How to Succeed in 3rd and 4th Year Computer Science Classes

Dr. Beth Simon

Science Learning and Teaching Fellow

Computer Science Department

esimon@cs.ubc.ca

Why listen to me?

- PhD in CS
  - Computer Architecture: branch predictors
  - San Diego Supercomputing Center: USA DoD
- Professor at UC San Diego
  - Computing Education Research
    - UBC Carl Wieman Science Education Initiative
- My job (and my passion) is to figure out how students struggle in learning computing and how to make it work better.

Oh yeah? What's your street cred?

- RateMyProfessor: (1-5, 5 is best)
  - Easiness: 3.8
  - Helpfulness: 4.8
  - Clarity: 4.8
  - Overall Quality: 4.8
- Professor Simon is an awesome professor. She is very enthusiastic and genuinely cares about her students' learning. She explains the material in a clear and concise manner and is always well prepared for class. Overall this prof is awesome and I hope to get her again.
- UCSD Instructor Evals: 97.93% of students recommend me as an instructor. (N=14)

What are you going to learn today?

By the end of class today you will be able to…

- Describe some of the expectations that faculty have of 3rd and 4th year CS students
- List resources and techniques to use in learning a new language.
- Explain at least 3 strategies for debugging a large code base.
- Explain how the theory of constructivism means that lectures are most valuable not as "content delivery" but as opportunities to "try out" your learning

How many of you have ever raised your hand and asked a question in a CS class?

How many of you have ever raised your hand and asked a question in a CS class?

- This is (almost) universally valued by (UBC CS) faculty.
  - They are NOT annoyed, upset, or intending to blow you off.
  - They DO want to know if you are confused.
- Instructors and Students do NOT experience the content of lecture the same way[1]
  - Instructors "see" more answers in a lecture than students do.

[1]Hrepic, et.al. Comparing students' and experts' understanding of the content of lecture. Journal of Science Education and Technology, 2007. V16 No 3.

You tell me…

- Why do you not raise your hand and ask a question in class?

Repeat after me:

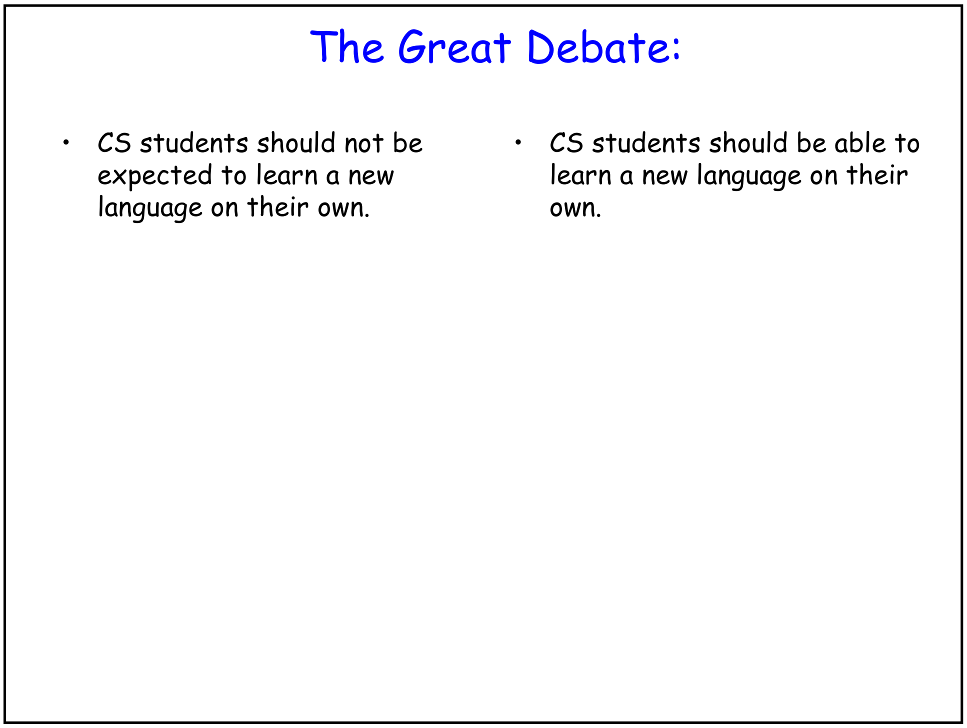
- “I have an question about that matrix on the right – why are we using that again?”

Repeat after me:

- “Wait, I didn’t quite get that – could you explain that again?”

By the end of your 3rd year in Computer Science you should be able to

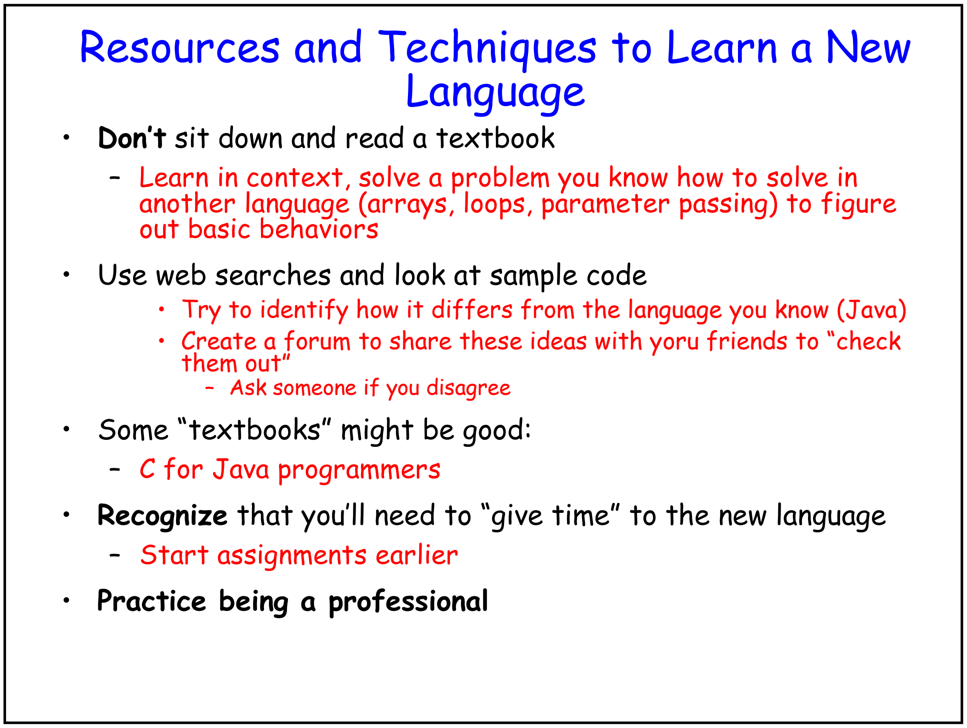
- Learn a new programming language
  - Without taking a class on it
- Engage advanced debugging skills for large-ish code bases
  - And be able to describe and defend
- Utilize with proficiency software development support systems
  - Version control systems
    - e.g. svn
  - Project management systems
    - e.g. Eclipse



The Great Debate:

- CS students should not be expected to learn a new language on their own.

- CS students should be able to learn a new language on their own.



Resources and Techniques to Learn a New Language

- **Don't** sit down and read a textbook
  - Learn in context, solve a problem you know how to solve in another language (arrays, loops, parameter passing) to figure out basic behaviors
- Use web searches and look at sample code
    - Try to identify how it differs from the language you know (Java)
    - Create a forum to share these ideas with yoru friends to "check them out"
      - Ask someone if you disagree
- Some "textbooks" might be good:
  - C for Java programmers
- **Recognize** that you'll need to "give time" to the new language
  - Start assignments earlier
- **Practice being a professional**

Things I want to know when I learn a new language

- How do I print? (key for basic playing around and debugging)
- How are variables “stored”?
  - Like primitive types or reference types?
- How are parameters passed?
  - By Value? (Java but with objects the value is a reference C++)
  - By Reference? (C++ if you use *)
- What’s the scope of a variable name (same)
- What are my available looping structures (same/mostly)
- How do data types differ (boolean, esp).
- What do all the OO-isms “look like”
- What built in libraries can I use to make my life easier?

Advice from someone who really knows:

- Khawaja Shams (ex-student works at NASA Jet Propulstion Lab in Pasadena, CA, USA)
- From chat:
  - i feel the best way to learn a language is to use it
  - recently, i had to learn actionscript for a project at work...and i needed to know more than just the basics
  - what worked for me was actually opening an IDE and starting to type and look for solutions online as i encountered problems
  - IDEs are a great tool for learning a new language btw, i think many underestimate how valuable of a teacher an IDE can be… things like auto complete...suggestions, warnings, etc...are great

Debugging Done Right

- Debugging as the Scientific Method
- Large Systems

The Scientific Method Applied to Debugging

- 1) Observe/describe the phenomenon
- 2) Form an educated guess (hypothesis) (may involve doing background research) about the cause of the phenomenon and make predictions based on hypothesis
- 3) Test your hypothesis with an experiment(s)
- 4) Check and interpret the results
- 5) Report results to the community

```
int foo(int y, int x)
{
  y++;
  x--;
}

int main()
{
  int x, y = 0;
  foo(x, y);
  cout << “x = “ << x;
  cout << “y = “ << y;
}
```

- 1) Phenom: x and y don’t change. DOH.
- 2) Educated guess
  - 1: Function not executed?
  - 2: Parameter values not “coming back” to main program
- 3) Experiment?
- 4) How check/interpret?

The step that will help you most

- Report results to the community.
  - Keep a personal bug log with the bug, description of error or how it exhibited itself, thoughts, understanding and fix.
  - Better yet, make a wiki with your friends and QUESTION each other about the bugs they report.
    - http://www.wikidot.com/
    - UBC wiki: brian.lamb@ubc.ca

Debugging Large Systems

- Locating the bug
  - Advanced Debuggers
  - Binary Search
- Crash or incorrect value/output
  - Working backwards/Working forwards
  - Explain it in plain English
    - Tell it to your cat, your friend (imaginary or not)
  - Separate known facts from “assumptions”

Utilizing support systems

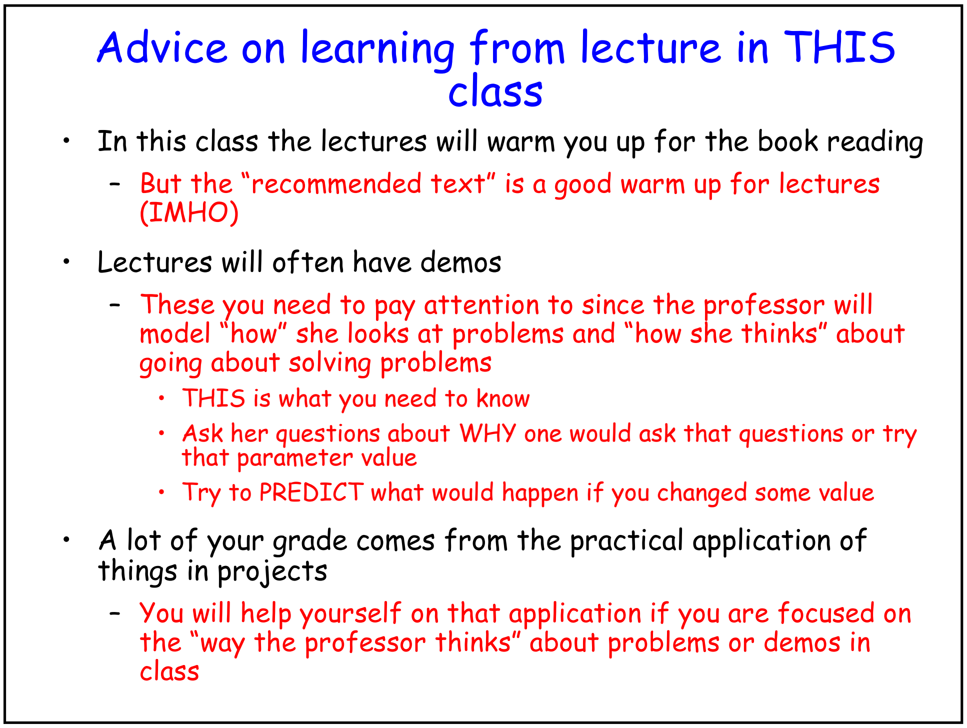
- SVN (or CVS) is your FRIEND
  - By FAR the most highly stated CRITICAL tool by students in their first “large programming” course.
  - Take snapshots of your code. About to:
    - Write a new method/function
    - Add a new class
    - Debug something
      - The most common problem of novice programmers is inserting new bugs when they are trying to fix other ones.
  - ROLL BACK
    - After that long night where you should have gone to bed…

Utilizing Support Systems

- Project Management Systems – Eclipse
  - Software development is not just about code
  - Planning/Design
    - Solve the RIGHT thing
    - What pieces should you build first
    - How will those pieces fit together to solve your problem?
  - Communication and Documentation
    - Among team members
    - So your code can be understood by others (or by you after 2 days)

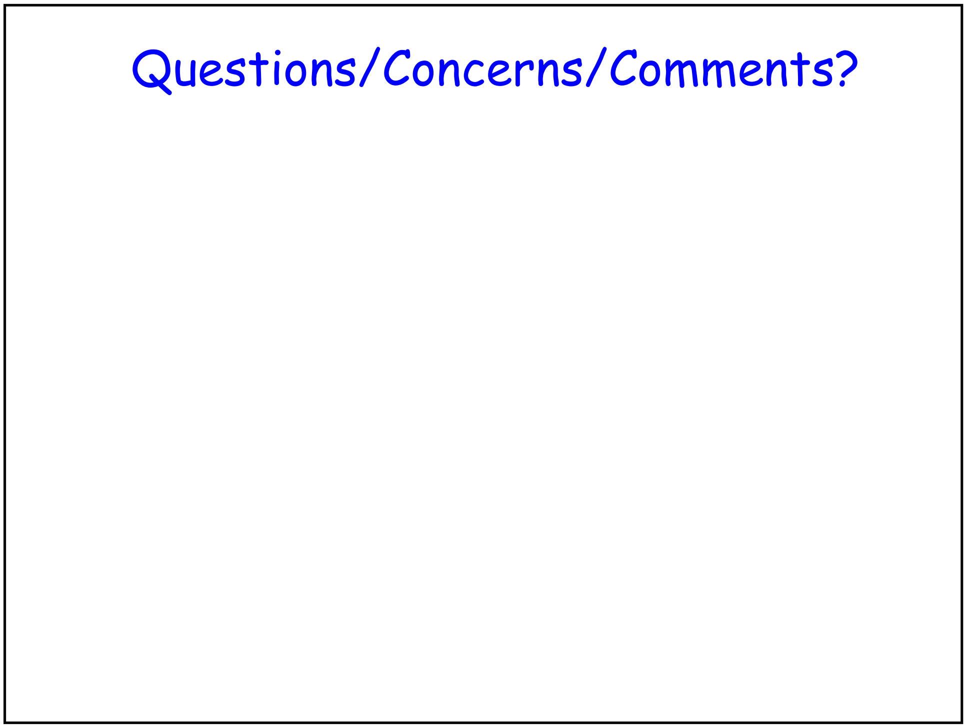
General advice (from the literature) on learning effectively in class

- At the beginning of class, know what you expect to be learning.
  - If you don't know what you are expected to learn…
    - How will you watch for it?
    - How will you know if you didn't "get it"?
- Don't rely on your notes for "fact"
  - People can't take good notes on material they are not yet familiar with
  - If you must, cross-compare with others' notes
- Always be "engaged"
  - You only learn by "constructing" new knowledge on top of things you already know
    - Think about how this new thing relates to something you know
    - Think about the scenarios you would use it or not use it
    - Ask yourself "why do I want to know this"?



Advice on learning from lecture in THIS class

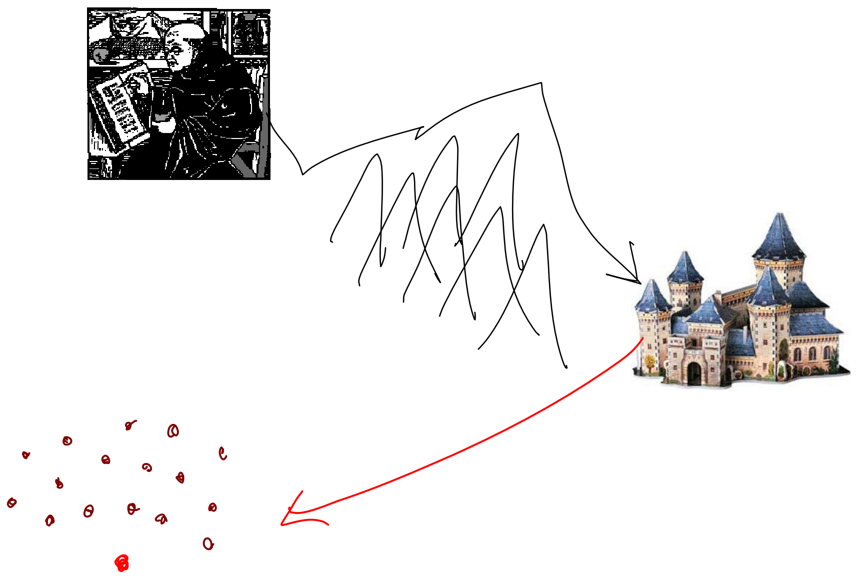
- In this class the lectures will warm you up for the book reading
  - But the “recommended text” is a good warm up for lectures (IMHO)
- Lectures will often have demos
  - These you need to pay attention to since the professor will model “how” she looks at problems and “how she thinks” about going about solving problems
    - THIS is what you need to know
    - Ask her questions about WHY one would ask that questions or try that parameter value
    - Try to PREDICT what would happen if you changed some value
- A lot of your grade comes from the practical application of things in projects
  - You will help yourself on that application if you are focused on the “way the professor thinks” about problems or demos in class

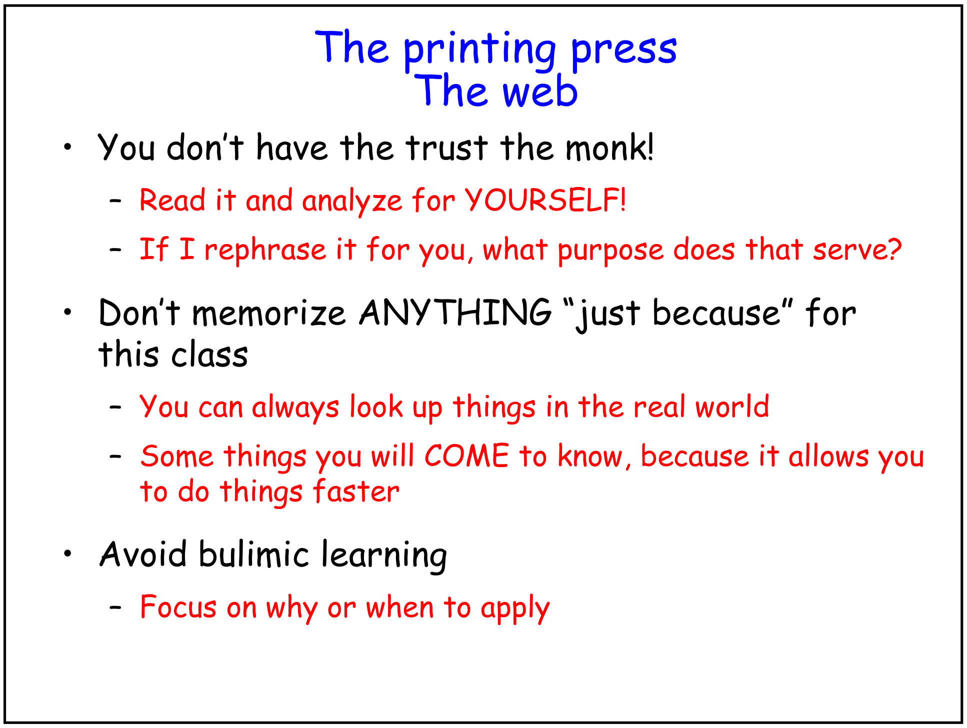


Questions/Concerns/Comments?

A word about lectures and medieval times

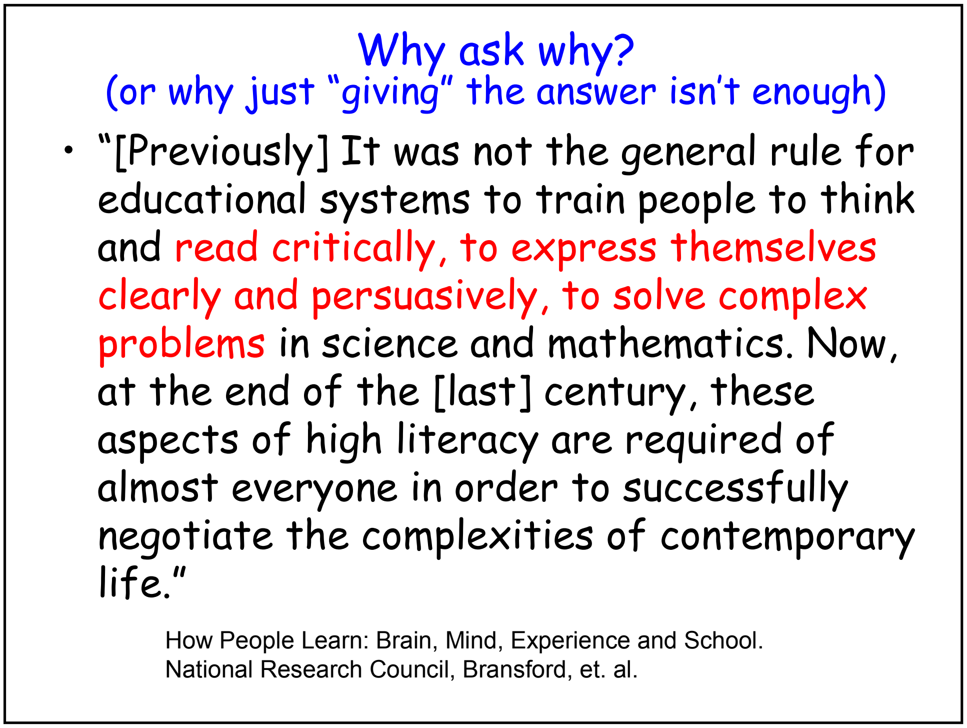
- Lecture: It’s a large part of what you pay for
  - Common: Highlight material in book, discuss complex issues, provide extra examples
- Why do we have the “lecture” format?
  - Why does someone stand at the front and tell you things?
  - Why do you take “notes” on what they say?





The printing press
The web

- You don't have the trust the monk!
  - Read it and analyze for YOURSELF!
  - If I rephrase it for you, what purpose does that serve?
- Don't memorize ANYTHING "just because" for this class
  - You can always look up things in the real world
  - Some things you will COME to know, because it allows you to do things faster
- Avoid bulimic learning
  - Focus on why or when to apply



Why ask why?
(or why just "giving" the answer isn't enough)

- "[Previously] It was not the general rule for educational systems to train people to think and read critically, to express themselves clearly and persuasively, to solve complex problems in science and mathematics. Now, at the end of the [last] century, these aspects of high literacy are required of almost everyone in order to successfully negotiate the complexities of contemporary life."

How People Learn: Brain, Mind, Experience and School.
National Research Council, Bransford, et. al.

Memorization: It's not what's for breakfast anymore

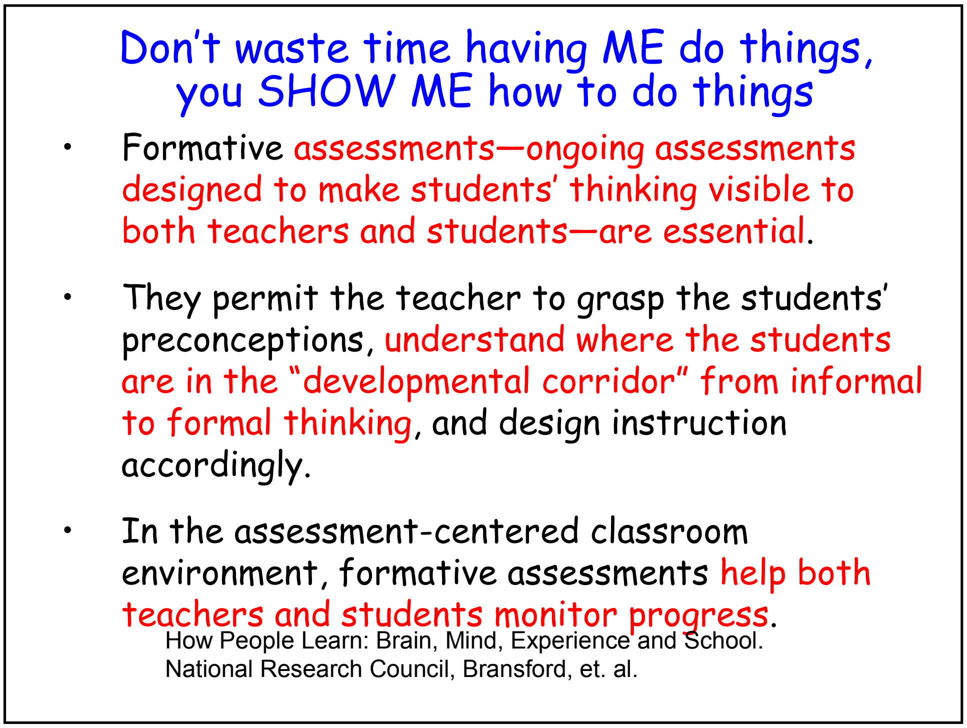
- As Nobel laureate Herbert Simon wisely stated, the meaning of "knowing" has shifted from being able to remember and repeat information to being able to find and use it (Simon, 1996).
  - More than ever, the sheer magnitude of human knowledge renders its coverage by education an impossibility; rather, the goal of education is better conceived as helping students develop the intellectual tools and learning strategies needed to acquire the knowledge that allows people to think productively about history, science and technology, social phenomena, mathematics, and the arts.
  - Fundamental understanding about subjects, including how to frame and ask meaningful questions about various subject areas, contributes to individuals' more basic understanding of principles of learning that can assist them in becoming self-sustaining, lifelong learners.

How People Learn: Brain, Mind, Experience and School.
National Research Council, Bransford, et. al.

Don't ask me to do it, tell me HOW to do it!

- Metacognition refers to people's abilities to predict their performances on various tasks (e.g., how well they will be able to remember various stimuli) and to monitor their current levels of mastery and understanding (e.g., Brown, 1975; Flavell, 1973).
- Teaching practices congruent with a metacognitive approach to learning include those that focus on sense-making, self-assessment, and reflection on what worked and what needs improving.
- These practices have been shown to increase the degree to which students transfer their learning to new settings and events (e.g., Palincsar and Brown, 1984; Scardamalia et al., 1984; Schoenfeld, 1983, 1985, 1991).

How People Learn: Brain, Mind, Experience and School.
National Research Council, Bransford, et. al.



Don't waste time having ME do things, you SHOW ME how to do things

- Formative assessments—ongoing assessments designed to make students' thinking visible to both teachers and students—are essential.
- They permit the teacher to grasp the students' preconceptions, understand where the students are in the "developmental corridor" from informal to formal thinking, and design instruction accordingly.
- In the assessment-centered classroom environment, formative assessments help both teachers and students monitor progress.

How People Learn: Brain, Mind, Experience and School.
National Research Council, Bransford, et. al.