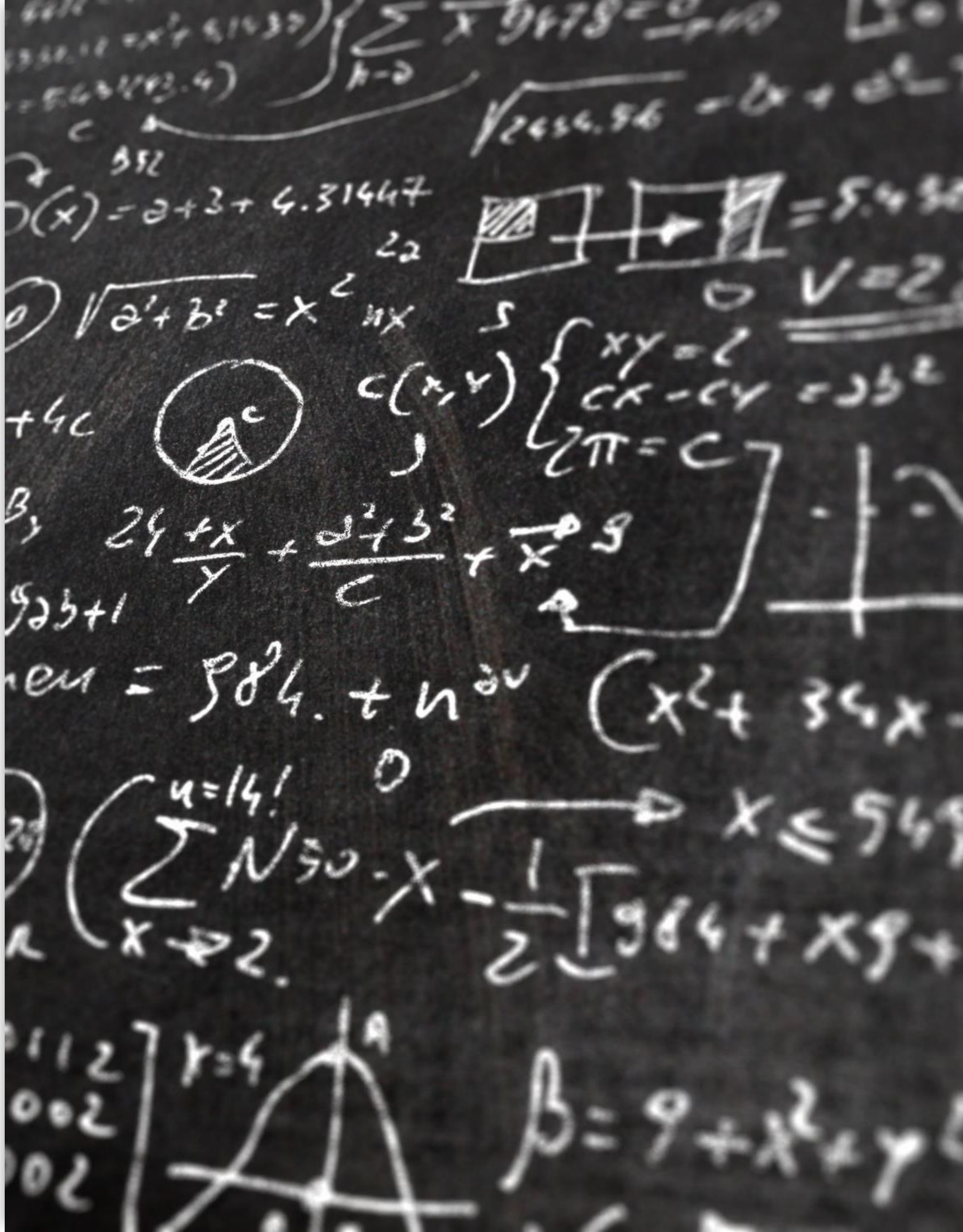
School of Computing & Communications | Lancaster University

SCC.111 Software Development – Lecture 6: Reading code

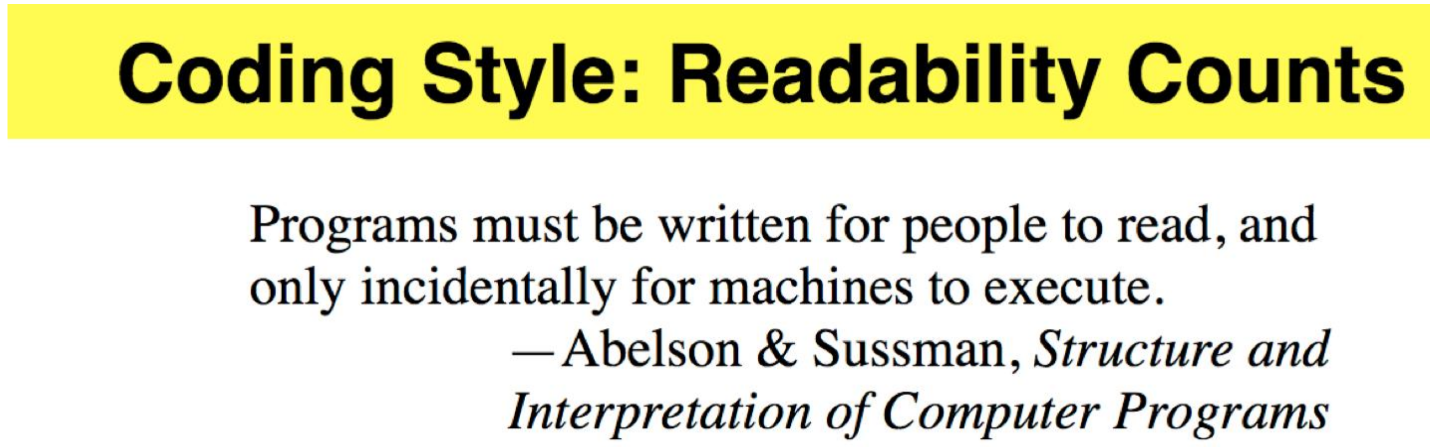
Adrian Friday, Nigel Davies, Hansi Hettiarachchi, Saad Ezzini

This lecture

- How to write code that's a pleasure to read and maintain!
- The do's and don't's of (our house) code style
- Some worked examples



Reading computer programs



Coding Style: Readability Counts

Programs must be written for people to read, and only incidentally for machines to execute.

—Abelson & Sussman, *Structure and Interpretation of Computer Programs*

“Reading great code is just as important for a programmer as reading great books is for a writer”

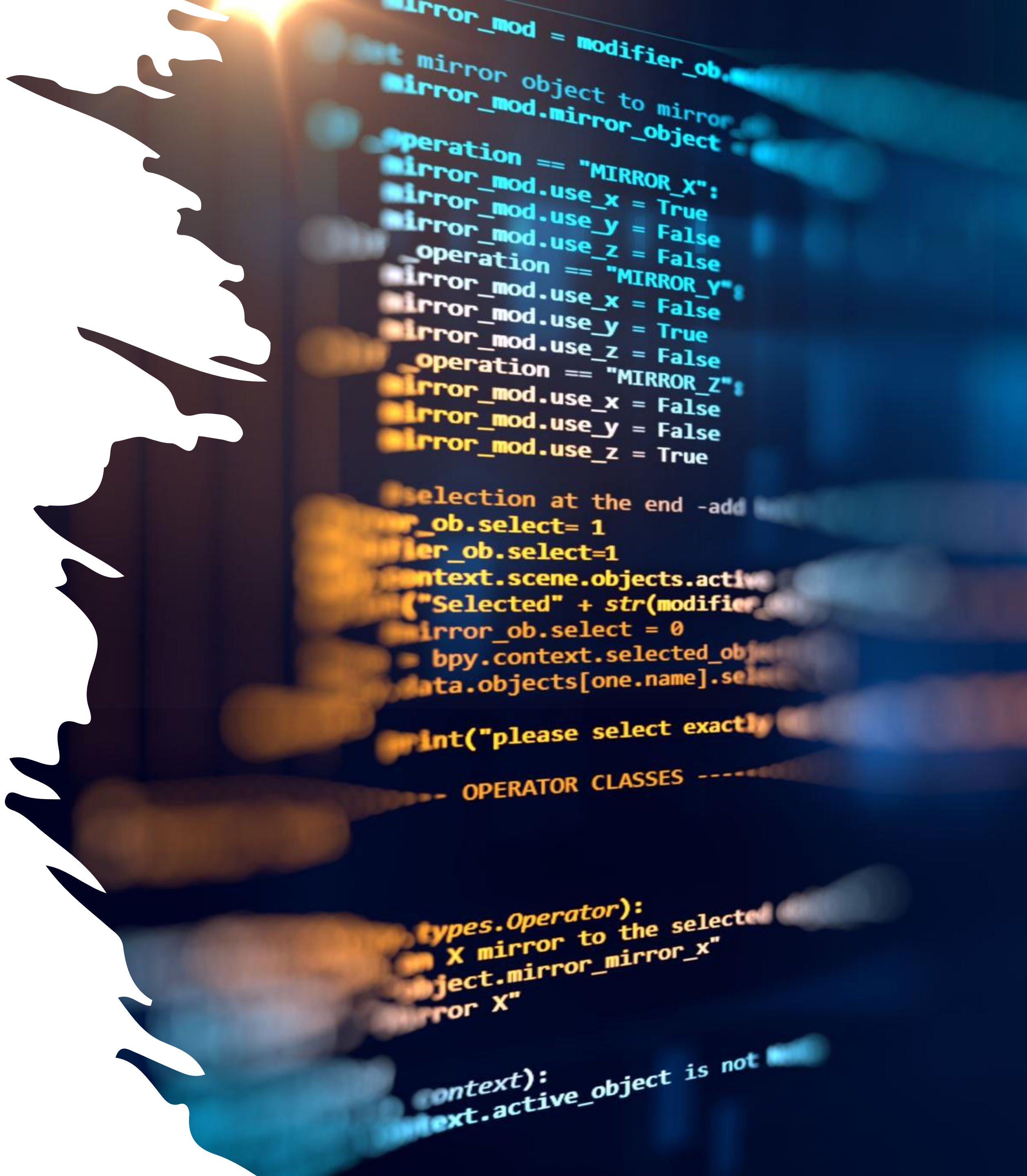
Peter Norvig, Director of Research, Google Inc.

as program writers we can do a lot to make things clearer

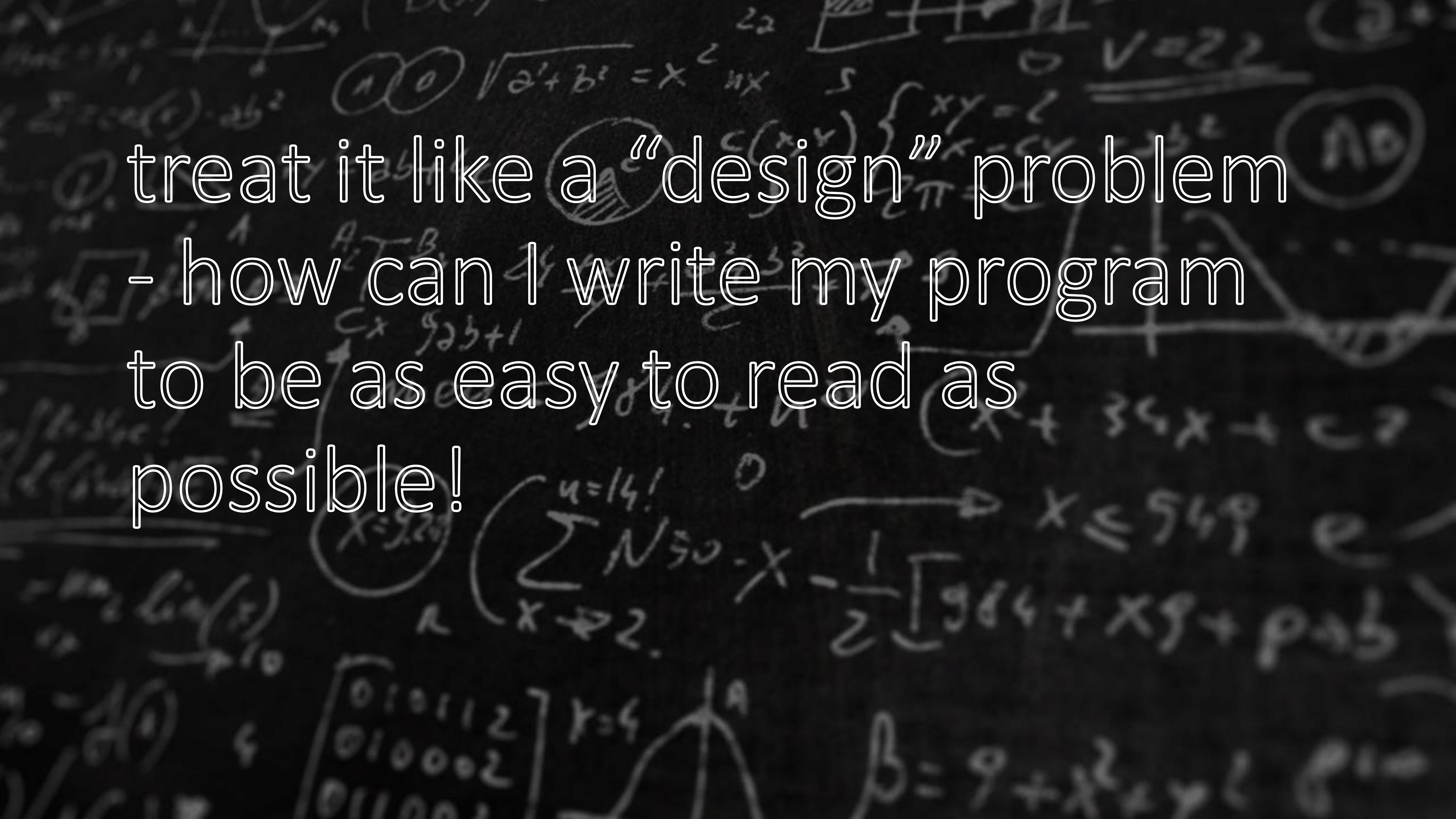
The difference between a tolerable programmer and a great programmer is not how many programming languages they know, and it's not whether they prefer Python or Java. **It's whether they can communicate their ideas.” Joel Spolsky**

The difference between a tolerable programmer and a great programmer is not how many programming languages they know, and it's not whether they prefer Python or Java. **It's whether they can communicate their ideas.”**

Joel Spolsky



```
irror_mod = modifier_ob.
 mirror object to mirror
irror_mod.mirror_object
peration == "MIRROR_X":
irror_mod.use_x = True
irror_mod.use_y = False
irror_mod.use_z = False
 operation == "MIRROR_Y"
irror_mod.use_x = False
irror_mod.use_y = True
irror_mod.use_z = False
 operation == "MIRROR_Z"
rror_mod.use_x = False
rror_mod.use_y = False
rror_mod.use_z = True

election at the end -add
_ob.select= 1
er_ob.select=1
ntext.scene.objects.activ
("Selected" + str(modifier
irror_ob.select = 0
 bpy.context.selected_ob
ata.objects[one.name].se

int("please select exactl

 OPERATOR CLASSES ----

types.Operator):
 X mirror to the selected
ject.mirror_mirror_x"
rror X"

ontext):
ext.active_object is not
```



treat it like a “design” problem - how can I write my program to be as easy to read as possible!

Adding Comments

- As we've already seen, we can add comments in C
- The compiler ignores them, the coder doesn't!
- Two different ways:
  - For a single line, everything after a // is ignored
  - For a block of code, begin with /* and end with */ - *handy for taking code in/out too!*
- *Warning: Care not to* attempt to *nest* block comments !

Beautiful, readable code

- Always indent the code contained within braces
- Set your editor to save tabs as spaces
- Use a consistent indentation size (eg, 2 or 4 spaces per level)
- Don’t leave white space at the end of lines
- Add comments to clarify your code
- Give your variables meaningful names
- Use ‘camelCase’ for variables

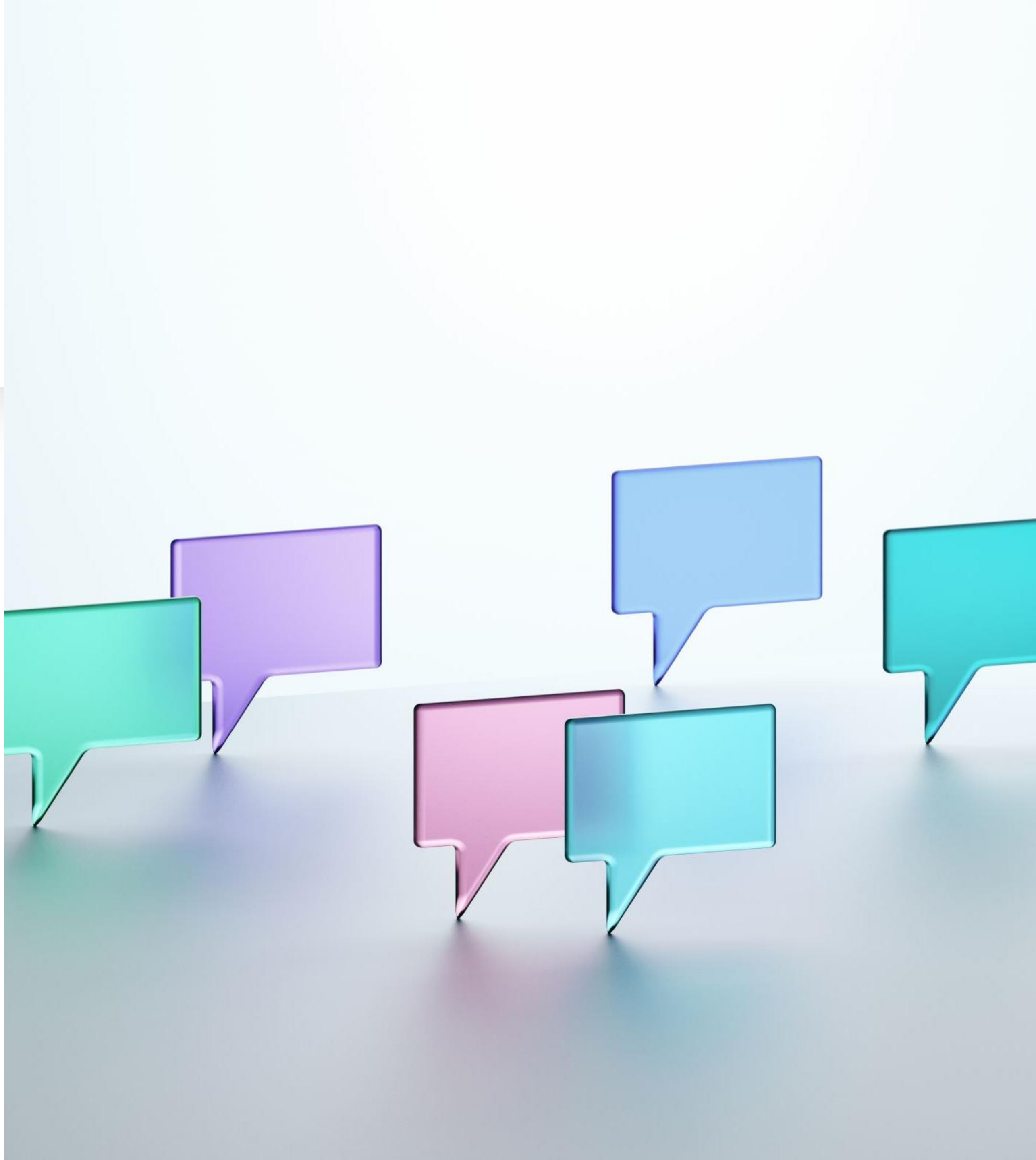
Code style examples with ‘for’

```
#include <stdio.h>
int main ()
{int zz, p, p2, temp;
                                scanf(“%d,%d”,&p2,&temp);
                    for (zz=0; zz<p2;zz++)
                                         {
            for (p=0; p<temp;p++) {
                       printf (“*”);} printf (“\n”);
                                                  }}
```

```
#include <stdio.h>

/* Name: Rectangle Drawer
 * Author: Nige
 *
 * Draws a rectangle of a size specified by the user
*/

int main ()
{
  int i, j, rows, columns;

  // read in the size from the user (N.B. input separated by a comma)

  scanf(“%d,%d”, &rows, &columns);

  for (i = 0; i < rows; i++) {
    for (j = 0; j < columns; j++)
      printf("*");
    printf ("\n");
  }
}
```

How many comments are enough?

- Code can be under commented
- Code can be over commented
- *Discuss! What is the purpose of a comment?*



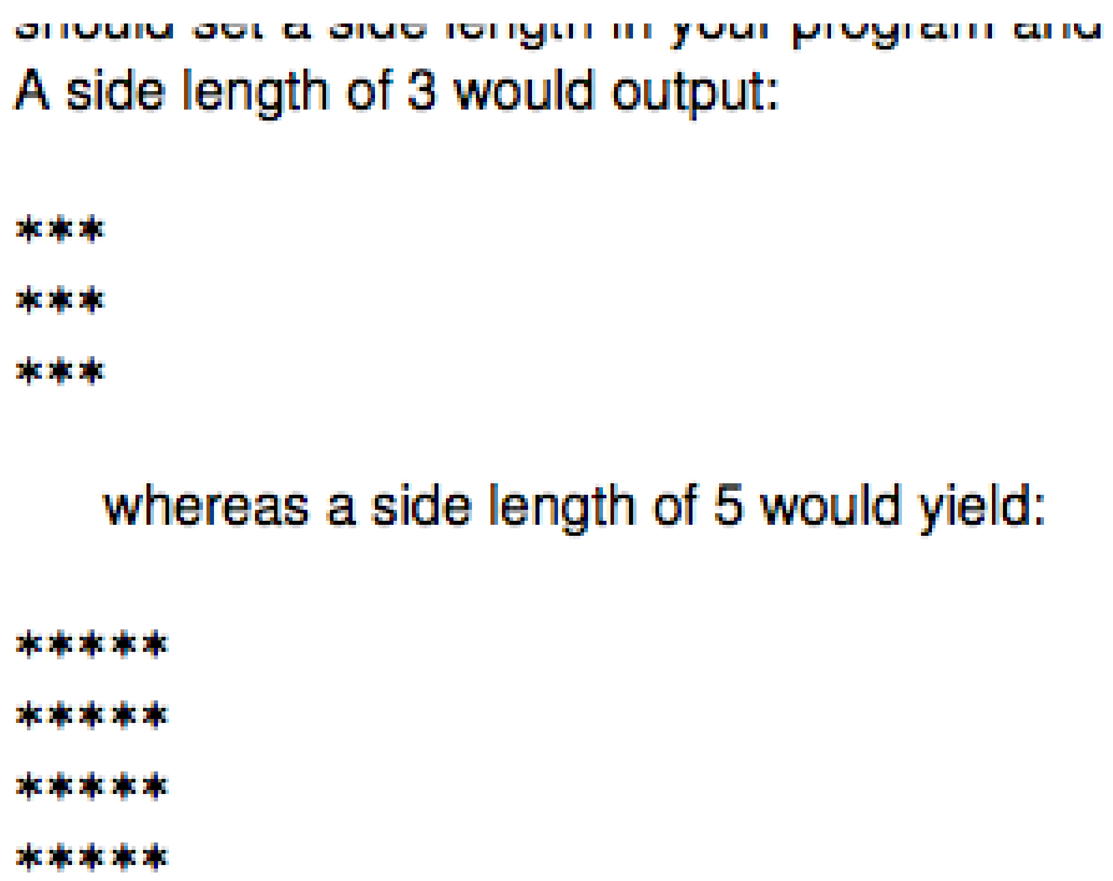
Comments

- Should add value, and shouldn’t need to *restate* that which is obvious from the code
    - Comments should be parsimonious, that is, enough to help you make sense of the approach being taken
    - Good variable and function names add to readability
    - Too many comments *obscure the code* and make it hard to maintain!
    - Comments shouldn’t be added later…! They’re also there to help you as a developer!

We set you a problem: drawing a square

Let’s design a solution step by step on paper

Now let's turn this into (**readable**) code



should set a side length in your program and

A side length of 3 would output:

```
***
***
***
```

whereas a side length of 5 would yield:

```
*****
*****
*****
*****
```

Note how our solution

- Makes good use of **comments throughout the coding process** (not just at the end as an afterthought!)
- Uses **vertical space** to separate lines and blocks of code to make it more readable
- Uses **spaces** around operators and line breaks to get more 'air' and 'readability' into each statement
- Uses horizontal space (**indents**) effectively to make the 'blocks' of related code and changes in 'flow' more obvious
- Uses sensible **human readable** variable names to make the representation of the problem 'as data' more obvious

Summary

- You should know about the importance of good code style
- Comments and how to add them
- “Proper” indentation (as we’d like to see it)
- Our house code style – *aim for ruthless consistency!*