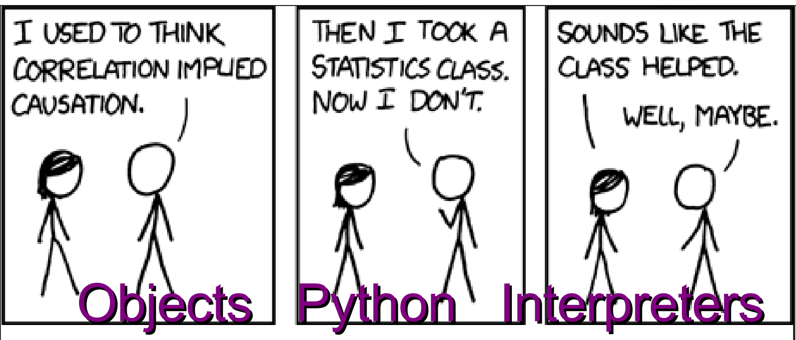
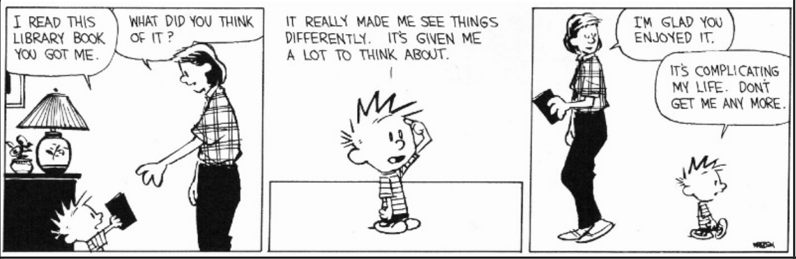

I USED TO THINK CORRELATION IMPLIED CAUSATION.
THEN I TOOK A STATISTICS CLASS. NOW I DON'T.
SOUNDS LIKE THE CLASS HELPED.
WELL, MAYBE.

## Objects Python Interpreters

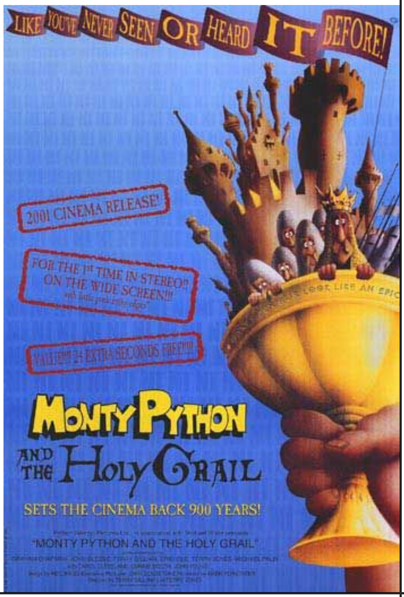

I READ THIS LIBRARY BOOK YOU GOT ME.
WHAT DID YOU THINK OF IT?
IT REALLY MADE ME SEE THINGS DIFFERENTLY. IT'S GIVEN ME A LOT TO THINK ABOUT.
I'M GLAD YOU ENJOYED IT.
IT'S COMPLICATING MY LIFE. DON'T GET ME ANY MORE.

## One-Slide Summary

- **Object-Oriented Programming** encapsulates state and methods together into objects. This hides implementation details (cf. inheritance) while allowing methods to operate on many types of input.
- **Learning new languages** has pragmatic value, expands our minds, deepens our understanding, builds confidence, and may even be fun.
- **Python** is a universal, imperative, object-oriented language.
- Building an **interpreter** is a fundamental idea in computing. Eval and Apply are **mutually recursive**.

## Outline

- Object Lessons
- Why Python?
- Ali G
- Interpreters

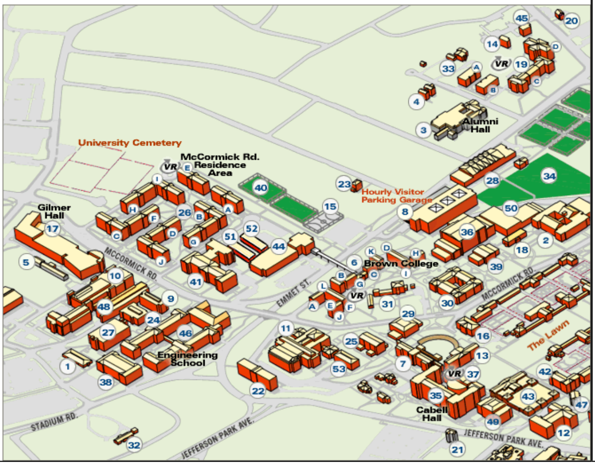

LIKE YOU'VE NEVER SEEN OR HEARD IT BEFORE!
2001 CINEMA RELEASE
FOR THE 1st TIME IN STEREO ON THE WIDE SCREEN!!!
MONTY PYTHON AND THE HOLY GRAIL
SETS THE CINEMA BACK 900 YEARS!

## Exam 2 Extra Credit

- 2 points on Exam 2: attend talk Thursday (tomorrow) in GILMER 190 from 3:30 to 4:30pm
- Scott Aaronson from MIT:
  - In the popular imagination, quantum computers would be almost magical devices, able to "solve impossible problems in an instant" by trying exponentially many solutions in parallel. In this talk, I'll describe various results in quantum computing theory that directly challenge this view. For example, at least in the "black-box model" that we know how to analyze, quantum computers would need exponential time to break cryptographic hash functions or find local optima, just as classical computers would. As time permits, I'll also describe how studying the limitations of quantum computers can lead to new insights even into classical computation.

## Where is Gilmer 190?

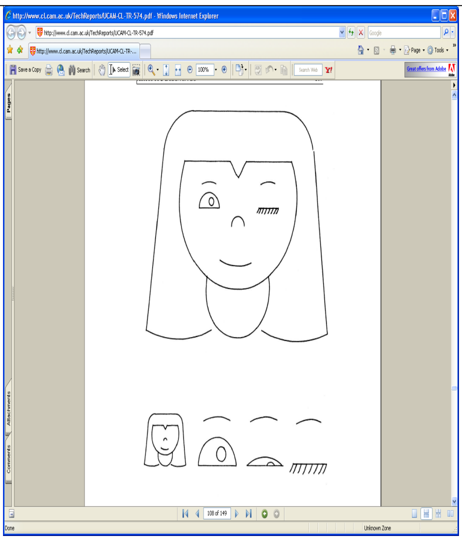
- (17)
  - upper
  - left
- We are in (27) now



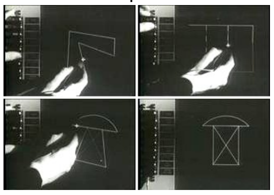
## BACS Info Session

- Thursday (tomorrow) 5-6pm Olsson 236D
  - Enter by the front steps: there will be signs up
- For students curious about the BA in CS
  - The Minor
  - The Major
  - Distinguished Majors Program
  - Required courses, etc.
- Can't make it but are interested?
  - Email Tom Horton (horton@cs.virginia.edu)

## Components in Sketchpad


http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-574.pdf - Windows Internet Explorer

Actual Sketchpad:



## Objects in Sketchpad

In the process of making the Sketchpad system operate, a few very general functions were developed which make no reference at all to the specific types of entities on which they operate. These general functions give the Sketchpad system the ability to operate on a wide range of problems. The motivation for making the functions as general as possible came from the desire to get as much result as possible from the programming effort involved. For example, the general function for expanding instances makes it possible for Sketchpad to handle *any* fixed geometry subpicture. The rewards that come from implementing general functions are so great that the author has become reluctant to write any programs for specific jobs.

Each of the general functions implemented in the Sketchpad system abstracts, in some sense, some common property of pictures independent of the specific subject matter of the pictures themselves.

Ivan Sutherland, *Sketchpad: a Man-Machine Graphical Communication System*, 1963 (major influence on Alan Kay developing OOP in 1970s)

## Simula

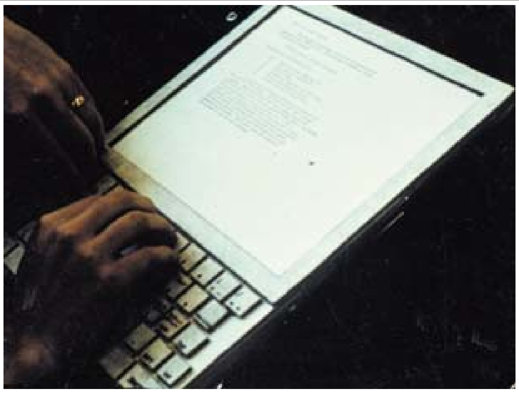
- Considered the first "object-oriented" programming language
- Language designed for *simula*tion by Kristen Nygaard and Ole-Johan Dahl (Norway, 1962)
- Had special syntax for defining classes that package state and procedures together

## Counter in Simula

```
class counter;
  integer count;
  begin
    procedure reset(); count := 0; end;
    procedure next();
         count := count + 1; end;
    integer procedure current();
        current := count; end;
  end
```

## XEROX Palo Alto Research Center (PARC)

- 1970s:
- Bitmapped display
- Graphical User Interface
  - Steve Jobs paid $1M to visit PARC (bought their stock), and returned to make Apple Lisa/Mac
- Ethernet
- First personal computer (Alto)
- PostScript Laser Printers
- **Object-Oriented Programming**



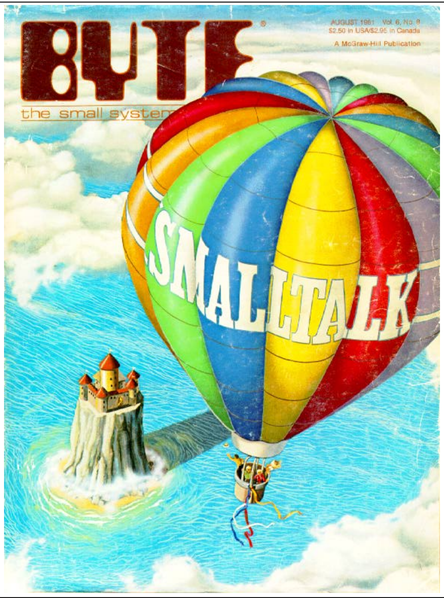
## Dynabook, 1972

(Just a model)

*"Don't worry about what anybody else is going to do… The best way to predict the future is to invent it. Really smart people with reasonable funding can do just about anything that doesn't violate too many of Newton's Laws!"*
*— Alan Kay, 1971*

## Dynabook 1972

- Tablet computer
- Intended as tool for learning
- Kay wanted children to program it also
- Hallway argument, Kay claims you could define "the most powerful language in the world in a page of code"
- Proof: Smalltalk
  - Scheme is as powerful, but takes two pages
  - Before the end of CS 150, we will see an equally powerful language that fits in ¼ page


BYTE
the small systems
AUGUST 1981 Vol. 6, No. 8
$2.50 in USA/$2.95 in Canada
A McGraw-Hill Publication
SMALLTALK

BYTE Magazine, August 1981

## Smalltalk

- Everything is an *object*
- Objects communicate by sending and receiving *messages*
- Objects have their own state (which may contain other objects)
- How do you do 3 + 4?

send the object **3** the message **"+ 4"**

## Counter in Smalltalk

**class name** counter
  **instance variable names** count
  new count <- 0
  next count <- count + 1
  current ^ count

## Counter in Python

```
class counter:
  def __init__(self): self._count = 0
  def reset(self): self._count = 0
  def next(self): self._count = self._count + 1
  def current(self): return self._count
```

counter() creates a new counter using the __init__ method
_count is the instance variable (_ is just a naming convention)

## Who was the first object-oriented programmer?

## First Object-Oriented Programmer?

By the word operation, we mean any process which alters the mutual relation of two or more things, be this relation of what kind it may. This is the most general definition, and would include all subjects in the universe. Again, it might act upon other things besides number, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations, and which should be also susceptible of adaptations to the action of the operating notation and mechanism of the engine... Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent.

Ada, Countess of Lovelace, around 1843

## Liberal Arts Trivia: Biology

- This family of non-venomous serpents contains the longest snake in the world. They have teeth, heat-sensing organs, and ambush prey. They kill by a process of constriction: sufficient pressure is applied to the prey to prevent it from inhaling, and the prey succumbs to asphyxiation and is swallowed whole.

## Liberal Arts Trivia: Chemistry

- This element is a ductile metal with very high thermal and electrical conductivity. When pure and fresh it has a pinkish or peachy color, but it turns green with age (oxidation). It has played a significant role in the history of humanity. In the Roman era it was usually mined on Cyprus; hence the provenance of its modern name (Cyprium to Cuprum).

## Implementing Interpreters

## Why learn Python?

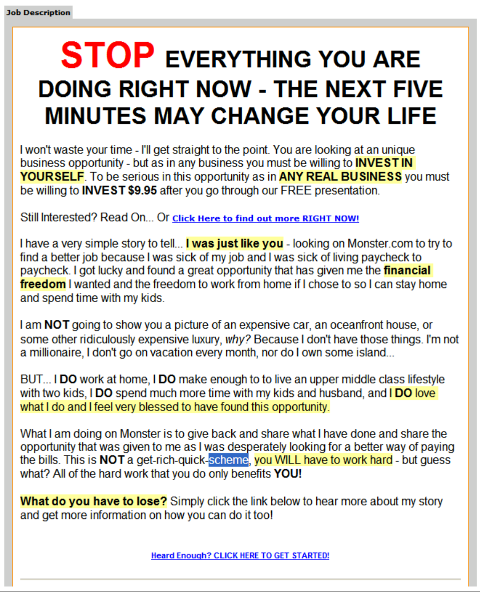
## Reason 1: Vocational Skill

Job listings at monster.com in Virginia (27 March 2007, postings in last 3 months):

| | | |
|---|---|---|
| Python | 27 | $40-200K |
| Java | 770 | $35-200K |
| SQL | 1138 | $60-400K |
| Scheme | 55 | $100-999K |

PS5, PS8 & 9

## "Scheme" Jobs



Job Description

**STOP EVERYTHING YOU ARE DOING RIGHT NOW - THE NEXT FIVE MINUTES MAY CHANGE YOUR LIFE**

I won't waste your time - I'll get straight to the point. You are looking at an unique business opportunity - but as in any business you must be willing to **INVEST IN YOURSELF**. To be serious in this opportunity as in **ANY REAL BUSINESS** you must be willing to **INVEST $9.95** after you go through our FREE presentation.

Still Interested? Read On... Or Click Here to find out more RIGHT NOW!

I have a very simple story to tell... **I was just like you** - looking on Monster.com to try to find a better job because I was sick of my job and I was sick of living paycheck to paycheck. I got lucky and found a great opportunity that has given me the **financial freedom** I wanted and the freedom to work from home if I chose to so I can stay home and spend time with my kids.

I am **NOT** going to show you a picture of an expensive car, an oceanfront house, or some other ridiculously expensive luxury, *why?* Because I don't have those things. I'm not a millionaire, I don't go on vacation every month, nor do I own some island...

BUT... I **DO** work at home, I **DO** make enough to to live an upper middle class lifestyle with two kids, I **DO** spend much more time with my kids and husband, and I **DO** love what I do and I feel very blessed to have found this opportunity.

What I am doing on Monster is to give back and share what I have done and share the opportunity that was given to me as I was desperately looking for a better way of paying the bills. This is **NOT** a get-rich-quick-scheme, you WILL have to work hard - but guess what? All of the hard work that you do only benefits **YOU!**

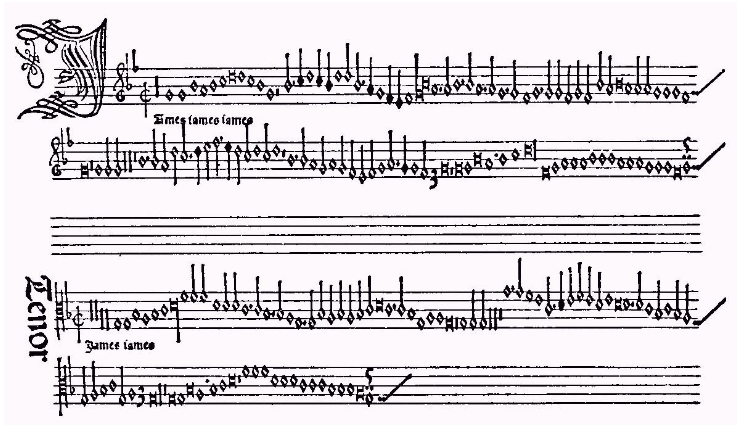
**What do you have to lose?** Simply click the link below to hear more about my story and get more information on how you can do it too!

Heard Enough? CLICK HERE TO GET STARTED!

## Reason 2: Expanding Minds

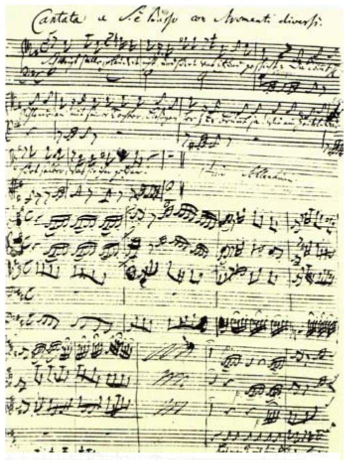
Languages change the way we think.

The more languages you know, the more different ways you have of thinking about (and solving) problems.



"Jamais Jamais Jamais" from *Harmonice Musices Odhecaton A*. Printed by Ottaviano Dei Petrucci in 1501 (first music with movable type)

"Jamais Jamais Jamais" from *Harmonice Musices Odhecaton A.* (1501)



J S Bach, "Coffee Cantata", BWV 211 (1732)
www.npj.com/homepage/teritowe/jsbhand.html

## Reason 3: Deepening Understanding

By seeing how the same concepts we encountered in Scheme are implemented by a different language, you will understand those concepts better (especially classes/objects, assignment, data abstraction).

## Reason 4: Building Confidence

By learning Python (mostly) on your own, the next time you encounter a problem that is best solved using a language you don't know, you will be confident you can learn it (rather than trying to use the wrong tool to solve the problem).

## Reason 5: Fun

Programming in Python is fun.

Especially because:
- It is an elegant and simple language
- Most programs mean what you think they mean
- It is dynamic and interactive
- It can be used to build web applications (PS8, PS9)
- It is named after *Monty Python's Flying Circus*
- It was designed by someone named Guido.

## Python

- A **universal** programming language
  - Everything you can compute in Scheme you can compute in Python, and vice versa
- **Imperative** Language
  - Designed to support a programming where most of the work is done using **assignment statements**: **x = e**
  - Means same thing as **(set! x e)**
- **Object-Oriented** Language
  - Every data thing is an object
  - Built in support for classes, inheritance

## Learning New Languages

- **Syntax**: Where the {, ;, $, etc. all go
  - If you can understand a BNF grammar, this is easy
- **Semantics**: What does it mean
  - Learning the evaluation rules
  - Harder, but most programming languages have very similar evaluation rules
- **Style**
  - What are the idioms and customs of experienced programmers in that language?
    - Takes many years to learn
    - Need it to be a "professional" Python programmer, but not to make a useful program

## Python If

*Instruction* ::= **if (***Expression***) :**
*Block*

Evaluate *Expression*. If it evaluates to true, evaluate the *Block*.

It is similar to (if *Expression* (begin *Statements*))

**Differences**:

Indenting and new lines matter!

Changing the indentation changes meaning of code

What "true" means:

Scheme: anything that is not **#f**.

Python: anything that is not **False**, **None**, **0**, and empty string or container

## Computability in Theory and Practice

(Intellectual Computability Discussion on TV Video)
(I hope this works!)

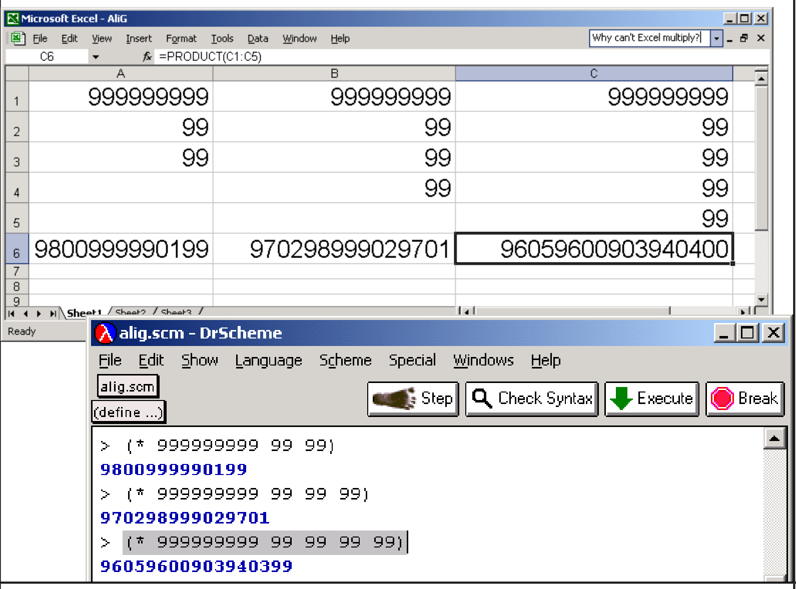
## Ali G Problem

- **Input:** a list of 2 numbers with up to $d$ digits each
- **Output:** the product of the 2 numbers

Is it computable?

Yes – a straightforward algorithm solves it. Using elementary multiplication techniques it is $O(d^2)$

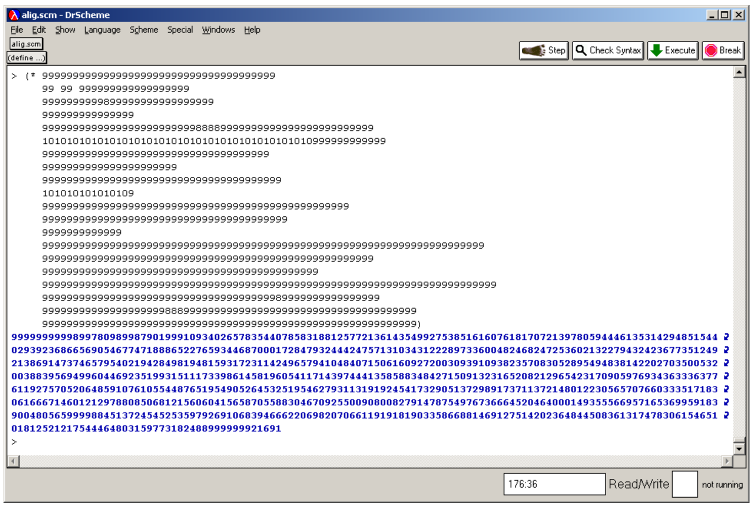
Can *real* computers solve it?


Microsoft Excel - AliG
File Edit View Insert Format Tools Data Window Help
Why can't Excel multiply?
Sheet1 / Sheet2 / Sheet3
Ready
alig.scm - DrScheme
File Edit Show Language Scheme Special Windows Help
alig.scm
(define ...)
Step
Check Syntax
Execute
Break

C6 =PRODUCT(C1:C5)

|   | A | B | C |
|---|---|---|---|
| 1 | 999999999 | 999999999 | 999999999 |
| 2 | 99 | 99 | 99 |
| 3 | 99 | 99 | 99 |
| 4 |  | 99 | 99 |
| 5 |  |  | 99 |
| 6 | 9800999990199 | 970298999029701 | 96059600903940400 |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |

```
> (* 999999999 99 99)
9800999990199
> (* 999999999 99 99 99)
970298999029701
> (* 999999999 99 99 99 99)
96059600903940399
```



```
> (* 99999999999999999999999999999999999999
     99 99 999999999999999999
     9999999998999999999999999999
     999999999999999
     999999999999999999999999988899999999999999999999999999
     10101010101010101010101010101010101010101010999999999999
     9999999999999999999999999999999999999
     9999999999999999999999
     999999999999999999999999999999999999999
     101010101010109
     99999999999999999999999999999999999999999999999999
     9999999999999999999999999999999999999999
     9999999999999
     999999999999999999999999999999999999999999999999999999999999999999999999
     999999999999999999999999999999999999999999999999999999
     999999999999999999999999999999999999999999999
     99999999999999999999999999999999999999999999999999999999999999999999999999
     9999999999999999999999999999999999999999899999999999999999
     9999999999999999999988899999999999999999999999999999999999999
     9999999999999999999999999999999999999999999999999999999999999)
99999999998997809899879019991093402657835440785831881257721361435499275385161607618170721397805944461353142948515 44
0293923686656905467747188865227659344687000172847932444247571310343122289733600482468247253602132279432423677351249
2138691473746579540219428498194815931723114249657941048407150616092720030939109382357083052895494838142202703500532
0038839569499604469235199315111733986145819605411714397444135858834842715091323165208212965423170905976934363336377
6119275705206485910761055448765195490526453251954627931131919245417329051372989173711372148012230565707660333517183
0616667146012129788085068121560604156587055883046709255009080082791478754976736664520464000149355566957165369959183
9004805659999884513724545253597926910683946662206982070661191918190335866881469127514202364844508361317478306154651
01812521217544464803159773182488999999921691
>
```

## Ali G was Right!

- Theory assumes ideal computers:
  - Unlimited, perfect memory
  - Unlimited (finite) time
- Real computers have:
  - Limited memory, time, power outages, flaky programming languages, etc.
  - There are many computable problems we cannot solve with real computer: the actual inputs ***do*** matter (in practice, but not in theory!)

## Liberal Arts Trivia: Philosophy

- In the philosophy of mind, *this* is used to describe views in which the mind and matter are two ontologically separate categories. In *this*, neither mind nor matter can be reduced to each other in any way. *This* is typically opposed to reductive materialism. A well-known example of this is attributed to Descartes, holding that the mind is a nonphysical substance.

## Liberal Arts Trivia: Statistics

- A t-test is a statistical hypothesis test in which the test statistic has a *This* distribution of the null hypothesis is true. The *This* distribution arises when estimating the mean of a normally distributed population when the sample size is small. It was first published by William Gosset in 1908 while he worked at a Guinness Brewery in Dublin. The brewery forbade the publication of research by its staff members (!), so he published the paper under a pseudonym.

## Implementing Interpreters

## Inventing a Language

- Design the grammar
  - What strings are in the language?
  - Use BNF to describe all the strings in the language
- Make up the evaluation rules
  - Describe what everything the grammar can produce means
- Build an evaluator
  - A procedure that evaluates expressions in the language

## Is this an exaggeration? (SICP, p. 360)

It is no exaggeration to regard this as the most fundamental idea in programming:

**The evaluator, which determines the meaning of expressions in the programming language, is just another program.**

To appreciate this point is to change our images of ourselves as programmers. We come to see ourselves as designers of languages, rather than only users of languages designed by others.

## Environmental Model of Evaluation

- To **evaluate** a combination, **evaluate** all the subexpressions and **apply** the value of the first subexpression to the values of the other subexpressions.
- To **apply** a compound procedure to a set of arguments, evaluate the body of the procedure in a new environment. To construct this environment, make a new frame with an environment pointer that is the environment of the procedure that contains places with the formal parameters bound to the arguments.

Eval and Apply are defined in terms of each other.

## Implementing meval

```
def meval(expr, env):
   if isPrimitive(expr):
      return evalPrimitive(expr)
   elif isConditional(expr):
      return evalConditional(expr, env)
   elif isLambda(expr):
      return evalLambda(expr, env)
   elif isDefinition(expr):
      evalDefinition(expr, env)
   elif isName(expr):
      return evalName(expr, env)
   elif isApplication(expr):
      return evalApplication(expr, env)
   else:
      evalError ("Unknown expression type: " + str(expr))
```

## Implementing mapply

```
def mapply(proc, operands):
   if (isPrimitiveProcedure(proc)):
      return proc(operands)
   elif isinstance(proc, Procedure):
         params = proc.getParams()
         newenv = Environment(proc.getEnvironment())
         if len(params) != len(operands):
            evalError ("Parameter length mismatch: ...")
         for i in range(0, len(params)):
            newenv.addVariable(params[i], operands[i])
         return meval(proc.getBody(), newenv)
   else:
      evalError("Application of non-procedure: %s" % (proc))
```

## Homework

- Read GEB Aria, GEB 13 for Monday
- Reading Quiz #2 Monday
- Problem Set 7 due Mon April 06