Strategies for Success in CS 106A

Brahm Capoor

Why are we here?

106A is a *hard* class, mostly because Computer Science requires a very unique style of thinking which takes some adjusting to (it took me until 106B!)

Having been around the block several times for 106A, I have some suggestions about how you can get the most from this class, and how you can minimize how much of a time sink and struggle it is.

What does success mean in an S/NC quarter of 106A?

- Pass the class (obviously, but also not a huge deal)
- Comfort with and an intuitive understanding of how course material fits together
- Muscle memory for the various ways you solve problems using programming principles
- A sense of how you might apply these tools to problems you're interested in

What are we talking about today?

Passing the Class

How to fit everything together

How to practice concepts

How to do the assignments

Your questions & an open discussion

Passing the class

Passing the class

Some math:

- The diagnostic is worth 15% of your grade, and assignments are worth 75% of your grade, with the later assignments being weighted more heavily
- This means that each assignment moving forward basically has equal weight to the diagnostic

Passing the class means that you’ve demonstrated a understanding of this material, and you have plenty of opportunities left to do that.

Fitting everything together

Fitting everything together

In CS 106A, we throw a lot at you, very quickly

Expecting you to deeply internalize everything as you learn it would be unfair, but we do want you to have at least a working knowledge of the material

Your target mindset is:

> "I don't necessarily know off the top of my head how I would do this, but I know I can figure it out *"

** some elbow grease required*

How do you get there?

Your goal is to understand how different parts of the course fit together, because understanding that context is the best way to understand why their internals work the way they do.

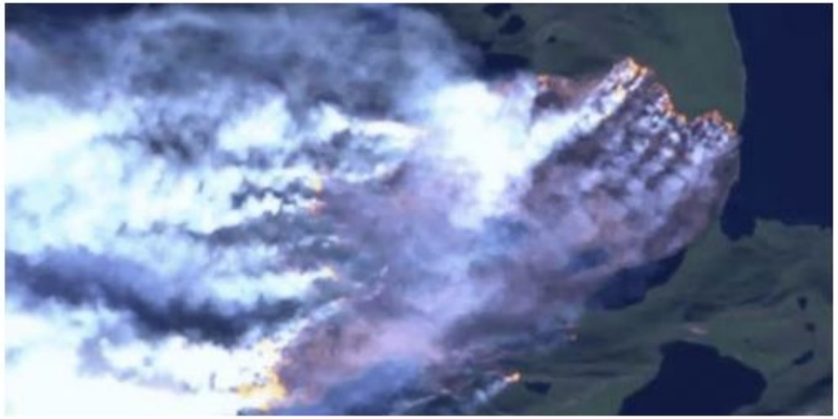
Let’s go through an example:

How to understand images

Rather than worry about all the syntax and code snippets associated with images, I think these two takeaways from lecture are sufficient:

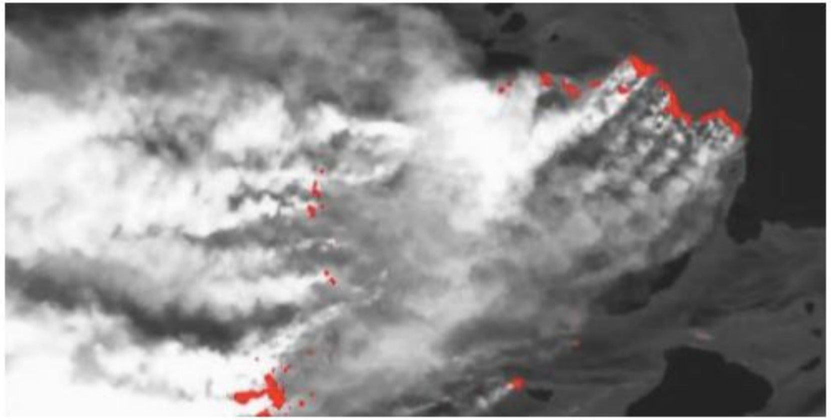
- Pixels have x and y coordinates
- Pixels have .red, .green and .blue variables

Let’s talk about how these takeaways are enough for an image problem



I need to iterate over all of the pixel x, y coordinates

- There are *ranges* of x and y coordinates, so I can use a for loop.



I need to adjust the color of pixels

- I can adjust the .red, .green and .blue variables

Use section materials as an opportunity to fill in the gaps

Section problems are a great way of applying your understanding of material to a new, safe, problem space

They're a fantastic way to understand how different parts of course material fit together.

A few other resources

- The Course Python Reader
- Asking conceptual questions on Ed
- Find a study group here!
  - Study groups are a *fantastic* way of reinforcing your understanding of material

Practicing

Where can you get practice?

- Section problems
- Worked examples on the course page (‘examples’ tab on the course page)
- Try reimplementing lecture code
- Write a program that interests you!
- Look over old assignments, and see if there’s a way you can improve them using techniques later on
    - Fun challenge: how could do Ghost for >3 images using lists?

Working on assignments

What’s your goal in assignments?

- To take materials we’ve covered in lecture, and apply them to a new problem with less hand-holding
- *This is the central task of Computer Science*
- Your ulterior motive is to convince us that you you’ve understood how the tools you’re learning can be applied to new problems.

High level advice

- Top down decomposition: plan a strategy before beginning to write code. What functions do you need? How will they fit together?
- Figure out what tools you need: What sort of loop structures are involved? What's the best way of representing the information in your program?
- Incremental Testing: Make sure a piece of code works before moving on to the next one
    - Use doctests liberally
    - Write minimal working examples
    - Think about edge cases

Working with others

The most common concern that students have about working with others is violating the Honor Code. Here's where we stand about this:

- Not only is discussing assignments and the class with others allowed, it's *encouraged*- collaboration is part and parcel of the experience of programming
- Discussing strategy and bouncing ideas off other students is a fantastic idea
- We want you to be responsible for the work you submit, so the actual code you submit should be your own work is (that is, the job of actually translating a strategy into code rests on your shoulders)

Debugging

- A bug is *not* a sign of failure, but a sign of *progress*
- The problem is not that the computer is not doing what you want it to do, but that it's doing *exactly* what you tell it to do, and you're not telling it the right thing.
- Here's a few techniques I use when I'm debugging:
    - Look at program output to gain a sense for *where* in your code the issue might be
    - Learn to love error messages (here's a fantastic guide one of our section leaders put together)
    - Call `print` to make sure my variables have the values I expect them to
    - In programs with a ton of output, calling `input` to make my program stop whilst I inspect the output

Getting Help

Needing help is *not* an admission of weakness. What takes me a minute to answer might save you hours of pain.

We provide so many different ways for you to get help, because we want you to take advantage of them.

I have some thoughts on the best ways to make use of these resources

Doing the legwork

The more legwork you’ve done before you get help, the more effective our help can be:

- If you have a debugging question, what have you already tried to fix it? What hypotheses do you have about what might be going wrong? Do you have any other approaches in mind?
- If you have a conceptual question, is there a specific program which demonstrates your confusion? What about the concept doesn’t fit in with your understanding?

Not having answers to these questions is totally fine, but thinking about them helps us, and more importantly, you.

Where to get help

The LaIR is a fantastic resource, but is time constrained. Use it to unblock yourself when you have a *specific* confusion or bug

Group Office Hours are a great place to discuss approach and to gain a better understanding of the different pieces of an idea or an assignment. You can also use Ed for this.

Individual Office Hours are an extended opportunity for you to go into depth on any of your questions

A question is always worth it.

Your questions