Setting up your computer for *CSSE 120*

Introduction to Software Development

November, 2020

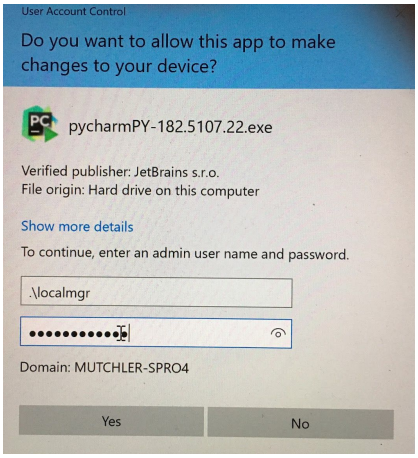
Goal

In this class we will use the ***Python*** programming language to introduce you to software development. We will also use tools like ***PyCharm*** and ***Git*** and cloud storage like that in ***gitter.csse.rose-hulman.edu*** to help make it easier to work with Python. Before we can start this course you need to install these tools on your computer.

Install ALL of this software, to make sure that you have up-to-date versions of everything. Older versions of Python (for example) will NOT work for everything that we do this term.

Work through this document, page by page, following the instructions for:

- Part 1: **Installing Python**
- Part 2: **Installing Git**
- Part 3: **Installing PyCharm Professional**
- Part 4: **Setting up the Python Interpreter in PyCharm**
- Part 5: **Setting up Git in PyCharm**
- Part 6: **Testing your settings**
- Part 7: **Tuning PyCharm for CSSE 120**



This setup requires downloading a total of roughly 450 MB so you should do it at Rose-Hulman or wherever you have reasonably fast internet access. When you are done, you can remove the installation files that were placed in your ***downloads*** folder.

Throughout, if you are asked to provide your administrator credentials (via a dialog similar to the one shown above and to the right):

- Set the ***User name*** to **`.\localmgr`**
- Provide your administrator (***localmgr***) password.[1]

Ask for help if you get stuck on anything!

These instructions are for Windows, but Mac or Linux users can proceed in a similar fashion. Most students take about 45 to 75 minutes to do all this setup.

[1] Page 11 of the instructions included when you received your laptop explains how to set your ***localmgr*** password. Email the EIT Service Desk servicedesk@rose-hulman.edu as needed for help.

Part 1: Installing Python

Python is the ***programming language*** in which we will write our programs. To install Python, you will install the ***Python interpreter*** that executes (runs) your programs, as follows:

Step #1 (of installing Python): Visit (click on the link):[2]

https://www.python.org/ftp/python/3.9.0/python-3.9.0-amd64.exe

to ***Save*** the file to your computer. (If you are using Chrome as your browser, it will appear at the bottom of the window. In any case, it has probably been saved in your ***Downloads*** folder.)

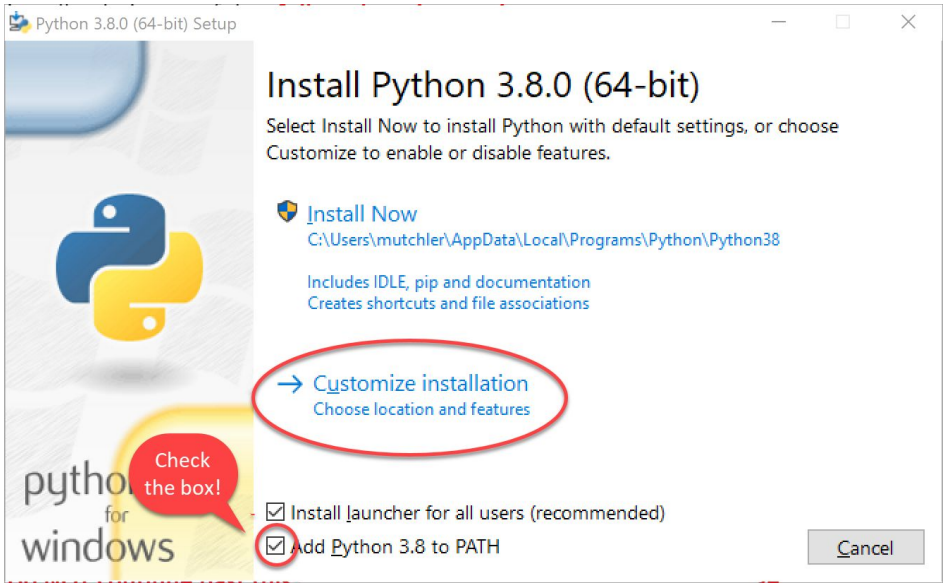
Step #2 (of installing Python): Double-click the downloaded file to run the installer, being careful to ***follow these instructions:***

Step #2a: At the ***initial installation window***, as shown below:

- ***Check the box*** for **"Add Python 3.9 to PATH"** (so both boxes on that page should end up checked).

  .

- Select **"Customize installation."**

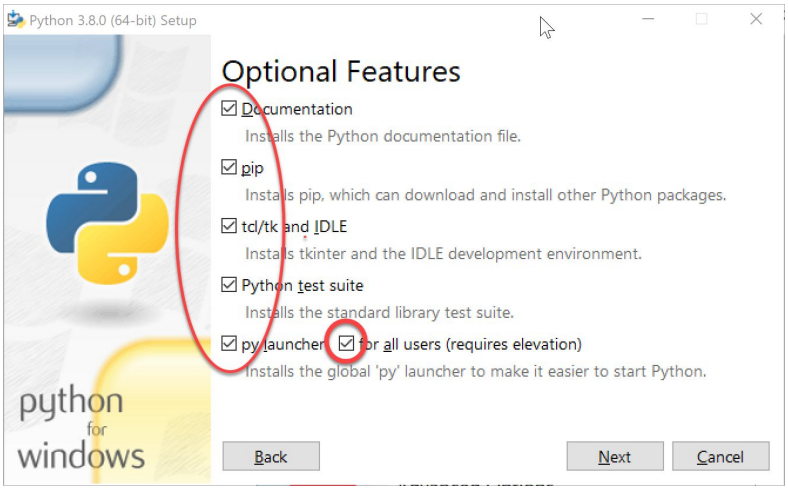


Do NOT continue past this point until you have turned to the next page (where these instructions continue).

The above picture and all the following ones show version 3.8.0 for Python. You will be installing ***3.9.0*** (the current version), so expect the pictures to reflect that small difference.

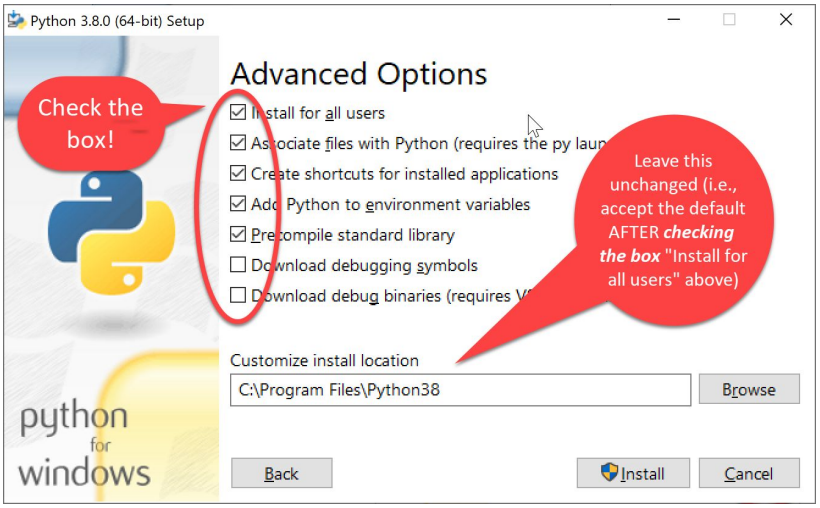
[2] Note: If you are using a Mac, instead use https://www.python.org/ftp/python/3.9.0/python-3.9.0-macosx10.9.pkg or whatever is relevant to your computer at https://www.python.org/downloads/release/python-390/.

Step #2b: On the window that appears next, all boxes should be checked.



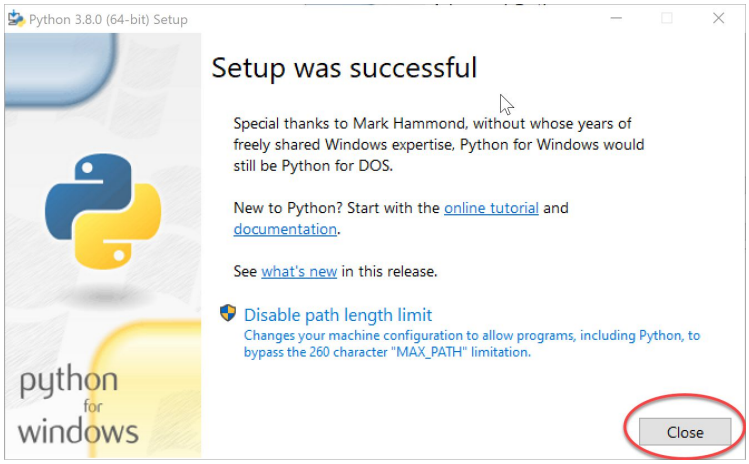
Step #2c: On the window that appears after that:

- ***Check the box*** for **"Install for all users."**



- Click ***Install*** and let the installation continue. Your computer will probably ask you for your admin (i.e., `.\localmgr`) username and password. The installation takes a few minutes on most modern computers.

Step #2d: At the end, select ***Close***.



Part 2: Installing Git

Professional software engineers use a workflow (i.e., a way to do their work) that includes using a **Version Code System** (**VCS**). We will use one called ***Git***.

Git allows software engineering teams to collaborate; it provides ways for each software engineer to work on parts of the software without fear of harming other team members' work, and it helps in the process of integrating a software engineer's code into the production version of the code when the time is ripe.

That said, we will use **Git** mostly to provide simple ways to:

1. **Get starting code** for each project ***from*** the "cloud" (where we put that starting code) ***to*** your computer (where you will add to that code).
2. **Get YOUR code** (as you are working on it) ***from*** your computer ***to*** the "cloud".

Here the "cloud" simply means a computer that is always running and allows both you and us to get and store information on it. By storing your code "in the cloud" (as well as on your computer):

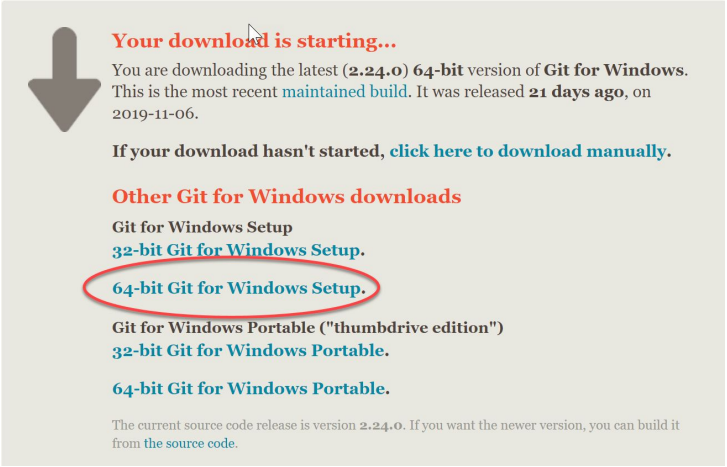
- Your code is automatically backed up, and
- You can collaborate with others on shared code (as you will do later in the course).

Step #1 (of installing Git): Visit
(right-click and open in a new tab)

https://git-scm.com/download/win

to download a Windows installer for Git:

That page should automatically start the download, but if for some reason it doesn't you can manually start the download via the **64-bit Git for Windows Setup** link on the page, as shown to the right.

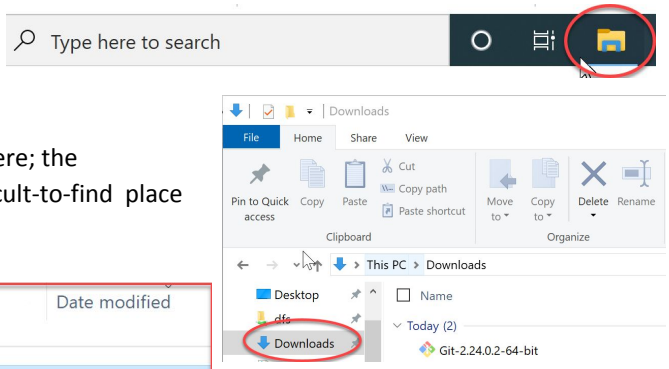


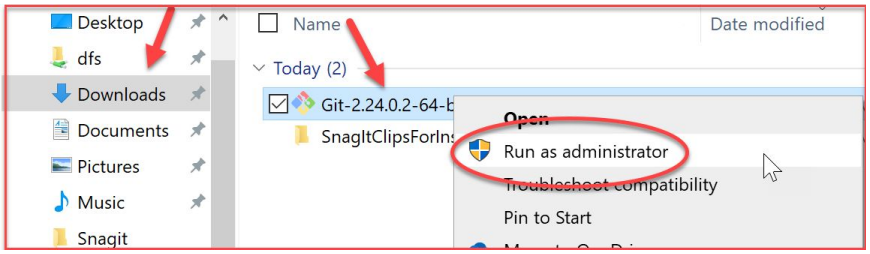
Check the download to be sure that it is the ***64*** **(not 32)** bit version.

Step #2 (of installing Git): Once downloaded, ***do NOT simply double-click on it***. ***Instead*** (instructions continue on the next page):

The above picture and all the following ones show version 2.24.0 for Git. You will be installing ***2.29.2*** (the current version), so expect the pictures to reflect that small difference.

Locate the downloaded file in your **Downloads** folder, ***right-click*** on the downloaded file, and select ***Run as administrator***. (If you lack such an option, just double-click to open the file and continue from there; the installation will still work, it just puts Git into an unusual, difficult-to-find place if you don't *Run as administrator*.)



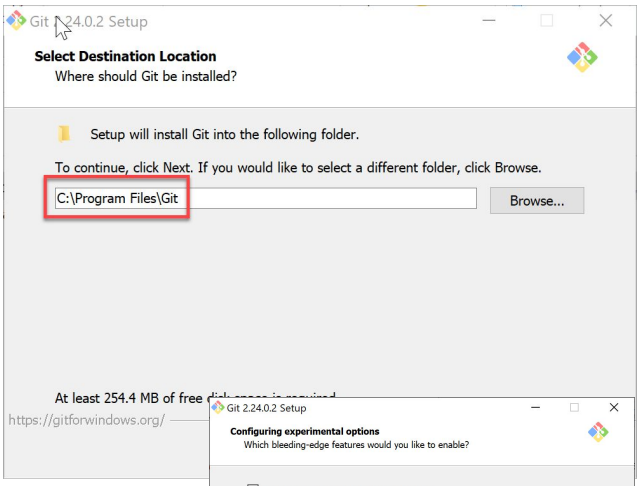


Doing so will start the Windows Setup for 64-bit Git.

Accept all defaults during the installation (just keep pressing *Next*, *Next*, *Next* …) **Especially do *NOT* change the folder where Git is to be installed -- it should be**

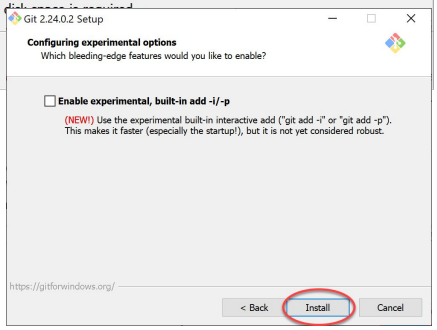
```
C:\Program Files\Git
```

as shown to the right. (If it shows some crazy-looking folder, then you are not successfully running the installation as Administrator. In that case, *Cancel* and try again by ***right-clicking*** on the installation file and selecting ***Run as administrator***. If you can't get that to work, continue but accept the crazy-looking folder that the Git installation chooses.)

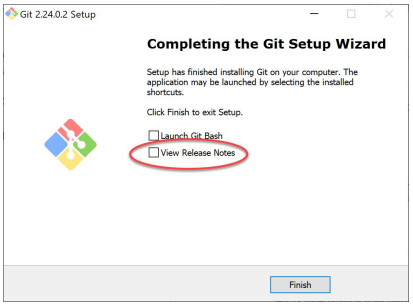


To repeat: **Accept all defaults** during the installation (just keep pressing *Next*, *Next*, *Next* …)

Eventually you will get to a screen that has an ***Install*** button (as in the picture to the right). Press ***Install*** to begin the actual installation (which will take a minute or so).



The installation is done when it reaches the page shown to the right.



Uncheck the "***View Release Notes***" checkbox on the final step since you don't need to read those, and click ***Finish.***

Part 3: Installing PyCharm Professional

PyCharm Professional is an ***IDE*** (**Integrated Development Environment)**. We will use it to write and run our programs. JetBrains is the company that makes PyCharm Professional. They provide their software free to students and faculty, requiring only that you renew the (free) license once a year until you graduate.

You will do the following steps, ***per instructions on the following pages***.

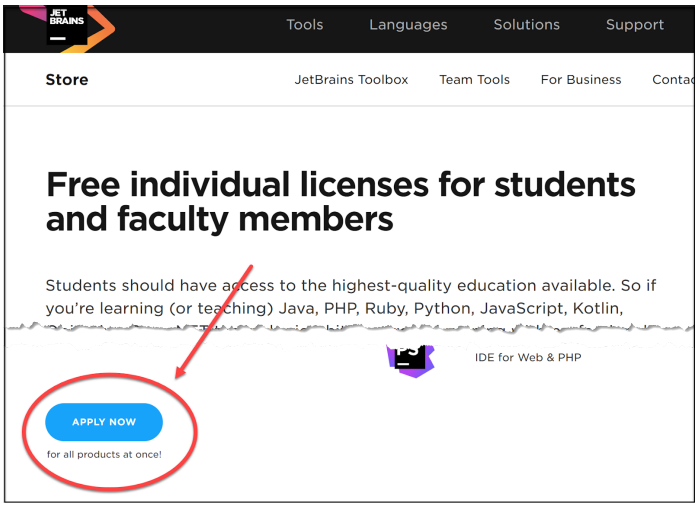
1. **Create a JetBrains account** and apply for a free educational license
2. **Activate** your educational license
3. **Download and install PyCharm Professional**

Warning before you start

Step #1 (and perhaps #2) requires an email confirmation. **For all email confirmations use your @rose-hulman.edu email** (even if you have other email accounts). When you complete a step in which JetBrains sends you an email, it should arrive within 1 minute. (Check your junk/spam folders if it does not.). To do a confirmation, just click the link that you will see in the email.

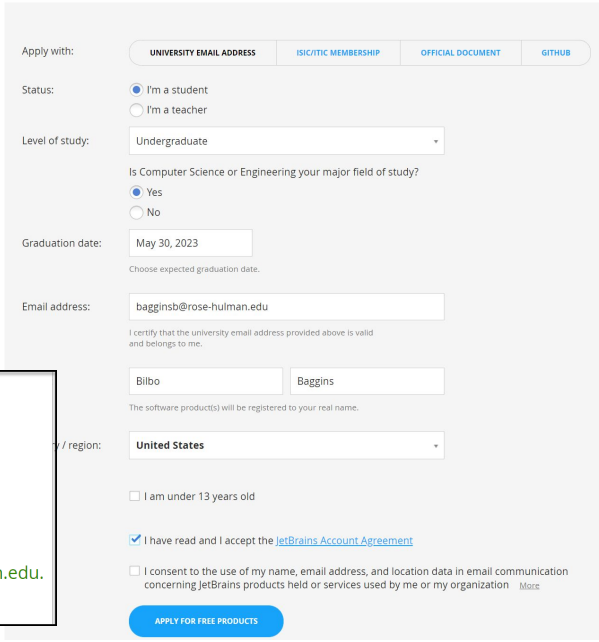
Step #1 (of installing PyCharm Professional): Create a JetBrains account and apply for a free Educational License, as follows:

Visit (right-click and open in a new tab) https://www.jetbrains.com/student/, scroll down the page and click on the ***Apply Now*** button that is about halfway down that page. (There is another ***Apply Now*** button higher on the page that might or might not be covered up by the "we use cookies" warning. Either button is fine.)

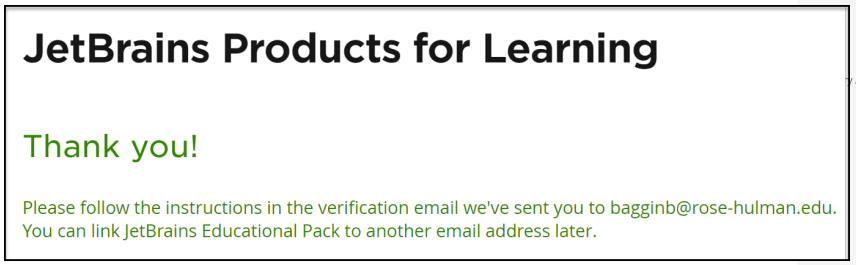


The instructions for this step continue on the next page.
Do NOT continue until you read them!

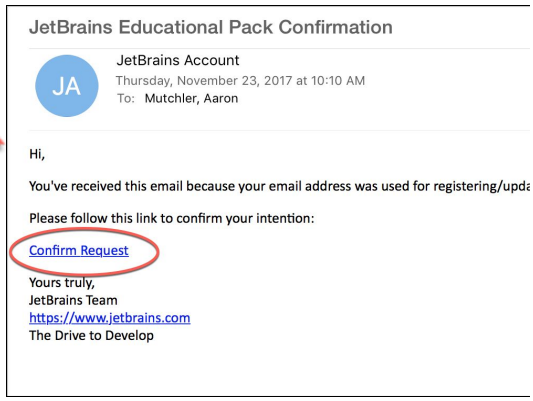
Fill out the form that appears ***using your @rose-hulman.edu email.***



After pressing the ***Apply For Free Products*** button, you should see a page that indicates that JetBrains has sent a confirmation email to you:

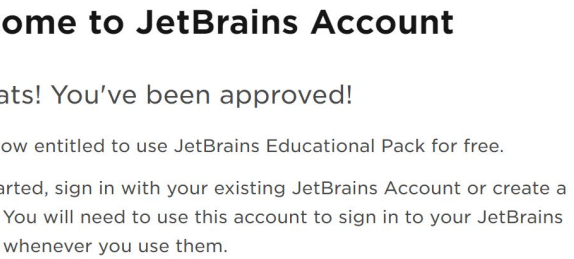


Check your email for a message from JetBrains (confirmation #1). Click on the ***Confirm Request*** link in the email.



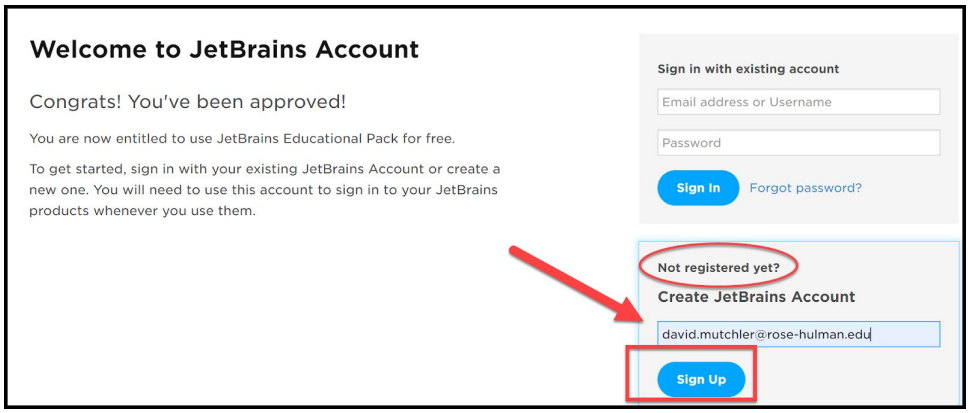
Step #2 (of installing PyCharm Professional): Activate your educational license

Clicking on the ***Confirm Request*** link in the ***first*** email that JetBrains sent you brings you to a page (shown to the right) that (after you accept the license) says that you have been approved (yay!). On that page:

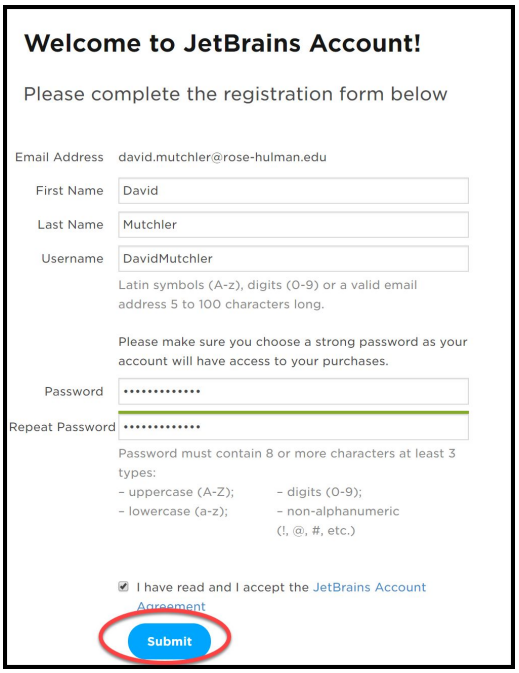


(Instructions continue on the next page.)

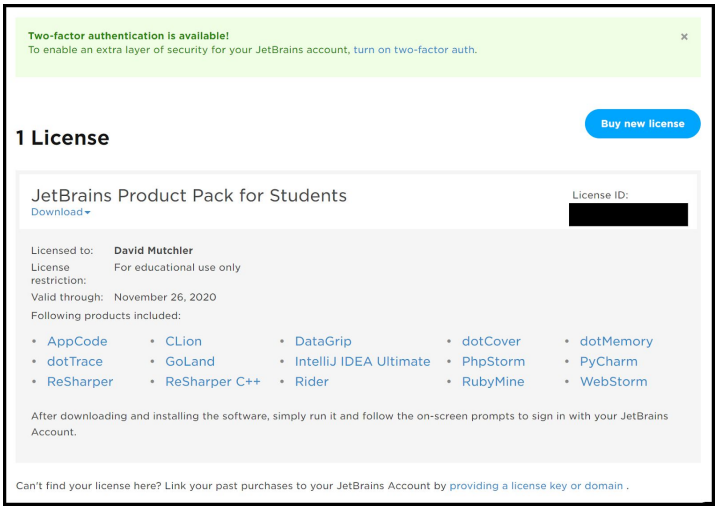
On the "Welcome to JetBrains Account" page, enter your **@rose-hulman** email account under the ***Create JetBrains Account*** section, as shown to the right. Then press the ***Sign Up*** button.



Now, you should be on a page at which you ***Complete JetBrains Account Registration*** by choosing a password and accepting the JetBrains Privacy Policy, as shown to the right. **Remember your password** since you will need it later!



After you press the ***Submit*** button on the ***Welcome to JetBrains Account*** registration page, you will have a JetBrains account!



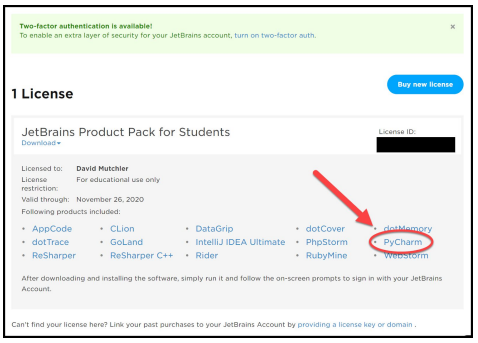
You are finished with this step only when you see a license, like the one shown to the left. If the above process did ***not*** get you to a page like the one shown to the right, you aren't done yet. :) In that case, visit

https://account.jetbrains.com/licenses and proceed from there.

Step #3 (of installing PyCharm Professional): Download and install PyCharm Professional, as follows.

You now have an account and a free Educational License. Next, you need to download PyCharm Professional to your computer.

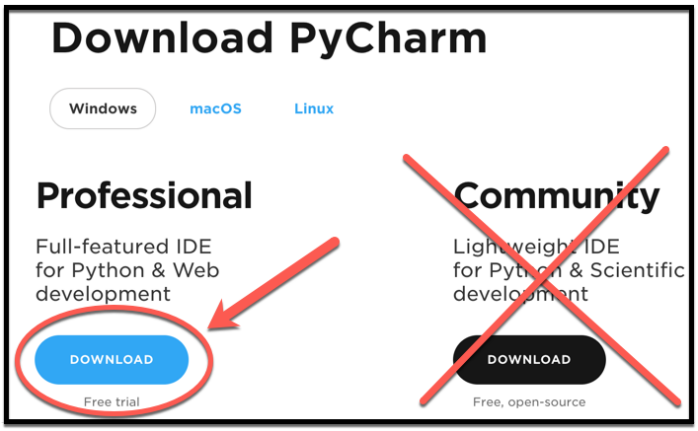
To do so, from the screen at which you ended the previous step, click on the ***PyCharm*** link. On the next page, click on the ***Download Now*** button to get to the Download page.



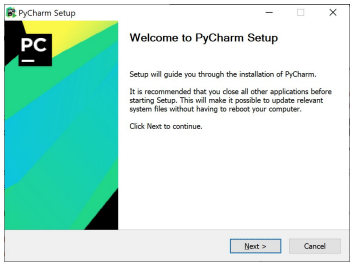
Or, just use this direct link (right-click and open in a new tab):

https://www.jetbrains.com/pycharm/download/

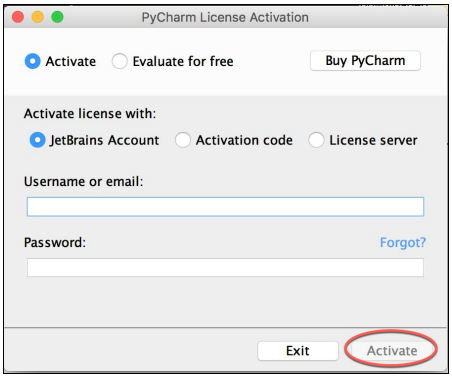
Once at the Download page, click the ***Download*** button that is below the **Professional** version of PyCharm to download the PyCharm installer.



After downloading the Professional version, **click on the downloaded file to run the installer**.



It may ask you to enter your username/email and password (as shown to the right) for the ***PyCharm License Activation***.

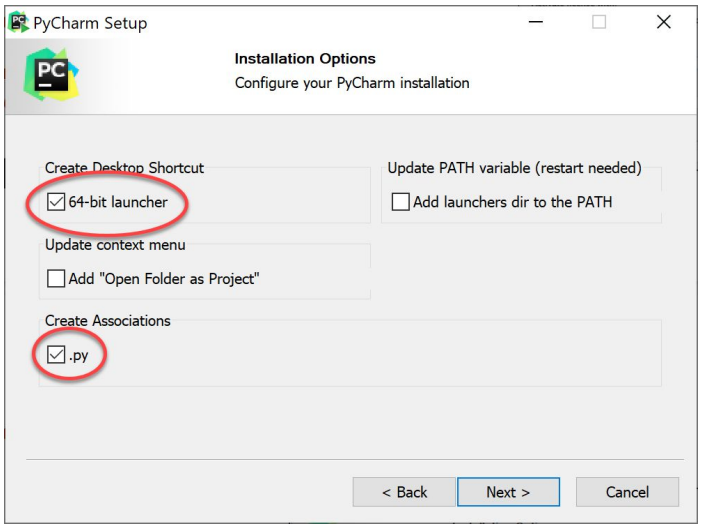


Accept all defaults during your installation, ***except***:

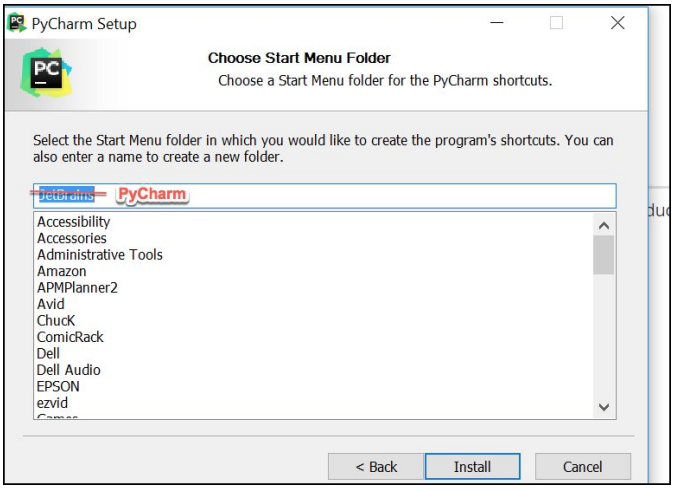
The instructions for this step continue on the next page.
Do NOT continue until you read them!

Accept all defaults during your installation, ***except***:

At the ***Installation Options*** page (shown to the right), check the boxes for ***64-bit launcher*** and ***.py*** association.



At the ***Choose Start Menu Folder*** page (shown to the right), change ***JetBrains*** to ***PyCharm*** (and then press ***Install***).

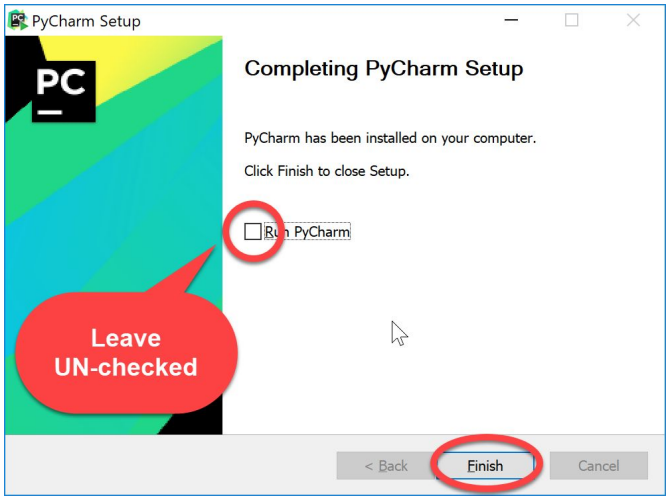


At the final page of the PyCharm installation, leave the box UN-checked and click on the ***Finish*** button.

If you accidentally open PyCharm, do not proceed past the first screen that appears. (If you accidentally do proceed past the first screen, use

`File ~ Close Project`

to get back to that first screen.)



ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Part 4: Setting up the Python *Interpreter* in PyCharm

You should have already done the following, per the previous instructions in this document:

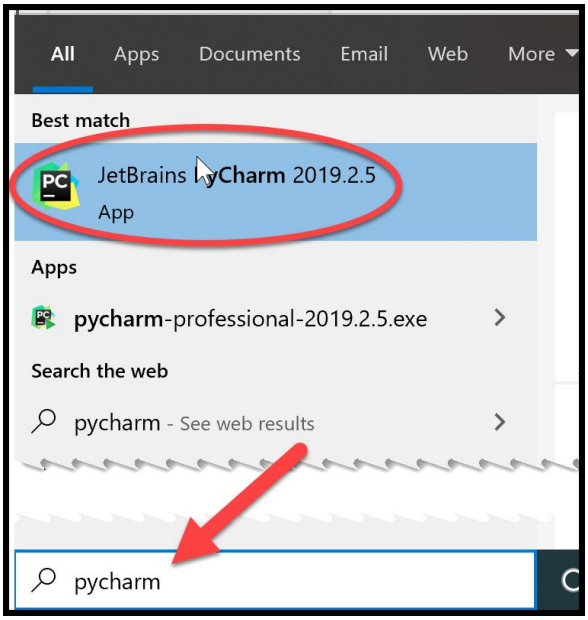
1. Install ***Python***
2. Install ***Git***
3. Install ***PyCharm Professional***

> Some of the following pictures show version 2019-2.5 for PyCharm. You will be installing ***2020.2.3*** (the current version), so expect the pictures to reflect that small difference.

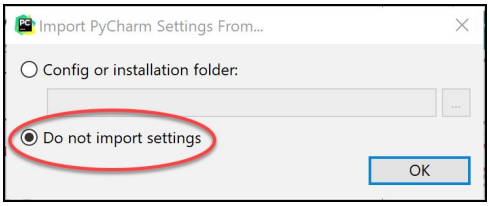
If you have not already done the above, do so now by finding the relevant section(s) in the previous pages of this document.

In this step, you will open PyCharm Professional and set it up to use your previously-installed Python interpreter and libraries that it needs for our work with robots.

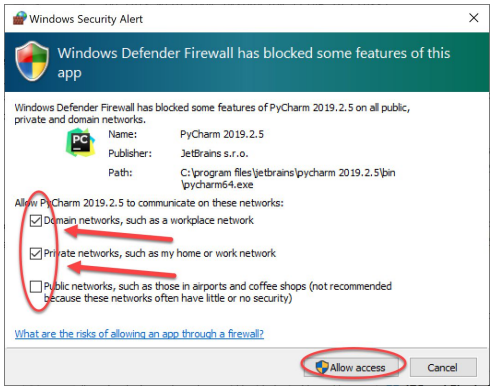
Step #1 (of setting up the Python Interpreter in PyCharm): Start PyCharm, as follows:



- **Run PyCharm** (use your Search tool as needed to find it).



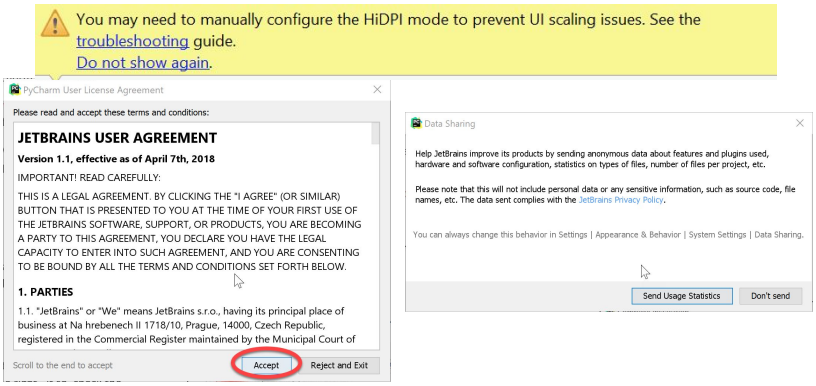
- When PyCharm itself starts up, you may see the dialog shown to the right. If so, check the ***Do not import settings*** box and then press ***OK.***



- If **at any point** you see a *Windows Defender Firewall* dialog like that shown to the right, choose options as you wish (the picture shows my own choices).

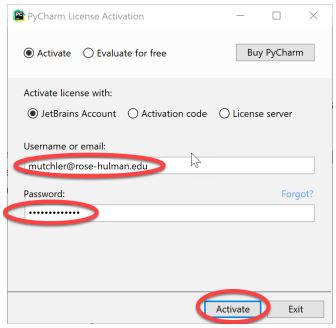
The instructions continue on the next page. *Do NOT continue past the next screen that appears until you read them!*

At some point, you might (but probably will *not*) see a message like that to the right. If you do see this message, click **Do not show again** and ignore it otherwise.



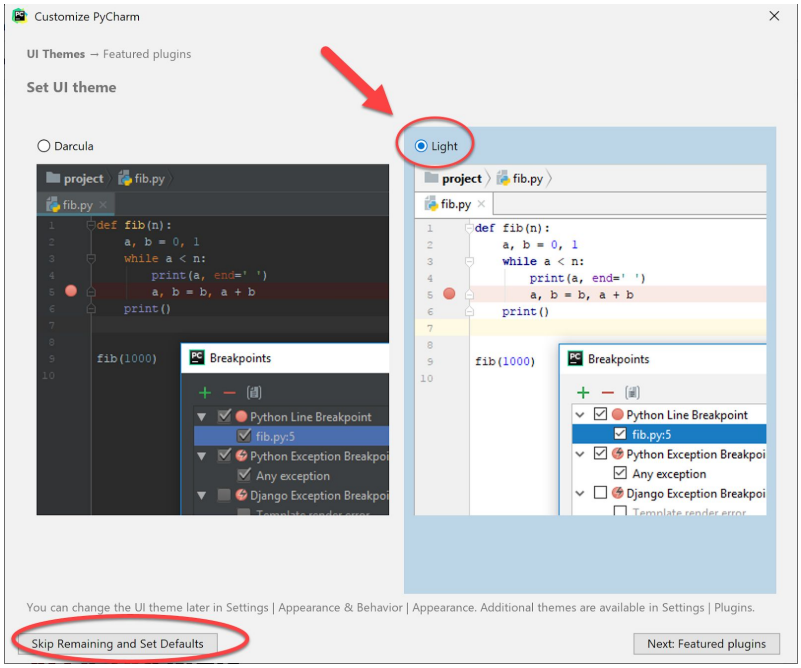
If asked, accept the license and choose whether or not to share your coding behavior with JetBrains (either choice is fine).

If **at any point** you are asked to ***Activate*** your license, enter the email address and password that you chose when you created your Jetbrains account, then press the **Activate** button.

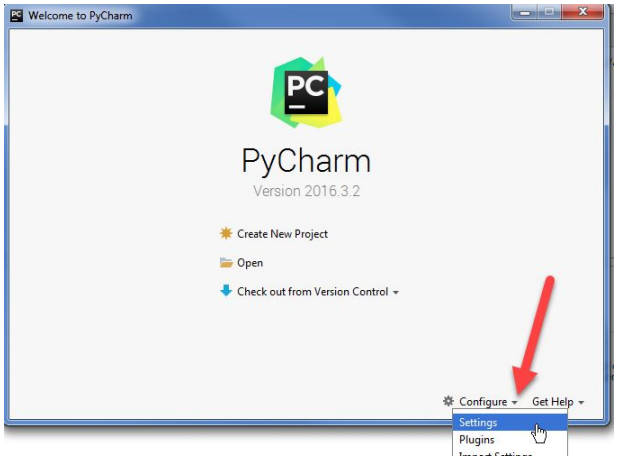


Soon you will see the page shown to the right. **Choose whichever theme you like**. It is easy to switch to another theme later if you wish.

Then press ***Skip Remaining and Set Defaults*** (the remaining default values are fine).



When you get to the screen shown to the right, *STOP* (and continue to the next step of these instructions, on the next page).

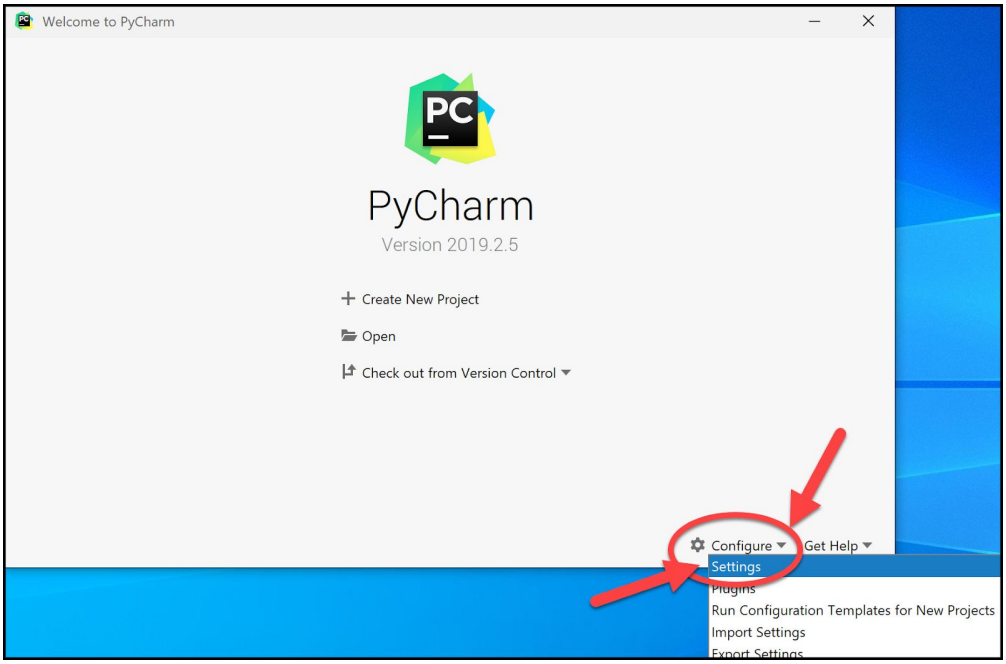


Step #2 (of setting up the Python Interpreter in PyCharm):

Tell PyCharm the location of your Python Interpreter, as follows:

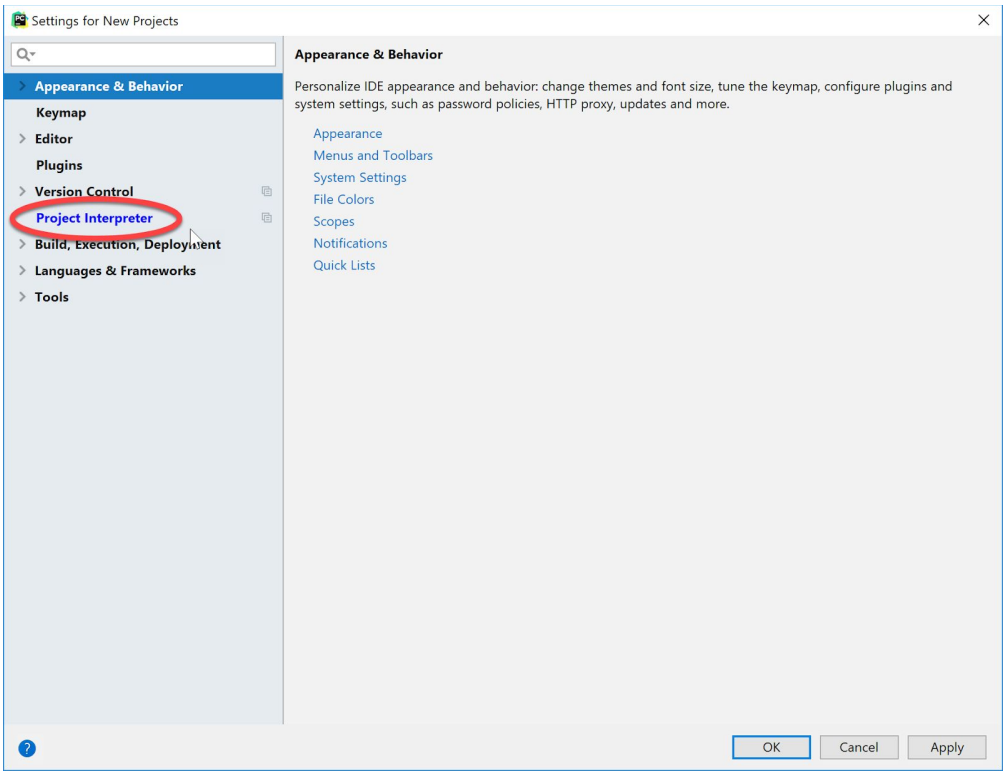
To start configuring your Python Interpreter:

From the main PyCharm main page, select the ***little pull-down arrow*** next to **`Configure`** toward the bottom-right of the window, then select ***`Settings`*** (or ***Preferences*** on a Mac) in the pop-up menu, as shown to the right.

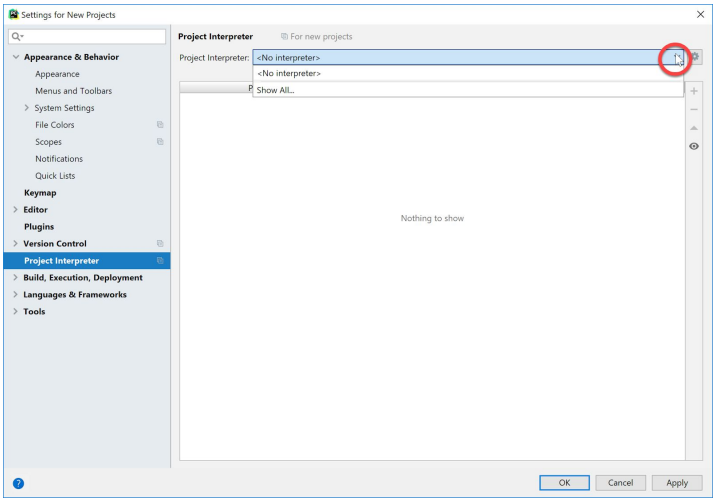


In the ***Settings*** dialog that appears, select **Project Interpreter** (a bit more than half-way down the list on the left-hand side, as shown in the picture to the right).

This brings up the the *Project Interpreter* pane, as shown on the next page:

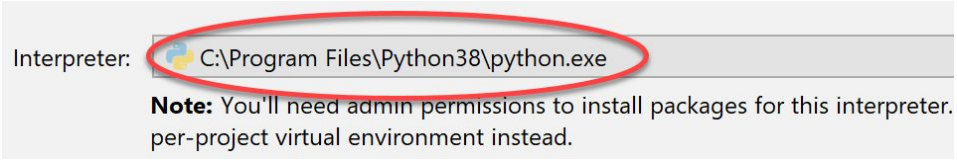


Click on the little pull-down arrow on the right-hand-side of the Project Interpreter text box (circled in ***red*** on the picture to the right).

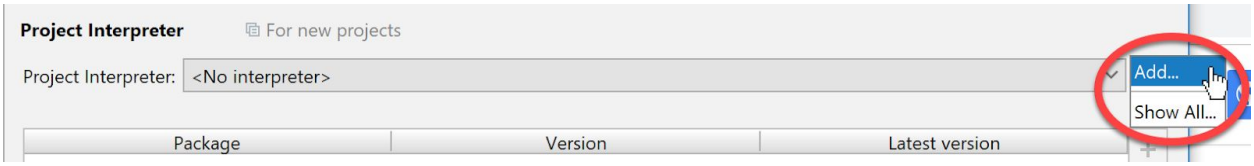


Hopefully it shows Python 3.9, as shown below. **If so, select *Python 3.9*** and then click ***OK*** to exit the Settings dialog and ***skip ahead to continue these instructions*** on **page 16**. It will take a minute or two for all the Python Interpreter files to get loaded.

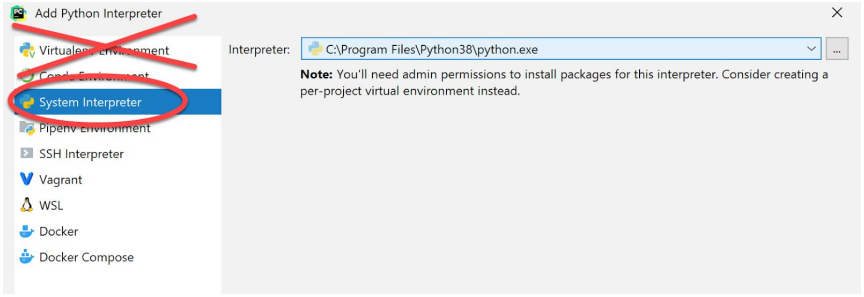
Reminder: The pictures show Python38, but you will see ***Python39*** when you do the installation (because you are installing Python 3.9, the current version).



If you do NOT see Python 3.9 in the pull-down shown above, then select the tiny "gear" symbol to the right of the pull-down symbol, and select ***Add...*** from the pop-up that appears, as shown below. (It takes a few seconds to react.)

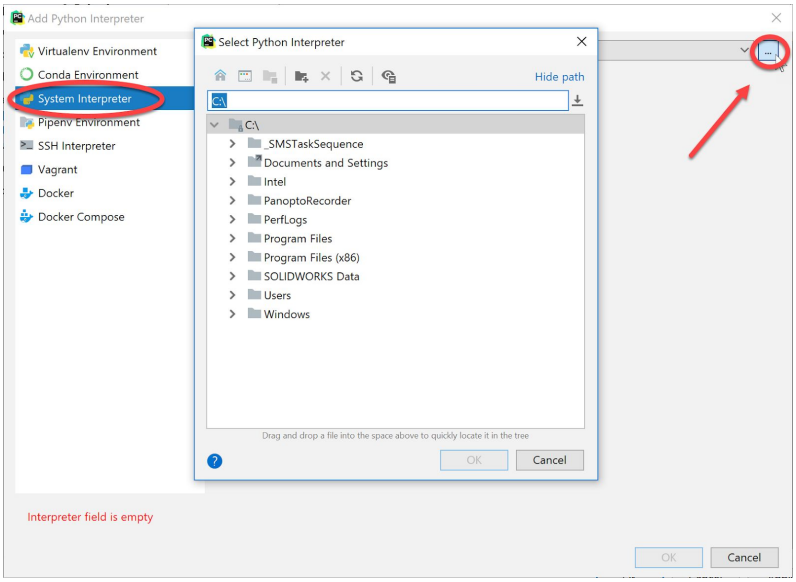


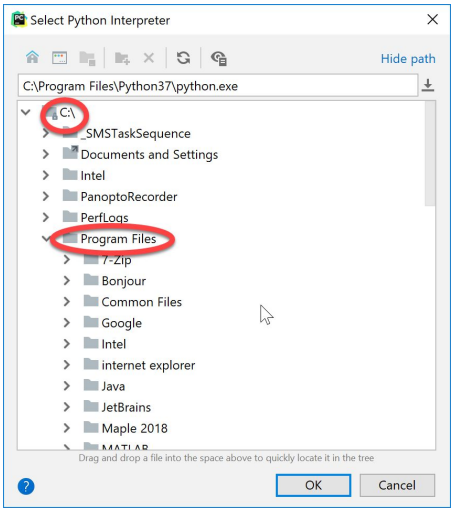
IMPORTANT: In the screen that appears then, select ***System Interpreter***, as shown to the right. Do NOT use the Virtualenv Environment that is the default option.



Now ***try the pull-down arrow again***. **If it shows a file for Python 3.9**, as shown above, ***select the* Python 3.9 *choice*** and then click ***OK*** to exit the Settings dialog and ***skip ahead to continue these instructions*** on **page 16**. It will take a minute or two for all the Python Interpreter files to get loaded.

If Python 3.9 *still* has not shown up, click on the three dots to the right of the Interpreter text box, as shown below and to the left. Doing so will bring up a file browser, as shown below and to the right.

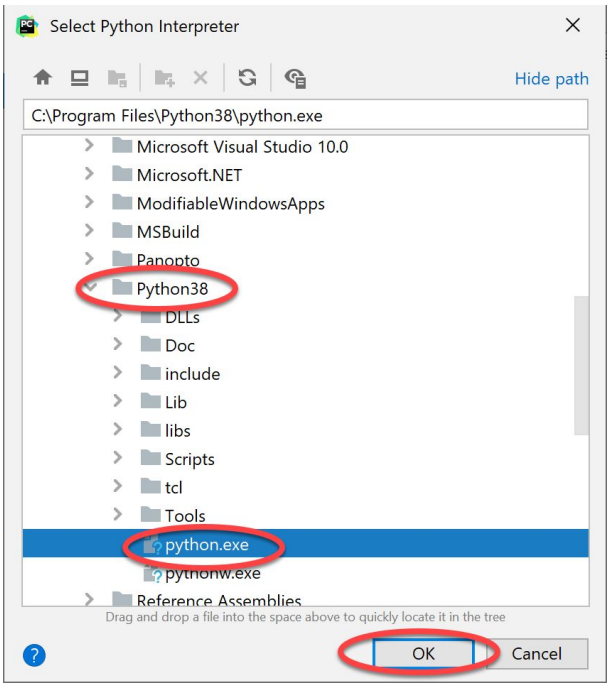




Use the file browser to locate and select where you installed Python (per instructions earlier in this document), presumably **`C:\Program Files\Python39\python.exe`**, as shown in the pictures to the right.

If you cannot find your Python installation, go back and redo the instructions for installing Python, then repeat the above instructions for telling PyCharm where you installed it.

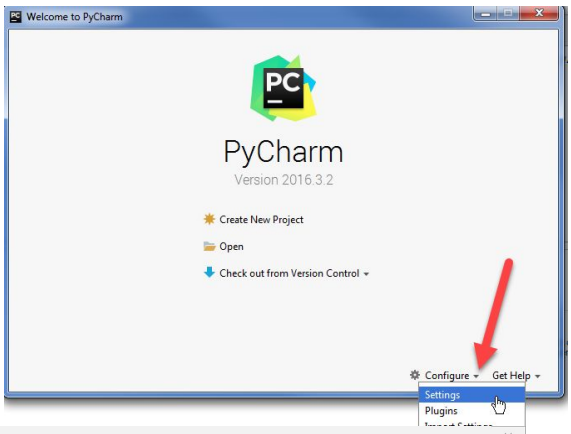
Throughout these instructions, get help from your instructor as needed!



Step #3 (of setting up the Python Interpreter in PyCharm): **Add packages (libraries) necessary for our projects**, as follows:

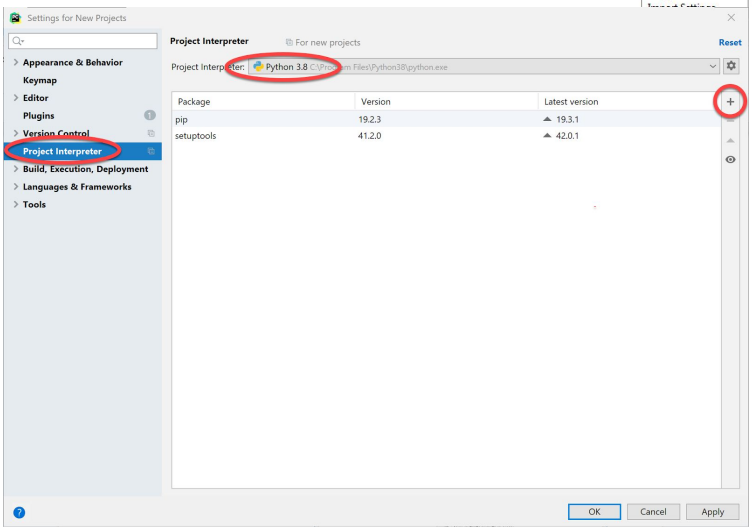
At this point, you should have configured PyCharm to know the location of your Python interpreter. (If not, then skip ahead to Part 5 on page 18 and get help from your instructor later on setting up your Python Interpreter.)

If you are back in the main PyCharm main page, select **`Configure ~ Settings`** (or ***Preferences*** on a Mac), as shown to the right, to return to the ***Settings*** page.



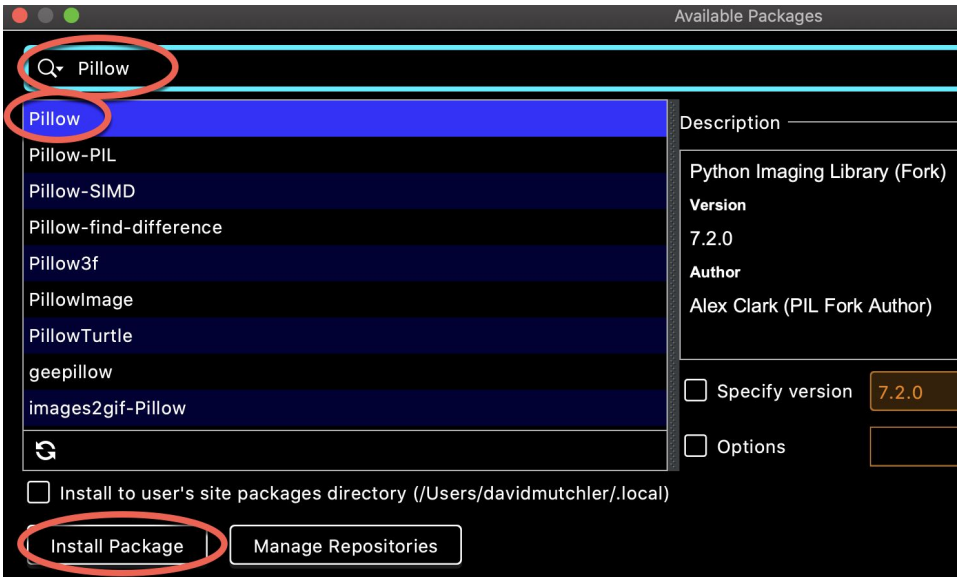
The *Project Interpreter* pane will probably still be open; if not, click on **Project Interpreter** in the list on the left-hand-side to open that pane.

Then click on the + sign on the right-hand-side near the top, as shown in the picture to the right.

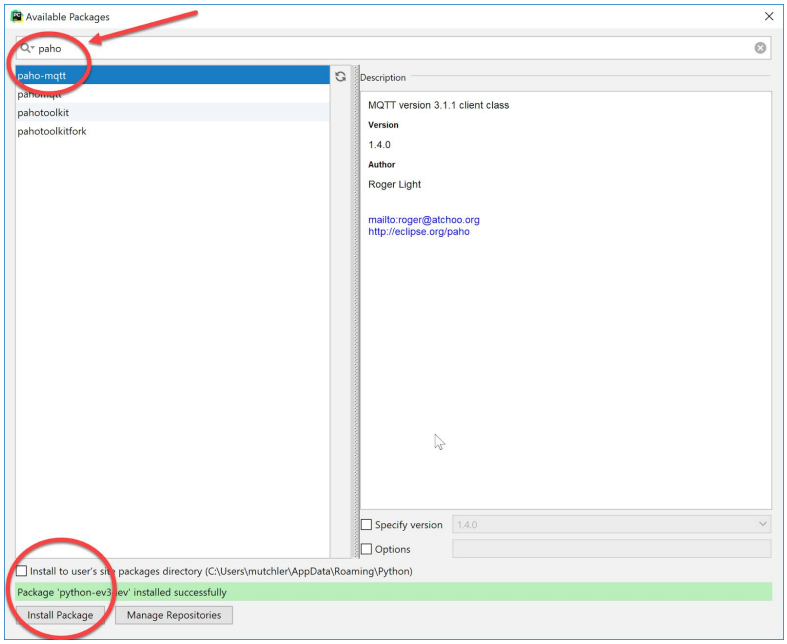


In the ***Available Packages*** pane that appears, type **`Pillow`** in the text box, as shown in the picture to the right.

Then click on the ***Install Package*** button (leave the checkbox unchecked). After about a minute, PyCharm will have installed the selected package.

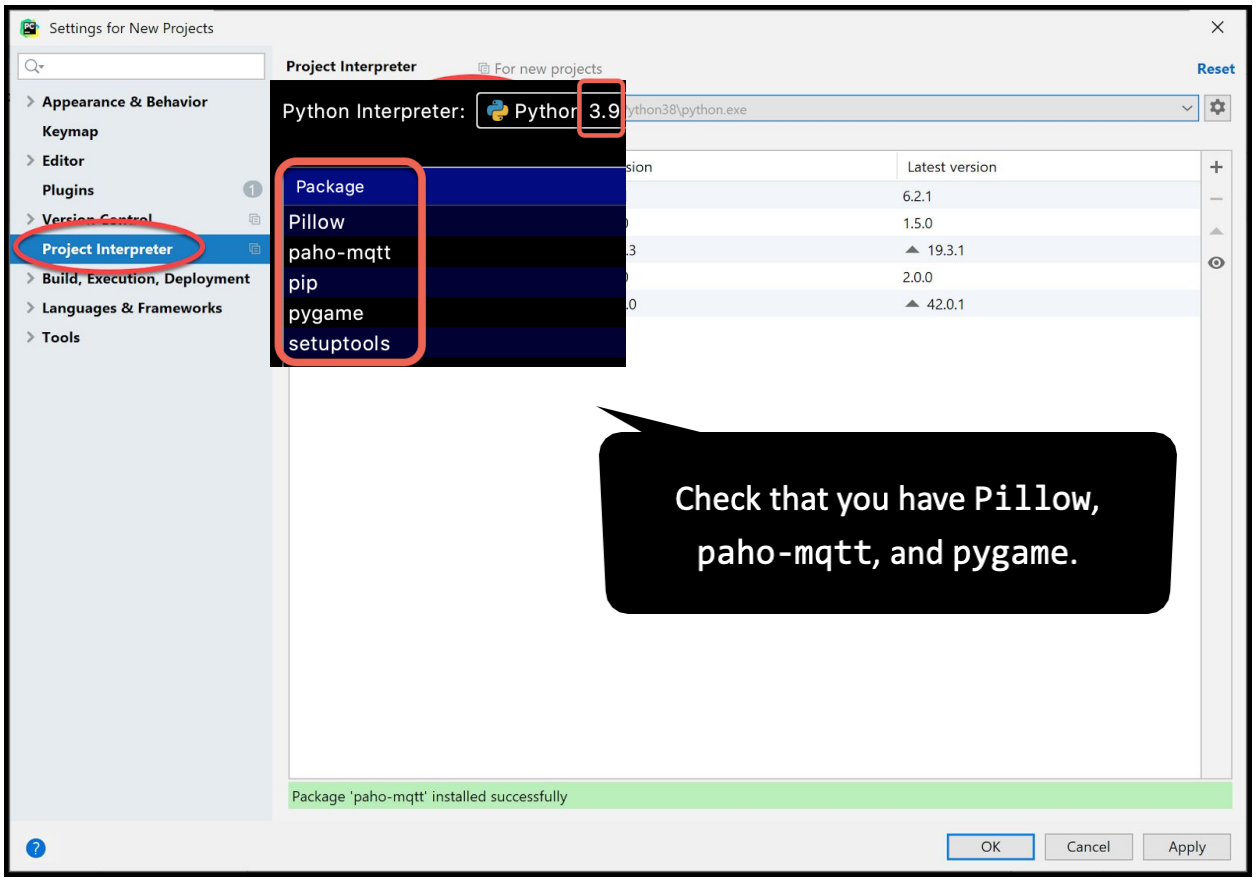


Now **repeat the process**, but this time installing the **`paho-mqtt`** package, as shown to the right. (Again be sure to have the box near the bottom UN-checked.)



Then **repeat the process**, but this time installing the **`pygame`** package. (Again be sure to have the box near the bottom UN-checked.)

After installing all three packages, **click on the X in the upper-right corner** to exit the *Available Packages* dialog and return to the *Project Interpreter* pane, which should now look as shown below.

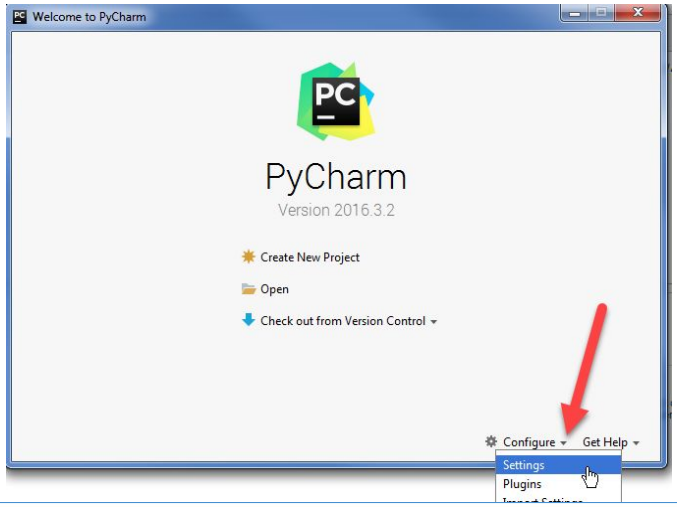


Press *OK* to complete this step.

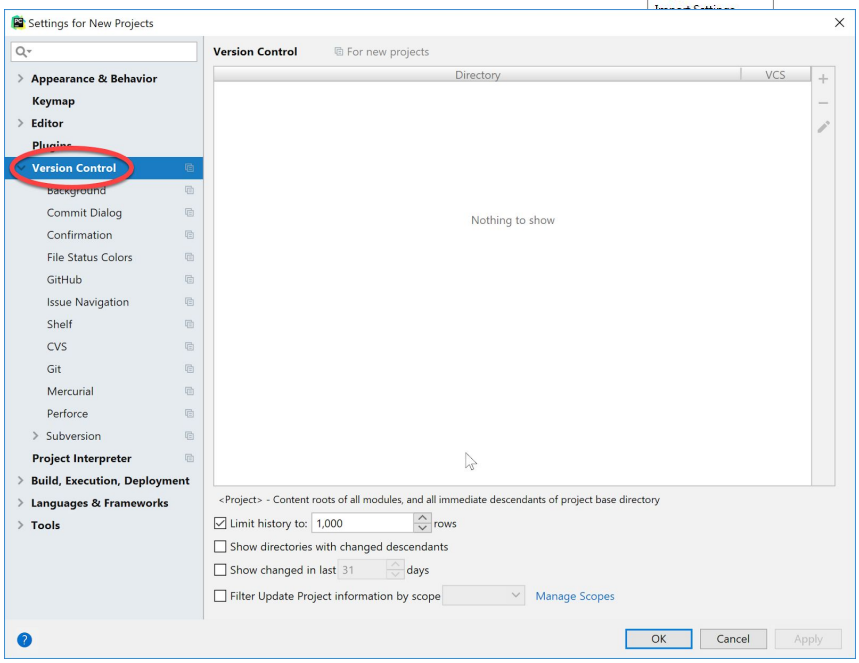
Part 5: Setting up Git in PyCharm

Set up Git in PyCharm, as follows:

If you are back in the main PyCharm main page, once again select **`Configure ~ Settings`** (or ***Preferences*** on a Mac), as shown to the right, to return to the ***Settings*** page.

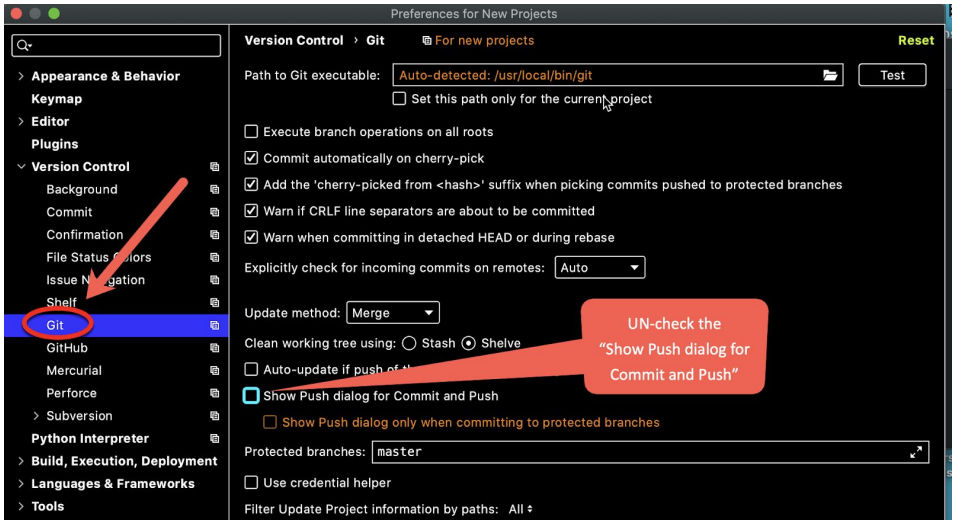


From the ***Settings*** page that appears, select and ***expand*** **Version Control**, as shown to the right.



From the expanded **Version Control,** select ***Git*** to get the screen shown to the right.

At that screen, UN-check the "Show Push dialog for Commit and Push", as indicated to the right. ***(Instructions continue on the next page.)***



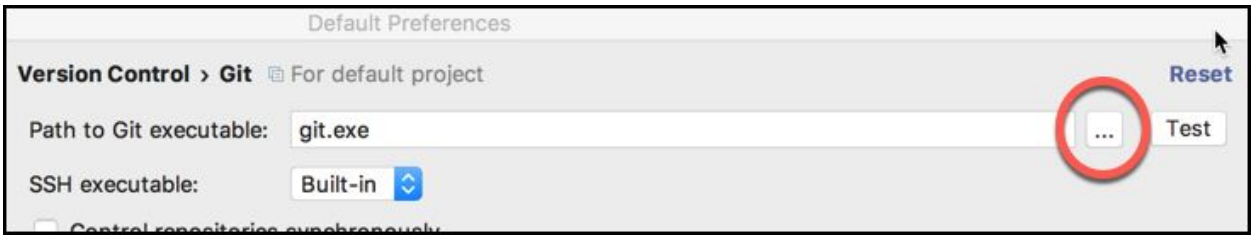
Near the top of that screen, you will probably see something like

the above, where PyCharm has already found where your ***git.exe*** file is stored on your computer and put it into the *Path to Git executable* as shown in the above screenshot.

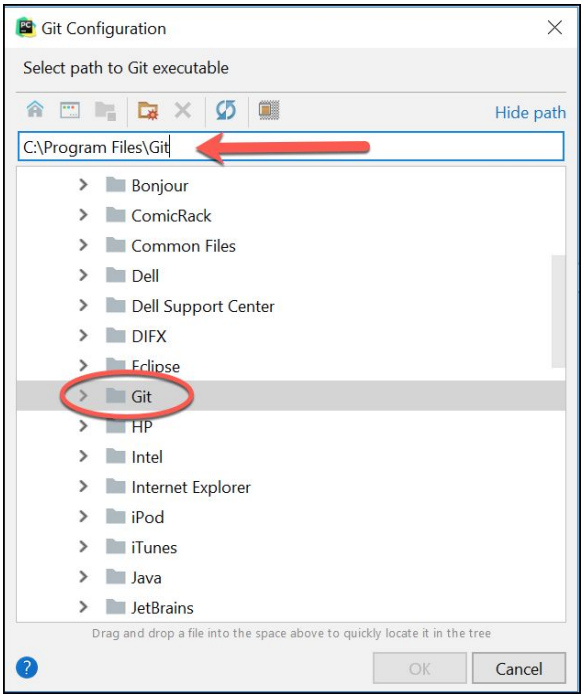
If so, **press the *Test* button** to check whether the path listed is indeed correct. If Git is installed successfully, it should then display the ***Git version***, as shown in the above.

If your Git appears to be successfully installed, as evidenced by showing the ***Git version*** as in the screenshot above, then **skip ahead to *Part 6: Testing your settings* on *page 22***.

But if you see nothing listed in the *Path to Git executable,* or if pressing the *Test* button does *not* display the Git version, then click the button that has the **three dots**, as shown below:

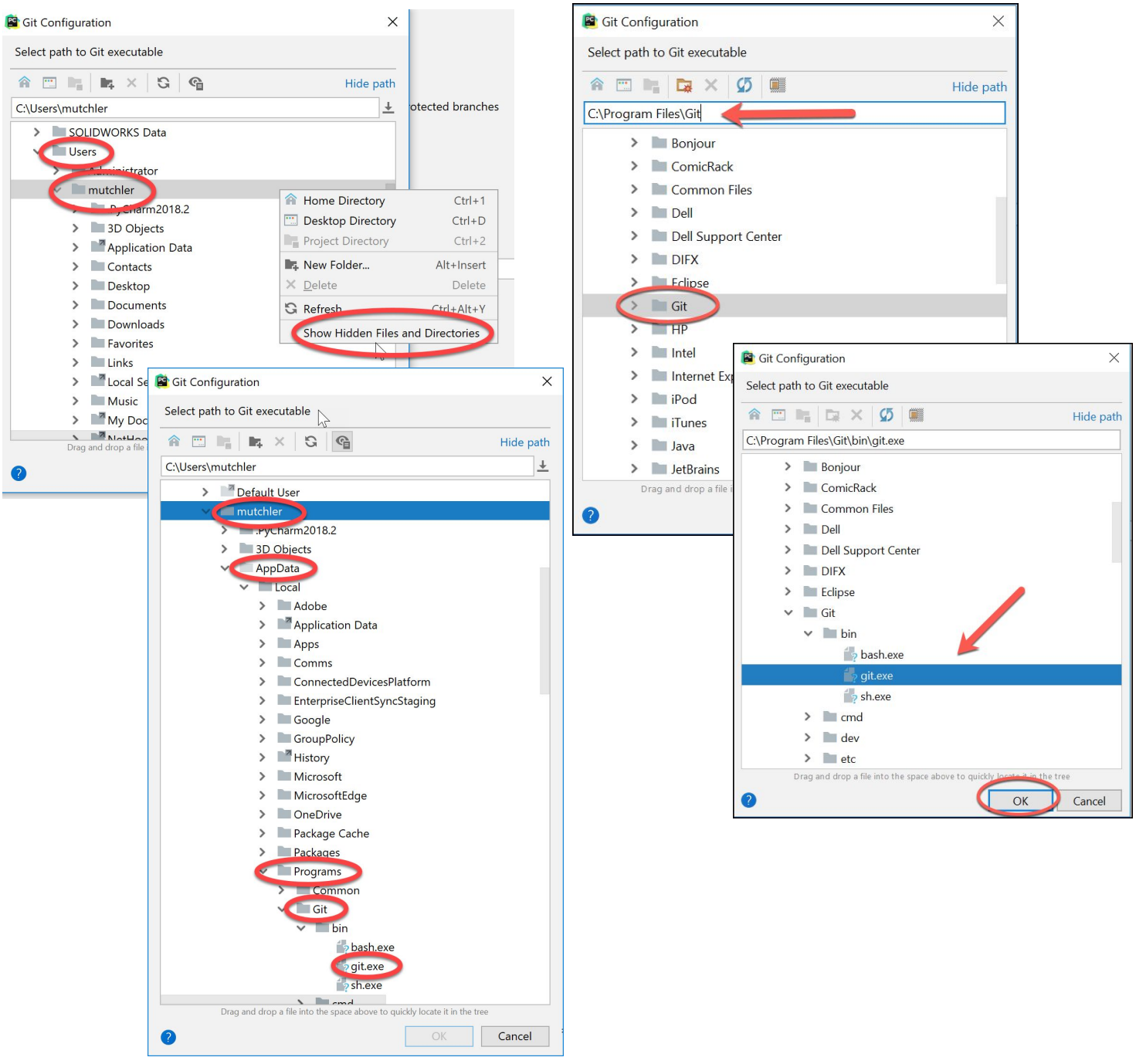


That will **pop up a submenu** (as shown to the right).



Expand folders as necessary to get to the *git.exe* file beneath the *Git* folder that you obtained when you installed Git, ***as shown on the next page.***

The file may be at **`C:\Program Files\Git\bin\git.exe`**, as shown below and to the right. Or, it may be beneath your (possibly hidden) **`AppData`** folder, as shown below and to the left. (Right-click on a folder to get a menu-item offering to *Show Hidden Files and Directories*.)



In any case, locate and select your **git.exe** file and then press *OK*.

You should now have a valid path to **git.exe**, something like that shown below. Once you have a valid path to **git.exe, press the *Test* button** to check whether the path listed is indeed correct. If Git is installed successfully, it should then display the ***Git version***, as shown in the above.

If you do NOT see the Git version shown, continue working through these instructions but get help from your instructor as needed. (If Git is NOT successfully installed, you will find out for sure in the next section.)

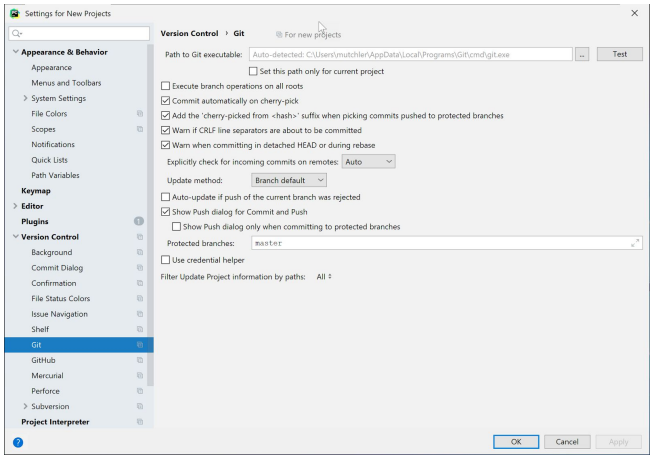
Part 6: Testing your settings

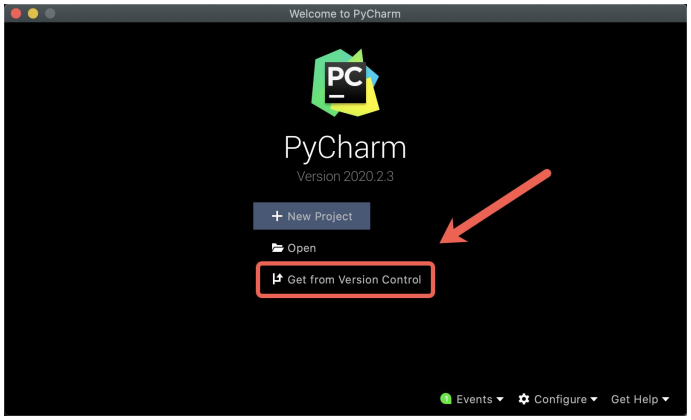
To test your settings, you will "checkout" the ***repository*** where all your work on CSSE 120 in-class projects will be stored, and then you will run a short test program to see whether all is well, as described below:

Important: If you are doing this step ***off-campus***, and more generally ***whenever you run PyCharm off-campus***, you should **run (i.e., connect to) the Rose-Hulman VPN first**. If you don't know how to do that, here are instructions for installing and running the Rose-Hulman VPN: https://servicedesk.rose-hulman.edu/knowledgebase/article/KA-01278/en-us.

Important: If this step (testing your settings) or any of the previous steps do not work for you, no worries. If you can get help from someone like your instructor before the first class session, that would be ideal, but if not, we'll help you get set up during the first class session.

If you are still on the ***Settings for New Projects*** page, as shown to the right, click ***OK*** to return to the PyCharm "home page"; it looks like that shown below.





Select ***Get from Version Control*** as shown to the left.

That pops up a ***Get from Version Control*** dialog that looks like this:

Be sure that ***Repository URL*** and ***Git*** are selected, as shown.

We recommend that you store your projects on your computer in the default Directory (folder), but if you want them somewhere else, change the ***Directory*** in the above as desired *and remember what you changed it to*.

Then type (you might start with a copy-and-paste) the following into the ***URL*** line:

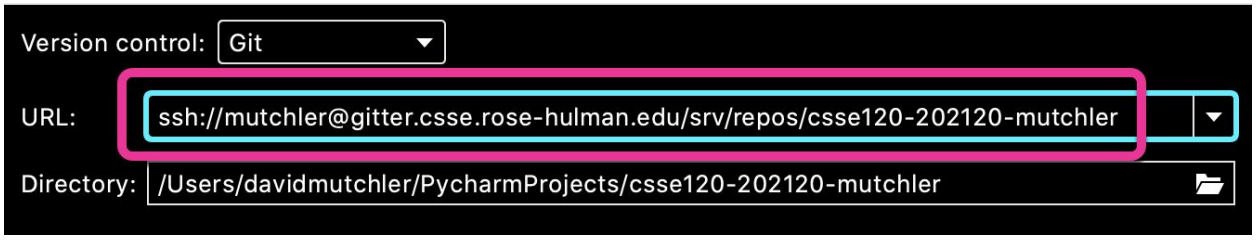
`ssh://`*`USERNAME`*`@gitter.csse.rose-hulman.edu/srv/repos/csse120-202120-`*`USERNAME`

where you replace ***USERNAME*** in ***both*** places with your ***own*** username.

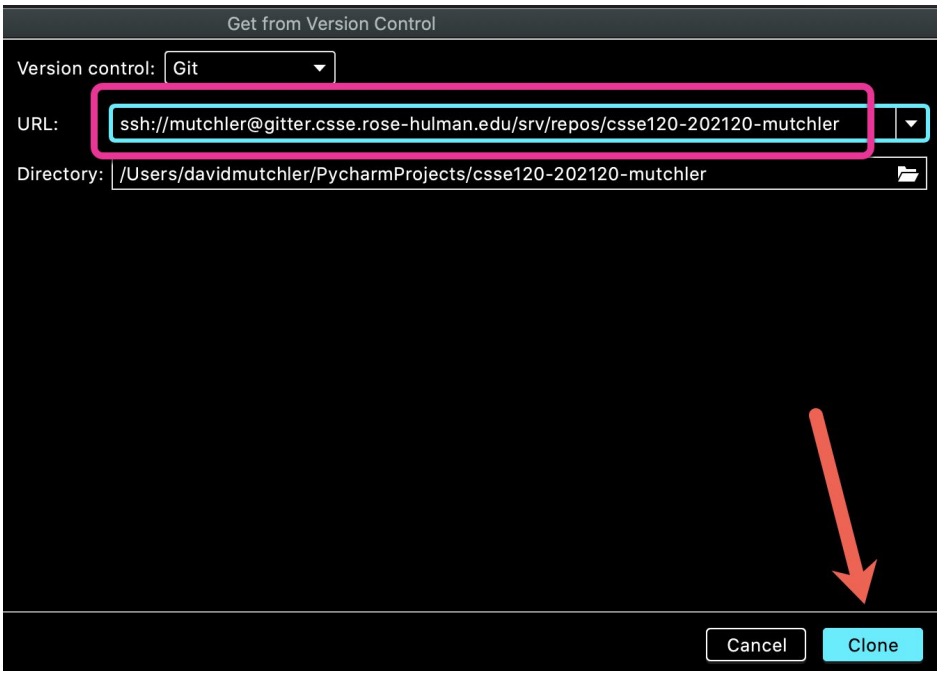
For example, if your Rose-Hulman email is **`swiftta@rose-hulman.edu`** then you would type **`swiftta`** in place of ***USERNAME*** in ***both*** places of the above. That is, you would type:

```
ssh://swiftta@gitter.csse.rose-hulman.edu/srv/repos/csse120-202120-swiftta
```

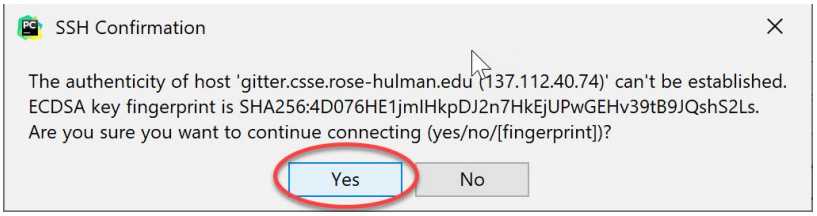
Here (below) is what it looked like when I (username **`mutchler`**) typed in the ***URL***. Note that it automatically puts the folder name at the end of the ***Directory***; you do NOT need to type anything on the ***Directory*** line.



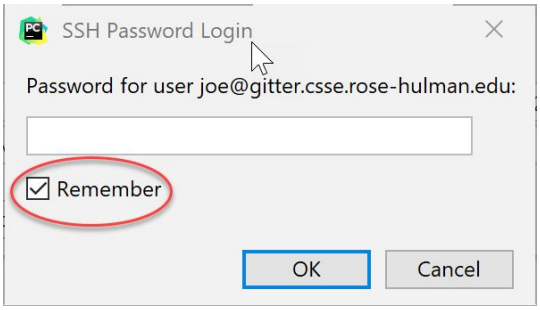
After typing in your URL (with your ***own*** username in ***both*** places), press the ***Clone*** button, as indicated in the picture below, and … (***instructions continue on the next page***)



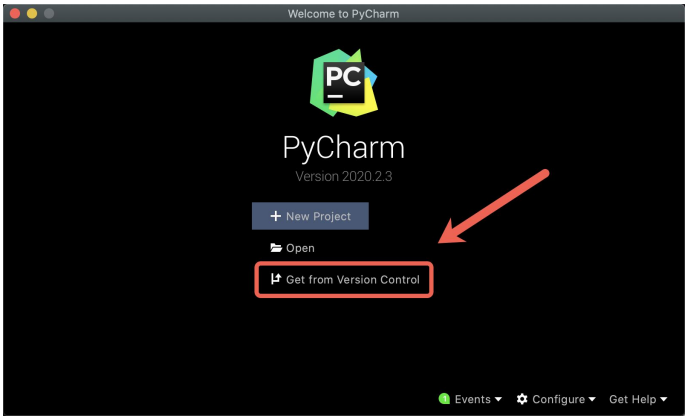
If something like that shown to the right (*asking about the authenticity of ...*) pops up at any point, click ***Yes***.



When it asks for your password, as shown to the right, type your Kerberos password there (that is, the same password that you use to log into anything Rose-Hulman). Also ***check the box*** to ***Remember*** that password.

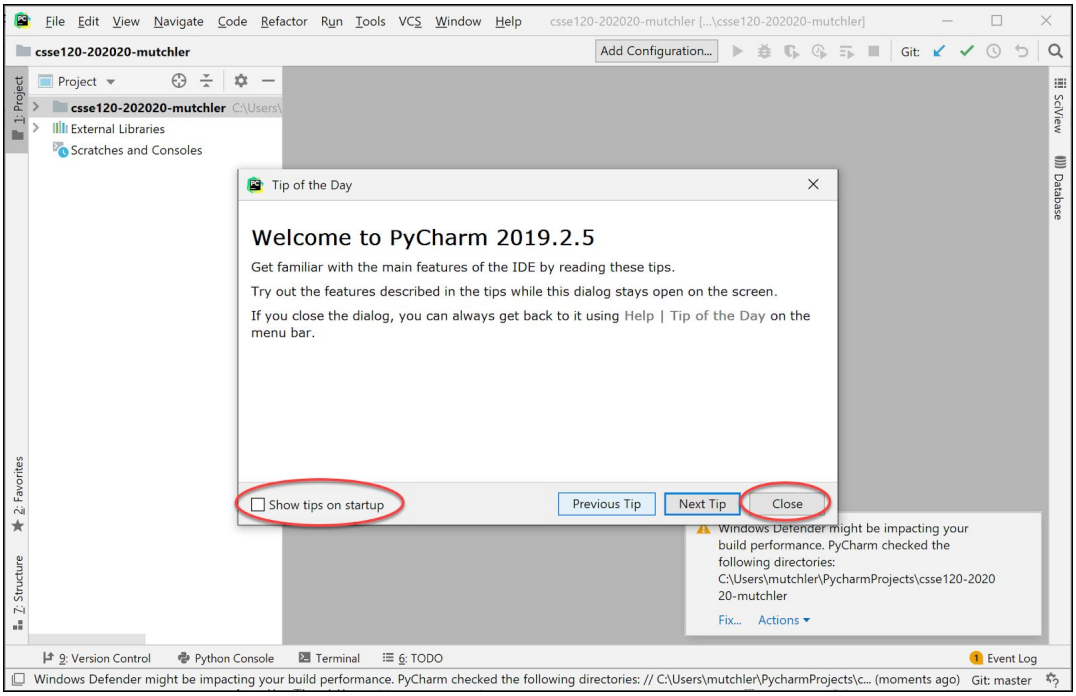


If the screen returns to the Home Page (as shown to the right) either before or after you entered your password, ***that means that something went wrong.***

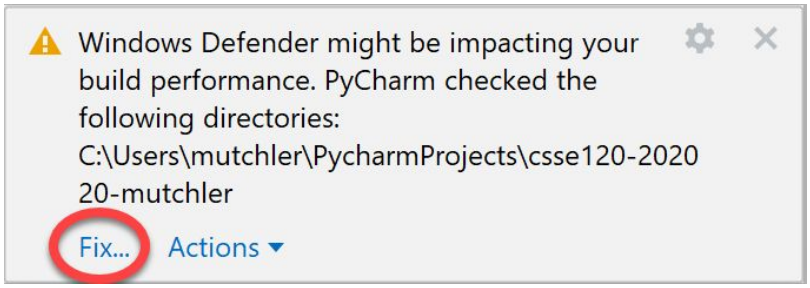


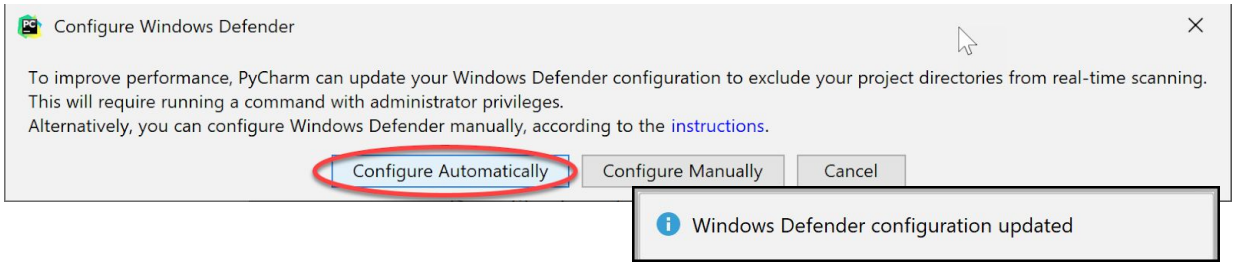
In that case, repeat the process from the previous page, being VERY CAREFUL when typing your URL and when typing your password. (If you copy-and-pasted the URL, try typing it by hand.) Try once or twice more and **if things still go wrong, no worries, CONTINUE TO PAGE 28** and ask for help on this step before or in the first class session.

But hopefully you have a successful connection and see a window like that shown the right, with a ***Tip of the Day***. If you don't want any more such tips, simply un-check the "*Show tips on startup*" box. In any case, ***close*** the TIp of the Day.

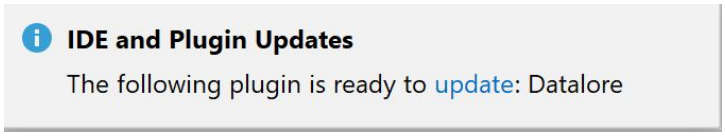


At some point, you might see a notification about ***Windows Defender***, like that shown to the right. If so, choose ***Fix***, then ***Configure Automatically*** as shown to the right and below, and proceed as directed from there. (After a few seconds you should see a "configuration updates" as shown below.)

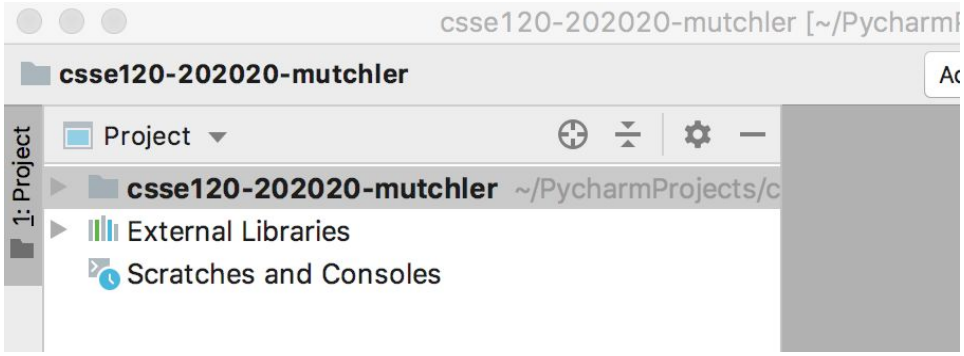




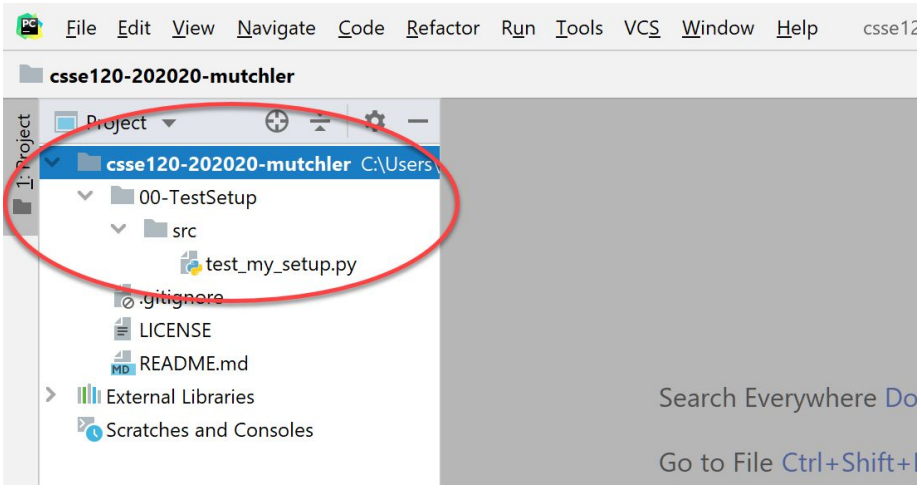
You also might see that one or more **plugins** are ready for updating, as shown to the right; if so, update them or not as you choose (we will not be using them).



The upper-left of your PyCharm should now look like the following, except with your own username (instead of *mutchler*). (If you don't see the "Project" part, click on the vertical *1: Project* on the far left sidebar.)

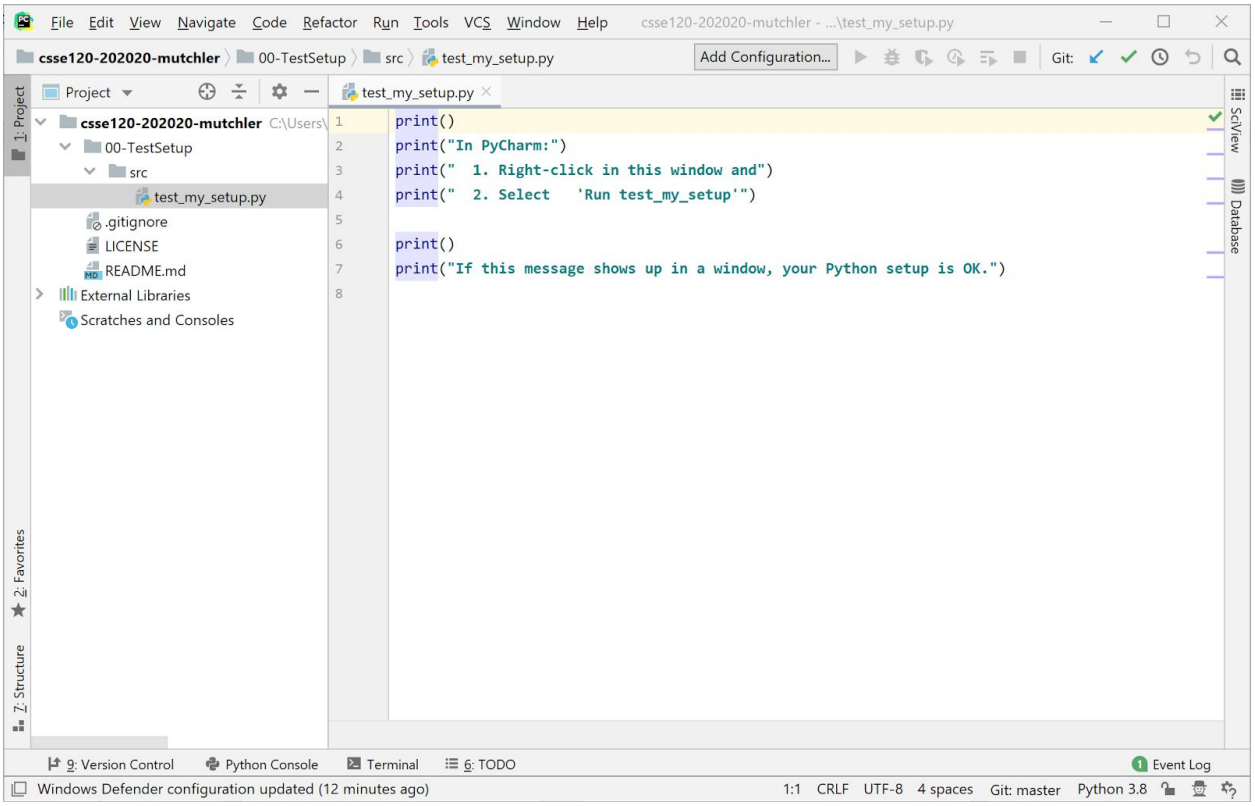


Expand (that is, click on the little arrow that points to the right) ***csse120-202120-username***, then expand the ***00-TestSetup*** that appears, then expand the ***src*** that appears below that, at which point you should see something like the picture to the right.

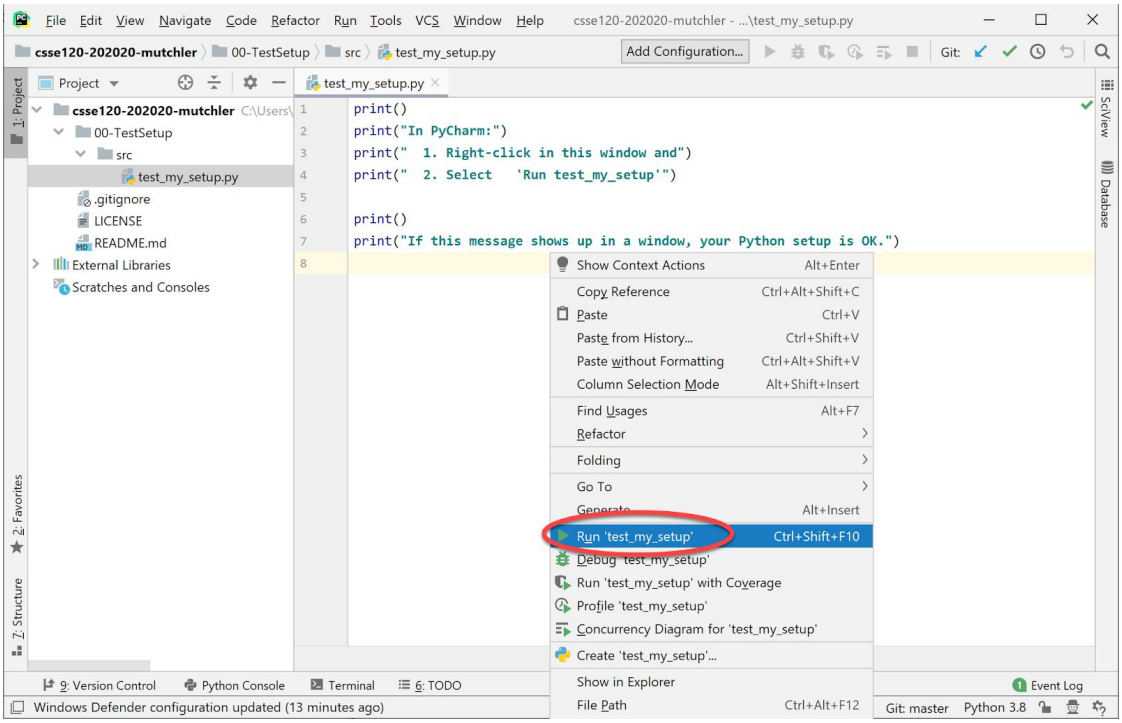


(You may also see a folder that begins with **`01-IntroductionTo…`**, or possibly even more folders, depending on when you are doing these instructions.)

Double-click on the ***test_my_setup.py*** file, at which point you will see its contents, like this:

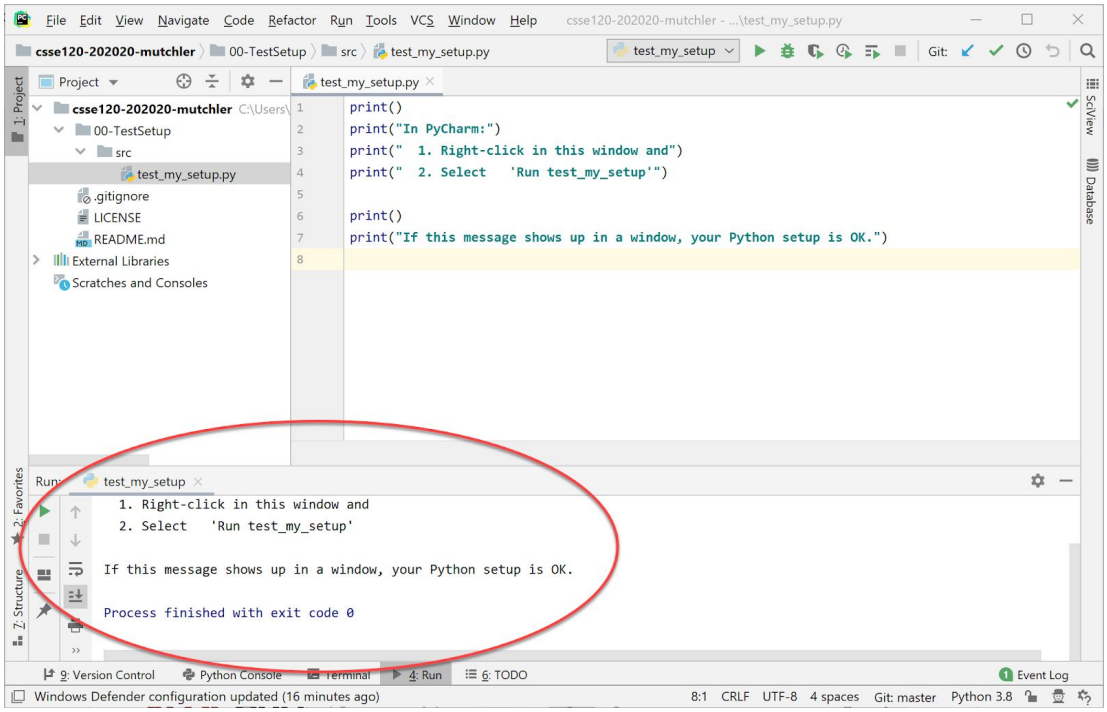


Now **right-click** anywhere inside the sub-window that shows the Python code and select ***Run test_my_setup***.



If a sub-window shows up with the output from the ***print*** statements (as shown to the right), your Python has been set up correctly!

Again, get help from your instructor if any of this does not seem to work for you.

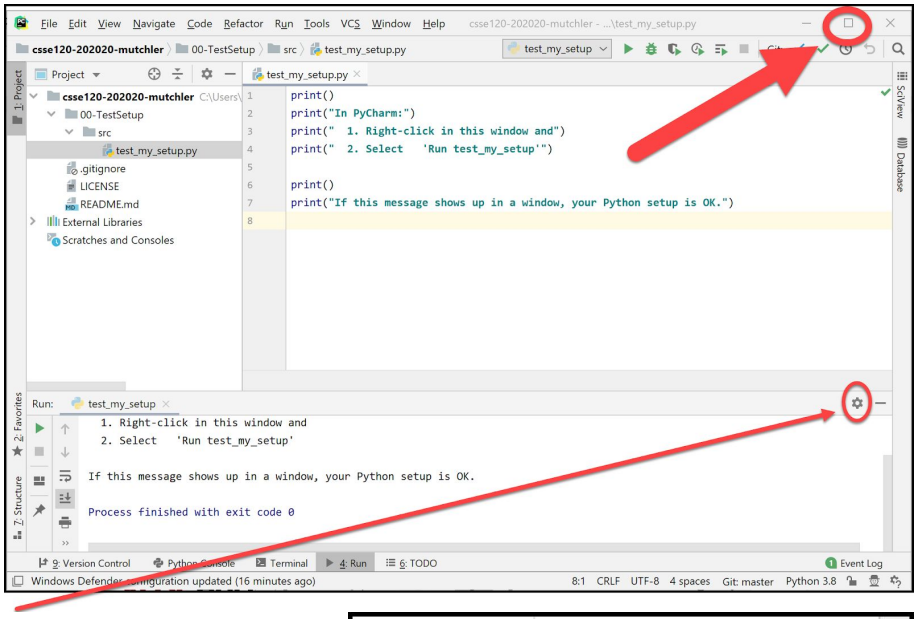


Part 7: Tuning PyCharm for CSSE 120

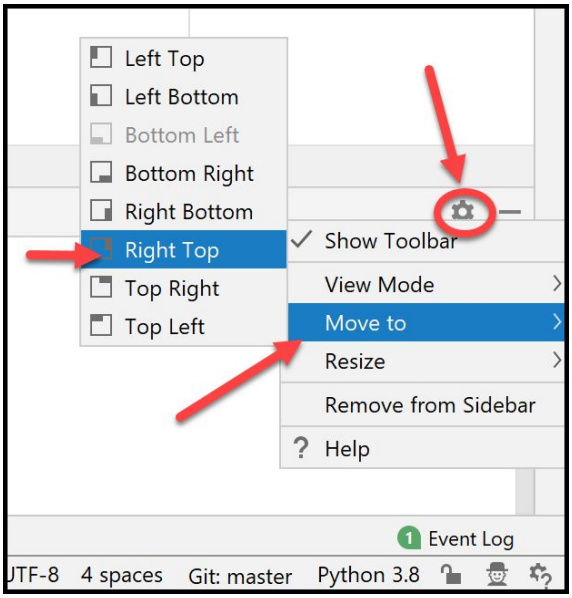
PyCharm works well "out of the box" but there are many, many changes that you can make to tailor it to your own needs. For example, if you are colorblind, you can easily change the color scheme. The following are the changes that CSSE 120 students have found most helpful.

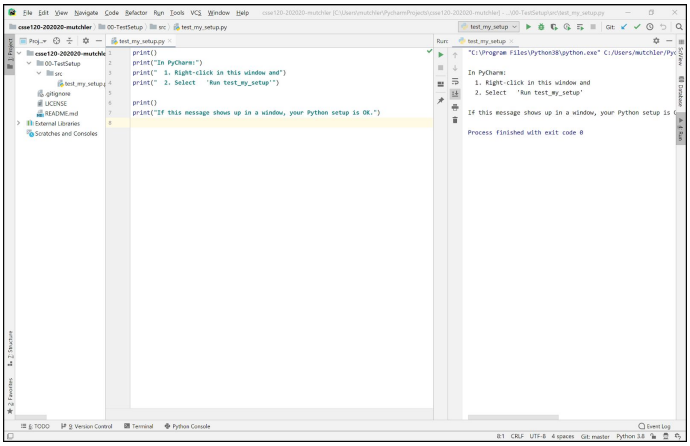
Make the changes as described below and then, *over time, tailor differently or additionally* as desired.

1. ***Maximize PyCharm*** if you have not already done so, since you will typically want lots of screen "real estate" for coding.



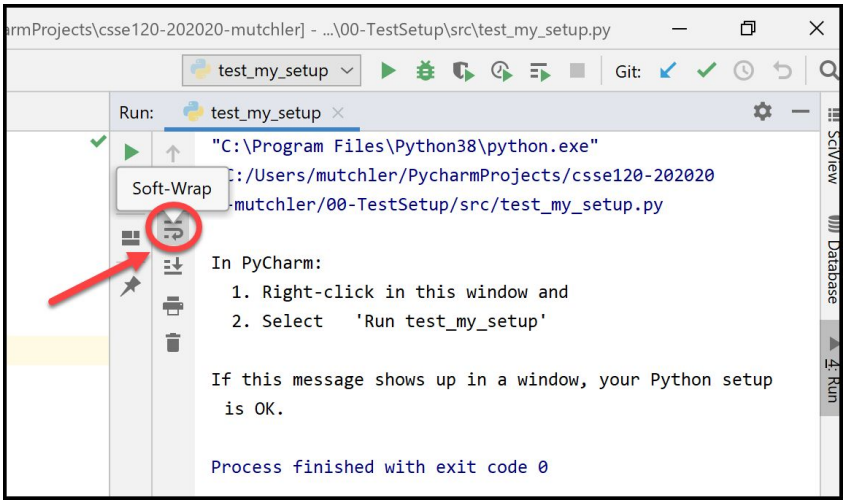
2. Find the **Console** window, where the ***print*** statements displayed their output. At the far right of the Console window, click on the little "***gear***" symbol (circled on the picture to the right), then select ***Move to***, then ***Right Top***. This will move the Console window from the bottom of the window to its right-hand-side, thereby allowing room for more output than if it were at the (default) bottom.





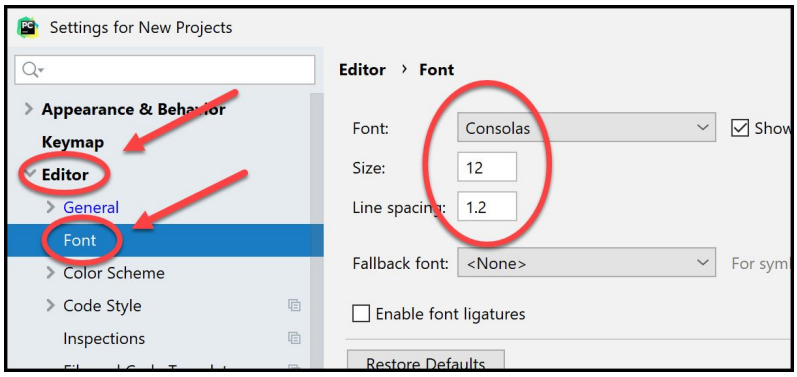
ROSE-HULMAN INSTITUTE OF TECHNOLOGY

3. Experiment with the ***"Soft-wrap"*** **button** on the left-hand-side of the Console window (see the picture to the right) until you understand how it "wraps" the text in the Console (or un-wraps it). Leave it in the "wrapping" or "non-wrapping" state, whichever you prefer.



4. Now do **`File ~ Settings`** to return to the "Settings for New Projects" dialog. ***Expand Editor*** and click on ***Font***, as shown to the right. Set:

   `Font` to **`Consolas`**
   `Size` to **`12`**
   (or bigger, if you prefer)
   `Line spacing` to **`1.2`**



(Continues on the next page.)

5. You might like to try the various Appearance ***Themes*** available, like this:

   Expand ***Appearance & Behavior***, then select ***Appearance***, then choose a Theme, as shown below.

Click ***OK*** at the bottom right of the "Settings" window to return to your code and see what the Theme looks like. Here (below) are samples of IntelliJ Light and Darcula, but really, you have to try them out to get a true sense of what they look like on your computer. (FWIW, I use ***High Contrast***, which is a higher-contrast version of *Darcula*.)

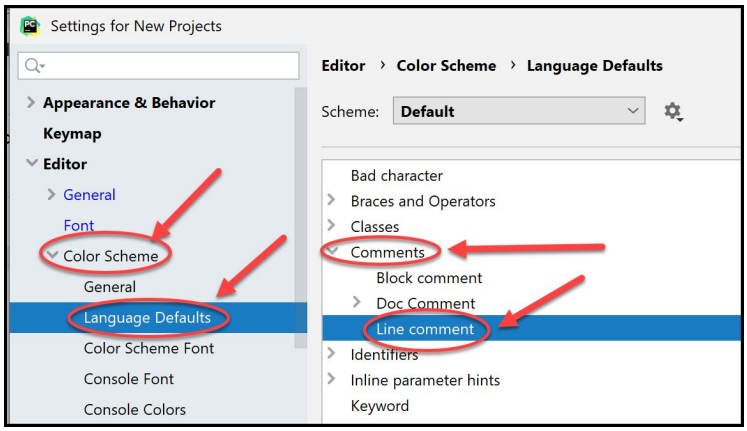
IntelliJ

```
def sum_more_cosines(m, n):
    """
    What comes in:  Integers m and n, with m <= n.
    What goes out:  Returns the sum
       cos(m) + cos(m+1) + cos(m+2) +  ...  cos(n)
    Side effects:   None.
    Examples:
      -- sum_more_cosines(0, 3)  returns
            cos(0) + cos(1) + cos(2) + cos(3)
         which is approximately 0.13416
      -- sum_more_cosines(-4, 1)  returns
            cos(-4) + cos(-3) + cos(-2) + cos(-1) + cos(0) + cos(1)
         which is approximately 0.02082.
    Type hints:
      :type m: int
      :type n: int
      :rtype:  float
    """
    # ----------------------------------------------------------------
    # TODO: 4. Implement and test this function.
    #   Note that you should write its TEST function first (above).
    #   That is called TEST-FIRST DEVELOPMENT (TFD).
    # ----------------------------------------------------------------
    total = 0
    for k in range(m, n + 1):
        total = total + math.cos(k)
    return total
```

Darcula

```
def sum_more_cosines(m, n):
    """
    What comes in:  Integers m and n, with m <= n.
    What goes out:  Returns the sum
       cos(m) + cos(m+1) + cos(m+2) +  ...  cos(n)
    Side effects:   None.
    Examples:
      -- sum_more_cosines(0, 3)  returns
            cos(0) + cos(1) + cos(2) + cos(3)
         which is approximately 0.13416
      -- sum_more_cosines(-4, 1)  returns
            cos(-4) + cos(-3) + cos(-2) + cos(-1) + cos(0) + cos(1)
         which is approximately 0.02082.
    Type hints:
      :type m: int
      :type n: int
      :rtype:  float
    """
    # ----------------------------------------------------------------
    # TODO: 4. Implement and test this function.
    #   Note that you should write its TEST function first (above).
    #   That is called TEST-FIRST DEVELOPMENT (TFD).
    # ----------------------------------------------------------------
    total = 0
    for k in range(m, n + 1):
        total = total + math.cos(k)
    return total
```

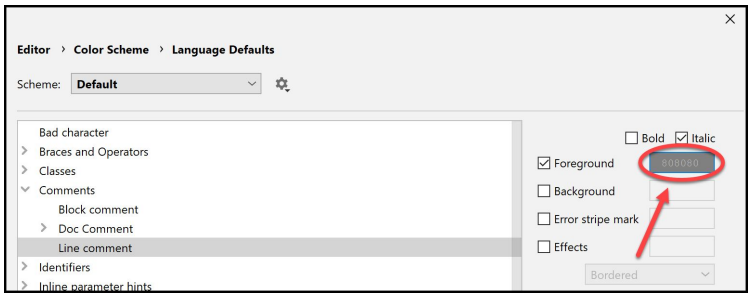
IMPORTANT: If you are using one of ***"Light"*** (white background) themes, we strongly recommend that you make Python'' so-called *Line comment* and *Block comment* symbols much more vibrant than the default, since we will use those to tell you what to do! Here is how to do so (on the next page):

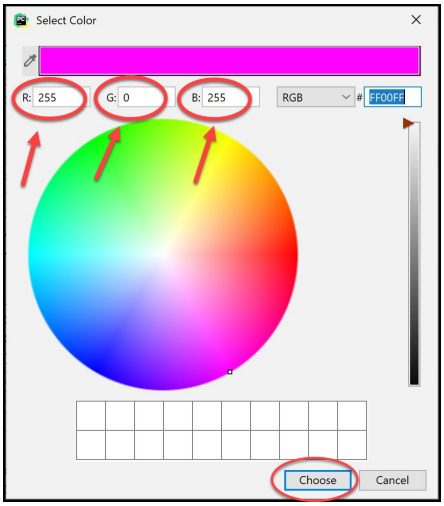


If you are using a "Dark" theme, leave the colors alone for now (and change them as desired once you use them a bit). **But for a "Light" theme, definitely try this:**

Do **`File ~ Settings`** to return to the "Settings for New Projects" dialog. Back under ***Editor***, expand ***Color Scheme***, then click on ***Language Defaults***, as shown to the left. In the pane that appears, expand ***Comments*** and select ***Line comment***, again as shown to the left.

On the right-hand-side of the pane, you will see ***Foreground*** checked. **Click on its color, as shown to the right.**





That will bring up a color-chooser, as shown to the left. Set the color to be nice and bright. If you are using a Light theme, you might choose:

```
def sum_more_cosines(m, n):
    """
    What comes in:  Integers m and n, with m <= n.
    What goes out:  Returns the sum
      cos(m) + cos(m+1) + cos(m+2) +  ...  cos(n)
    Side effects:   None.
    Examples:
      -- sum_more_cosines(0, 3)  returns
           cos(0) + cos(1) + cos(2) + cos(3)
         which is approximately 0.13416
      -- sum_more_cosines(-4, 1)  returns
           cos(-4) + cos(-3) + cos(-2) + cos(-1) + cos(0) + cos(1)
         which is approximately 0.02082.
    Type hints:
      :type m: int
      :type n: int
      :rtype:  float
    """
    # ----------------------------------------------------------
    # TODO: 4. Implement and test this function.
    #   Note that you should write its TEST function first (above).
    #   That is called TEST-FIRST DEVELOPMENT (TFD).
    # ----------------------------------------------------------
    total = 0
    for k in range(m, n + 1):
        total = total + math.cos(k)
    return total
```

For ***Line comment***:

R (red):	**255**
G (green):	**0**
B (blue):	**255**

For ***Block comment ~ text***:

R (red):	**0**
G (green):	**125**
B (blue):	**0**

to get the effect shown to the right. In this and all these tunings, start with the suggested choice as above, try it out for a while, and then change as desired.

Later, we will suggest other tunings that you might like, but the above are the ones that have proved most important to previous students.