Beyond the Mouse – A Short Course on Programming

1. Thinking programs

Jeff Freymueller and Ronni Grapenthin

Geophysical Institute, University of Alaska Fairbanks

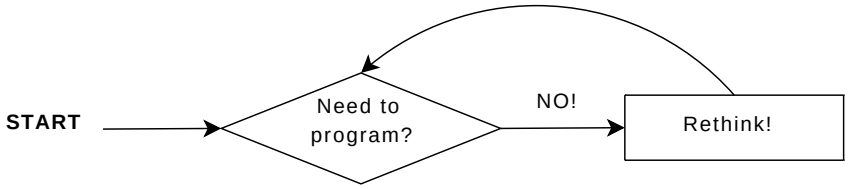
September 4, 2012

"The Uncomfortable Truths Well", http://xkcd.com/568 (April 13, 2009)

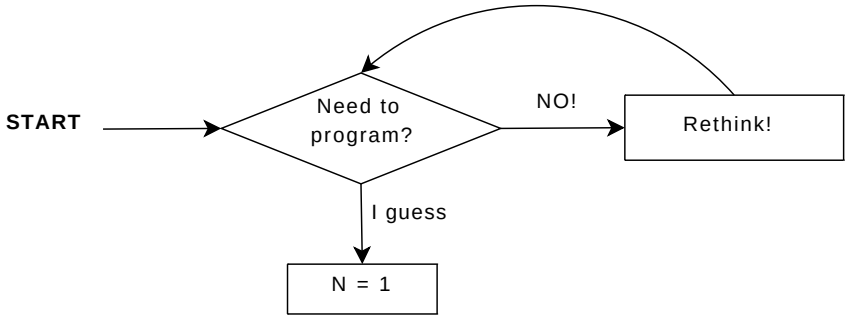
Outline

Outline

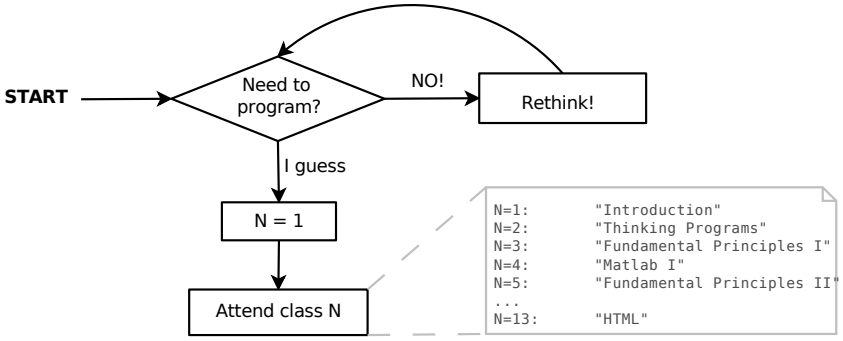
The Program . . .



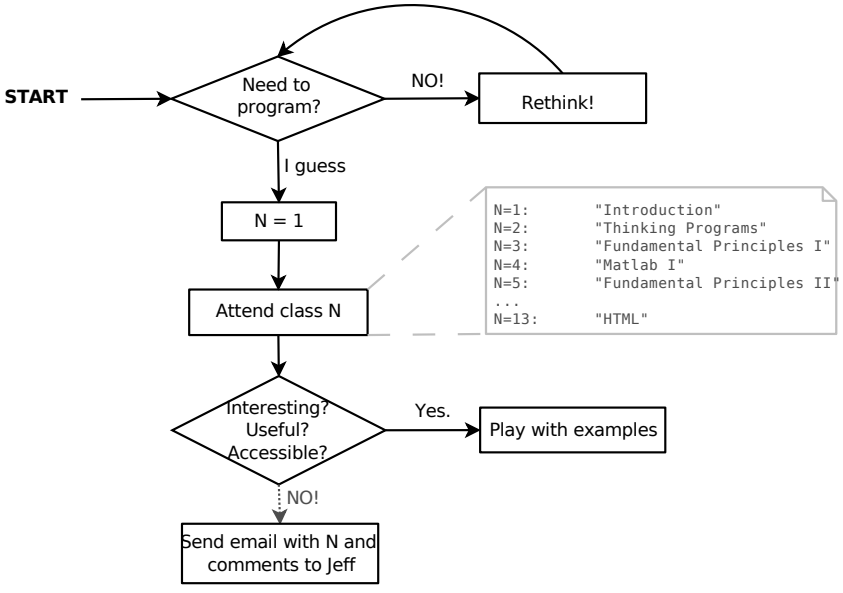
The Program . . .



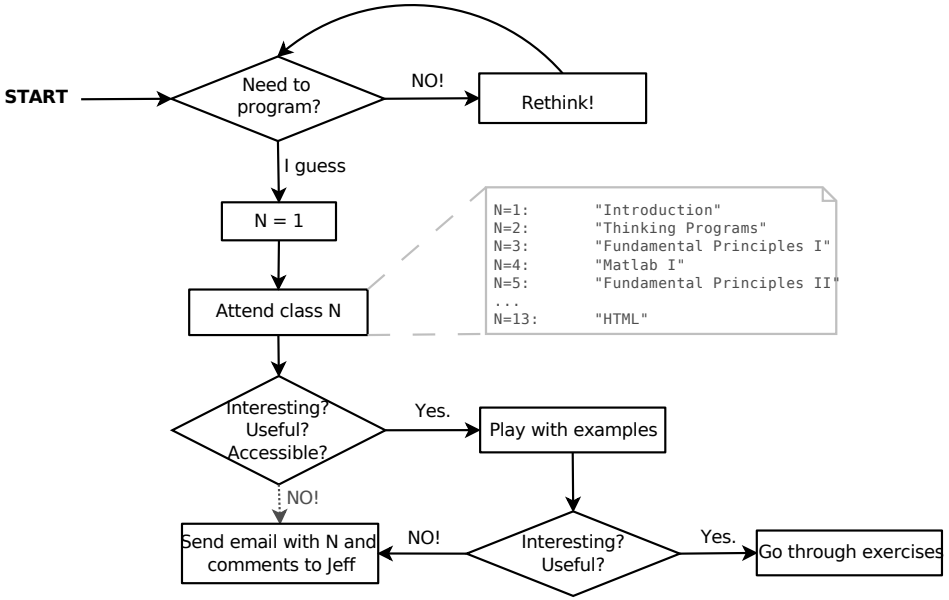
The Program . . .


START
Need to program?
NO!
Rethink!
I guess
N = 1
Attend class N
N=1: "Introduction"
N=2: "Thinking Programs"
N=3: "Fundamental Principles I"
N=4: "Matlab I"
N=5: "Fundamental Principles II"
...
N=13: "HTML"

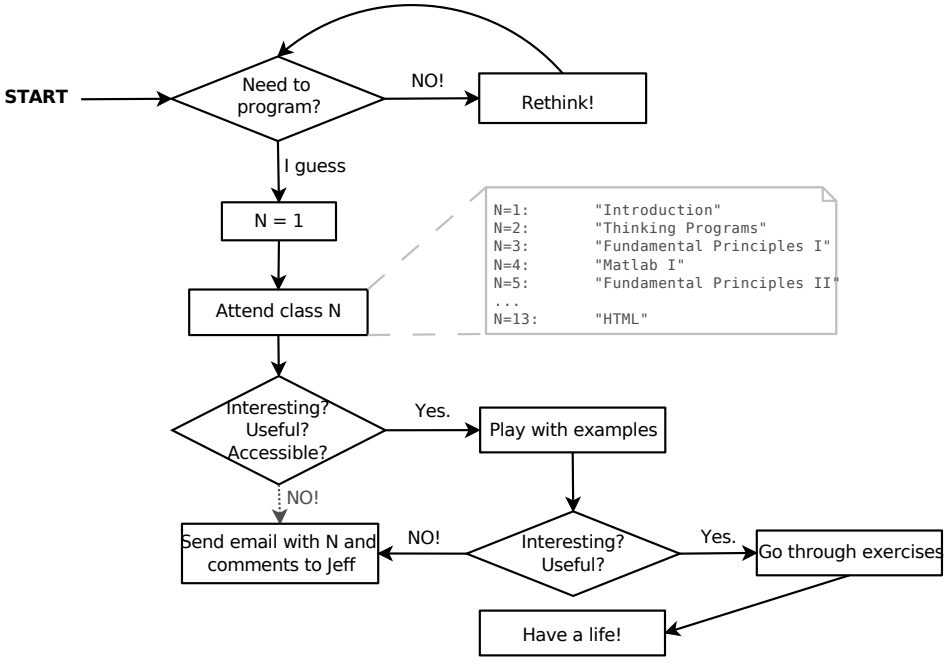
The Program . . .


START
Need to program?
NO!
Rethink!
I guess
N = 1
Attend class N
N=1: "Introduction"
N=2: "Thinking Programs"
N=3: "Fundamental Principles I"
N=4: "Matlab I"
N=5: "Fundamental Principles II"
...
N=13: "HTML"
Interesting? Useful? Accessible?
Yes.
Play with examples
NO!
Send email with N and comments to Jeff

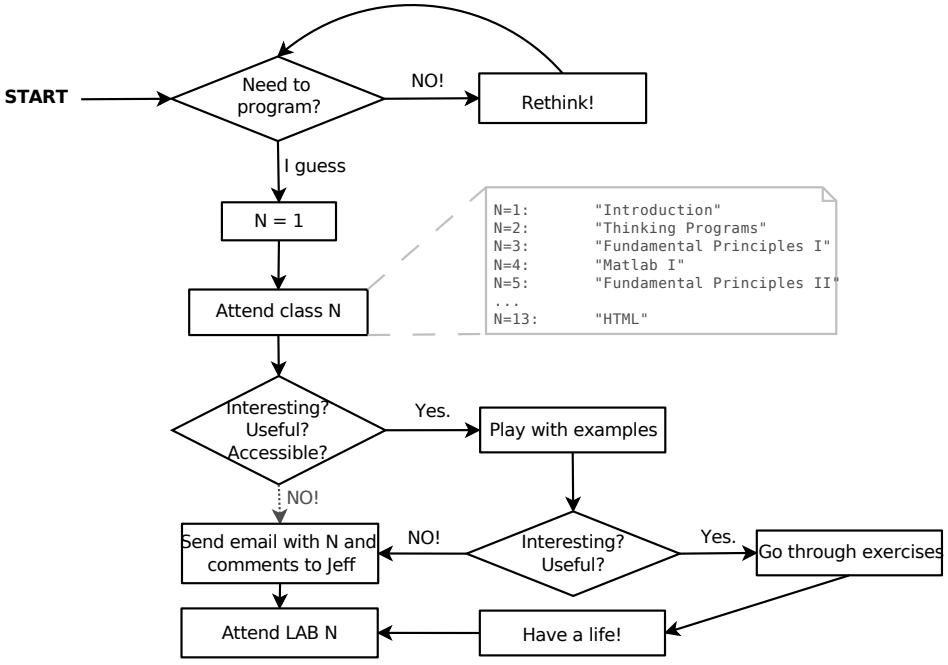
The Program . . .


START
Need to program?
NO!
Rethink!
I guess
N = 1
Attend class N
N=1: "Introduction"
N=2: "Thinking Programs"
N=3: "Fundamental Principles I"
N=4: "Matlab I"
N=5: "Fundamental Principles II"
...
N=13: "HTML"
Interesting? Useful? Accessible?
Yes.
Play with examples
NO!
Send email with N and comments to Jeff
NO!
Interesting? Useful?
Yes.
Go through exercises

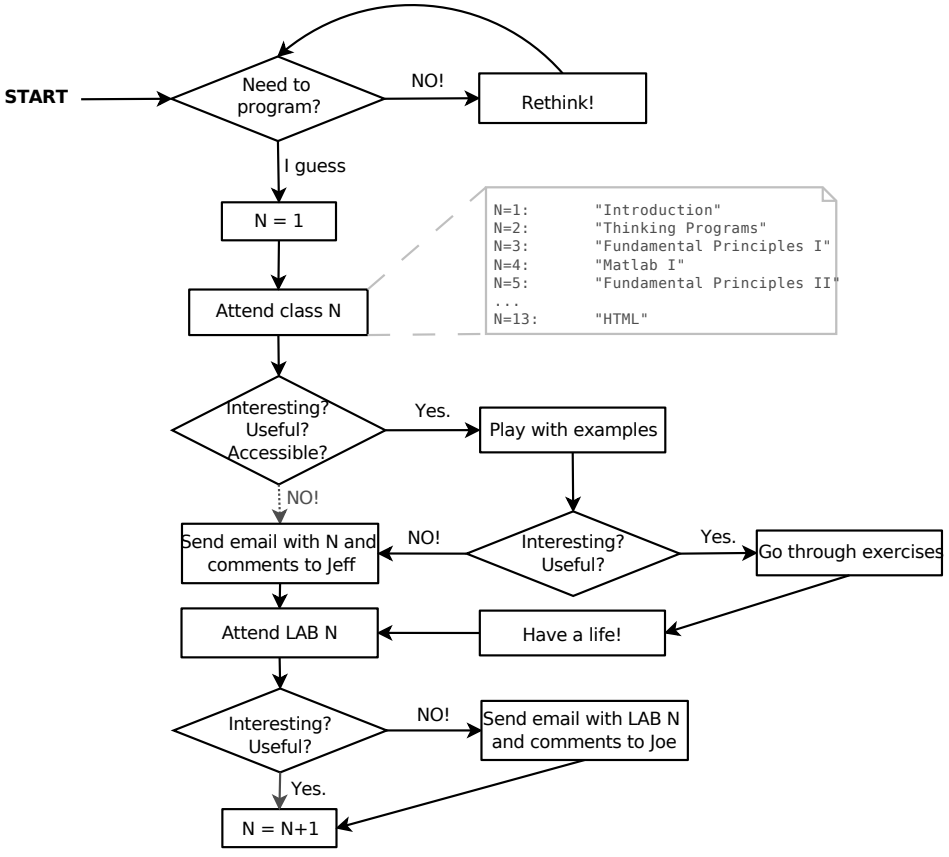
The Program . . .


START
Need to program?
NO!
Rethink!
I guess
N = 1
N=1: "Introduction"
N=2: "Thinking Programs"
N=3: "Fundamental Principles I"
N=4: "Matlab I"
N=5: "Fundamental Principles II"
...
N=13: "HTML"
Attend class N
Interesting? Useful? Accessible?
Yes.
Play with examples
NO!
Send email with N and comments to Jeff
NO!
Interesting? Useful?
Yes.
Go through exercises
Have a life!

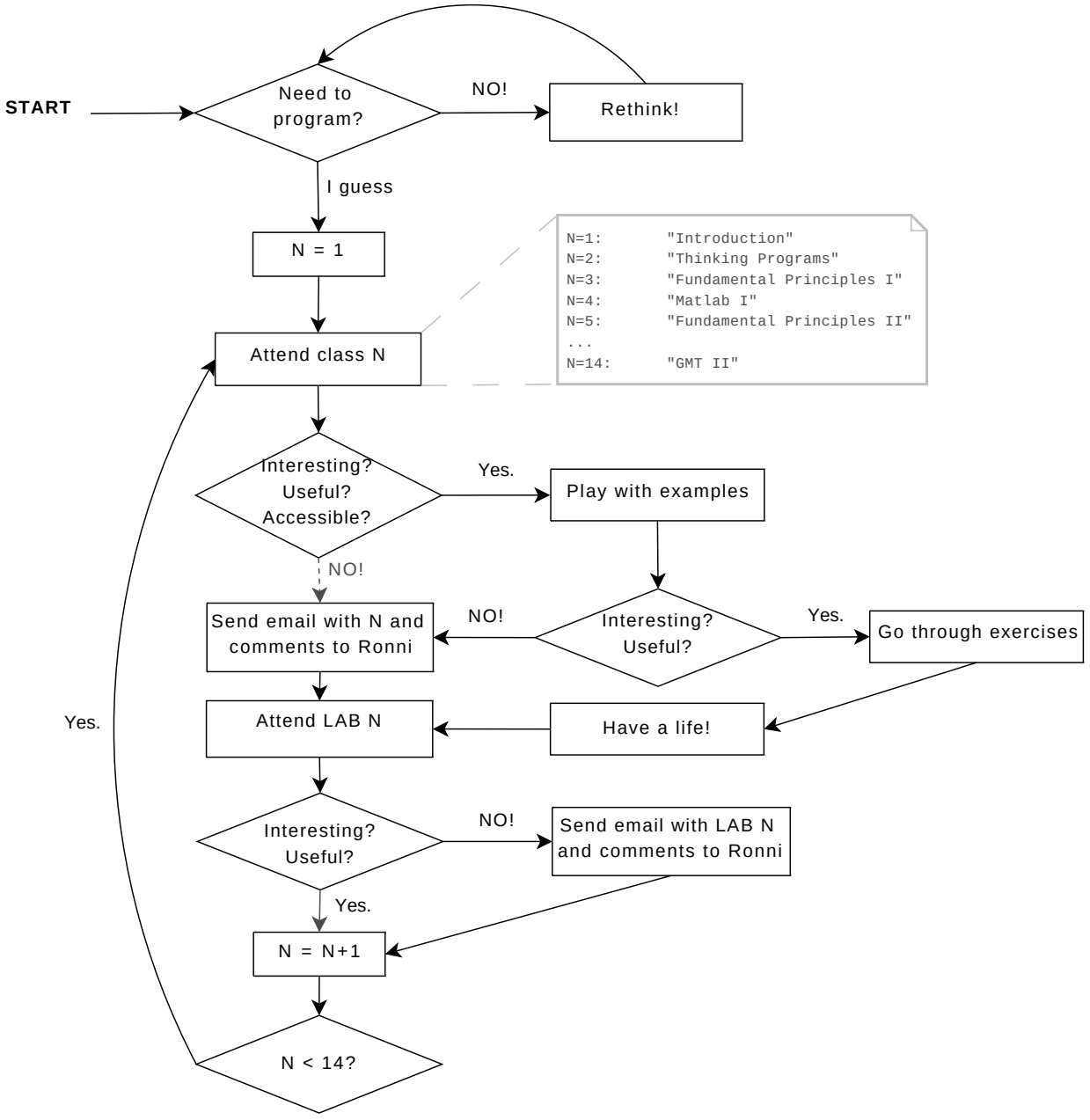
The Program . . .


START
Need to program?
NO!
Rethink!
I guess
N = 1
N=1: "Introduction"
N=2: "Thinking Programs"
N=3: "Fundamental Principles I"
N=4: "Matlab I"
N=5: "Fundamental Principles II"
...
N=13: "HTML"
Attend class N
Interesting? Useful? Accessible?
Yes.
Play with examples
NO!
Send email with N and comments to Jeff
NO!
Interesting? Useful?
Yes.
Go through exercises
Attend LAB N
Have a life!

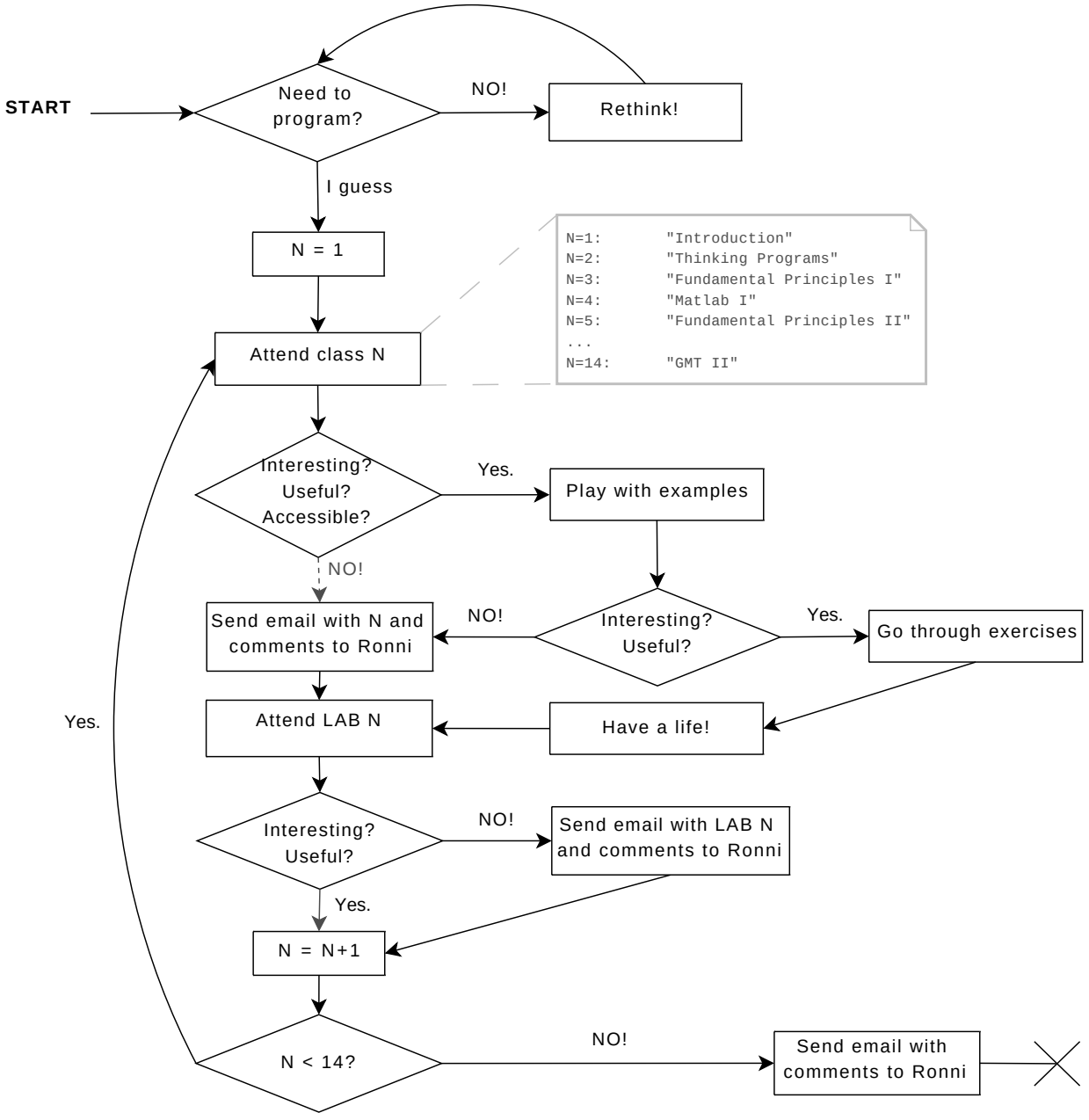
The Program . . .


START
Need to program?
NO!
Rethink!
I guess
N = 1
N=1: "Introduction"
N=2: "Thinking Programs"
N=3: "Fundamental Principles I"
N=4: "Matlab I"
N=5: "Fundamental Principles II"
...
N=13: "HTML"
Attend class N
Interesting? Useful? Accessible?
Yes.
Play with examples
NO!
Send email with N and comments to Jeff
NO!
Interesting? Useful?
Yes.
Go through exercises
Attend LAB N
Have a life!
Interesting? Useful?
NO!
Send email with LAB N and comments to Joe
Yes.
N = N+1

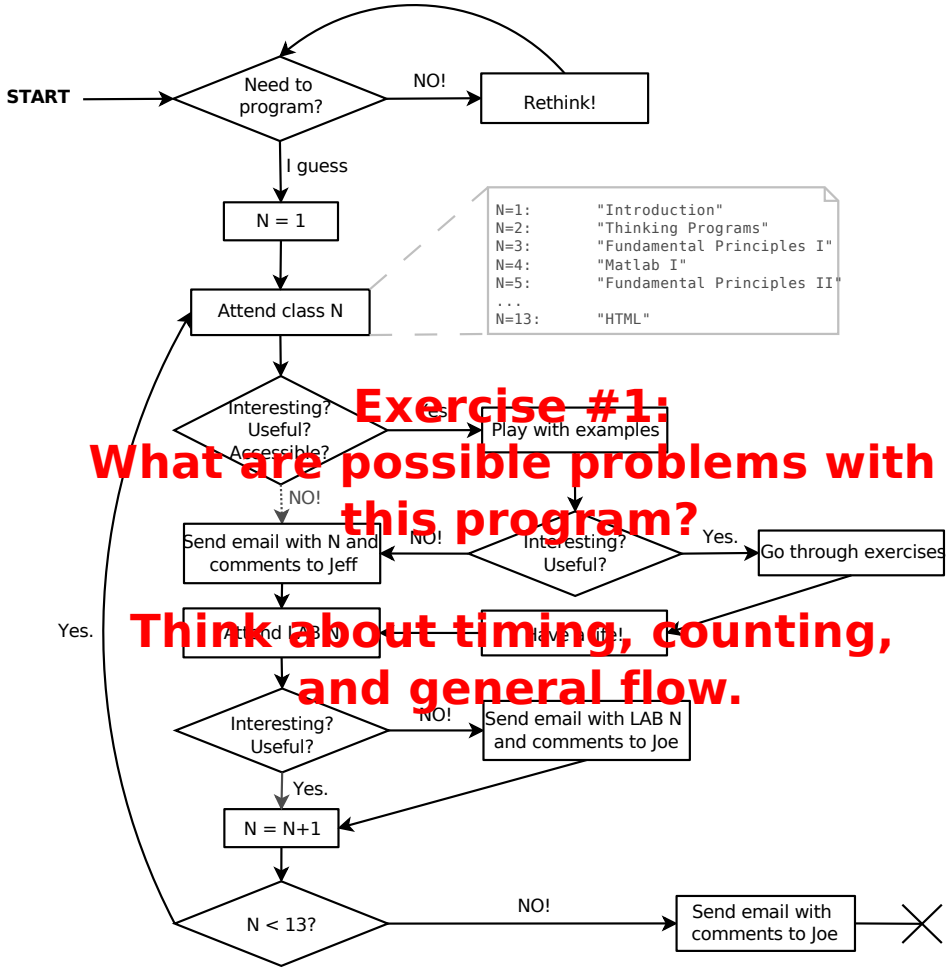
The Program . . .


START
Need to program?
NO!
Rethink!
I guess
N = 1
N=1: "Introduction"
N=2: "Thinking Programs"
N=3: "Fundamental Principles I"
N=4: "Matlab I"
N=5: "Fundamental Principles II"
...
N=14: "GMT II"
Attend class N
Interesting? Useful? Accessible?
Yes.
Play with examples
NO!
Send email with N and comments to Ronni
NO!
Interesting? Useful?
Yes.
Go through exercises
Attend LAB N
Have a life!
Yes.
Interesting? Useful?
NO!
Send email with LAB N and comments to Ronni
Yes.
N = N+1
N < 14?

The Program . . .


START
Need to program?
NO!
Rethink!
I guess
N = 1
N=1: "Introduction"
N=2: "Thinking Programs"
N=3: "Fundamental Principles I"
N=4: "Matlab I"
N=5: "Fundamental Principles II"
...
N=14: "GMT II"
Attend class N
Interesting? Useful? Accessible?
Yes.
Play with examples
NO!
Send email with N and comments to Ronni
NO!
Interesting? Useful?
Yes.
Go through exercises
Attend LAB N
Have a life!
Yes.
Interesting? Useful?
NO!
Send email with LAB N and comments to Ronni
Yes.
N = N+1
N < 14?
NO!
Send email with comments to Ronni

The Program . . .


START
Need to program?
NO!
Rethink!
I guess
N = 1
N=1: "Introduction"
N=2: "Thinking Programs"
N=3: "Fundamental Principles I"
N=4: "Matlab I"
N=5: "Fundamental Principles II"
...
N=13: "HTML"
Attend class N
Interesting? Useful? Accessible?
Yes.
Play with examples
NO!
Send email with N and comments to Jeff
NO!
Interesting? Useful?
Yes.
Go through exercises
Yes.
Attend LAB N
Have a life!
Interesting? Useful?
NO!
Send email with LAB N and comments to Joe
Yes.
N = N+1
N < 13?
NO!
Send email with comments to Joe
Exercise #1:
What are possible problems with this program?
Think about timing, counting, and general flow.

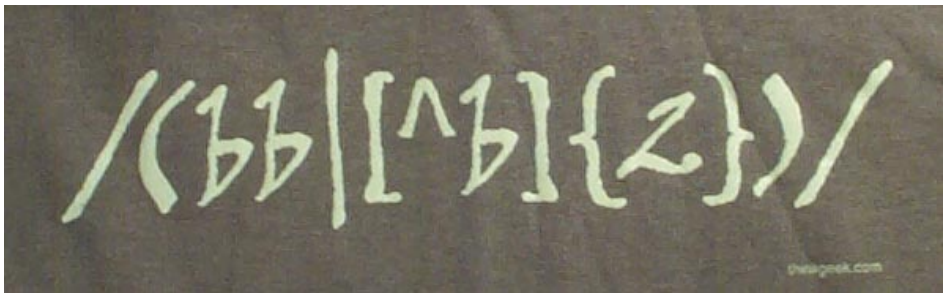
The very basics (1)

From ‘The Conscience of a Hacker’, The Mentor (1986):

[. . .] I made a discovery today. **I found a computer.** Wait a second, this is cool. It does what I want it to. **If it makes a mistake, it’s because I screwed it up. Not because it doesn’t like me . . .**
Or feels threatened by me . . .
Or thinks I’m a smart ass . . .
Or doesn’t like teaching and shouldn’t be here [. . .]

The very basics (2)

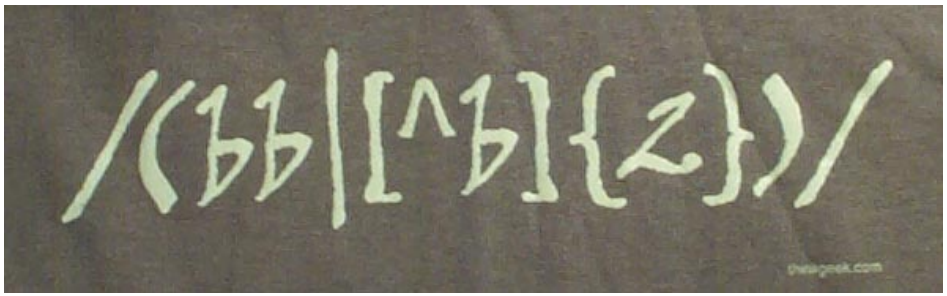
- Programming is beyond language.



http://thinkgeek.com

The very basics (2)

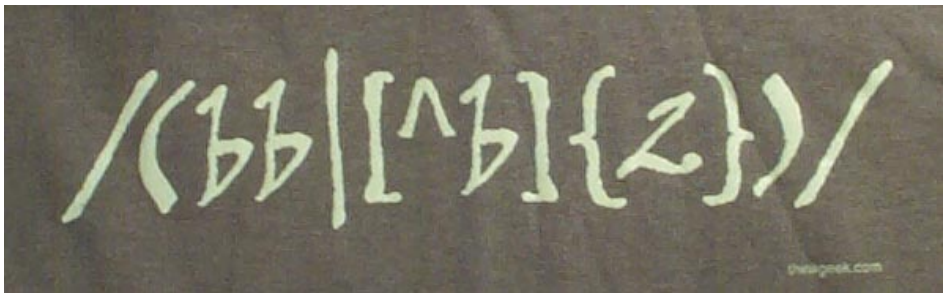
- Programming is beyond language.
- Programming is about writing code that people can read.



http://thinkgeek.com

The very basics (2)

- Programming is beyond language.
- Programming is about writing code that people can read.
- Code is poetry.



http://thinkgeek.com

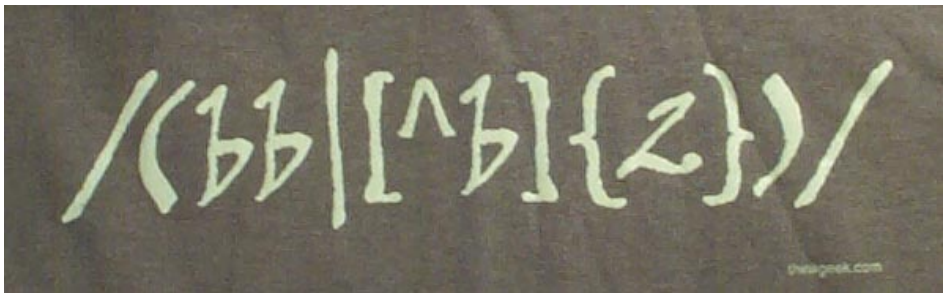
"When I'm writing poetry, it feels like the center of my thinking is in a particular place, and when I'm writing code the center of my thinking feels in the same kind of place."

Richard Gabriel,

Distinguished Engineer at Sun Microsystems

The very basics (2)

- Programming is beyond language.
- Programming is about writing code that people can read.
- Code is poetry.
- RTFM *and/or* the internet



http://thinkgeek.com

"When I'm writing poetry, it feels like the center of my thinking is in a particular place, and when I'm writing code the center of my thinking feels in the same kind of place."

Richard Gabriel,

Distinguished Engineer at Sun Microsystems

More Philosophy . . .

Jon Claerbout (a geophysicist), as quoted in “WaveLab and Reproducible Research”:

More Philosophy . . .

Jon Claerbout (a geophysicist), as quoted in “WaveLab and Reproducible Research”:

An article about computational science in a scientific publication is **not** *the scholarship itself, it is merely* **advertising** *of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.*

More Philosophy . . .

Jon Claerbout (a geophysicist), as quoted in “WaveLab and Reproducible Research”:

An article about computational science in a scientific publication is **not** *the scholarship itself, it is merely* **advertising** *of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.*

Implications . . .

- publications should include data and code (example: Okada)
- figures should be reproducible by readers
- write code that others can use!

What does that mean?

Good

```
function fp = screw2d(x, xf, d, sdot)
% function fp = screw2d(x, xf, d, sdot)
%
% Computes fault-parallel slip rate for 2D screw dislocation
% with fault located at xf, with locking depth d and slip rate sdot.
% Will compute at one or many locations x.
%
% x     column vector
% xf    scalar
% d     scalar
% sdot  scalar
%
if ( d == 0 )
    fp = sdot*0.5*sign(x-xf*ones(size(x)));
else
    fp = sdot*atan2((x-xf*ones(size(x))),d)/pi;
end
```

What does that mean?

Good

```
function fp = screw2d(x, xf, d, sdot)
% function fp = screw2d(x, xf, d, sdot)
%
% Computes fault-parallel slip rate for 2D screw dislocation
% with fault located at xf, with locking depth d and slip rate sdot.
% Will compute at one or many locations x.
%
% x     column vector
% xf    scalar
% d     scalar
% sdot  scalar
%
if ( d == 0 )
    fp = sdot*0.5*sign(x-xf*ones(size(x)));
else
    fp = sdot*atan2((x-xf*ones(size(x))),d)/pi;
end
```

Bad

```
function fp = screw2d(x, xf, d, sdot)
if(d==0)fp=sdot*0.5*sign(x-xf*ones(size(x)));else fp=sdot*atan2((x-xf*ones(size(x))),d)/pi;
end
```

Outline

Example 1:

Getting into grad school ... and out.

Thinking programs – Breaking down complex tasks

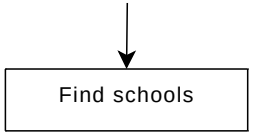
Example 1:

Getting into grad school ... and out.

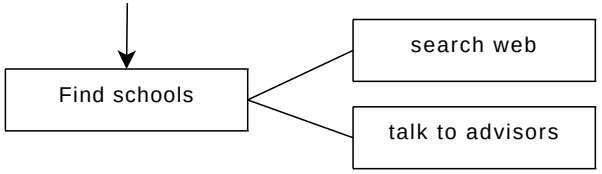
things to do:

apply, figure out where to go, visa stuff, class work, research, thesis ...

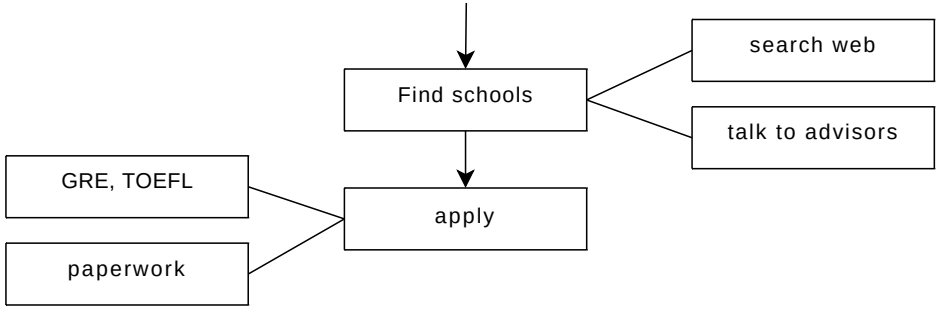
Thinking programs – Breaking down complex tasks



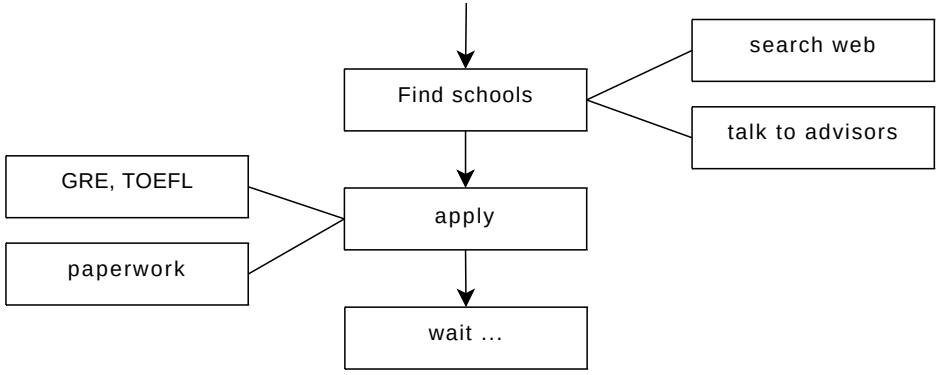
Thinking programs – Breaking down complex tasks



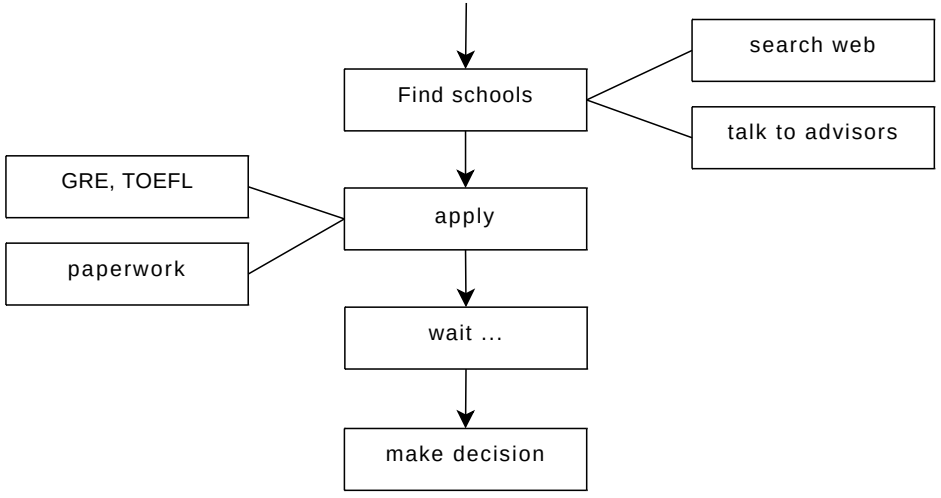
Thinking programs – Breaking down complex tasks



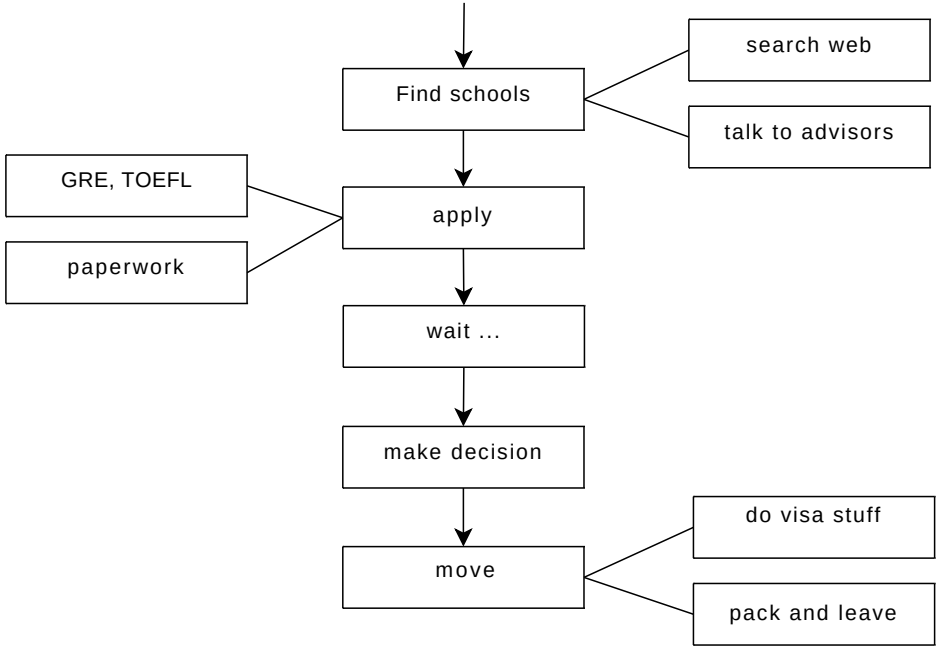
Thinking programs – Breaking down complex tasks



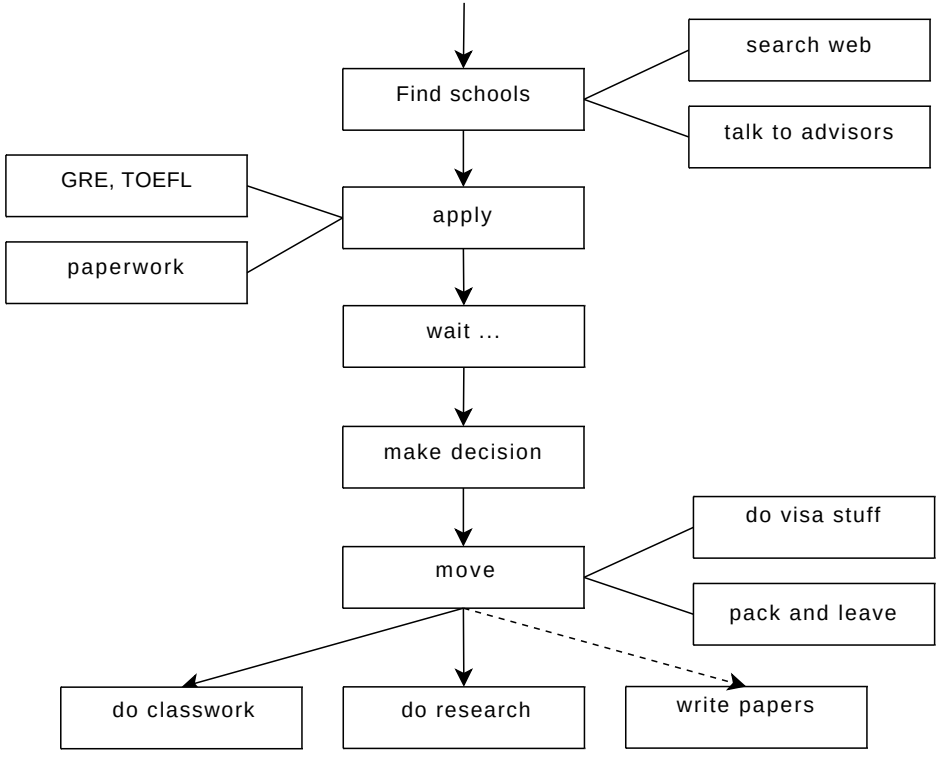
Thinking programs – Breaking down complex tasks



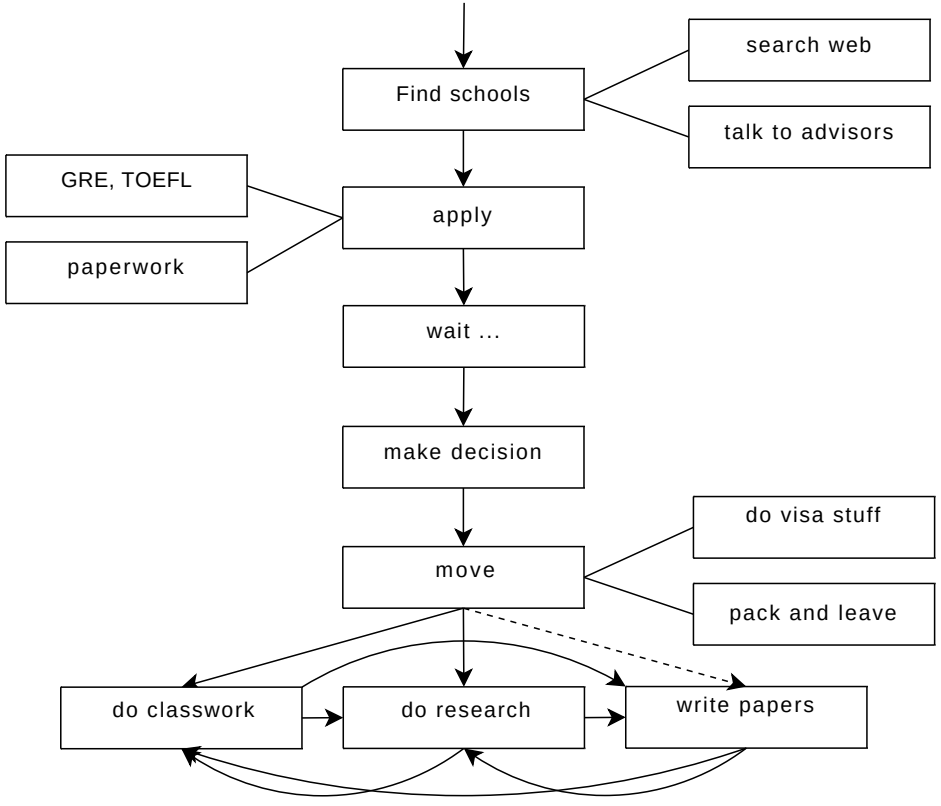
Thinking programs – Breaking down complex tasks



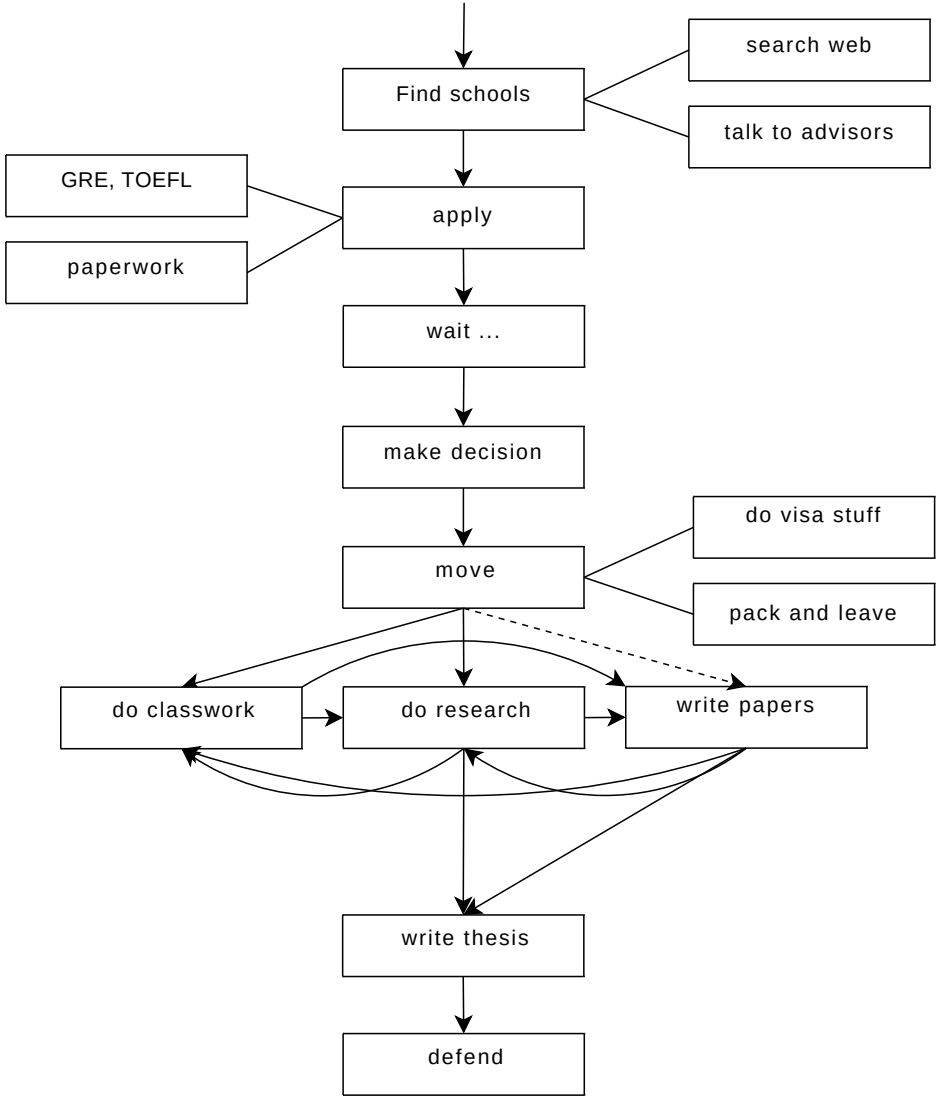
Thinking programs – Breaking down complex tasks



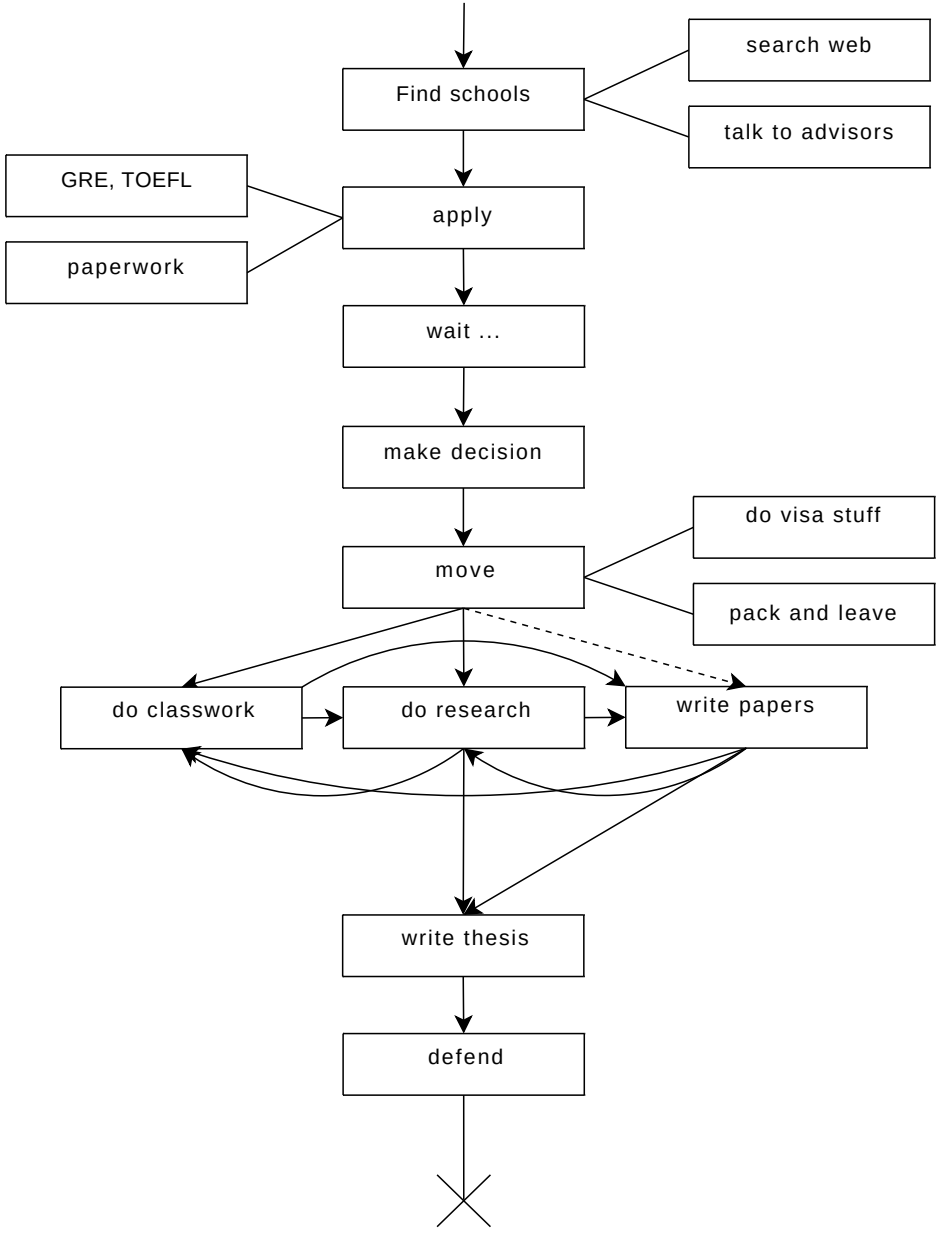
Thinking programs – Breaking down complex tasks



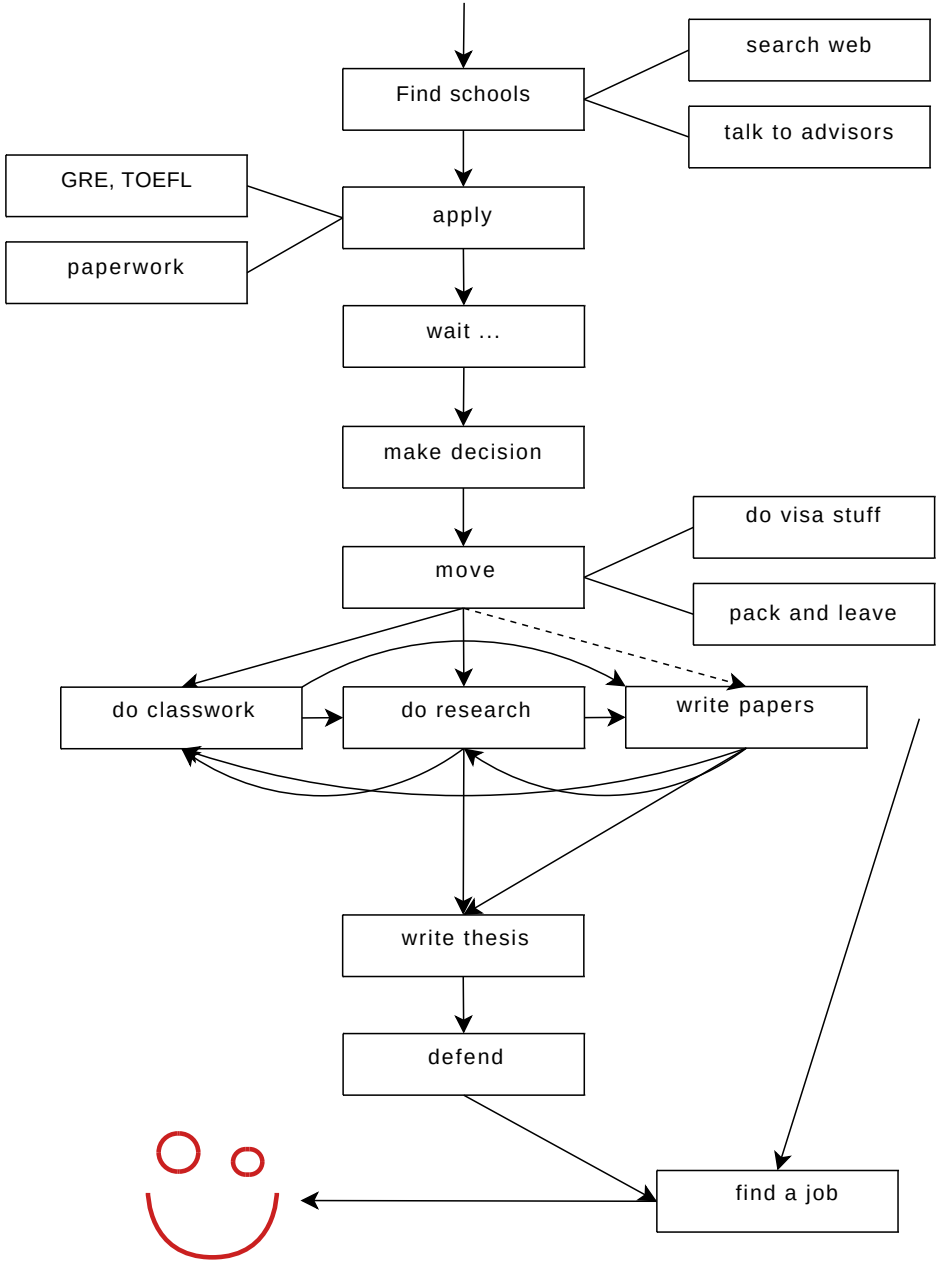
Thinking programs – Breaking down complex tasks



Thinking programs – Breaking down complex tasks



Thinking programs – Breaking down complex tasks



Example 2:

Grad student's Average Day

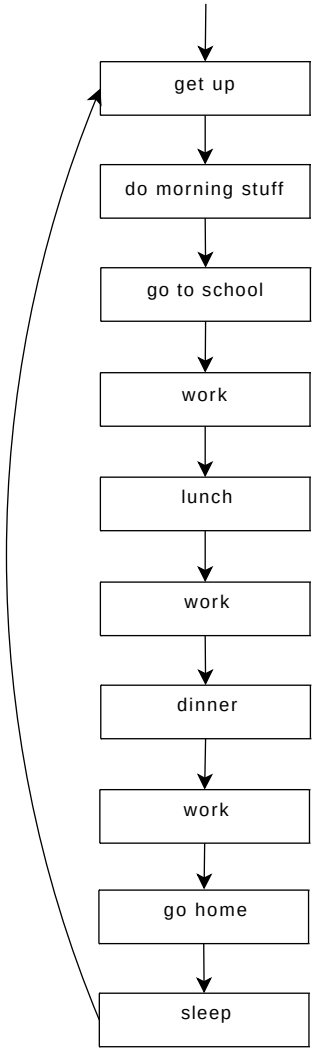
Example 2:

Grad student's Average Day

possible activities:

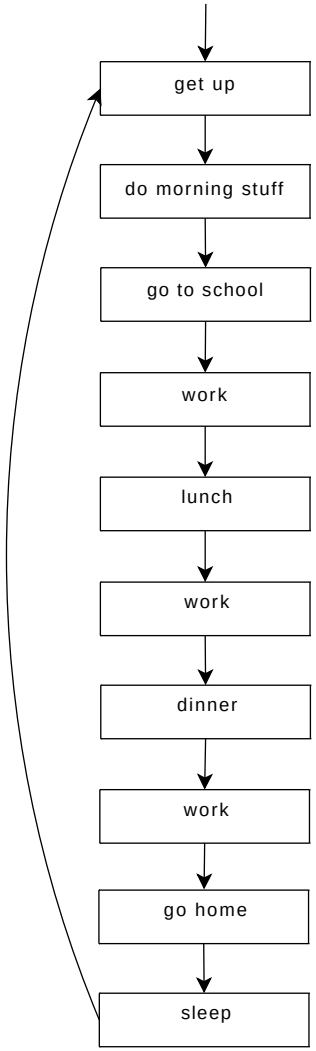
eat, sleep, work, do stuff, . . .

Thinking programs – Breaking down complex tasks



Listing 1: make_my_day

Thinking programs – Breaking down complex tasks



possible implementation

```
% make_my_day.m
%_____________
% program that shows how much fun
% live as a grad student is :)

clc;

getUp;
eat('breakfast');
walk('school');
work;
eat('lunch');
work();
eat('dinner');
work();
walk('home');
haveLife;
sleep;
```

Listing 1: make_my_day

Outline

Building programs – One black box at a time

Strategies to implement a program:

Top down

Same as the examples above:

- start with the big picture
- identify reasonable subtasks
- try to divide things to a level of managable complexity (atoms)
- implement atoms
- implement main routine (flow control)

Building programs – One black box at a time

Strategies to implement a program:

Top down

Same as the examples above:

- start with the big picture
- identify reasonable subtasks
- try to divide things to a level of managable complexity (atoms)
- implement atoms
- implement main routine (flow control)

Bottom up

- problems accumulate
- implement an atom at the time
- at some point you figure out that things could go together
- revise main routine constantly
- add necessary subroutines

Building programs – One black box at a time

Bottom line

- Try building tools that solve a set of similar problems in a generic way. Use Parameters!
- Build and test each atom individually, test all scenarios (and more) with synthetic input.
- Treat atoms as black boxes that implement desired functionality. Don’t care about them once they’re working

Building programs – One black box at a time

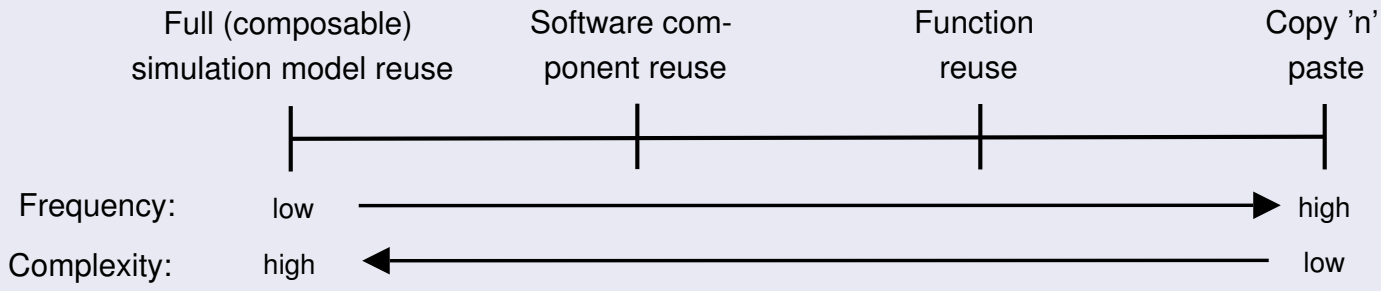
Keys to good programs

- **Modularity**: split problem in manageable tasks, implement and test one at a time

Building programs – One black box at a time

Keys to good programs

- **Modularity**: split problem in manageable tasks, implement and test one at a time
- **Reusability**: write functions, avoid redundancy, avoid monolithic code (theoretically one loop would be enough)

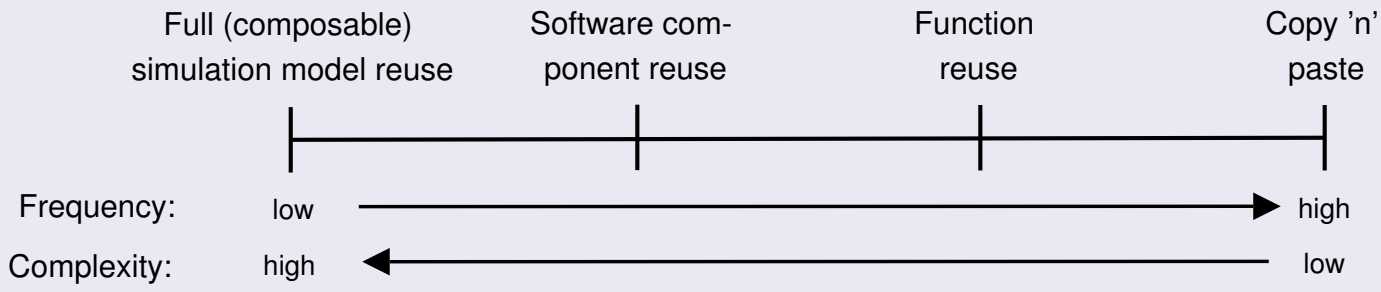


Pidd, 2002

Building programs – One black box at a time

Keys to good programs

- **Modularity**: split problem in manageable tasks, implement and test one at a time
- **Reusability**: write functions, avoid redundancy, avoid monolithic code (theoretically one loop would be enough)



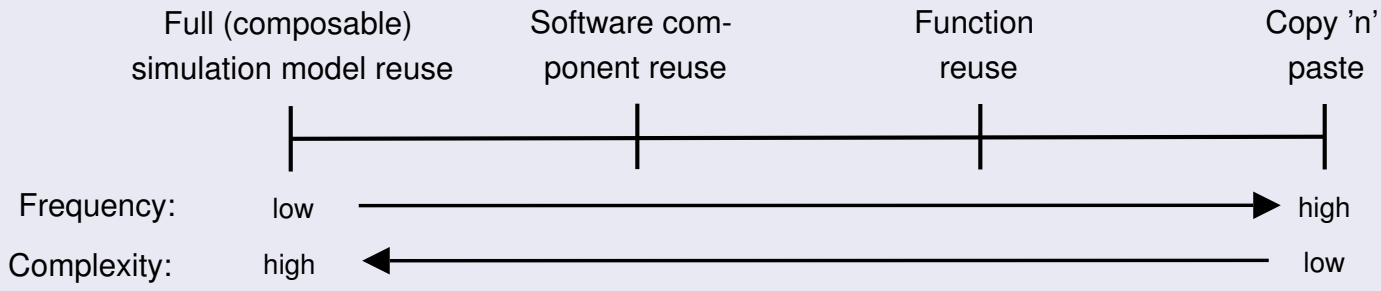
Pidd, 2002

- **Generalize**: use variables instead of hard coded values, hand parameters to functions

Building programs – One black box at a time

Keys to good programs

- **Modularity**: split problem in manageable tasks, implement and test one at a time
- **Reusability**: write functions, avoid redundancy, avoid monolithic code (theoretically one loop would be enough)



Pidd, 2002

- **Generalize**: use variables instead of hard coded values, hand parameters to functions
- Functionality, then efficiency

Building programs

The Control Routine

```
% make_my_day.m
%----------------
% program that shows how much fun
% live as a grad student is :)

clc;

getUp;
eat('breakfast');
walk('school');
work;
eat('lunch');
work();
eat('dinner');
work();
walk('home');
haveLife;
sleep;
```

Using Parameters

```
% eat.m
%----------------
function eat(what)
    fprintf(1,'%s: yummy ... %s\n', ...
            mfilename, what);
    pause(1);
end
```

Outline

Summary – Take home messages

Thinking . . .

- Think modular
- Think in general cases
- Think non-redundant
- Think about reuse
- Think about reproducibility

Exercising . . .

- Read other peoples' code . . . critically
- The first version is for the trash bin (unintentionally)

Summary – Take home messages

Thinking . . .

- Think modular
- Think in general cases
- Think non-redundant
- Think about reuse
- Think about reproducibility

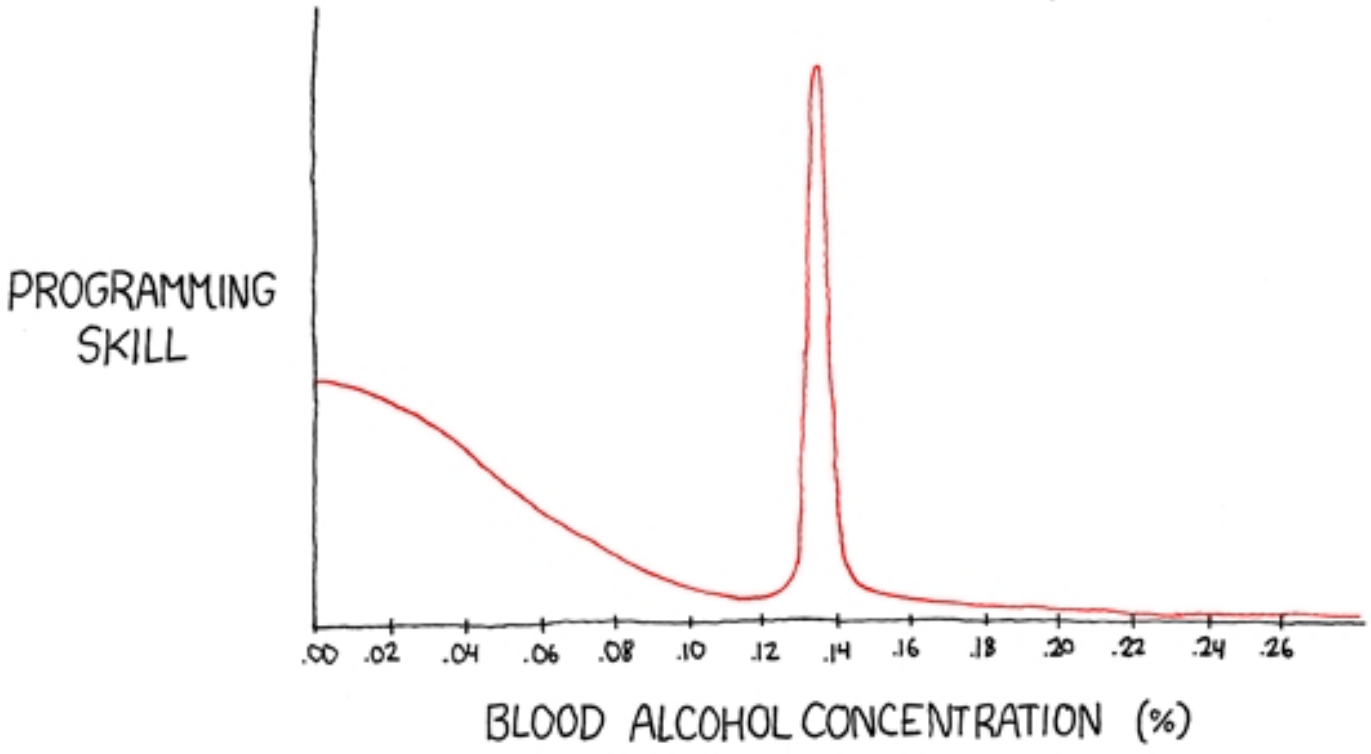
Exercising . . .

- Read other peoples' code . . . critically
- The first version is for the trash bin (unintentionally)

Truth . . .

Your working environment will change, concepts likely survive! Be flexible in the choice of languages and tools.

If all fails . . .



“The Ballmer Peak”

http://www.xkcd.com/323/