Lecture 11: Interfaces

Midterm rules and advice ...

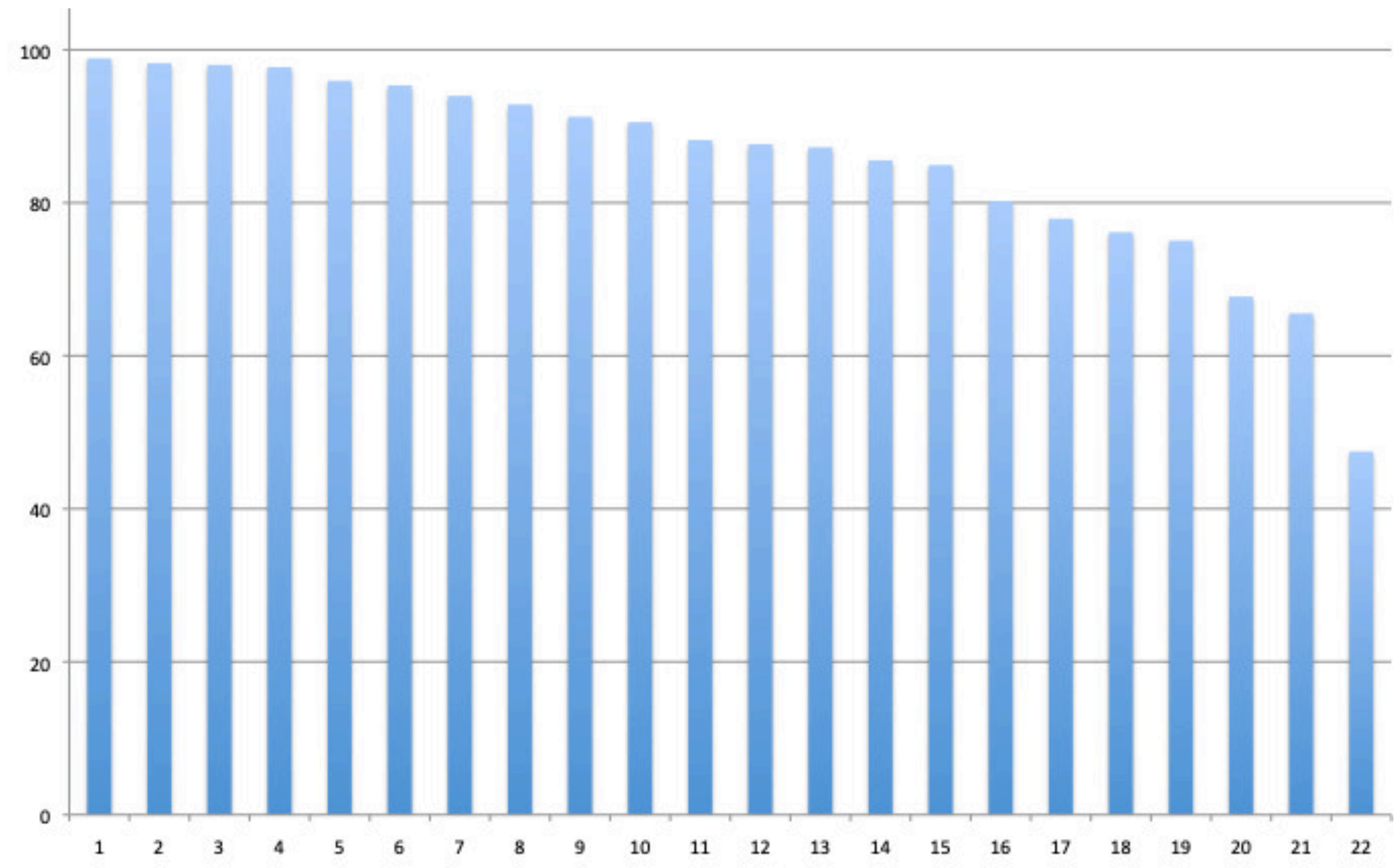
- community auditors welcome; I will grade you too
- open book: notes, text, old exams, etc., all ok
- 90 minutes in a single sitting

- read the rules (posted on web page)

- scan and return as soon as possible after completion
- by 9 PM Friday at the latest – no exceptions, no extensions

- I'm trying to see if you understand; it's not meant to be tricky
- think straightforwardly; don't deconstruct; think about course topics
- if you're writing or computing a lot, you're on the wrong track
- know the powers of 2, powers of 10 and hex digits
- don't make careless arithmetic errors

Problem sets 1-4 (cumulative, out of 100)



Interfaces

- In computing, an interface is a shared boundary across which two or more separate components of a computer system exchange information. The exchange can be between software, computer hardware, peripheral devices, humans and combinations of these.
  - (Wikipedia, the source of all truth)

- there has to be agreement about what information is exchanged and how

- lots of technical issues
- surprisingly, some important legal issues

Reprise: what an operating system does

- **manages CPU(s), schedules and coordinates running programs**
  - switches CPU among programs that are actually computing
  - suspends programs that are waiting for something (e.g., disk, network)
  - keeps individual programs from hogging resources
- **manages memory (RAM)**
  - loads programs in memory so they can run
  - swaps them to disk and back if there isn't enough RAM (virtual memory)
  - keeps separate programs from interfering with each other
  - and with the operating system itself (protection)
- **manages and coordinates input/output to devices**
  - disks, display, keyboard, mouse, buses, network, ...
  - provides fairly uniform interface to disparate devices
- **manages files on disk (file system)**
  - provides hierarchy of folders/directories and files for storing information

How applications use the operating system

- **operating system provides services to be accessed by application programs**
  - Unix "system calls", Windows Application Programming Interface ("API")

    "what is the exact time?"

    "allocate M more bytes of RAM to me"

    "read N bytes from file F into memory starting at location M"

    "write N bytes from memory locations starting at M into file F"

    "set up a network connection to www.princeton.edu"

    "write N bytes to the network connection"

    “I’m all done; get rid of me”
- **operating system provides an interface for applications to use**
  - programs access machine capabilities only through this interface
  - different physical hardware can provide the same interface
  - programs can be moved to any system that provides the same interface
  - different operating systems can provide the same interface
  - one operating system can simulate the interface provided by another
- **operating system hides details of specific hardware**

Example of system-call level coding

- C program to copy input to output ("copy" command)
- read, write, exit are system calls

```
main() {
  char buf[8192];
  int n;
  while ((n = read(0, buf, sizeof(buf))) > 0)
    write(1, buf, n);
  exit(0);
}
```

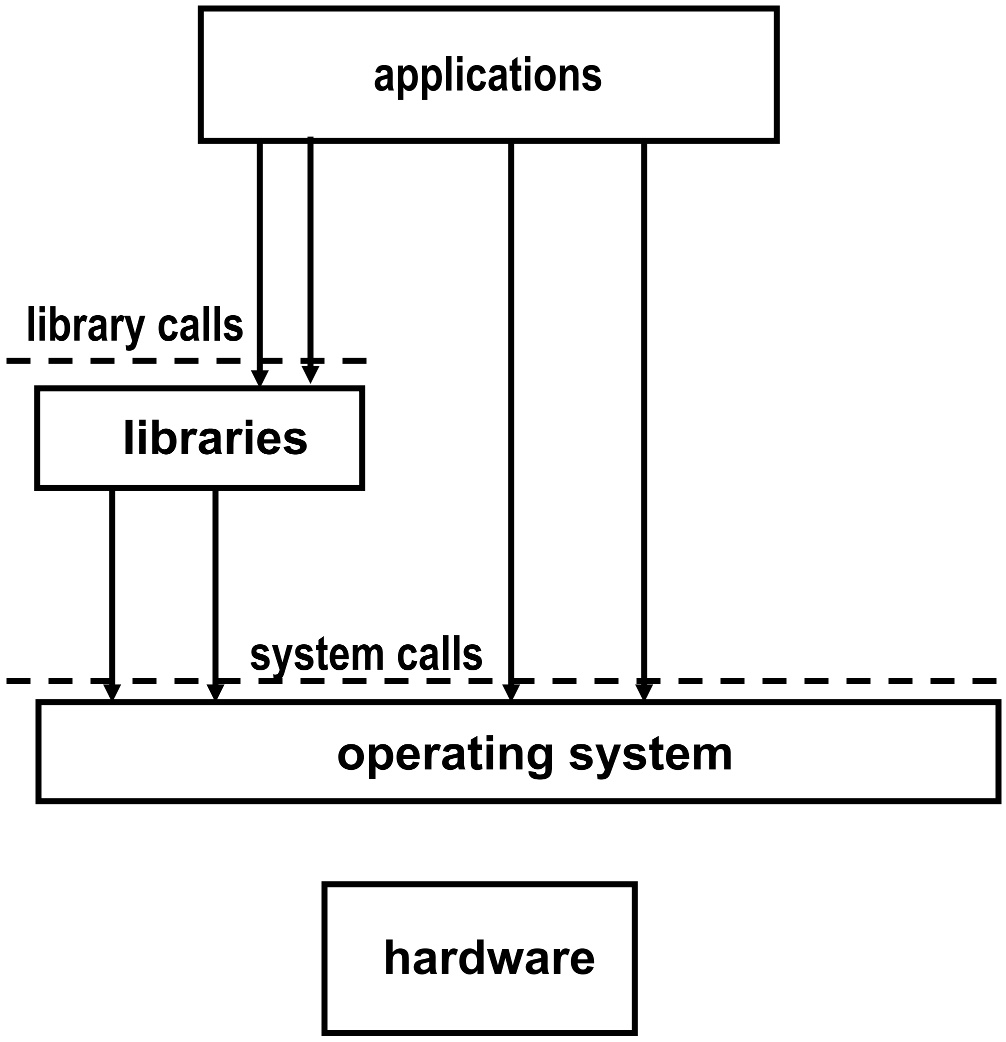
Software is organized into "layers"

- **each layer presents an interface that higher layers can use**
  - defines a "platform" for putting more on top
  - insulates the higher layer from how the lower layer is implemented
  - often called "Application Programming Interface" or API
- **operating system ("kernel")**
  - lowest software layer, on top of hardware

    (usually: virtual machine is on top of another program, e.g., an operating system)
  - presents its capabilities as system calls
- **libraries**
  - code to be used as building blocks in programs
  - present their capabilities as APIs
- **applications**
  - e.g., browser, word processor, mailer, compiler, directory lister, ...
  - use libraries and system calls through APIs

Layering

- an application generally calls multiple libraries
  - might not make direct system calls
- a library generally calls other libraries
- library and system call levels define interfaces (APIs)
- programmers may not know what is "library" and what is "system call"



What's an API?

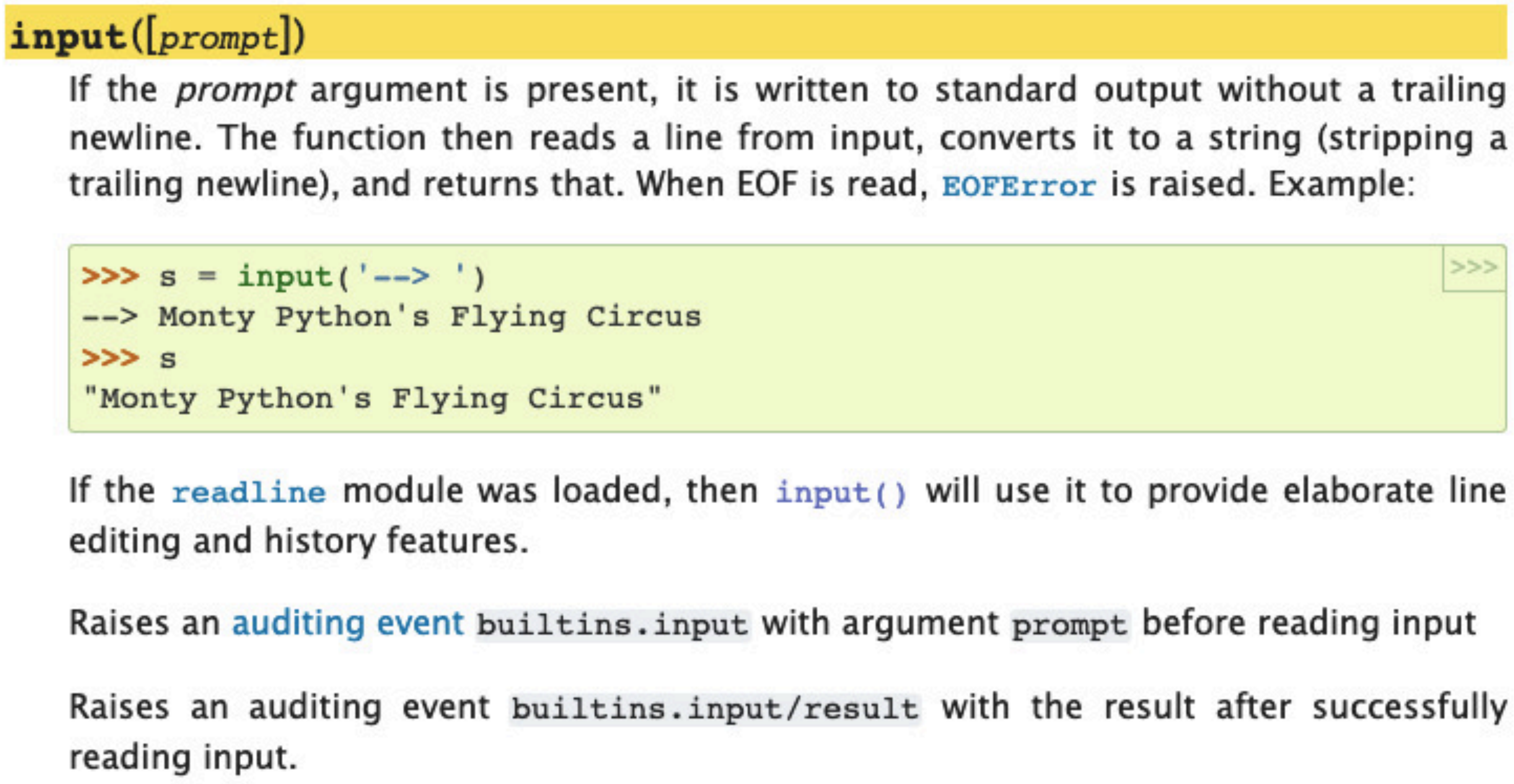
Operating systems perform many functions, including allocating computer memory and controlling peripherals such as printers and keyboards. Operating systems also function as platforms for software applications. They do this by "exposing" — i.e., making available to software developers — routines or protocols that perform certain widely-used functions. These are known as Application Programming Interfaces, or "APIs."

Excerpted from Final Judgment

State of New York, et al v. Microsoft Corporation

US District Court, District of Columbia, Nov 1, 2002

Sample Python API



`input``([`*`prompt`*`])`

If the *prompt* argument is present, it is written to standard output without a trailing newline. The function then reads a line from input, converts it to a string (stripping a trailing newline), and returns that. When EOF is read, `EOFError` is raised. Example:

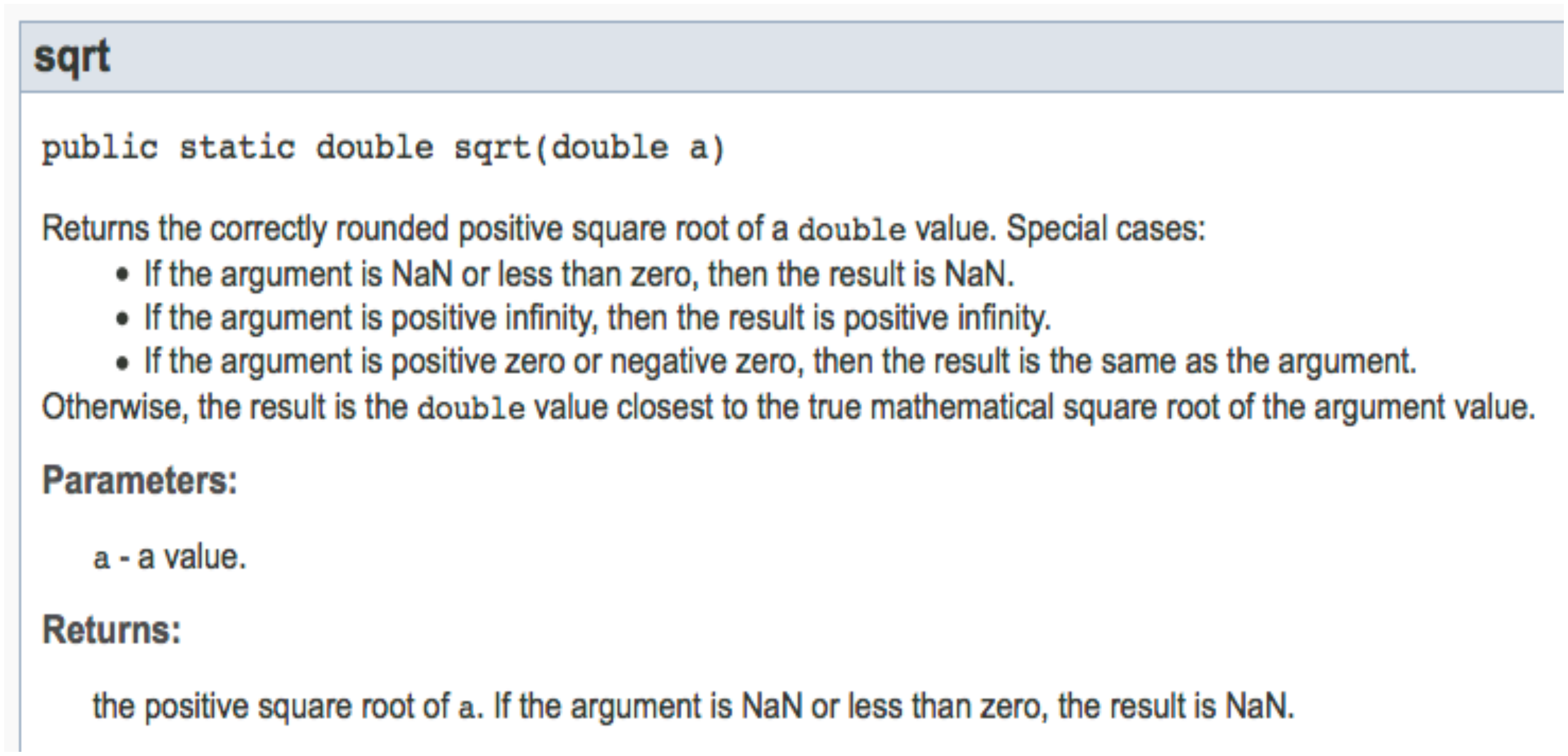
```
>>> s = input('--> ')
--> Monty Python's Flying Circus
>>> s
"Monty Python's Flying Circus"
```

If the `readline` module was loaded, then `input()` will use it to provide elaborate line editing and history features.

Raises an auditing event `builtins.input` with argument `prompt` before reading input

Raises an auditing event `builtins.input/result` with the result after successfully reading input.

Sample Java API (tiny excerpt)



sqrt

```
public static double sqrt(double a)
```

Returns the correctly rounded positive square root of a `double` value. Special cases:

- If the argument is NaN or less than zero, then the result is NaN.
- If the argument is positive infinity, then the result is positive infinity.
- If the argument is positive zero or negative zero, then the result is the same as the argument.

Otherwise, the result is the `double` value closest to the true mathematical square root of the argument value.

Parameters:

`a` - a value.

Returns:

the positive square root of `a`. If the argument is NaN or less than zero, the result is NaN.

Independent implementations of an interface

- can interfaces be owned?

- company A sells something (hardware or software)
- company A publishes (widely) the API for programming it
  - with the intent that third parties will develop applications for the thing
  - and thus make it more attractive so company A will sell more

- company B uses A's interface definition to make a cheaper version of the thing that works the same
  - so all the third-party applications will run on B's cheaper version
  - thus cutting into A's market

- company A sues company B for copying A's interface

- who should win?

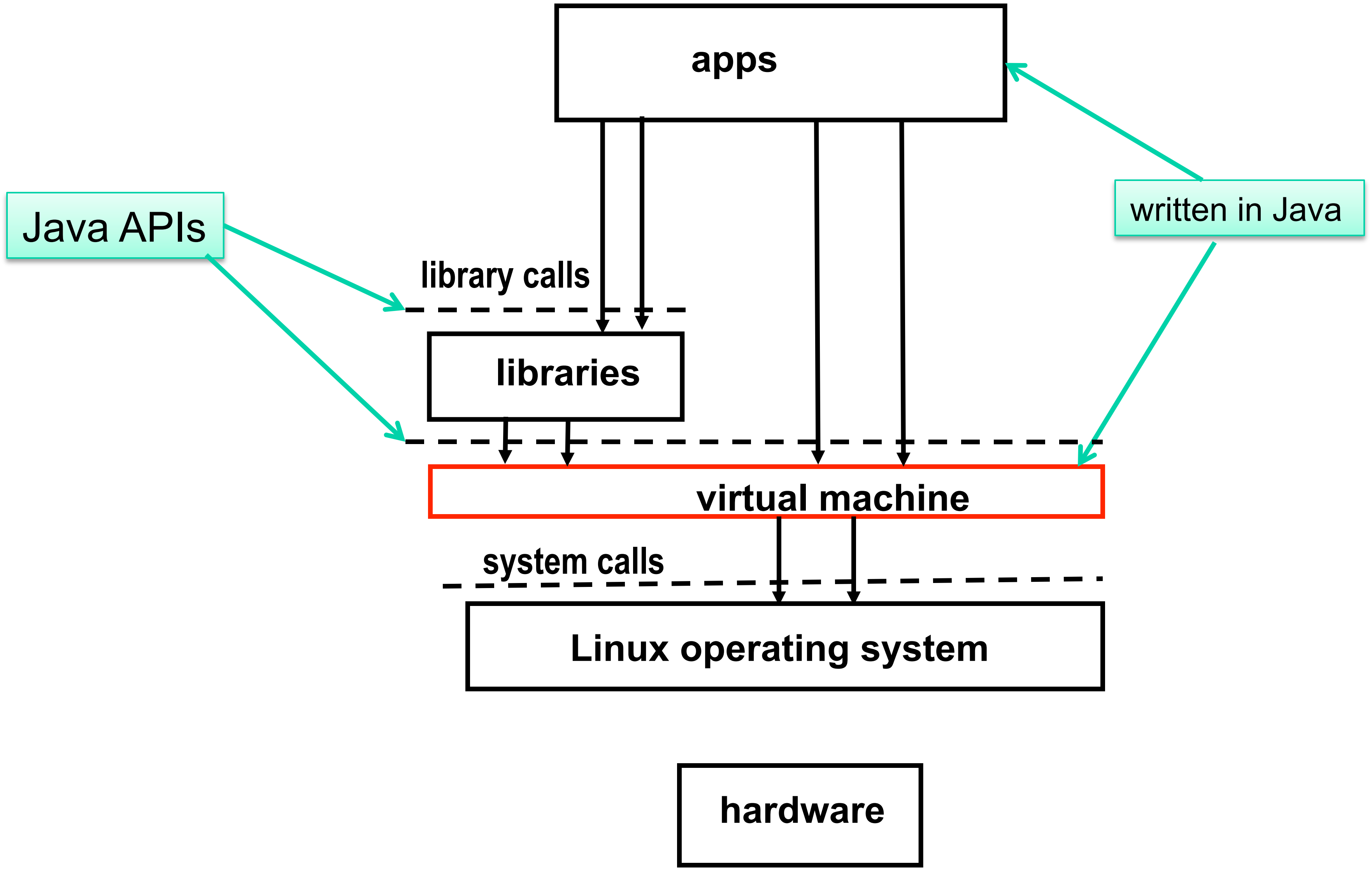
Oracle v Google

- August 2010: Oracle sues Google for use of Java API
  - partly patent, partly copyright
  - patent part: jury said “non-infringing”, eventually thrown out

- remaining copyright part mostly about Java APIs:

  did Google violate Oracle’s copyright by re-using the Java APIs verbatim?
  - re-implementing the code behind them was not at issue

Android phone organization



Oracle v Google (from the decision in May, 2012)

The Java language, like C and C++, is a human-readable language. Code written in a human-readable language — “source code” — is not readable by computer hardware.

Only “object code,” which is not human-readable, can be used by computers. Most object code is in a binary language, meaning it consists entirely of 0s and 1s. Thus, a computer program has to be converted, that is, compiled, from source code into object code before it can run, or “execute". In the Java system, source code is first converted into “bytecode,” an intermediate form, before it is then converted into binary machine code by the Java virtual machine.

RangeCheck

```
private static void rangeCheck(int arrayLen,
                    int fromIndex, int toIndex) {
  if (fromIndex > toIndex)
     throw new IllegalArgumentException("fromIndex(″
          + fromIndex + ") > toIndex(" + toIndex+")");
  if (fromIndex < 0)
     throw new ArrayIndexOutOfBoundsException(fromIndex);
  if (toIndex > arrayLen)
     throw new ArrayIndexOutOfBoundsException(toIndex);
}
```

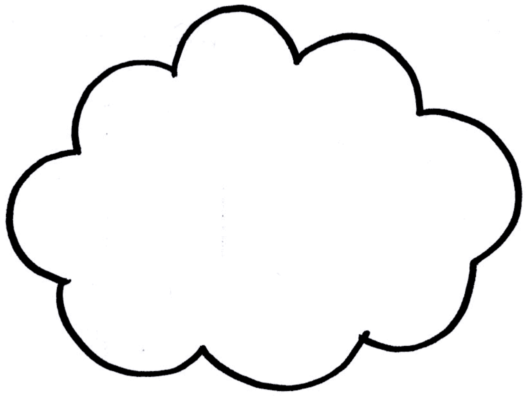
RangeCheck (simpler version, in Python)

```
def rangeCheck(len, from, to):
  if from > to or from < 0 or to > len:
     return 0
  else:
     return 1
```

Cloud computing APIs

- 'Cloud' has been a go-to metaphor for almost as long as the Internet has existed, conveying a sense that the Internet was intangible and bigger than the sum of its parts."

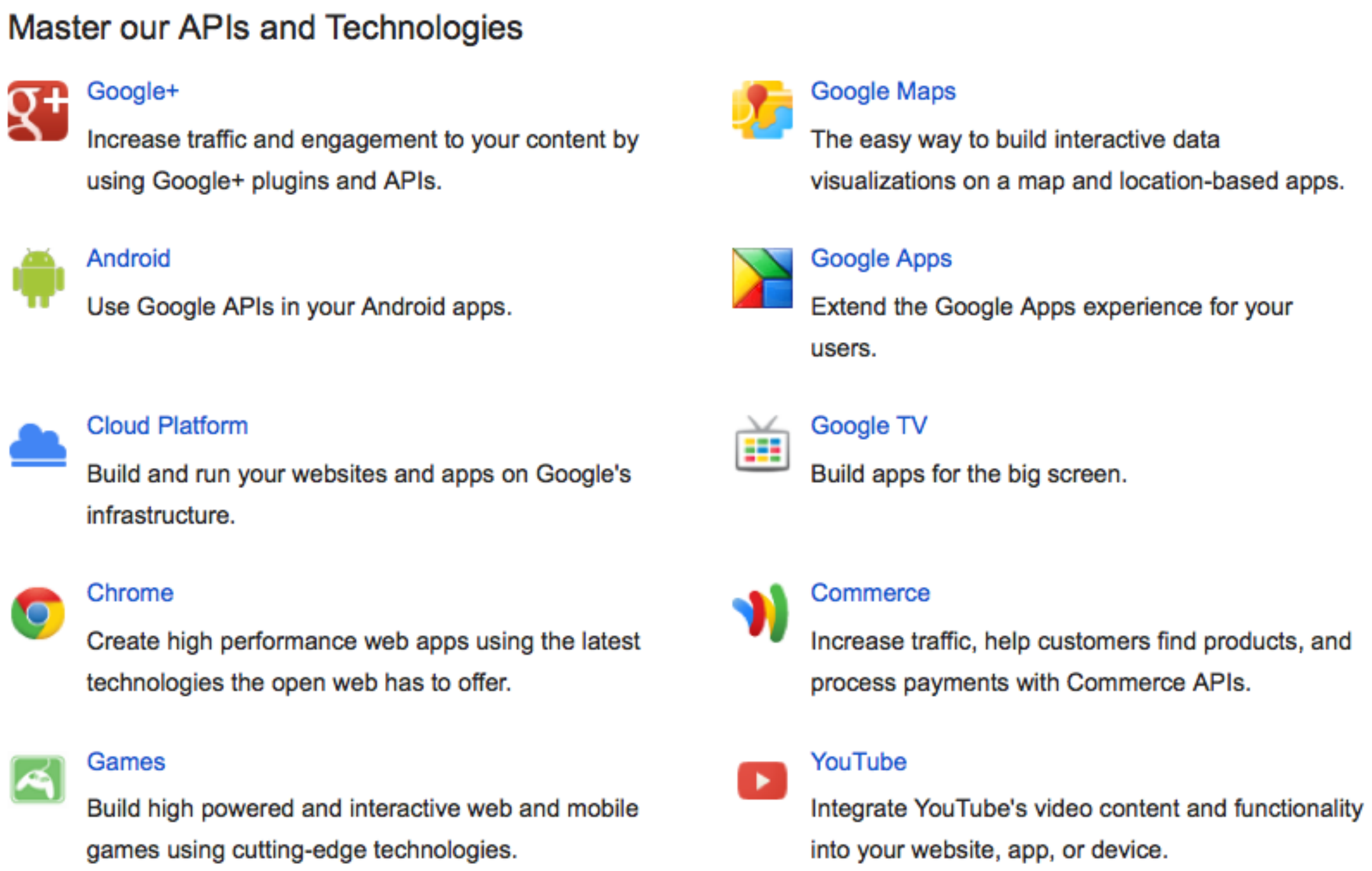
  (Wall Street Journal, 9/23/08)



- software services delivered via the Internet
  - Gmail, ...
  - Facebook, Twitter, Instagram, …
  - Google Docs, calendar,
  - Windows Live, Office 360
  - Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform

- most cloud services have an API for access by programs

Google APIs



Master our APIs and Technologies

Google+
Increase traffic and engagement to your content by using Google+ plugins and APIs.

Google Maps
The easy way to build interactive data visualizations on a map and location-based apps.

Android
Use Google APIs in your Android apps.

Google Apps
Extend the Google Apps experience for your users.

Cloud Platform
Build and run your websites and apps on Google's infrastructure.

Google TV
Build apps for the big screen.

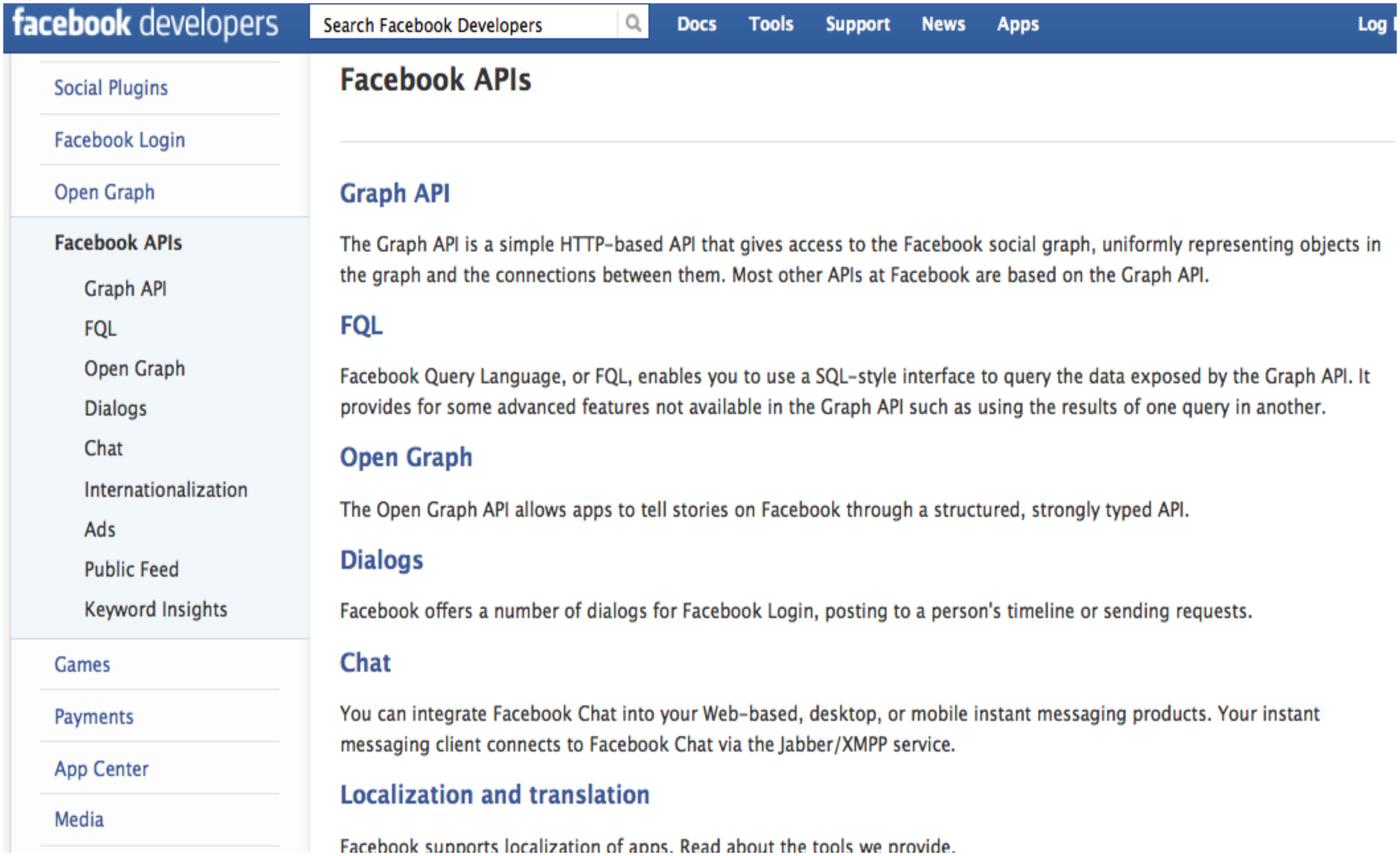
Chrome
Create high performance web apps using the latest technologies the open web has to offer.

Commerce
Increase traffic, help customers find products, and process payments with Commerce APIs.

Games
Build high powered and interactive web and mobile games using cutting-edge technologies.

YouTube
Integrate YouTube's video content and functionality into your website, app, or device.

Facebook APIs



Facebook APIs

Graph API

The Graph API is a simple HTTP-based API that gives access to the Facebook social graph, uniformly representing objects in the graph and the connections between them. Most other APIs at Facebook are based on the Graph API.

FQL

Facebook Query Language, or FQL, enables you to use a SQL-style interface to query the data exposed by the Graph API. It provides for some advanced features not available in the Graph API such as using the results of one query in another.

Open Graph

The Open Graph API allows apps to tell stories on Facebook through a structured, strongly typed API.

Dialogs

Facebook offers a number of dialogs for Facebook Login, posting to a person's timeline or sending requests.

Chat

You can integrate Facebook Chat into your Web-based, desktop, or mobile instant messaging products. Your instant messaging client connects to Facebook Chat via the Jabber/XMPP service.

Localization and translation

Facebook supports localization of apps. Read about the tools we provide.