0.3-win.exe C:\Python_Projects
1 file(s) moved.

C:\>move C:\Users\Patrick\Downloads\RStudio-1.4.1103.exe C:\Python_Projects
1 file(s) moved.

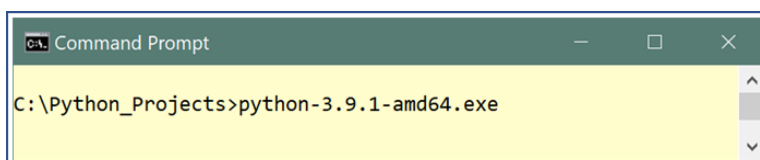
C:\>cd C:\Python_Projects
C:\Python_Projects>
```

#### 4. Installing Python-related Software in the Python\_Projects Directory

In this section we install Python-related software in the C:\Python\_Projects 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**

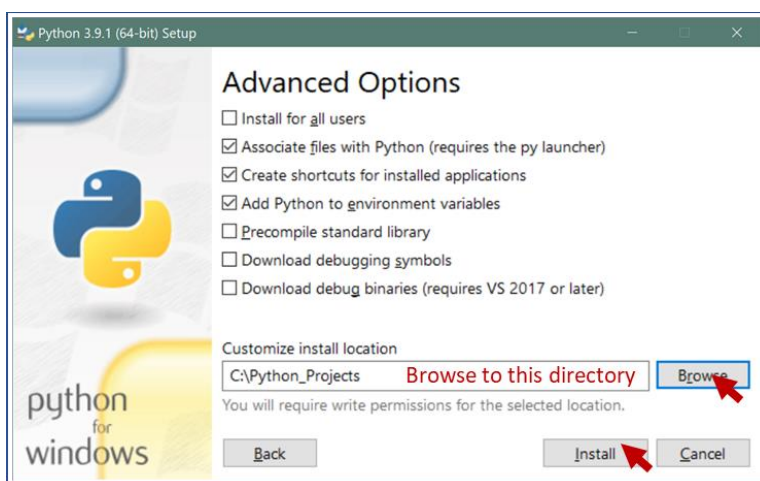
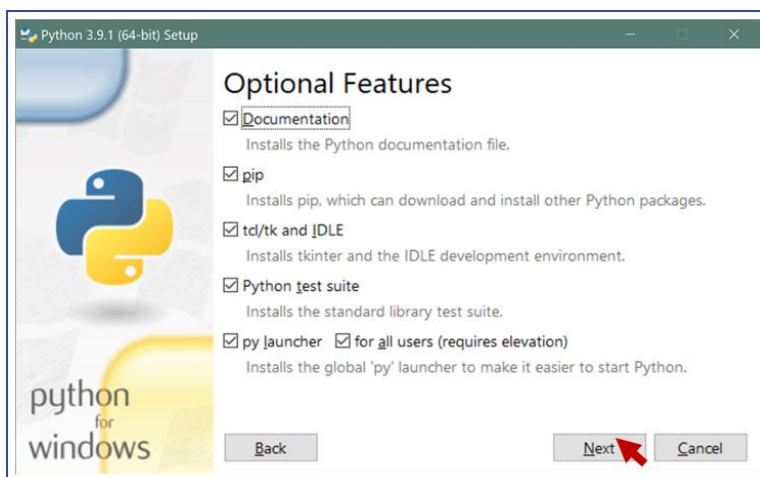
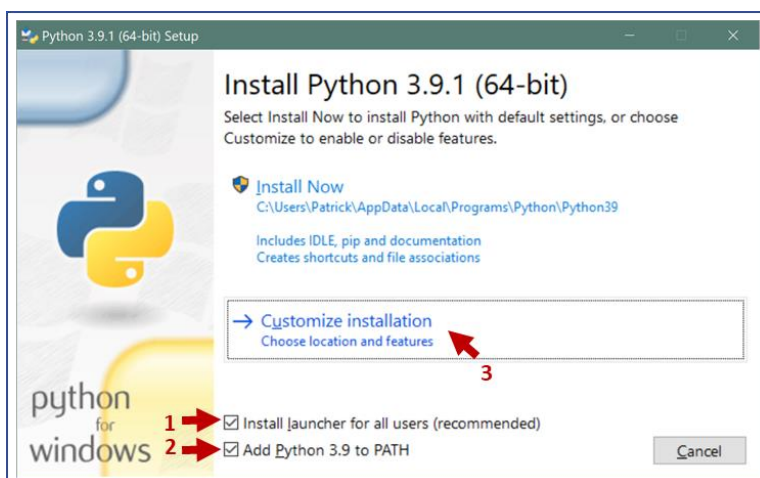


```
Command Prompt

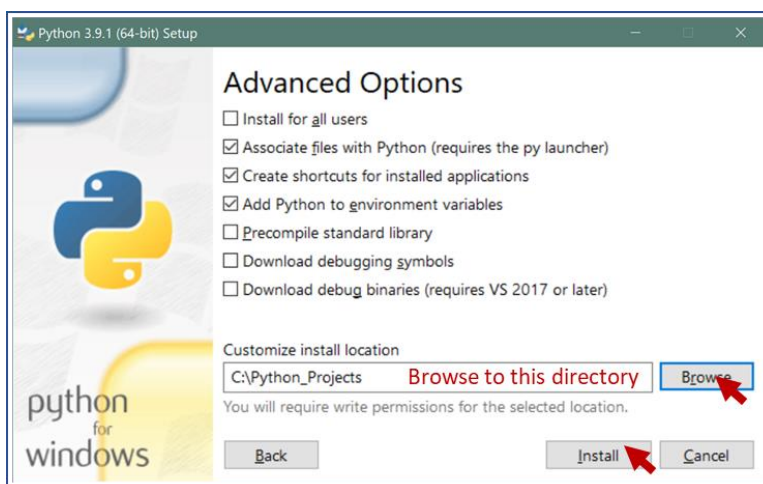
C:\Python_Projects>python-3.9.1-amd64.exe
```

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.

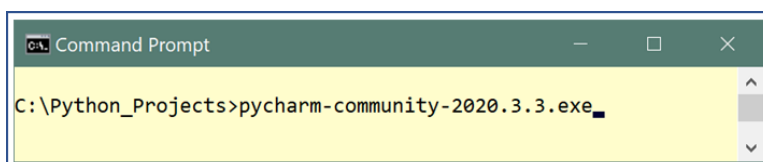


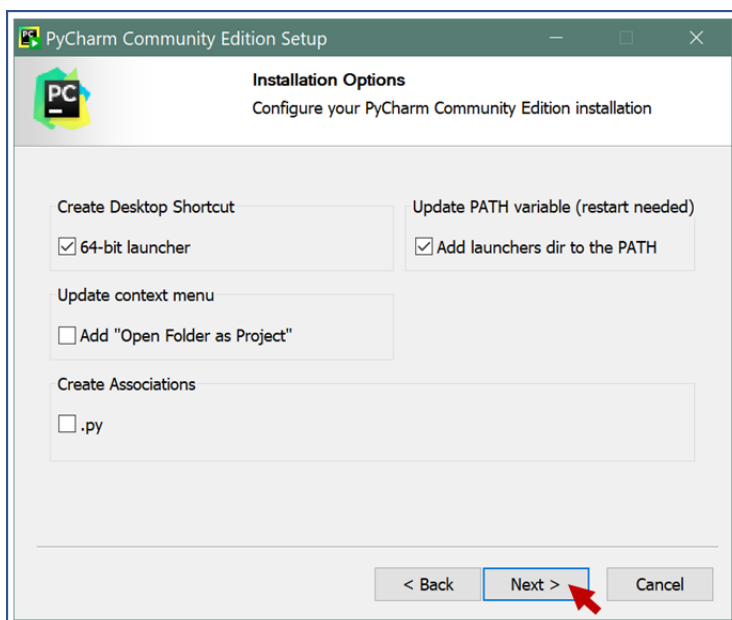
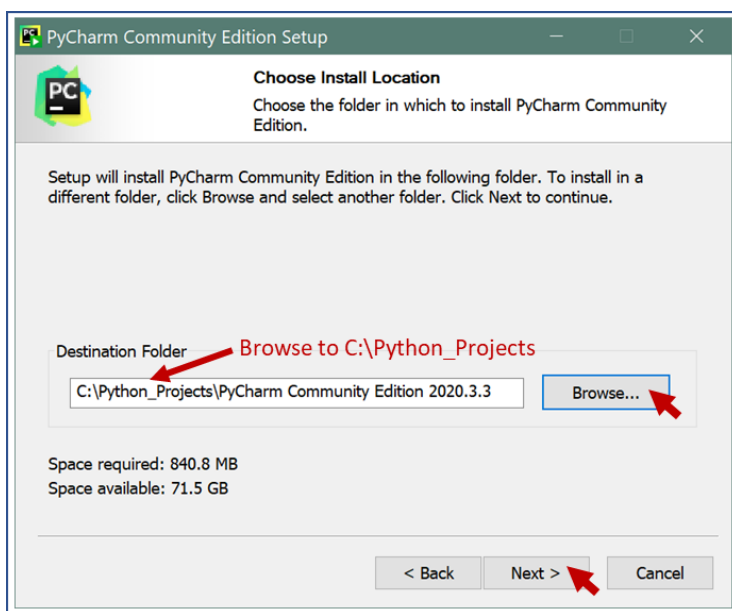


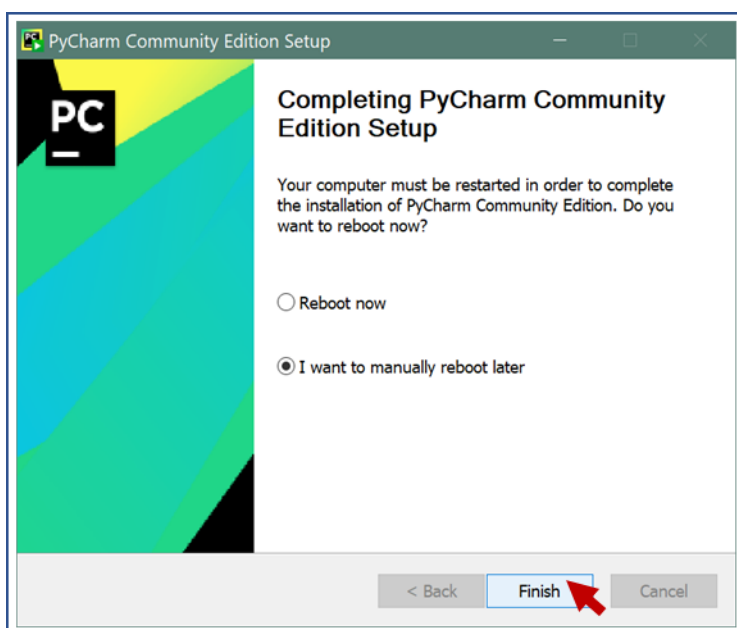
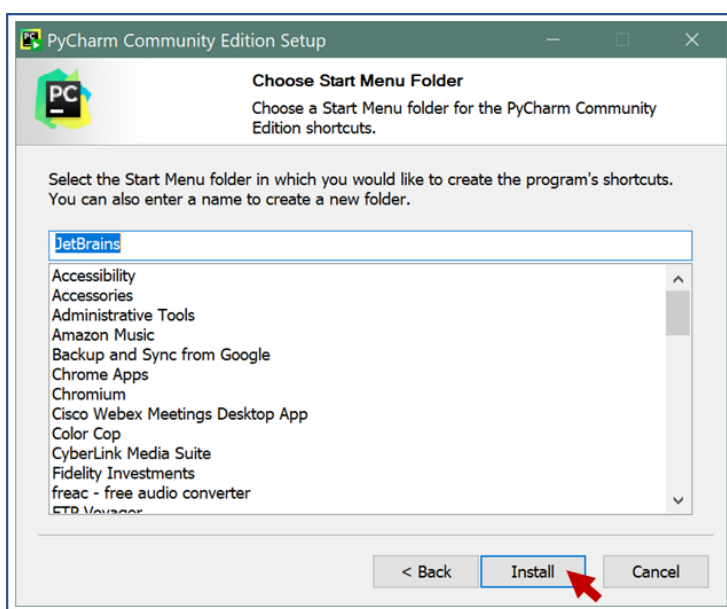




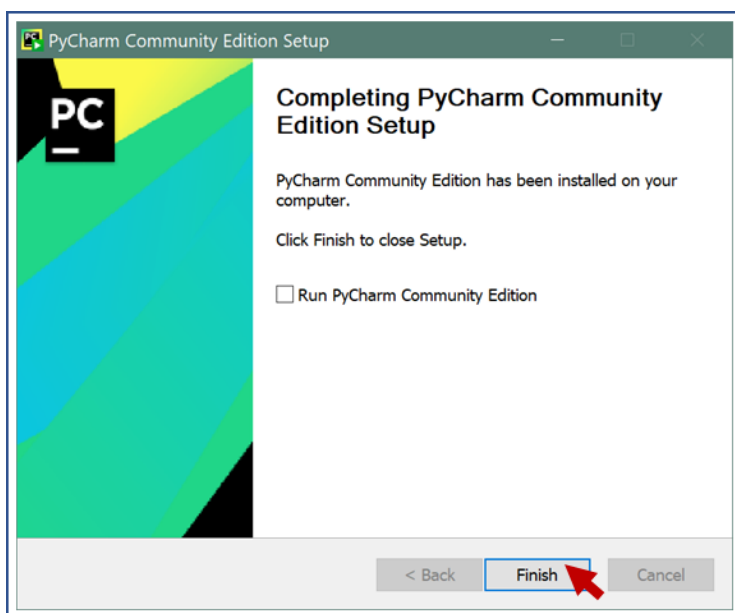
- **PyCharm** – from the Command Prompt in the C:\Python\_Projects directory, execute the following command: **pycharm-community-2020.3.3.exe**, and follow the instructions in the graphics below.







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



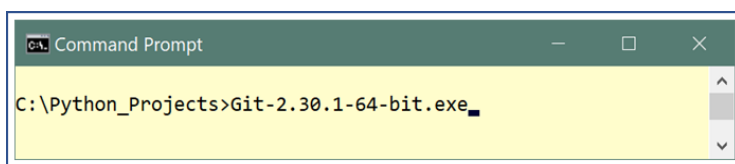
After PyCharm is installed, pin it to the taskbar, as you did for the Command Prompt.

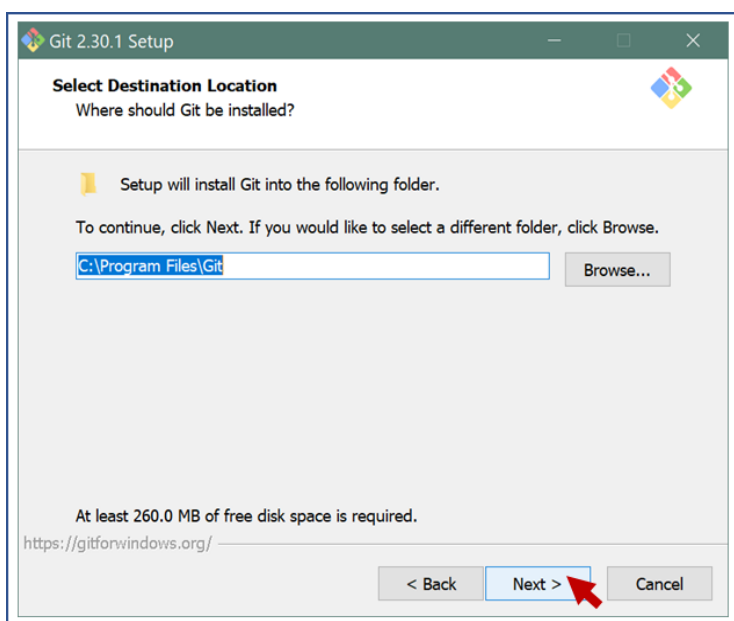


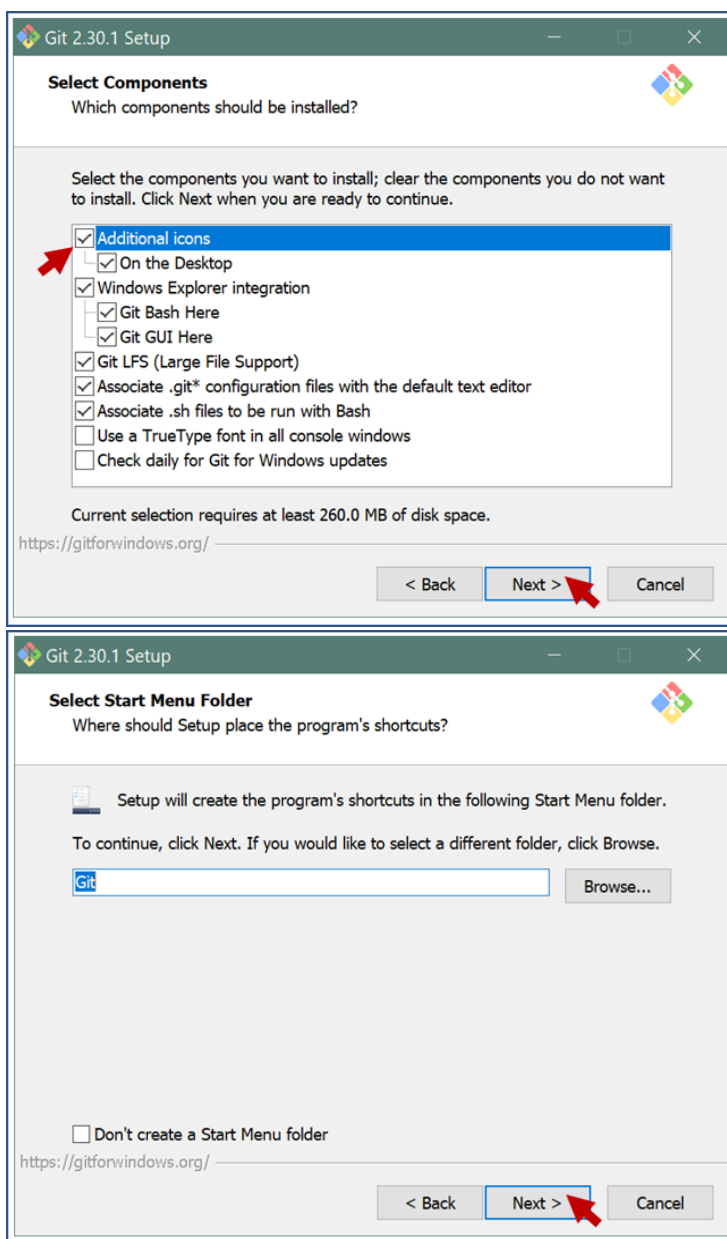
## 5. Installing Non-Python Software in Default Directories

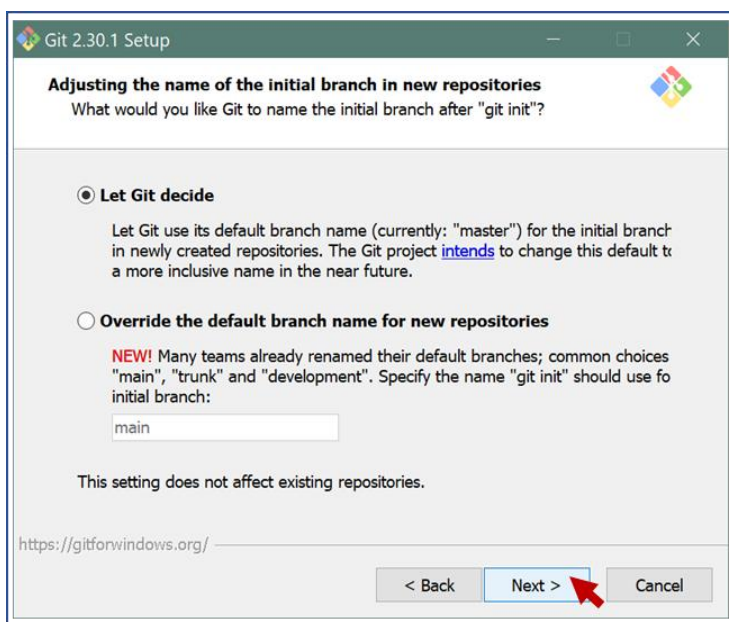
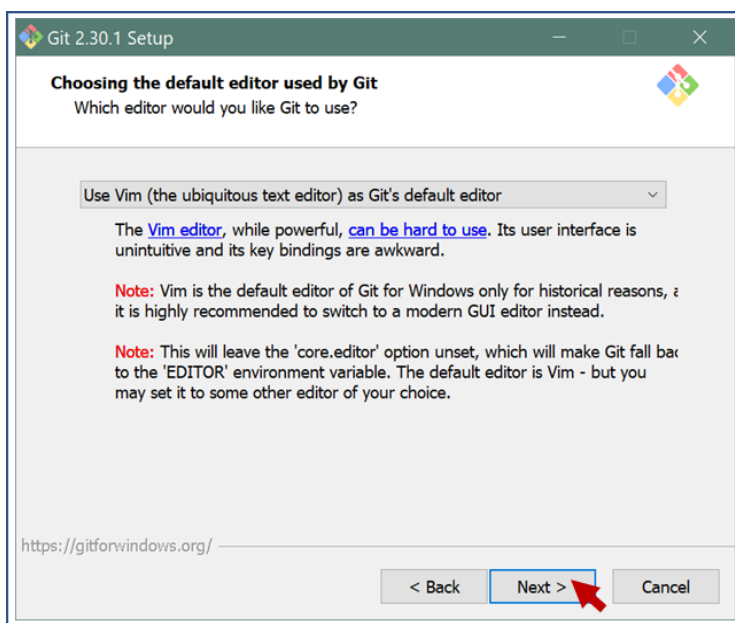
In this section we install non-Python related software in their default install directories. These include Git, GitHub, R and RStudio.

- **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.

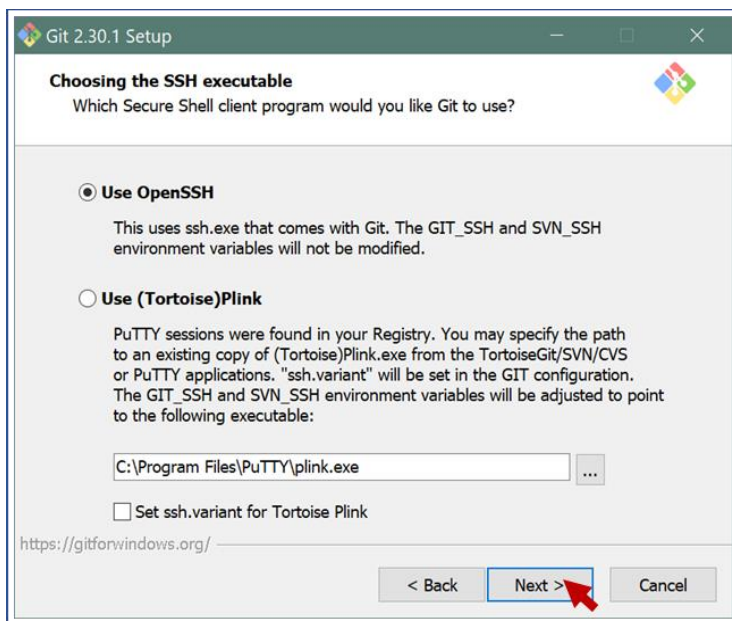
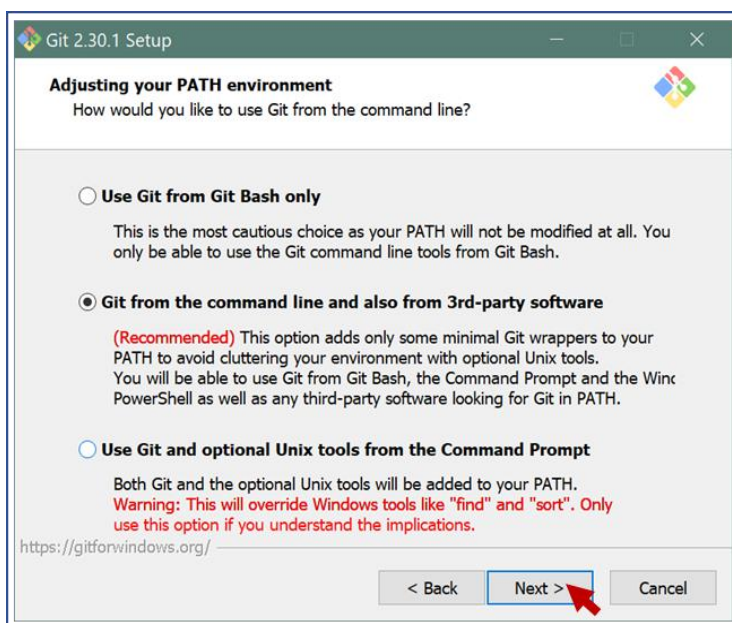


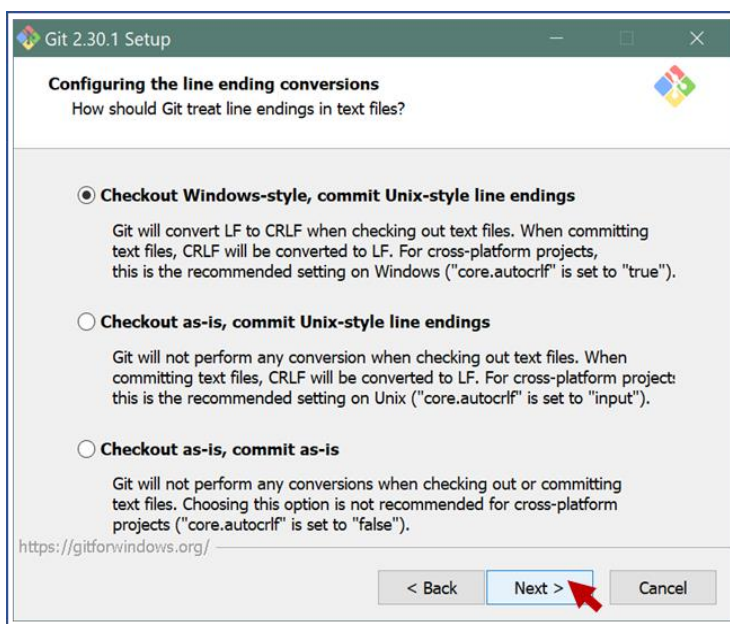
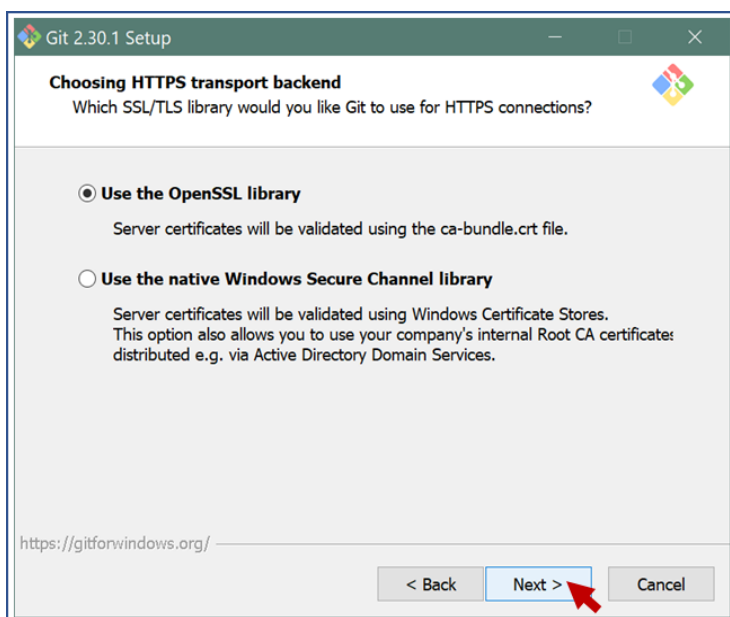


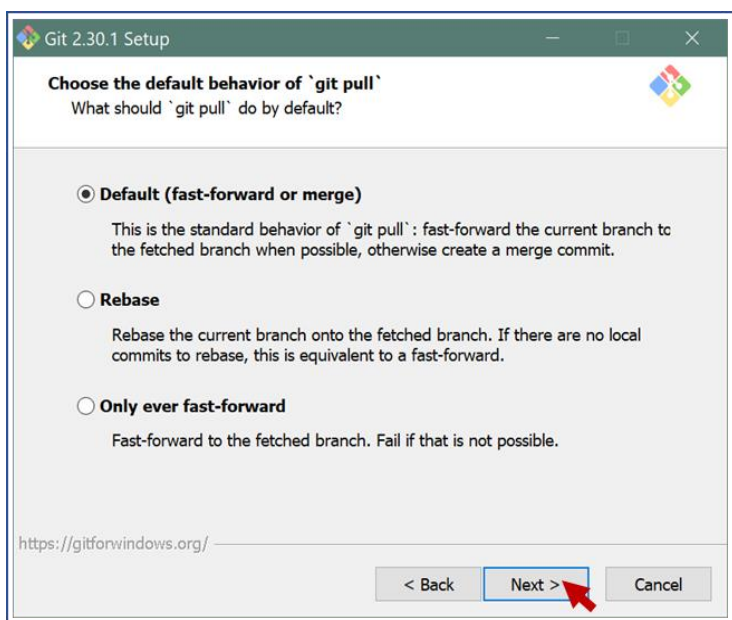
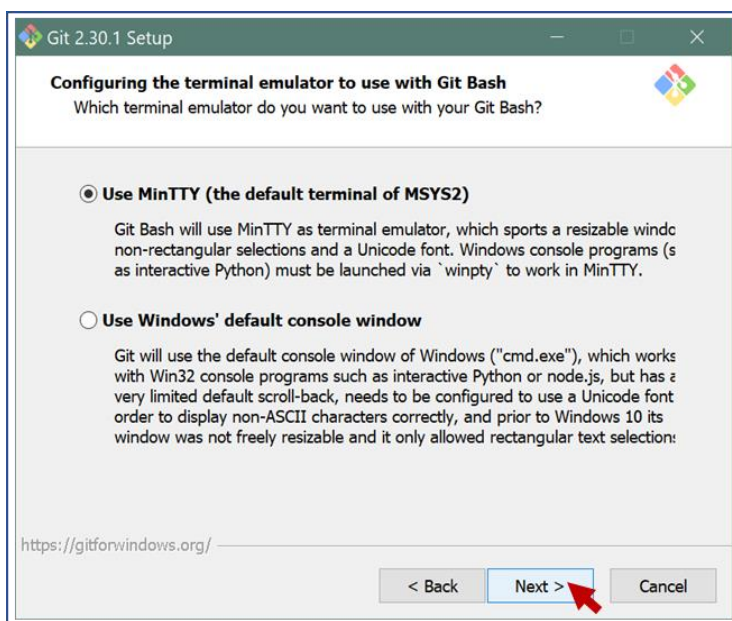


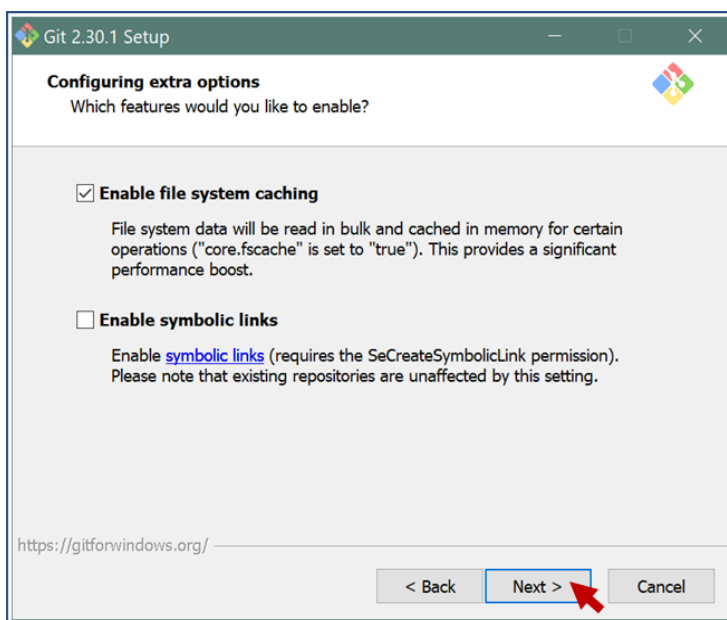
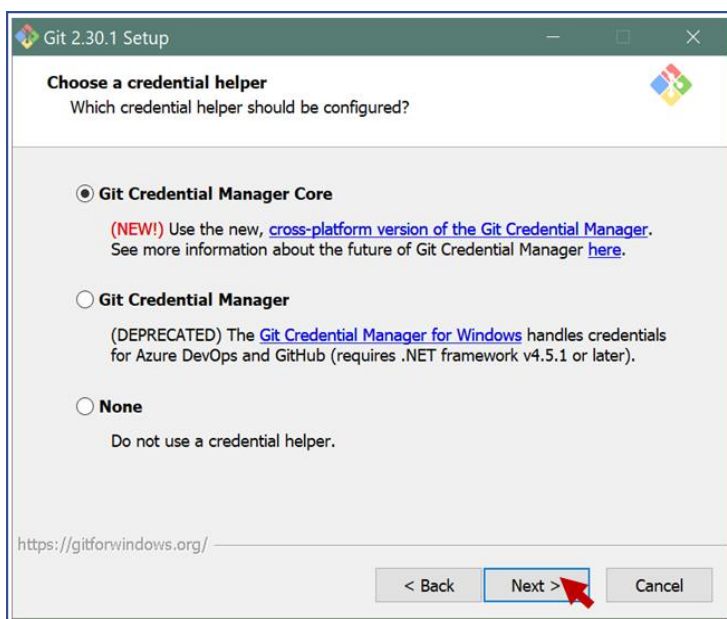


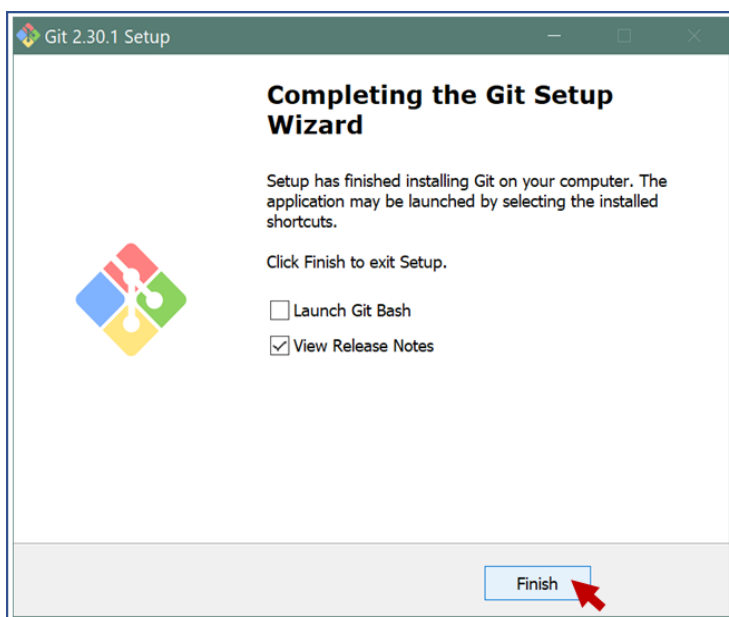
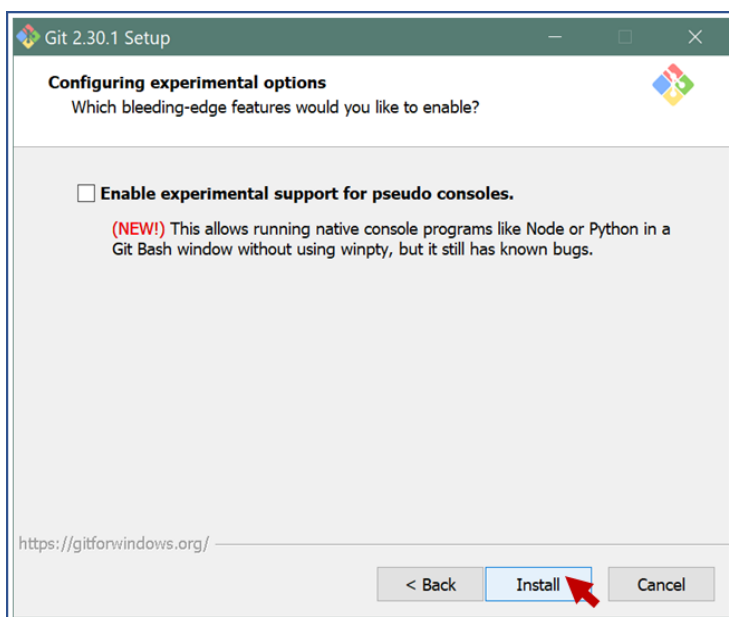




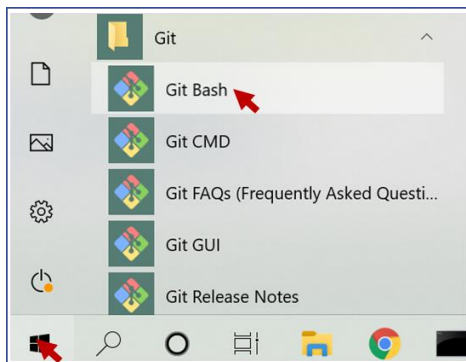






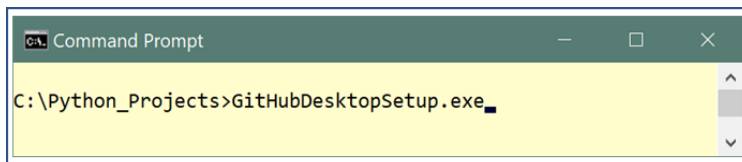


Notice that when using the Start button to verify installation, Git has installed a number of applications.



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

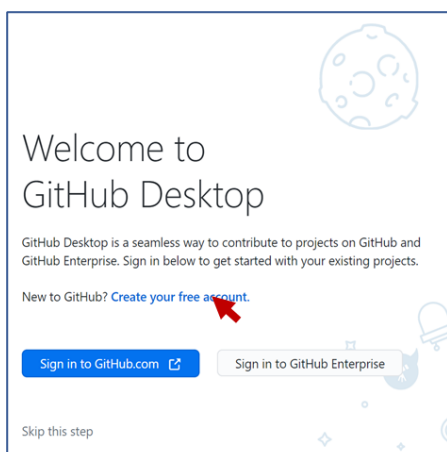
- **GitHub Desktop Studio** – from the Command Prompt in the C:\Python\_Projects directory, execute the following command: **GitHubDesktopSetup.exe**.



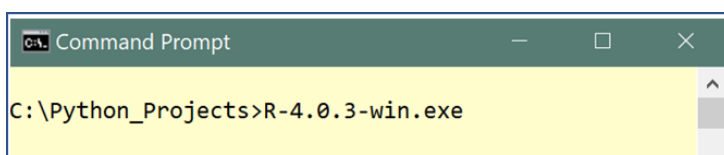
The following screen will appear, and the software will be installed with no further input from you. (Nice, huh?)



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**.



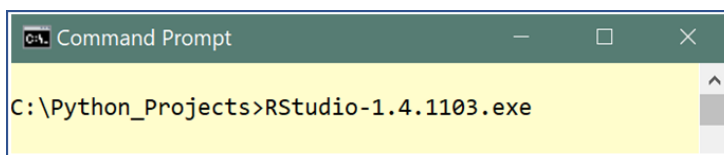
On the screens that appear next, do the following:

- 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**.

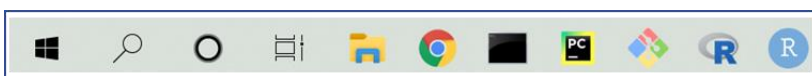




On the screens that appear next, choose the:

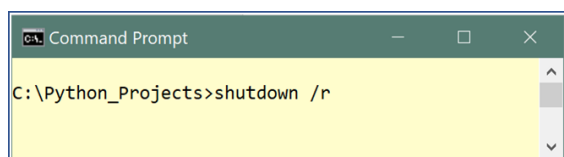
- **Next** button on the **first two** screens;
- **Install** on the **third** screen;
- Finish on the final screen, which indicates that setup is complete.

Pin RStudio to the taskbar, next to R.

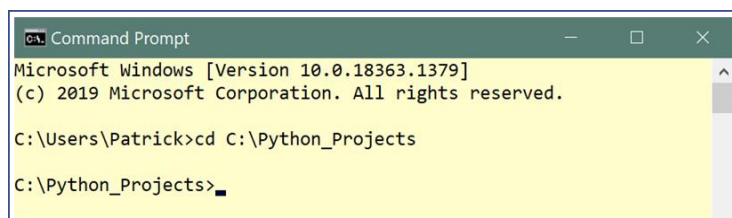


## 6. Reboot the Machine

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.

```

C:\Python_Projects>python
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MS
C v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more inf
ormation.
>>> exit()
C:\Python_Projects>
  
```

DOS Command Line

Python Command Line. Type exit() and Enter to go back to DOS

[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.

```

C:\Python_Projects>pip -V
pip 20.2.3 from c:\python_projects\lib\site-packages\pip (python 3.9)
  
```

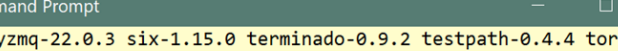
## 8. Installing Jupyter Notebook

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.

```
c:\Python_Projects>pip install notebook
Collecting notebook
  Using cached notebook-6.2.0-py3-none-any.whl (9.5 MB)
Collecting tornado>=6.1
  Using cached tornado-6.1-cp39-cp39-win_amd64.whl (422 kB)
Collecting jupyter_core>=4.6.1
  Using cached jupyter_core-4.7.1-py3-none-any.whl (82 kB)
Collecting jupyter-client>=5.3.4
  Using cached jupyter_client-6.1.11-py3-none-any.whl (108 kB)
Collecting nbconvert
  Using cached nbconvert-6.0.7-py3-none-any.whl (552 kB)
Collecting prometheus-client
  Using cached prometheus_client-0.9.0-py2.py3-none-any.whl (53 kB)
Collecting terminado>=0.8.3
  Using cached terminado-0.9.2-py3-none-any.whl (14 kB)
Collecting jinja2
  Downloading Jinja2-2.11.3-py2.py3-none-any.whl (125 kB)
    [REDACTED] 125 kB 1.3 MB/s
Collecting argon2-cffi
```

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.

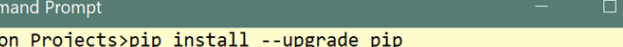
[Note: When using PIP to install software, occasionally, as happened to me in writing this, your return to the DOS command line will be preceded by a warning that a newer version of PIP is available, as below.



```
C:\ Command Prompt
05.7 pyzmq-22.0.3 six-1.15.0 terminado-0.9.2 testpath-0.4.4 tornado-
6.1 traitlets-5.0.5 wcwidth-0.2.5 webencodings-0.5.1
WARNING: You are using pip version 20.2.3; however, version 21.0.1 is
available.
You should consider upgrading via the 'c:\python_projects\python.exe
-m pip install --upgrade pip' command.

c:\Python_Projects>pip install --upgrade pip
```

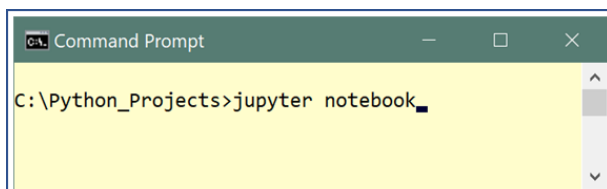
To upgrade pip, and since you are already in the Python install directory, simply enter the DOS command **pip install --upgrade pip**.



```
C:\Python_Projects>pip install --upgrade pip
Collecting pip
  Using cached pip-21.0.1-py3-none-any.whl (1.5 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.2.3
    Uninstalling pip-20.2.3:
      Successfully uninstalled pip-20.2.3
  Installing pip-21.0.1:
    Successfully installed pip-21.0.1
```

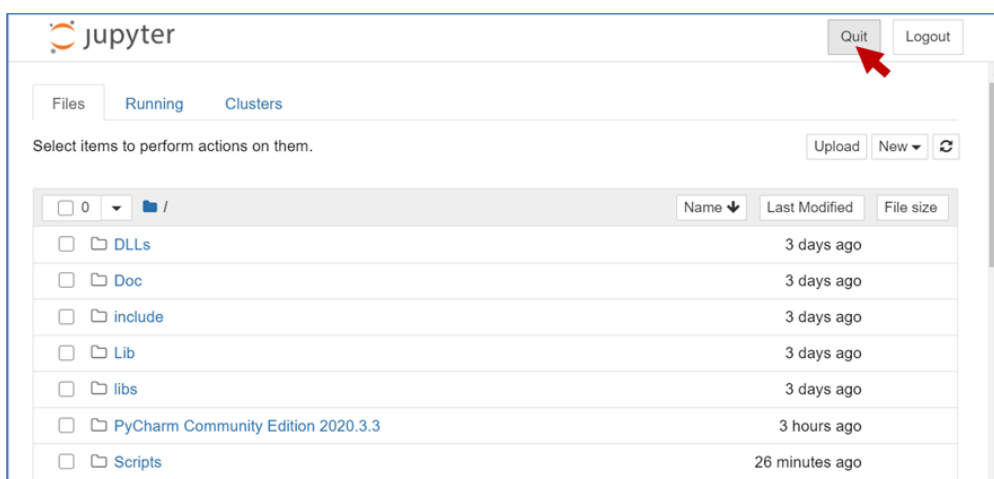
1

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



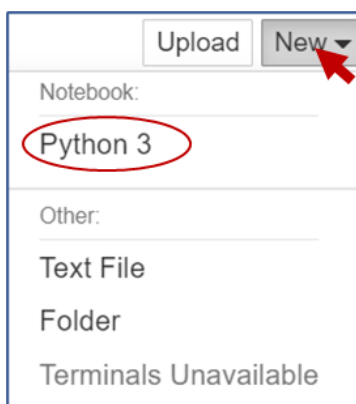
(Note: While you are working with Jupyter Notebook, do not close the Command Prompt window from which you started Jupyter, which will shut down the Jupyter server.)

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.

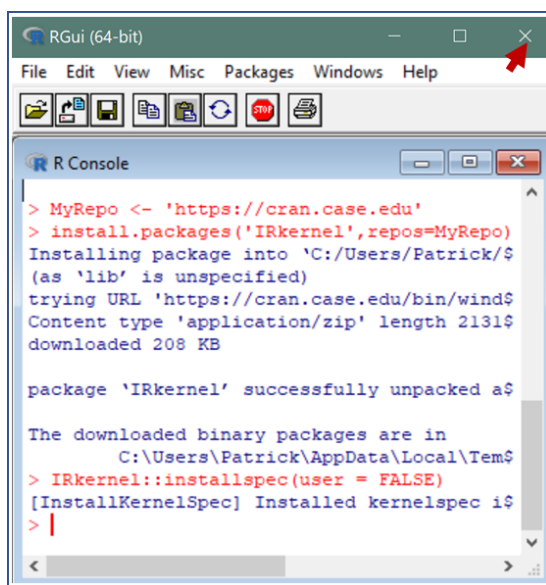
Note also that, currently, Jupyter Notebook can only support Python3 (or what is called the Python3 kernel). This is shown when trying to create a new notebook. When we choose **New**, our only choice is Python3, as below.



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)`

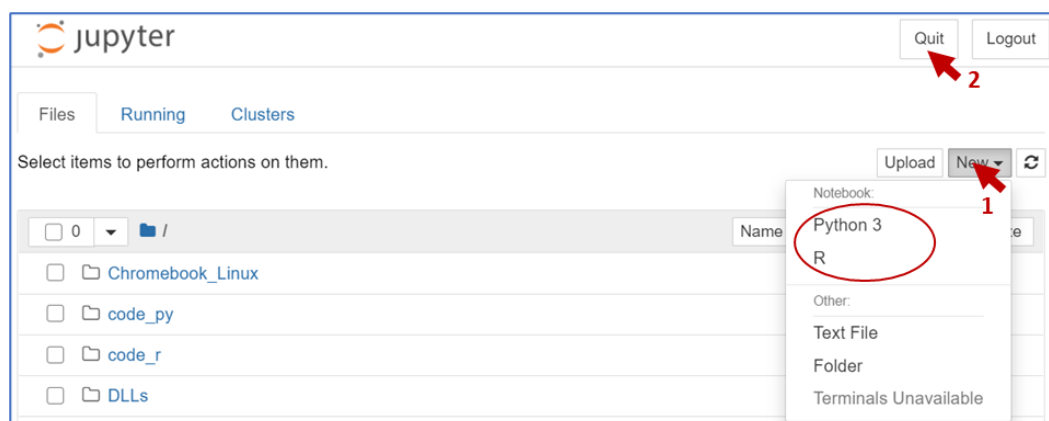


- The R Console can be closed.
- Just for fun, let's start up Jupyter Notebook using Git Bash, which uses Linux commands. Start Git Bash from its taskbar shortcut.
- In the Git Bash command line, enter the following two commands, pressing enter after each:
  - **cd /c/Py\*\_\*** (use wildcards)
  - **jupyter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

```

MINGW64:/c/Python_Projects
Patrick@PSWPRO3 MINGW64 ~
$ cd /c/Py*_*
Patrick@PSWPRO3 MINGW64 /c/Python_Projects
$ jupyter notebook
[W 17:07:58.745 NotebookApp] Terminals not available (error
was No module named 'winpty.cwinpty')
[I 17:07:58.946 NotebookApp] Serving notebooks from local di
rectory: C:\Python_Projects
[I 17:07:58.946 NotebookApp] Jupyter Notebook 6.2.0 is runni
  
```

- 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!