

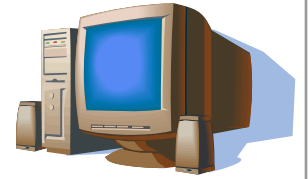
Computer Programming

Telling Computers What To Do

Jan Hannemann, Hidehiko Masuhara

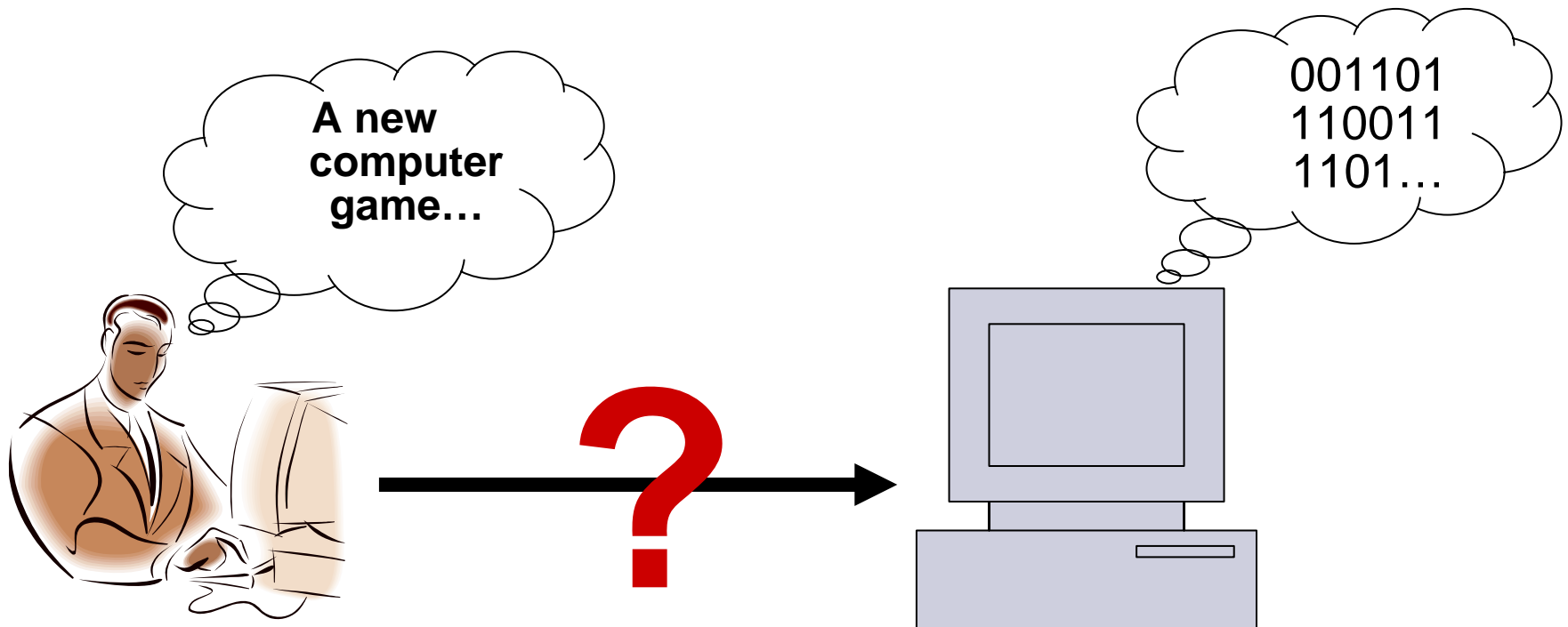
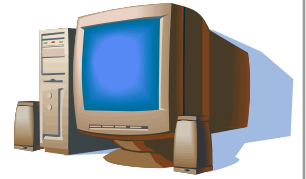
University of Tokyo

Computers Are Useful For...

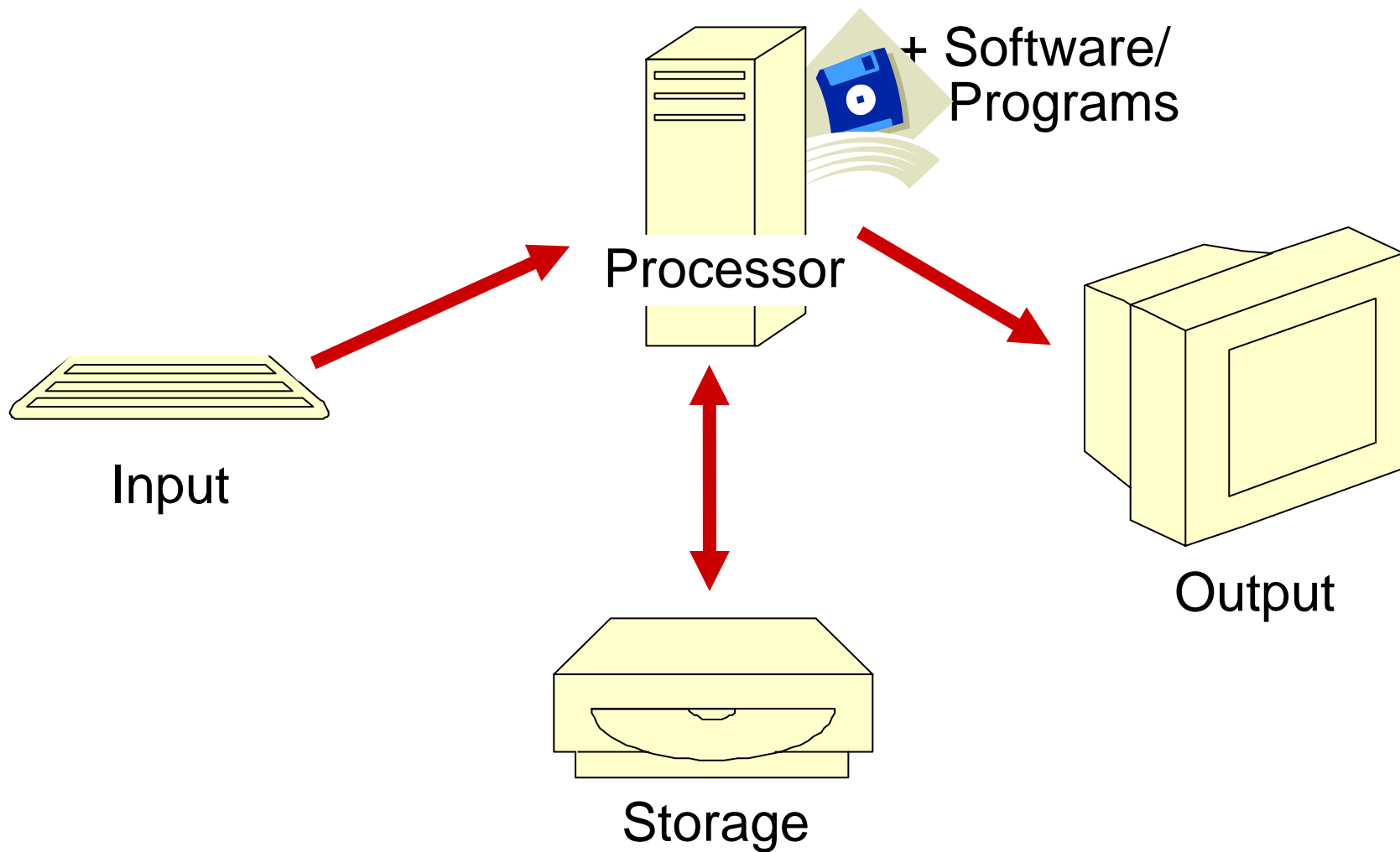
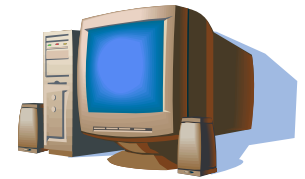


- Animations
 - Games & movies
- Computations
 - Weather forecasts & earthquake predictions
- Applications
 - Text processing, presentations
- Electronic Devices
 - Cell phones, rice cookers
- Etc.

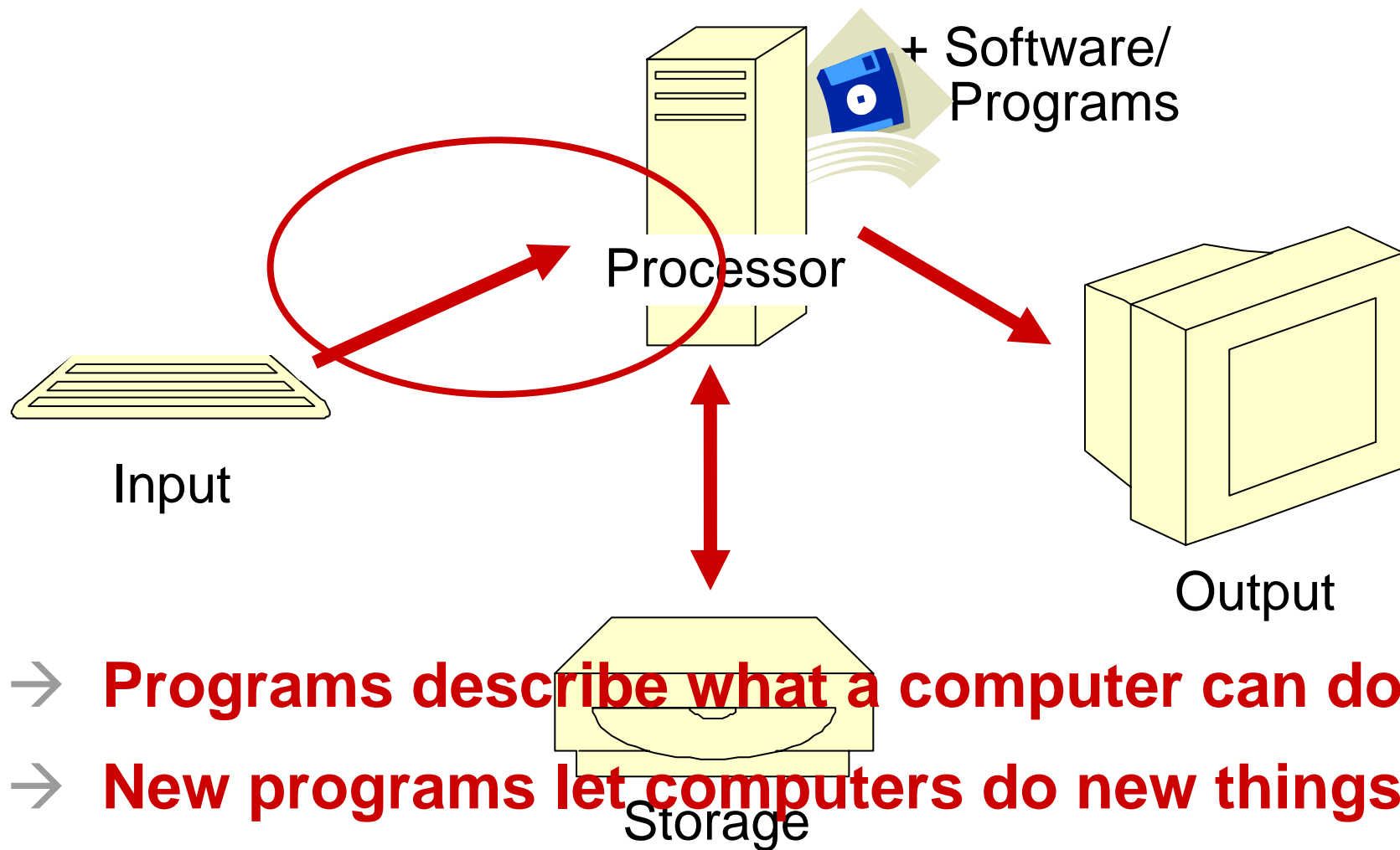
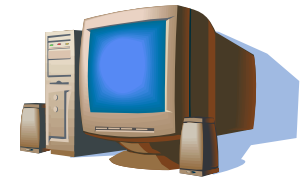
How to Tell Them What to Do?



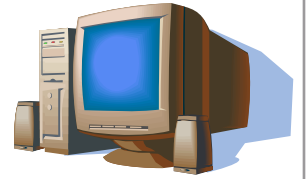
How Computers Work



How Computers Work

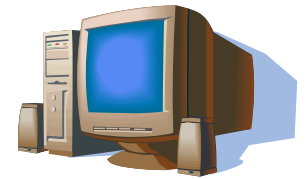


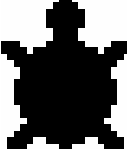

Now **YOU** Can Try!




- Now you can try for yourself what programming is like!
- Use the **Logo** programming language to create shapes on the screen
- This is just *one* example of hundreds of programming languages


Logo – Turtle Programming



- A programming language for simple graphics
- Imagine a turtle  with a pen  on its back

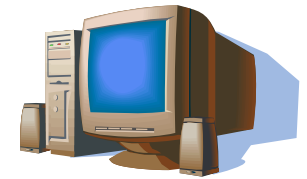
- It can move... 

FD 100

- ... and change it's direction 

FD 100
RT 35
FD 40

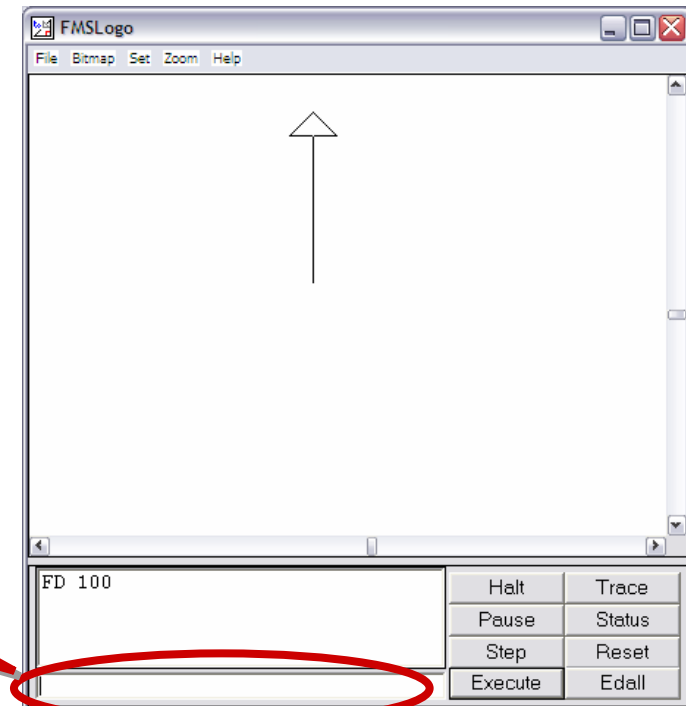
- **All you need to know is on the handouts!**

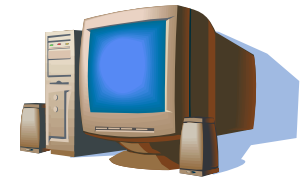


Exercise 1






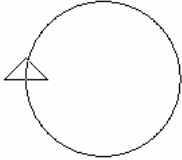
- Read the instructions
- Try to write a program that draws
 - a triangle
 - a square
- Write down the solution!

**Enter your
Program
commands
here!**

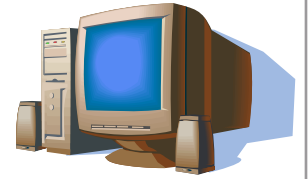




Exercise 2

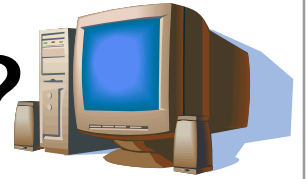
- Use the **REPEAT** *loop* to make your programs simpler
- Write **REPEAT** programs to draw
 - A Triangle 
 - A Rectangle 
 - A Pentagon 
 - A Hexagon 
 - An Octagon 
- Think about the similarities of your programs!
- Try to draw a circle! 

Exercise 3



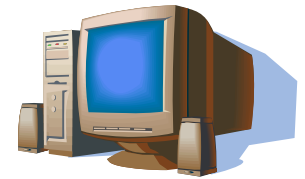
- Your turtle can **LEARN!**
- Use *procedures* to make your turtle **LEARN** more commands!
- Write a TRIANGLE procedure
- Try to understand procedure ARGUMENTS!
- If there's time, experiment with procedures!

So, What's Programming Like?



- Programming can be ...
 - Fun
 - Difficult
 - ...
- Other observations
 - Computers follow instructions,
even if they are wrong!
- Initial and final versions of programs
 - Final versions shorter
 - Final versions easier to understand?

Programming “Abstractions”



- “Tricks” to make programs better ...

Triangle #1

Many Instructions

```
FD 100
RT 120
FD 100
RT 120
FD 100
RT 120
```

Triangle #2

Using a “loop”

```
REPEAT 3 [FD 100 RT 120]
```

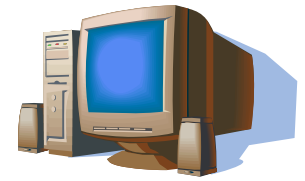
Triangle #3

Using a “procedure”


```
TRIANGLE
```

Which one is the easiest to understand?

Guess-the-Size!

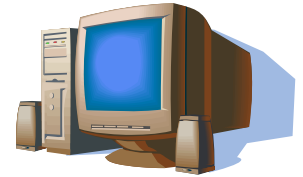


- Your first *Triangle* program had 6 lines
- How many lines do you think these have:

- | | | |
|---------------------------------------|---|-------------------|
| – Cellular phone software (old) |  | 30,000 |
| – Car sunroof control | | 50,000 |
| – <i>The Sims Online</i> game | | 3,000,000 |
| – Commercial airline control software | | 4,000,000 |
| – The Windows™ XP operating system | | 40,000,000 |

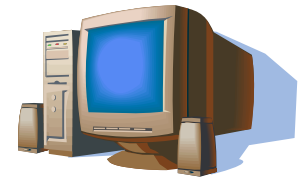
→ It's important to make programs easy to understand!

Guess-the-Size!

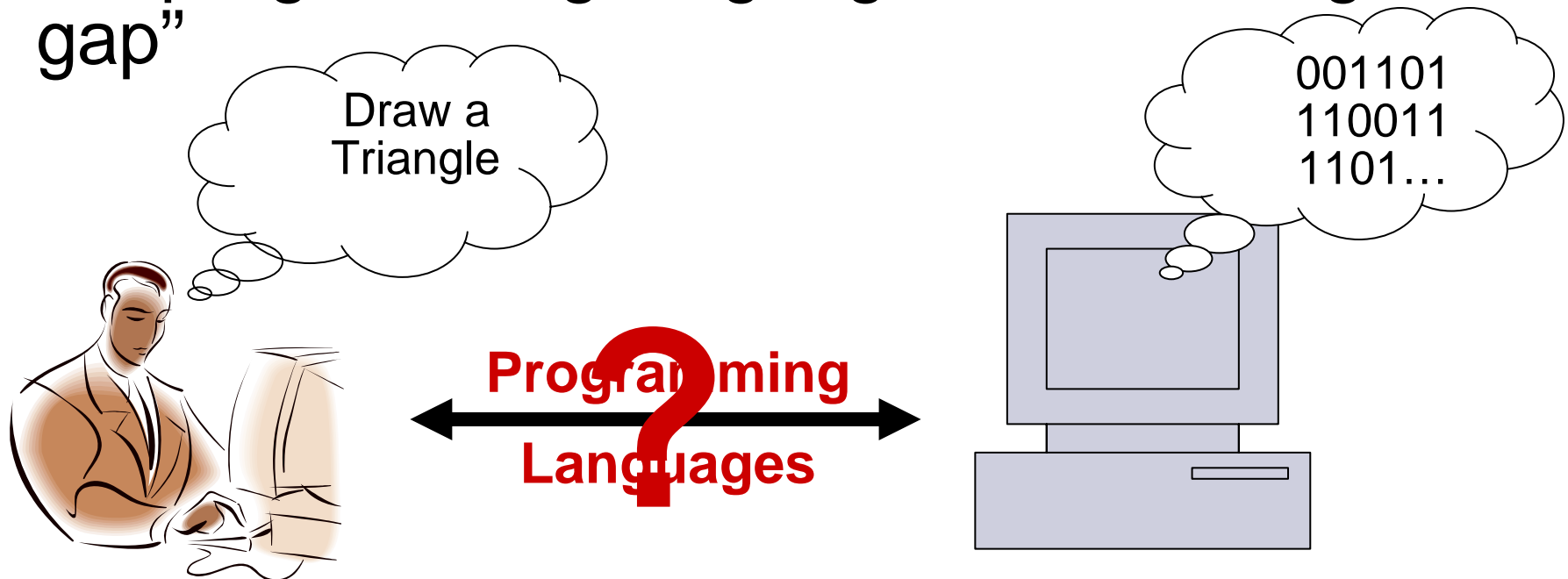


- The Windows™ operating system, for example
 - 40,000,000 lines of program “code”
 - Printed out (2 lines per centimeter), *it would cover more than the distance from here to Tokyo or Nagoya!*
 - For large programs, how do you...
 - Make sure it’s correct?
 - Make sure it does all it’s supposed to do?
- *It’s important to make programs easy to understand!***

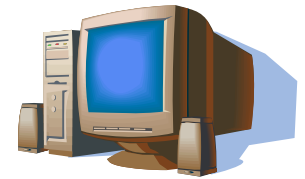
Abstractions in Programming



- To make programs easy to understand, programming languages use *abstractions*
- Computers only understand “on” and “off”, so the programming language has to “bridge the gap”



Abstractions in Programming



???

???

This is what we are working on!

object-oriented:
ca. 1980

MyTriangle.setSize(); MyTriangle.draw()

procedural:
ca. 1960

TRIANGLE

imperative:

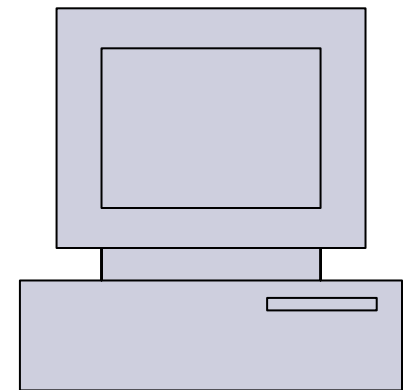
FD 100 RT 120 FD 100 ...

“assembly”:
ca. 1950

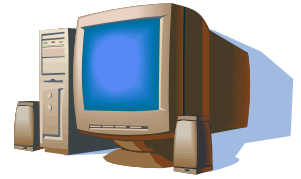
reg 2 add 4 push 5 ...

basic instructions:

0111011011101000010... →

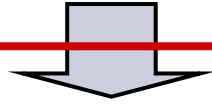


Abstractions in Programming



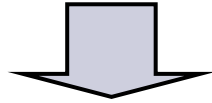
aspect-oriented?

MyTriangle; MySquare; DrawAspect



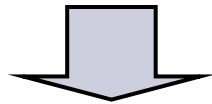
object-oriented:

MyTriangle.setSize(); MyTriangle.draw()



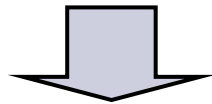
procedural:

TRIANGLE



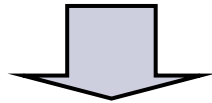
imperative:

FD 100 RT 120 FD 100 ...



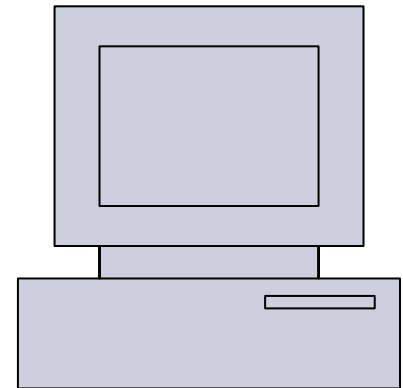
“assembly”:

reg 2 add 4 push 5 ...

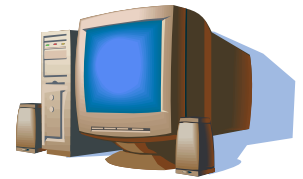


basic instructions:

0111011011101000010...

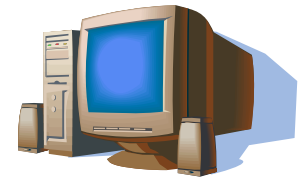


Summary



- Computers do what they are told
- Computers only understand “on” and “off”
- To tell a computer do what we want, we need a **programming language!**
 - To translate our ideas into computer instructions
- The better the language, the easier it is to describe what we want
- The better the language, the easier it is to understand what we have written

Thank You!



Any Questions?

