# PAIR PROGRAMMING MANUAL: LEARN Y'ALL KICKFLIPS ON THE TI-84

COLTON GRAINGER

Person 1: ______________________________

Person 2: ______________________________

Please find a partner. This is y'all's manual for the next week. *Bring it to each class before the midterm.*

## README

**Please read these wikiHow[1] instructions with your partner.** To get in the right mindset, you might offer to read aloud one section, then switch with your partner for them to read the next section.

> **Pair programming** is a method of programming in which two programmers work together at one workstation. One, **the driver**, writes code while the other, **the navigator** reviews each line of code as it is typed in. The two programmers switch roles frequently.[2]

**Step 1: Start with a reasonably well-defined task before you sit down.** The task should be something you are confident that you can complete in an hour or two. For example, "run a T-test" on some legit numerical data. You may find it helpful to outline what you plan to do before you begin to code.

**Step 2: Agree on one tiny goal at a time.** Something you can complete within a few minutes. Stating the problem in words to another person helps focus your mind and helps engage your partner's mind. It also ensures that you both know what you are working on right now.

**Step 3: Rely on your partner, support your partner.** When you're the driver, complete the current tiny goal as quickly as you can, ignoring larger issues. Trust the observer to be your safety net.

When you're the observer, read the code that the driver is writing as they write it. Your job is code review. You should pay total attention, aiming to let nothing get by you. Think about possible bugs, larger issues, and ways to simplify or improve the design. Bring up errors and code that you find unreadable right away.

When you're the observer, don't dictate the code. The driver should be actively thinking about how to achieve the current task, not just typing passively. And as the observer, you should exploit the fact that you don't need to invent the small details; you can and should think at a higher level. Saying "That looks right. How about handling the case once $\sigma$ is known?" is better than "OK, now type 'STAT, TESTS, 1, ENTER, ...' "

**Step 4: Talk a lot!** Say what you are about to do, ask for an implementation idea, ask for a better way to solve the problem at hand, bring up alternative ideas, point out possible inputs that the question didn't state, tell the driver that little bit of mathematical knowledge that they need right at the moment they need it, etc. Listen a lot, too, of course. When people are pairing well, they are talking back and forth almost non-stop. Here are some common things to say while pairing:

- "Do you think this is a valid test?"
- "Does that look correct to you?"
- "What's next?"
- "Trust me" (when it's easier to write a little code to make your point than to say it out loud).

*Date*: 2019-11-12.

[1]`https://www.wikihow.com/Pair-Program`

[2]`https://en.wikipedia.org/wiki/Pair_programming`

**Step 5: Sync up frequently.** As you are working together, you will find yourself getting out of sync: becoming unsure what your partner is doing, or becoming unclear about the current task. This is normal. When it happens, sync up again. The key to good pairing is to sync up very frequently—within seconds or a minute of noticing that you're out of sync. If you are spending five minutes (or more) out of sync, you might as well be coding solo, because it's the frequent re-syncing that creates the synergy of pairing.

- When you can, say what you are about to do before you do it. Better yet, ask your partner; for example, "Shall we create a confidence interval to support our conclusion now?"
- When your partner asks if you agree with something, like "Shall we perform a `1-PropZTest` on the sample data now?" or "I think this problem is just conceptual. Do you agree?", say "Yes" or "No" clearly and immediately.

**Step 6: Take a moment to celebrate as you complete tasks and overcome problems.** For example, each time you get a test to pass, give each other a high five. If you also high-five each time you get a new test to fail, you'll really get into the groove of collaborative programming and test-driven design.

**Step 7: Switch roles often—at least every half hour.** This keeps you both fully engaged, both of you in tune with the low-level details and the big picture. Also, driving full-blast can tire you out, and it's hard to maintain the vigilance required of the observer role for longer than half an hour. Switching roles recharges you.

TODO

You have three 30 minute blocks of minutes of class time (and as much time outside of class as you plan to meet) to "learn to do kickflips" with **each of the following TI-84 functions**.[3]

1. `Z-Test`: Test for 1 $\mu$, known $\sigma$
2. `T-Test`: Test for 1 $\mu$, unknown $\sigma$
3. `2-SampZTest`: Test comparing 2 $\mu$'s, known $\sigma$'s
4. `2-SampTTest`: Test comparing 2 $\mu$'s, unknown $\sigma$'s
5. `1-PropZTest`: Test for 1 proportion
6. `2-PropZTest`: Test comparing 2 proportions
7. `ZInterval`: Confidence interval for 1 $\mu$, known $\sigma$
8. `TInterval`: Confidence interval for 1 $\mu$, unknown $\sigma$
9. `2-SampZInt`: Confidence interval for difference of 2 $\mu$'s, known $\sigma$'s
10. `2-SampTInt`: Confidence interval for difference of 2 $\mu$'s, unknown $\sigma$'s
11. `1-PropZInt`: Confidence interval for 1 proportion
12. `2-PropZInt`: Confidence interval for difference of 2 proportions

Your "pair programming goal" is to **implement and understand** each of the above TI-84 functions by **searching for relevant problems** (e.g., a problem matching the calculator function) in the midterm 2 study guide[4] and **programming on the TI-84 with the relevant function** to obtain the problem's "correct answer". These "correct answers" are in the margin of the midterm 2 study guide.

I have attached the TI-84 manual pages to this guide for "the navigator/observer". The "driver" should be on the TI-84, and the "observer" should be in the "man[5] pages" looking at the section outlining the correct implementation of the relevant calculator function. You both should (collaboratively) pick problems from the midterm study guide, then make a plan, **before starting into the calculator programming**.

There are video instructions (scan QR code, see playlist videos 8–15) for eight of these functions (`Z-Test`, `T-Test`, `ZInterval`, and `TInterval` don't have videos). *You should watch these videos on your own, outside of class.*

[3]If you don't have a TI-84, you'll need to find the analogous function on your calculator of choice.

[4]Everyone should have received their own midterm 2 study guide. Ask Colton if you need an extra copy.

[5]Short for manual.