The Main Ideas from Session 21

1. SVN: working as a team
    a. When you commit to your team project, always include a comment that tells your teammates what you have done, and particularly tells them if there are ways they will need to change their code as a result if what you have done.
    b. You can see the most recent commit comments from yourself and your team members by selecting your project ion Package Explorer and then doing SVN→Show History
    c. SVN->Update to HEAD will copy everyone's changes from your team's repository to your computer.
    d. You should never change or try to commit your teammate's code files. This will cause conflicts that will be hard to resolve.
    e. Whenever you are about to make changes to the **c0.py** file, you should let your teammates know, so no one else will be trying to change it simultaneously. After you commit, everyone else should update. IN general you should do Update to Head for his file immediately before editing and immediately before committing.
    f. If you do get a conflict, it is best to get help from your instructor or someone else who is knowledgeable about SVN.
2. The goal for Friday's class (and your team's follow-up)
    a. Take all of the code that you wrote after session 20, integrate it with the other team members' code, and get it all working together.
    b. If you did not complete this in class, get together as a team and spend some time working on it this weekend. If you get completely stuck, get help. It is okay if this is not completed until Monday's class, or even sometime Tuesday.
3. Integrating your project.
    a. Things that everyone's program needs to use should go in your **gui()** function, or in something called from there.
    b. The program should construct exactly *one* **Tkinter.Tk** object and only one **mainloop**. If either of these is done in **gui()**, it will be a problem. Both should happen in **main()**.
    c. For now, each person's function should make a ttk.Frame, add it to the window, and put all of your widgets (buttons, labels, etc.) in that frame.
    d. Later, your team will probably want to integrate the various GUI features into a nice-looking interface. For now, concentrate on functionality.
    e. The program should construct exactly one **new_create.Create** object. This should happen when the human user of the program presses the Connect button, and not before this.
4. Shared data
    a. There is certain information that everyone's code needs to know about and be able to modify. For now, there are three such things: The robot object, the speed, and the time (number of seconds to run). These things should be fields of an object from a class that you create.
    b. It is not obvious what the name of this class should be; pick something that works for your team. For definiteness in this document, I will call the class **Robot_and_its_Data**.

c. Your program should call the constructor for the **Robot_and_its_Data** class exactly once, in **main**. It should be passed as an argument to everyone's **gui()** function, which should in turn pass it as an argument to any event handler that needs to use or change its fields.
d. You will need to modify the event-handling functions that you wrote so that they expect this object as a parameter (perhaps with a local name like **robot_with_data**), and access the data they are supposed to use as fields of that object (e.g., **robot_with_data.speed**). If you prefer can also write methods to manipulate the data in this object.