How to Ask for Help

Adam Blank

The First Time You Need Help

In the interest of starting out honestly, there will probably be a first time you need help. For many of you, it won't have happened yet. When it does, it will be incredibly frustrating, demotivating, and demoralizing. To make matters worse, you probably won't see it coming, and you'll probably delay dealing with it for longer than you should.

When this happens to you, *take a deep breath*. The first thing you need to do is *calm down*.

Needing help doesn't mean you aren't smart. It doesn't mean you're a failure or you aren't as capable as your friends. It doesn't necessarily mean your instructor or TAs suck. It probably doesn't mean "the homework is too difficult".

All that it means is that it's time to seek out the help of someone who understands *the problem at hand* better than you do. This person can be a friend, an upperclassman, a TA, or an instructor. The most important thing is that you seek out help rather than struggling for too long and getting stressed and freaked out until it's too late to ask for help.

The first time you need help will likely be in your *very first semester* of college. Barring that option, it will pretty much certainly be in your second one. This is true regardless of whether you were your high school valedictorian or not. It's true despite your straight A's in math, and your "1337 h4x0r skillz".

When it happens, don't freak out. *Take a deep breath*, and figure out how to ask for help.

Getting the Most Out of Resources

Many of the resources that you are given are intended to help you deal with problems as they arise. What follows is some ways you can best take advantage of those resources.

- *Be active in course meetings.* If you have a question, ask it. If you're afraid of asking the instructor, you can always ask whomever is sitting next to you. Don't be scared to answer the instructor's questions–he/she is asking them for a reason! (One caveat: Don't try to answer *every* question the instructor asks. If you've already answered more than a couple questions in a particular lecture, give other people a chance.)

  If you were up late working, and you won't be able to pay attention (or worse think you will fall asleep), seriously consider if going to lecture is a good idea. (This is assuming that the course does not require you to miss only a couple course meetings.) You can always (and if you use this strategy, *ABSOLUTELY SHOULD*) go to office hours later if you don't understand something. Not all instructors will be okay with this–some would rather you show up to lecture regardless. I, personally, would rather students come to office hours later if they are just going to fall asleep–it's better for your learning, and it's less disrespectful. One other thing I will stress is that if you miss lecture and want to know what happened, a good strategy is to apologize for missing lecture, since you're asking the instructor to repeat the lecture for you personally.

- *Review your notes / the course notes before asking for help.* It is really frustrating for someone on course staff (and inefficient for you!) for you to ask questions about things that are explicitly stated in notes. Importantly, if you look and honestly can't find it, or you know where it is, but don't understand the explanation, we'd *love* to help you. This tip is just that you should make an effort first.

- *Use Google/Wikipedia (except where explicitly disallowed).* When it comes to weird compiler errors, definitions, general concepts, etc. Google and Wikipedia are fine (and *encouraged*!) resources. They will often help you get your answer faster than a TA or instructor could.

  Some courses will disallow these resources for specific problems/topics/etc. These policies often stem from the fact that there are a limited number of good problems, and many of them are somewhere on the internet. *Please* respect this request. If you're not sure if looking something up violates the cheating policy or not, it can't hurt to shoot an e-mail to a TA or the instructor before looking it up.

- *Start the homework before office hours; so, you can come with questions.* It's a bad situation when students don't start the homework before office hours, and end up with questions too late. An easy way to avoid this is to plot out when office hours are and make sure (regardless of if you think you'll need them or not) that you've started the homework early enough to still go.

- *If office hours are very full, it's probably within a day of the assignment being due. If you go to office hours earlier, they won't be as crowded.* Along the same line as the previous point, people get desperate right before the homework is due. You probably won't be able to get great advice or help the night it's due. Sometimes, office hours will be packed the night before as well. Make sure to plan ahead.

Ultimately, instructors and TAs really want to help you. But remember that they are people too. They have their own quirks, just like you do. It's highly unlikely that an instructor or TA doesn't want to see you succeed. They might have a weird personality. They might make weird or sarcastic jokes. We try as hard as possible to not play favorites or "dislike" students. If you think that an instructor or TA doesn't like you, there's an extremely large chance that you're taking something personally when you shouldn't. Everyone gets grouchy or has a bad day once in a while.

What *not* to do

There are many common behaviors/questions/approaches to getting help that at best frustrate the person helping you and at worst cause you to not get the help you need. I've outlined several common gotchas here to help you avoid them.

- **For Your Learning**
  - *Don't be afraid to go to office hours or wait to ask for help until the day the assignment is due.* Nobody will think you're stupid for going to office hours. TAs and instructors wouldn't be teaching if they didn't want to help you!
  - *Try out different people on course staff, not just one.* Often times, different TAs and instructors will have different styles. Different styles often work best for different students. TAs/instructors will also have pet favorite topics which they are either better at or will go into more depth with. If you get to know everyone on course staff, then you have a better chance of knowing where to look for help!
  - *Don't ask for "the answer".* The "answer" isn't going to help you learn. This is an (unfortunately) common strategy among students. It doesn't work, and everyone involved gets frustrated. If you need help, figure out what exactly it is you don't understand, and ask about that. One of the later sections of this guide explains exactly how to do that.

  - *Don't ask for "a hint".* This question is almost as bad as asking for the answer. What kind of hint? Where are you stuck? Have you actually tried to solve the problem, or did you just read it and go to office hours? Are you asking where to look for help? Or are you just asking for the answer in a different way?
  - *Don't ask for test cases.* If test/edge cases aren't provided, it's almost certainly because you are expected to generate your own. Telling a TA that "you don't know what is wrong" isn't going to help them help you. Neither is asking for more test cases. Ask yourself where your code/proof might be failing. Then, go ahead and ask your TA if the edge cases you came up with make sense if you want to.
- **For Courtesy**
  - *Be respectful.* Your TA or instructor is trying to help you. Try to say please and thank you. Don't get angry at them if you aren't getting the problem–they're trying. Showing a little respect goes a long way toward getting them to sympathize with you.
  - *Don't be entitled.* If you are asking for help, an extension, a re-grade, or anything else that involves your TA or instructor doing something especially for you, please don't assume you will get the extension or get points back. We will follow the course policy (generally to the letter) in providing these sorts of leniences. This generally means that unless a significant portion of the class has a similar conflict, or you have a medical or major personal problem (like you have the flu), we won't give you the leniency. This doesn't mean you shouldn't ask! It's almost always worth asking. Just do it respectfully, and don't be upset if we have to follow the course policy to be consistent among students.
  - *Listen to what your instructor/TA is saying. Don't cut them off. Don't try to skip people in line.* This goes hand in hand with being respectful. If you ask a question, try to listen to and process the complete response before responding. Also, if there's a line for getting help, trying to cut it is *not* a good idea.
  - *Don't ask the same question multiple times hoping for a different answer.* There is a reason we gave the vague or otherwise unsatisfactory answer we did. It's usually related to us wanting you to think more before helping you further. There is one *major* exception to this rule: "could you explain this topic/idea/proof from lecture?" paired with "I still don't understand". Please please *please* ask these questions until you understand. There's often more than one way of explaining a topic, and we will try different ones until you get it.
  - *Don't run to ask for help immediately after getting a mysterious compiler message.* Remember to try to use the tools at your disposal. In this particular circumstance, if the programming language is commonly used, just copy/paste the error message into Google, and you will get your answer very quickly.
- **For Getting a Good Answer**
  - *Avoid pointing to your proof/code as much as possible when you're asking for help.* We try to avoid looking at your proof or code, because it leads to poorly targeted questions like "why doesn't my code pass the tests?" or "it won't compile, see!" Explain the higher level ideas ("this is the algorithm I wrote, and I think it's failing at *this* part", "I'm not sure my proof works, because in *this* case, I don't think it makes sense") instead. Occasionally, you really will need to point to code ("It doesn't like the syntax I used here, and I'm confused", "I got this equation, but it conflicts with this other thing I saw"). Figuring out the line between these two is often difficult, and you shouldn't be afraid of asking us to look at your code/proof–you just shouldn't be upset if we refuse and want to help you in a different way.
  - *Don't even ask them if "your proof/code is good (enough)".* We will not grade you at office

hours. If you have a specific piece you aren't sure about, that's totally fine. But asking us to find errors for you is totally unacceptable, and we will *never* do it.

- *Don't expect them to know exactly what your problem is.* You need to describe the situation you're in to us. We won't magically know what's wrong with your code/argument. Tell us your approach and where you got stuck.
- *Don't just say "my code isn't working" or "I don't know how to solve this problem".* These questions indicate that you haven't tried to debug your code/proof and want us to do it for you. We will not do this. You need to provide us with what you've tried so far, a couple ideas as to why it isn't working, and maybe a few ideas of how to proceed that you're not quite sure how to do.

Different Kinds of Help

Depending on the outcomes of whatever course you're taking, you might have problems gaining different types of knowledge. This section covers three common ones (conceptual, procedural, and descriptive) and how to approach them.

- **Descriptive Knowledge ("what is it?").** This is the type of question where you can usually look up the answer in notes, a textbook, or the internet. If you can't find it after looking, then ask a friend or someone on course staff.
- **Procedural Knowledge ("how do I do it?").** These questions are usually answered by *worked examples* in lecture, notes, or a text book. It is often easy to create similar, but different, examples for these questions, which means it's totally legitimate to go to office hours and ask someone to work through another example with you. It can often be illuminating to see more examples when you can ask questions and it goes at your own pace.
- **Conceptual Knowledge ("what does it mean?", "where does it come from?").** These are the hardest types of questions to answer. Usually, this type of knowledge comes after the related descriptive and procedural knowledge are both mastered. Once you're sure you understand the other two, asking someone to walk you through the lecture notes again or derive something for you can help you master this.

Some Good Self-Prompts

Here's some questions that we will often ask that you should be prepared to answer before asking for help:

- What have you tried so far?
- What other strategies could you use to solve the problem?
- Why do you think your solution is wrong/doesn't work?
- What is the high-level idea you're trying?
- What's an edge case that your solution could be failing on?
- If you don't know where to start, try doing small examples.
- If the question has to do with a particular topic, make sure you understand the notes on that topic. If you don't, ask about those first.
- If you're stuck, think about what gaps you would need to fill in to finish the problem.

- What have you tried to verify your answer with?
- Why have you chosen the strategy you did? What about the problem made you try it?
- Did you read through all the note/examples/solutions we've provided?
- Are you working in a group (if it's allowed)? Where are your group members?
- Why is your answer correct?

Make sure that you are able to answer most or all of these before you ask for help. It will speed up the process.

A Process

Here's a handy little procedure for what you should do when you get stuck on a homework question.

(1) **Isolating the Problem**

Figure out what *exactly* your problem is. Do you understand everything you've done? If your code isn't compiling, what are a couple reasons that could be the case? If your proof strategy isn't working, why not? Is there another strategy you've learned that might work better?
Are you missing underlying knowledge that you need? Did you understand lecture? recitation? Have you tried some of the other questions to see if your problem is the topic or just this question?

(2) **Trying for Yourself One Last Time**

Now that you know what the problem is, try out your thoughts for yourself. If you thought of a couple reasons your code might not be compiling, give an honest effort to trying them out. If you thought of another proof strategy to use, try it for a little bit and see if you get anywhere.

(3) **Who to Ask?**

So, you've tried out all your ideas and they didn't work. Who should you come to with the question? If you're working in a group, you might try explaining the problem to your groupmate. Otherwise, think about what type of knowledge you're missing. Is it descriptive? Maybe try Google, an e-mail, or Piazza/AnnotateMyPDF.
Is it procedural? Think about which person on course staff is best at explaining how to do procedures *for you*. This sort of question tends to take a while; so, I recommend going to office hours that are usually fairly unpopulated.
Is it conceptual? Here, you might try splitting your question up into different things you don't understand, and going to office hours for each one separately. Again, try to choose a person who has a history of being good at helping you.

(4) **Gathering Your Thoughts**

Now that you know who's office hours you're going to, you should gather your thoughts about your problem and questions. Bring with you a list of things you've tried and notes to yourself about why they didn't work. Think about what you want to ask in advance. Think about how you want to explain your approach. Figure out the essential parts of your question and the non-essential ones, in case there's not enough time to ask about everything.

(5) **Asking Questions**

When you get to office hours, use your notes about what you've tried to prompt your questions. As you're asking, draw diagrams, write down code or phrases you think will help, and explain what you've tried and why you're stuck.

(6) **Mulling Over the Responses**

Generally, one of two things happens upon getting help: (1) you solve your problem immediately, and (2) you need to think about the response for a while. If the first one happens, great! You're done! If the second one happens, avoid the urge to just keep on asking more questions or respond immediately saying you still don't get it.

(7) Regrouping and Trying Again

Sit down, and actually think about what the TA or instructor said. Try to integrate it with your view of the problem that you have written down. Think about how it might help you solve the problem. If after trying for a while, you still don't get it, do an abbreviated version of this procedure again to figure out your next questions.

Keep in mind that "a while" is always at least five minutes and usually at least ten to fifteen minutes.