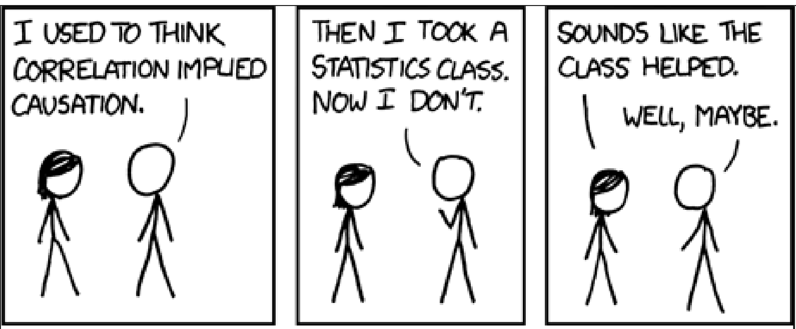

I USED TO THINK CORRELATION IMPLIED CAUSATION.
THEN I TOOK A STATISTICS CLASS. NOW I DON'T.
SOUNDS LIKE THE CLASS HELPED.
WELL, MAYBE.

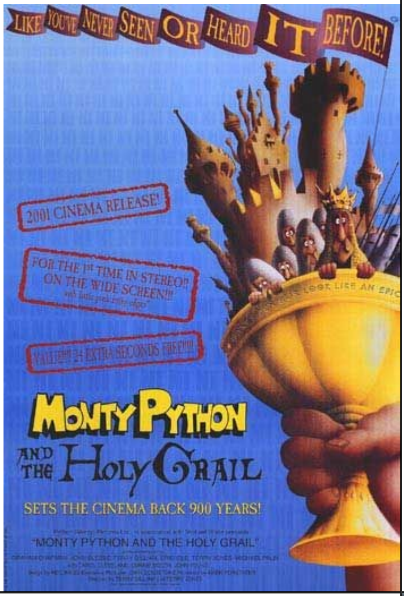
Objects Python Interpreters

One-Slide Summary

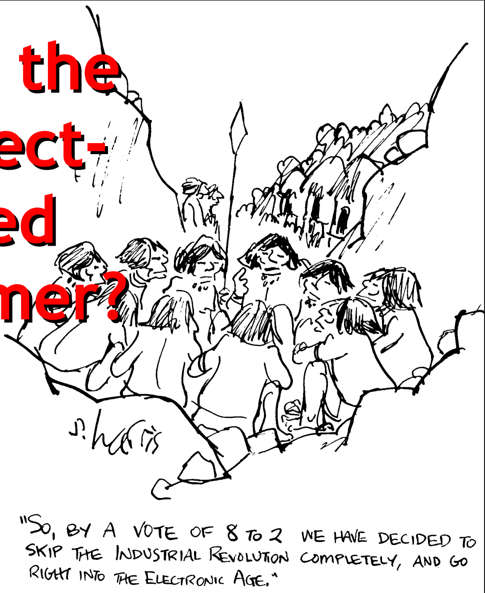
- **Object-Oriented Programming** encapsulates state and methods together into objects. This hides implementation details (cf. inheritance) while allowing methods to operate on many types of input.
- **Python** is a universal, imperative, object-oriented language. **Scheme** is a universal, functional language with imperative features.
- Building an **interpreter** is a fundamental idea in computing. Eval and Apply are **mutually recursive**.
- We can write a Python program that is an interpreter for Scheme programs.

Outline

- Object Lessons
- Ali G
- Interpreters
- Eval
- Apply


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!

Who was the first object-oriented programmer?


S. Harris
"So, by a vote of 8 to 2 we have decided to skip the Industrial Revolution completely, and go right into the Electronic Age."

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.

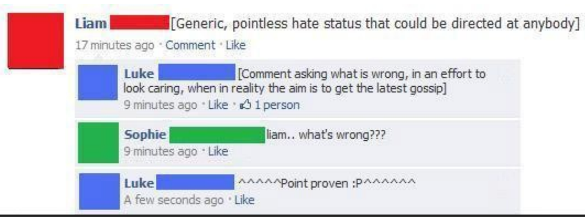
Ada, Countess of Lovelace, around 1843

Implementing Interpreters

Learn Languages: Expand Minds

Languages change the way we think.

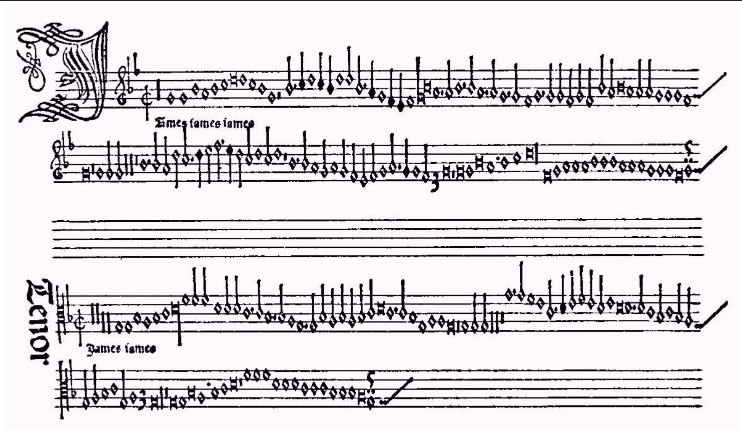
The more languages you know, the more different ways you have of thinking about (and solving) problems.


Liam [redacted] [Generic, pointless hate status that could be directed at anybody]
17 minutes ago · Comment · Like

Luke [redacted] [Comment asking what is wrong, in an effort to look caring, when in reality the aim is to get the latest gossip]
9 minutes ago · Like · 1 person

Sophie [redacted] liam.. what's wrong???
9 minutes ago · Like

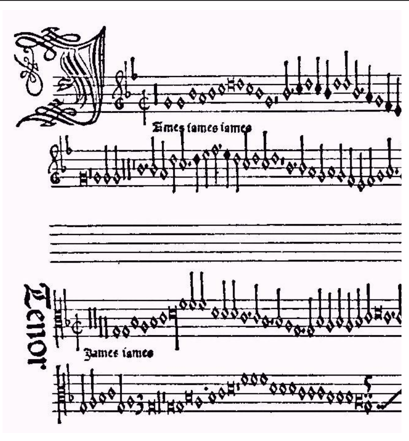
Luke [redacted] ^^^^^Point proven :P^^^^^^
A few seconds ago · Like



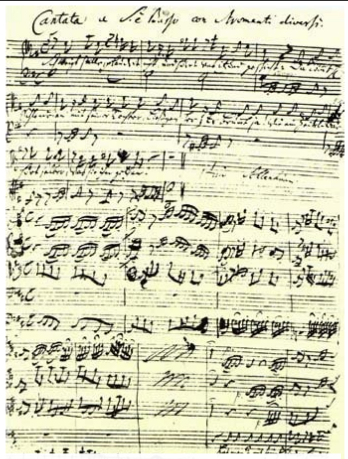
“Jamais Jamais Jamais” from *Harmonice Musices Odhecaton A*. Printed by Ottaviano Dei Petrucci in 1501 (first music with movable type)

Music with Movable Type?

- *Odhecaton*: secular songs published in 1501 Venice
  - First book of music ever printed using movable type,
  - Hugely influential both in publishing in general, and in dissemination of the Franco-Flemish musical style.
  - Seeing business potential, in 1498 Petrucci had obtained an exclusive 20-year license for all printing activities related to music anywhere in the Venetian Republic.
  - He printed two parts on the right-hand side of a page, and two parts on the left: four singers or instrumentalists could read from the same sheet.
  - Petrucci's publication not only revolutionized music distribution, it contributed to making the Franco-Flemish style the international musical language of Europe for the next century.



“Jamais Jamais Jamais” from *Harmonice Musices Odhecaton A*. (1501)



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

Computability in Theory and Practice

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

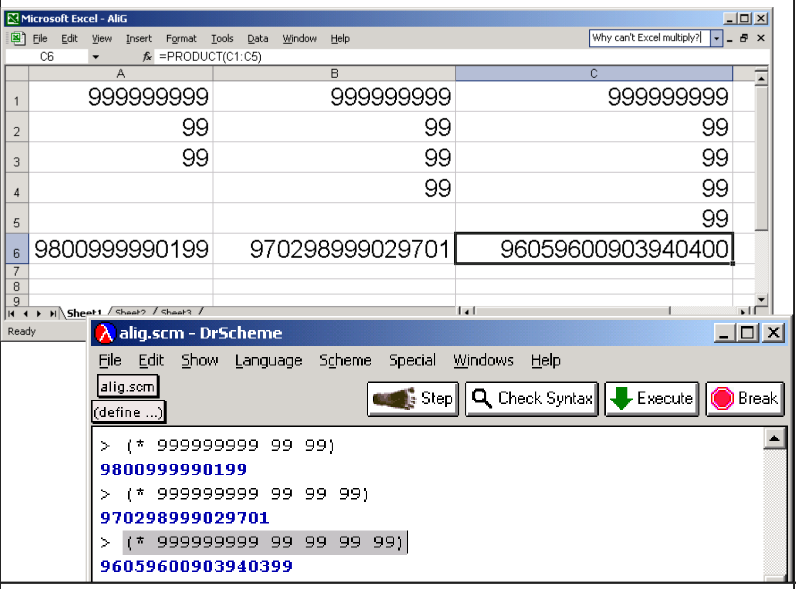
Ali G Problem

- **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?

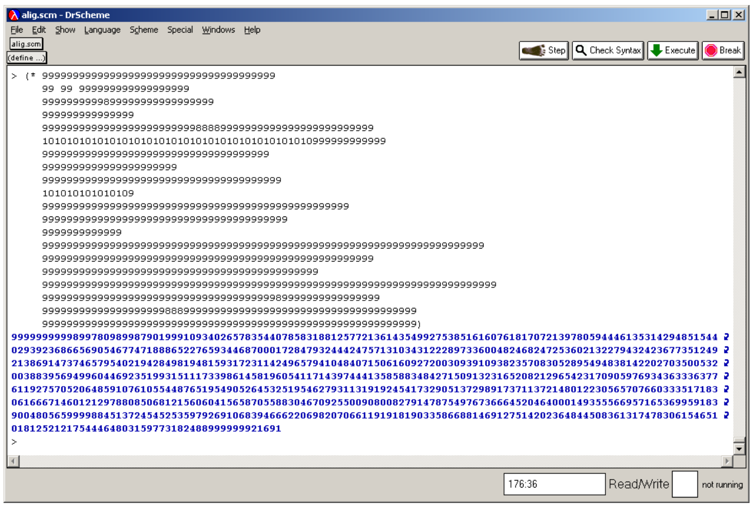


Microsoft Excel - AliG
File Edit View Insert Format Tools Data Window Help
Why can't Excel multiply?
C6
fx =PRODUCT(C1:C5)
Sheet1 / Sheet2 / Sheet3 /
Ready

	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			

alig.scm - DrScheme
File Edit Show Language Scheme Special Windows Help
alig.scm
(define ...)
Step
Check Syntax
Execute
Break

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



```
> (* 99999999999999999999999999999999999999
     99 99 999999999999999999
     9999999999899999999999999999
     999999999999999
     999999999999999999999999998889999999999999999999999999
     10101010101010101010101010101010101010101010109999999999999
     9999999999999999999999999999999999999
     9999999999999999999999
     999999999999999999999999999999999999999
     101010101010109
     99999999999999999999999999999999999999999999999999
     9999999999999999999999999999999999999999
     9999999999999
     9999999999999999999999999999999999999999999999999999999999999999999999999
     999999999999999999999999999999999999999999999999999999
     999999999999999999999999999999999999999999999
     999999999999999999999999999999999999999999999999999999999999999999999999999
     9999999999999999999999999999999999999999999998999999999
     9999999999999999999988899999999999999999999999999999999999999
     99999999999999999999999999999999999999999999999999999999999999)
999999999899780989987901999109340265783544078583188125772136143549927538516160761817072139780594446135314294851544 ↵
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!)

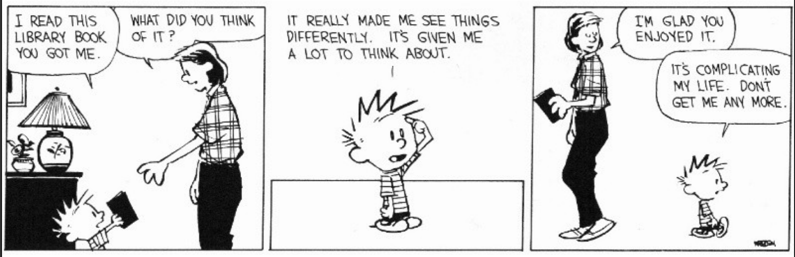
Liberal Arts Trivia: Biology

- 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


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.

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

#19

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.

#20

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.

#21

Eval and Apply are defined in terms of each other.

#22

The Plan – Front End

- We are given a string like “(-   (+ 5   6) 10)”
- First, we remove whitespace and **parse** it according to our Scheme grammar.
- We will represent Scheme expressions internally as either
  - Primitive elements, like '5' or '+'
  - Or lists of functions and their arguments
  - ['-', ['+', '5', '6'], '10']
  - (Plus some other details in PS7.)

#23

The Plan – Back End

- So now we have Python representations of Scheme expression, such as:
  - '123'   # 123
  - ['-', ['+', '5', '6'], '10'] # (- (+ 5 6) 10)
- We want to **evaluate** such expressions. We will write a procedure, **meval**(), that does this.
  - meval('123') = 123
- We'll also write **mapply()** to handle functions.
  - meval(['+','5','6']) = 11

#24

Evaluation and Environments

- Recall the Environment Model for Scheme
  - Go back to Class 13 or Book Chapter 9 if not!
- Let's think about some meval inputs and outputs together:
  - meval('12') = 12
  - meval('x') = ??
- So meval will need an **environment**:
  - meval('12',env) = 12
  - meval('x',env) = *value-of-x-in-env*

#25

meval basics

- So we might try to write meval by cases:

```
def meval(exp,env):
  if isPrimitive(exp):            # 3, x, etc.
    return evalPrimitive(exp,env)
  if isApplication(exp):          # (+ 3 4)
    return evalApplication(exp,env)
  # What other kinds of expressions are there?
  # Brainstorm now!
```

#26

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))
```

#27

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.

#28

Liberal Arts Trivia: Chemistry

- This exothermic chemical process involves the rapid oxidation of a fuel material, releasing light, heat and various reaction products. Fuels of interest often include organic compounds and hydrocarbons. Slower oxidation processes, like rusting, are not part of this process.

#29

Liberal Arts Trivia: Statistics

- A t-test is a statistical hypothesis test in which the test statistic has a *This* distribution if 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.

#30

mapply basics

- Let's think about mapply input and output together:
  - meval(['+', '1', '2']) = 3
  - meval(parse("(lambda (x) (+ x 2)) 5")) = 7
  - meval(['sqrt', '4']) = 2
- So we'll have separate handling for
  - **Primitive** procedures
  - Procedures made with **lambda**
    - Have parameters ("x") and bodies ("+ x 2")
    - May live in the environment ("sqrt")

Implementing mapply

```
def mapply(proc, operands):
  if (isPrimitiveProcedure(proc)):
    return proc(operands)
  elif isinstance(proc, Procedure):
      params = proc.getParams()
      newenv = Environment(proc.getEnvironment())
      # sanity check: no (sqrt 5 6)
      if len(params) != len(operands):
        evalError ("Parameter length mismatch: ...")
      # evaluate arguments first -- e.g., (sqrt (+ 3 1))
      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))
```

Simple Calculations

```
>>> parse(" 55 ")
'55'                         # or very close
>>> parse(" (- (+ 5 6) 10) ")
['-', ['+', '5', '6'], '10']         # or very close
    ...     # recall from meval() ...
elif isApplication(expr):
    return evalApplication(expr, env)
    ...
```

- So how do we define isApplication() ?

Detecting Applications

```
>>> parse(" (- (+ 5 6) 10) ")
['-', ['+', '5', '6'], '10']         # or very close

def isApplication(expr):
   # Applications are lists [proc, oper, oper, ...]
   return isinstance(expr, list)
```

- So how do we do evalApplication()?

Evaluating Applications

```
>>> parse(" (- (+ 5 6) 10) ")
['-', ['+', '5', '6'], '10']         # or very close
```

- So how do we do evalApplication()?

```
def evalApplication(expr, env):
   # To evaluate an application, evaluate all the subexpressions
   subexps = expr
   subexpvals = map (lambda sub: meval(sub, env), subexps)
   # then, apply the value of the first subexpression to the rest
   return mapply(subexpvals[0], subexpvals[1:])
```

Applying Primitives

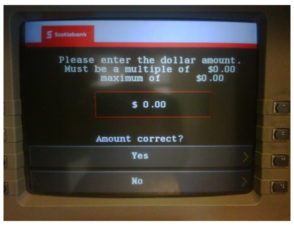
- >>> parse(" (- (+ 5 6) 10) ")
- ['-', ['+', '5', '6'], '10']   # or very close

```
  ...
def mapply(proc, operands):
   if (...):       # look up in environment, etc.
      return PrimitivePlus(operands)
  ...
```

- So how do we define PrimitivePlus() ?

Recall Scheme Addition

```
> (+ )
0
> (+ 5)
5
> (+ 6 7)
13
> (+ 6 7 8)
21
```



**Hint:
what are the basic
problem solving
strategies in this class?**

- Now write Python code to do it!

PrimitivePlus

```
def primitivePlus (operands):
   if (len(operands) == 0):
      return 0          # base case
   else:
      return operands[0] +
             primitivePlus (operands[1:])
```

- PrimitiveMinus is similar

Tracing

```
>>> parse(“ (- (+ 5 6) 10) ”)
['-', ['+', '5', '6'], '10']        # or very close
```

- meval([-, [+, 5, 6], 10])
  - evalApplication([-, [+,5,6], 10])
  - meval(-)   meval([+,5,6])   meval(10)
  - ...   evalApplication([+,5,6])   ...
  - ...   meval(+), meval(5), meval(6)   ...
  - ...   primitivePlus(5,6)   ...
  - ...   11   10
  - PrimitiveMinus(11,10)
  - 1

Homework

- Work on Problem Set #7
  - If we give you a long time before a problem set is due, what does that imply?