

#### Foundations of Style: Whence The Way? A program is a document: – *Some* of it is read by a computer.

- All of it is read by people.

"Style" is a catch-all term for *people*-oriented programming encompassing all documentation:

 comments, spacing, indentation, names, clear & straightforward & well-organized code

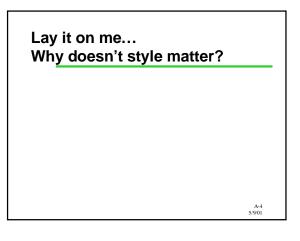
> A-2 5/9/01

#### Style "Extremism": Literate Programming

...we can best achieve this [better documentation] by considering programs to be works of literature... Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.

-Donald Knuth

A-3 5/9/01

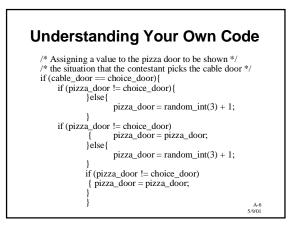


## But, I don't need good style because the *computer* doesn't!

Remember the "obfuscated" code?

- The computer "understands" fundamentally differently from you!
- Just because code is executable, doesn't mean it's comprehensible.. even to the author!

A-5 5/9/01



### But *now*, my bare code is crystal clear to others!

...the code itself... is something we can run but not exactly understand... Even if you have the source code in front of you, there are limits to what a human reader can absorb from thousands of lines of text designed primarily to function, not to convey meaning.

-Ellen Ullman (programmer of 20 years)

A-7 5/9/01

# And again... It is the difference between performing and exposing a magic trick. -Ross Williams

### But, my documentation tells exactly what the code does! /\* subtract one from sheep \*/ sheep = sheep - 1; /\* account for the sheep that the big bad wolf just ate \*/ sheep = sheep - 1;

A-9 5/9/01

#### But, my style is better than {yours, this code's, everyone's} /\* init the double-buffer windows \*/ the\_hdc = GetDC(hDrawWnd); offscreen\_bitmap = CreateCompatibleBitmap(the\_hdc,

2\*GP142\_XMAX+1, 2\*GP142\_YMAX+FUDGE); offscreen\_DC = CreateCompatibleDC(the\_hdc);

> A-10 5/9/01

### But, this code is so elegant, no one could *fail* to understand it!

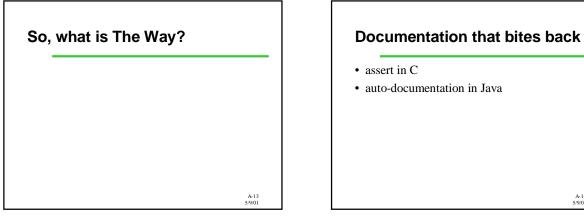
reveal\_door =

(prize\_door == door\_choice) \* (door\_choice + random\_int(2) + 1) + (prize\_door != door\_choice) \* (6 - prize\_door - door\_choice);

reveal\_door = (reveal\_door - 1) % 3 + 1; switch\_door = 6 - reveal\_door - door\_choice;

> A-11 5/9/01





# · auto-documentation in Java

A-14 5/9/01

#### Syntactic "Salt" (and the Python language)

def left\_child(self, location):

child = location \* 2 + 1

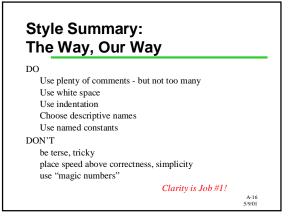
if child >= self.pairs\_size:

return None

else:

return child

So... indentation is syntactic salt in Python. But Python doesn't have types... therefore? A-15 5/9/01



#### **QOTD: Guerilla Style Wars**

Think of a common bug/problem you have in your code.

Now try to imagine a *stylistic* convention that would overcome that.

Example:

– I might often type = rather than ==.

- If I never write (x == 3) but instead always write (3 == x), the compiler will find my bug!

Is the 3 = x convention really good style? Would it really help? What about your convention? A-17 5/9/01

3