Step 1: Find your teammates and practice tkinter, as follows:

1. If you have not already done so, check out the tkinter_ttk and tkinter_ttk_part2 projects, in the usual way.
2. Find out which of your teammates are online (perhaps by emailing each other for starters). I will attempt to make a Lync Chat Room for each team, and I will invite you to join your team's Chat Room. If that does not work, do something else to communicate with your team and with me.
3. While you are waiting for teammates, open the tkinter project. Review its m00, m01, m02 and m03, which the Session 19 Prep discussed in a video. Practice tkinter by doing stuff in m99. Go back to m04 and m05 when needed.

Keep practicing until either your entire team is present or you have decided to move on without them. (Give them a reasonable length of time to show up.)

In the next video, I will assume that your team is all working at the same time. ** WHEN YOU ARRIVE, SEE WHERE YOUR TEAMMATES ARE IN THIS PROCESS ** and adjust as best you can.

Step 2: Get your shared repo and learn how you avoid stepping on each other's toes, as follows:

1. You SHARE a repository.
2. Check it out now: SVN ~ Checkout, Use a NEW Repository Location. Use the team name that I sent you.

   http://svn.csse.rose-hulman.edu/repos/csse120-201540-team00-larry-curly-moe [but use your OWN team!]
3. M1 is the first person in your team name, m2 the second, and m3 the third. 1 uses m1, 2 uses m2, 3 uses m3. M0 is shared (more on that soon). 2-person teams, just ignore m3 throughout and just do 2 things whenever I say "all 3".
4. Run your M.
5. Put YOUR OWN NAME in YOUR OWN m file. Run and be sure all is still OK.
6. Commit your M, indicating that you put your name in it. COMMENTS ARE KEY from here on out, as you will shortly see. PUT A SHORT COMMENT IN EVERY COMMIT FROM HERE OUT.
7. After you commit, do SVN ~ Update to Head. This gets the changes YOUR TEAMMATES made! Keep doing this until you see a change in the date for a teammate's m. Then look at their M. DO NOT CHANGE IT. But notice that you got the change that your teammate made.
8. You avoid stepping on each other's toes by working in YOUR OWN M. If you commit a teammates' M, even if the change was just to put an extra space, bad things happen. (I'll give separate instructions on those, but best to avoid the bad.)
9. M0 is shared. Decide which ONE of you will do ALL the work in m0 tonight.

Step 3: Work collaboratively to make your project have three frames, one for each of you, as follows:

1. M0 person, add names of all teammates to it. Commit. Others Update to Head. End of this M0 person.
2. Each of you, work in your own M.
3. Make your main call the main in m0: m0.main(). Note that the dot trick works! Run. Make sure you understand what is printed and why.
4. In your own M, put the following code into your my_frame function, but use your OWN name (not 'Moe'). You will NOT be able to test it effectively quite yet. As you saw in the Prep, it puts a button on a frame, but there is not yet a root window for the frame. Commit your code.

```
frame = ttk.Frame(root)
button = ttk.Button(frame, text='Moe')
button.grid()
return frame
```

5. Whoever finishes first, be the new M0 person – communicate with teammates to make SURE that you don't have 2 of you being M0. M0 person, put into M0 the following code in main:

```
root = tkinter.Tk()
frame1 = m1.my_frame(root, None)
if frame1 is not None:
    frame1.grid()
root.mainloop()
```

6. M0, run M0. Make sure nothing breaks. If something breaks, then it is probably in M1 and the M1 person must fix it!

   If M1 has finished, their frame should appear.

   M0, you can use M2 (or M3) if M1 is having trouble – just change m1.my_frame... to m2.my_frame in M0 code.
7. M0, once nothing is breaking (and no red Xs), commit your code. Tell your teammates to Update to Head.
8. Teammates, Update to Head. Run YOUR M.
9. Keep going until everyone's M displays M1's frame.
10. Then, M0 add lines to M0, where the new lines are highlighted.

```
root = tkinter.Tk()
frame1 = m1.my_frame(root, None)
frame2 = m2.my_frame(root, None)
frame3 = m3.my_frame(root, None)
if frame1 is not None:
    frame1.grid()
if frame2 is not None:
    frame2.grid()
if frame3 is not None:
    frame3.grid()
root.mainloop()
```

11. M0, test. You should see ALL three frames show up now. There may be errors in M2 or M3's code – THEY must fix it.
12. M0, once all seems OK, commit your code.
13. Tell your teammates to Update to Head.
14. Teammates, Update to Head. Run YOUR M. You should now see ALL 3 frames.

Step 4: See your teammates work and comments, as follows:

As you continue to add items to YOUR frame in YOUR M, the new items will show up to your teammates when you commit and they Update to Head! So:

- Do NOT commit if your code has any red Xs or you think that it will break a teammate's code.
- Every time you BEGIN a work session, select the entire project and:
    - Do an Update to Head to your teams' work.
    - Do a SVN ~ Show History. It shows you the comments. See why they are important!
- Somehow, you have to avoid two people committing M0 "at the same time". To that end, if you need to change M0 (as in the NEXT video):
    - If a team member is online, tell your team member you are working on M0, do an Update to Head on M0, do the work, commit your work, and tell your team member to Update to Head on M0.
    - If a team member is NOT online, just minimize the time between your Update to Head on M0 and your Commit on M0.
    - If a team member screws up their code and hence M0 breaks, you can comment out the call to their frame in YOUR M0, but BE SURE NOT TO COMMIT YOUR M0 without coordination (and letting your teammate know that their code is broken). You can do whatever you want to M0 TEMPORARILY. Just be careful NOT to commit it! (Hit Cancel when asked to put a message.)

Exercise: Practice the above by making some more buttons or things in your frame in your M, testing, and committing when your code runs OK. Do periodic Update to Head of teammates M. Nobody changes M0 during this exercise.

That exercise shows how you can work independently, effortlessly get each other's work, and not step on anyone else's toes.

<u>Step 5</u>: Show the answer to the previous work and review the following:

1. Your team has a shared repository with a single project in it called Z_Project.
2. That project has a src folder that contains the code that your team will write.
3. It also has a process folder. It has a place for you to put contact information if you want, and for you to put THE HOURS YOU WORK ON THIS PROJECT. All your time counts, including watching this video. At every work session, put a number into that file. (The computer can add up the numbers for you.) To nearest half-hour is plenty good enough.

   BUT: Be sure that if you are Person 1, you put your work in hours1.txt. If you are …
4. Back to the src folder. Each of you has a file – Person 1 uses m1, etc. The "work flow" process that you will ALWAYS use is:
   a. Check the time.
   b. Look at your m0, m1, m2, m3 (and perhaps the ones under process). If YOUR M has black beside it, that is OK – it means that you have done work that you are not ready to give to your teammates. If any OTHER has black beside it, then you have accidentally changed one of your teammate's file or the shared m0. That is probably a mistake – fix it BEFORE doing the next step.
   c. Select the src folder and do SVN ~ Update to Head. That gets the work your teammates committed since your last work session.
   d. Do SVN ~ Show History to see a quick summary of what your teammates did.
   e. Do your work on YOUR M (and sometimes M0, as we will see in the NEXT video). Every time you get to a "stable" place where your code does not break things, COMMIT your M with a short message.

      In grading, I will look for frequent commits with reasonable messages.
   f. When you are done with your work session (and perhaps committed your M if it is at a stable point), check the time. Open the hours-M.txt in the process file. Put your hours in it. COMMIT that file by selecting it and doing SVN ~ Commit.
5. Pause the video and do the preceding if you have not already done so.

6. Let's review the code that we have so far. First let's run it.
7. First m0:
   a. Makes a Tk (root window) and does a mainloop on it. (Show that alone.) SITS THERE waiting for humans to do things.
   b. Calls the 3 M's my_frame functions sending them the root (so that they can put their frame on the root) and eventually another thing. Each of the my_frame functions should return that person's frame with whatever Buttons, etc that person has put on it. They will do whatever that person told them to do.
8. Now let's look at M1 (and M2 and M3) are similar.
   a. Main calls m0.main() so that you all are running the SAME PROGRAM. You ALWAYS get your teammates work, in the most recent state that she has committed her work.
   b. My_frame constructs and returns a Frame (which m0 grids to make it show up). Currently, M1's frame just puts up a Button and then returns to M0 which enters its mainloop.
9. I will make one improvement now, which each of you should ALSO make to YOUR M. Let's make the frames stand out a little from each other. I'll use M2.

```
frame = ttk.Frame(root, padding=10, relief='groove')
```

and a bit more:

```
button = ttk.Button(frame, text='More stuff')
button.grid()
button = ttk.Button(frame, text='Curly')
button.grid()
```

10. At this point you should understand:
    a. You share a repository and the project in it.
    b. Your workflow is:
       i. Check the time.
       ii. Select src and SVN ~ Update to Head (but not if black dots by teammates code or M0)
       iii. SVN ~ Show History to see what your teammates did.
       iv. Do your work. FREQUENT commits, but always at a point at which your code does not break. NEVER if your code has syntax errors (red).
       v. When done, add your hours to your M-hours.txt file.
    c. Almost all of your work is in YOUR M.
    d. Your M makes a frame on which you can put buttons, menus, sliders and more.
    e. Those buttons, menus, sliders and more will do whatever you tell them to do for the project.
    f. The document reachable from the course home page lists the MANY options (and REQUIREMENTS) you have for your capstone robotics project. Note also Graphics and Resources links.

Questions about any of that? If so, post to Piazza or email as appropriate.

Step 6: How you SHARE information via M0, while practicing "safe M0", as follows:

1. In the previous step, you "hard-coded" your name into your Button. Now we are going to make your names become SHARED DATA.
2. Let's start in m0. Just watch this entire video, don't type, yet.
3. DataContainer for shared data.

```
    self.our_names = ['alpha',beta 2'
'gamma]
```

4. In m0 main (which everyone's main runs), we will make an instance of the data container..

   ```
   dc = DataContainer()
   ```

   It contains our names, see the DOT trick

   dc.

It's just like a circle has a radius and a fill_color and you reference those with the dot notation. A dc will have whatever shared information you want and access with the dot notation. For example, all 3 may want to do things with the robots speed, but there should be a SINGLE speed variable that you share.

5. M0 main sends that single dc to each frame:

   None -> dc

6. At some point tonight, your team should decide which ONE of you is going to make your main look just like this. That person will want to look at this screen to do so. The changes that need to happen at this time are to construct a DC and send it to each of the 3 frames. Communicate with each other to determine who will make and commit those changes.
7. Now let's turn to the Ms. First, let's USE that shared data.
8. Dc. Shows ournames.
9. So I could put it on my button. Run.
10. You should now see that if data is in the dc object that the three frames share, any of you can USE that data. Now let's see how you can SET and change that data.

Rest is not 100% as we will do it this term. Idea is to end with 3 buttons:

1. Person 1: Connect to SIM
2. Person 2: Connect to Robot with Port (using instance variable in DataContainer object, later to have an Entry to set that instance variable)
3. Person 3: Test Robot (run it for 2 seconds).

11. To do so, I will do the beginning steps of making a button that connects to a robot. Your team will collaboratively complete this work however your team sees fit. Just be sure that all of you UNDERSTAND what you do.
12. I'll work in m1, but you could work in any of the m1, m2 or m3 – but not all 3, decide who is going to do this.
13. I'll make a new button and I will make that button do something. As you saw in the Session 19 Prep, the way to do that is:

    ```
    button['command'] = lambda: connect_to_simulator()
    ```

14. Left-hand side means ..
15. Right-hand side means…
16. Make the connect_to_simulator function. Quick Fix.
17. Remember from session 5 that to connect to the simulator:

    robot = new_create.Create('sim')

18. Test.
19. BUT: local variable! Need to get that data to other functions and modules. SO: Pass dc in and MUTATE dc.
20. Add dc to call. Back to m0 and add self.hal = None . Back to m1, change robot to self.hal = … Once you have this, each of you should make a simple button that makes the robot move for 2 seconds and stop (see Session 5). If you get that right, you probably are good for sharing data.
21. Ask questions! Project is all about learning HOW TO LEARN, using: links from home page, Google (but you must UNDERSTAND whatever you use! And say where you got the idea from in a comment in the code!), teammates, Piazza, and ME!