CS61A Discussion 2:
Environment Diagrams and Recursion

TA: **Jerry Chen**
Email: **jerry.c@berkeley.edu**
TA Website: **jerryjrchen.com/cs61a**

Agenda

1. Feedback!
2. Week in Review
3. Environment Diagrams
4. Lambdas (brief)
5. Recursion

Feedback!

Thanks for your feedback! Some changes:

- Trying some minor format changes
  - **All lec first, then problems later**
- **Better clarity** on which problems will be covered (disc is too long to do all the questions!)

Week In Review

CSM sections for 61A are available for signup!

Hog is due this Thursday!

- How many have **started/finished** with Hog?
- How many are **finished** with Hog?

How was lab 2? (Lambdas/HOFs, Recursion)

Attendance

Form: **tinyurl.com/jerrydisc**

Weekly question is the quiz.

Optional: add what you think of last week's and this week's quiz.

(Weekly question is **not judged based on correctness**)

Environments

Q: What is an **environment?**

A: Environments represent a **context** for execution.

- Environments store things such as name-value bindings

- Visualize environments using **environment diagrams**

Environment Diagrams

Consists of many frames that track program state

Some rules:

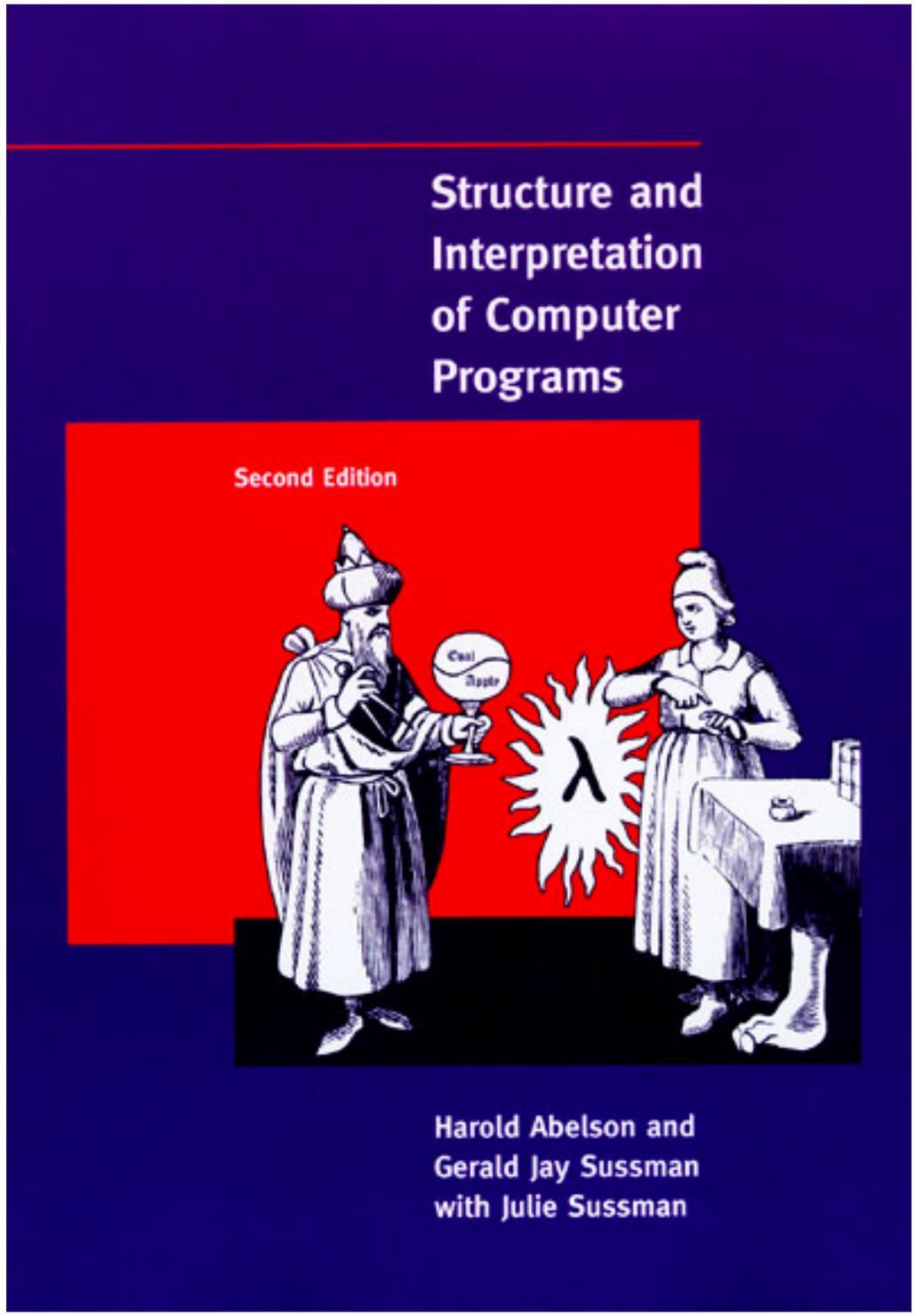
- **Function call: create and number new frame** (f1, f2, etc.) — always start in global frame

- **Assignment:** write variable name and expression value

- **Def statements:** record function name and bind function object. Remember parent frame!

- **Frames return values** upon completion (Global is special)

A Lambda Detour



A Lambda Detour

```
(lambda x, y: x + y * y)(4, 5)
```

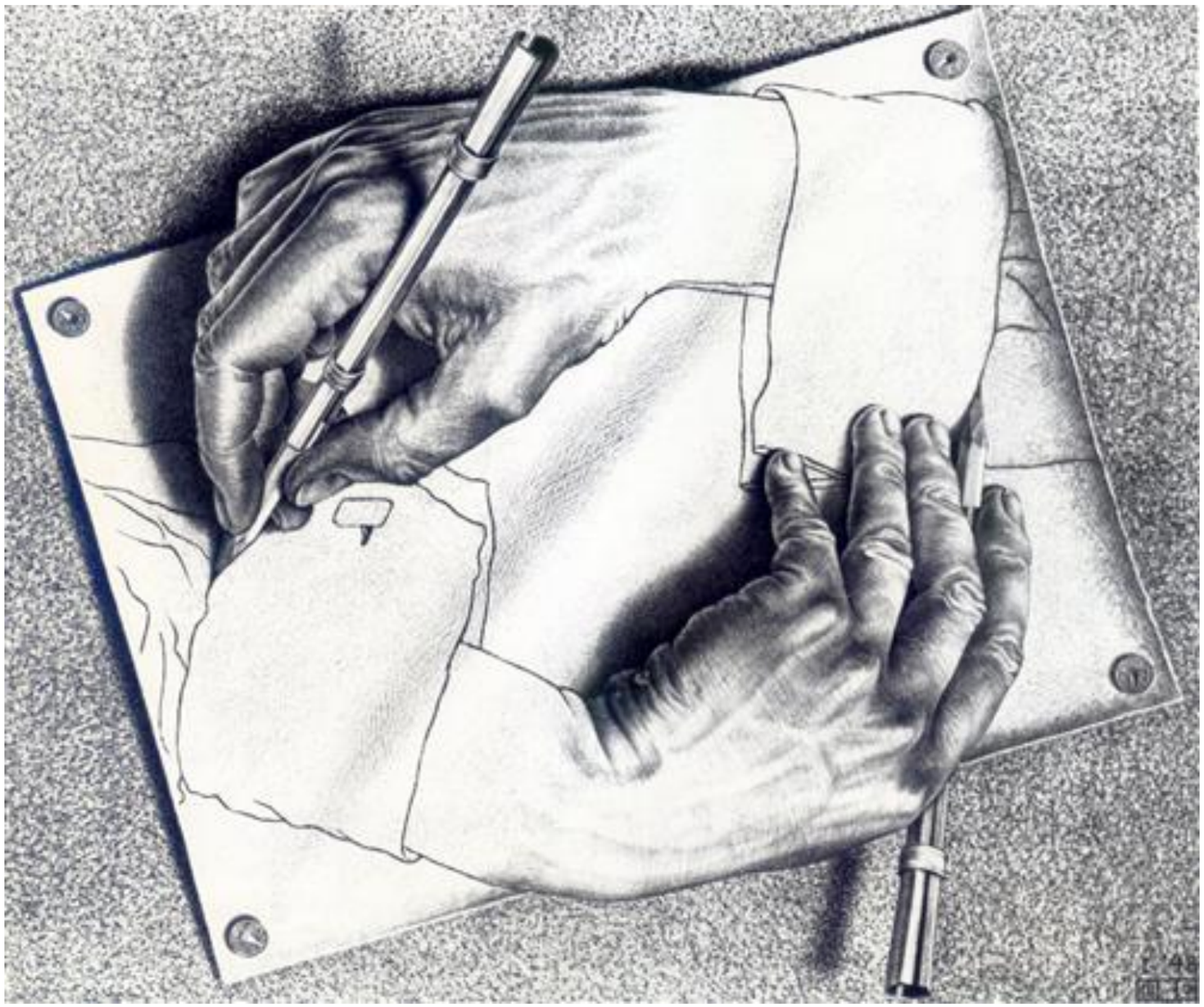
Lambda definition

Lambda call

Result (after currying):

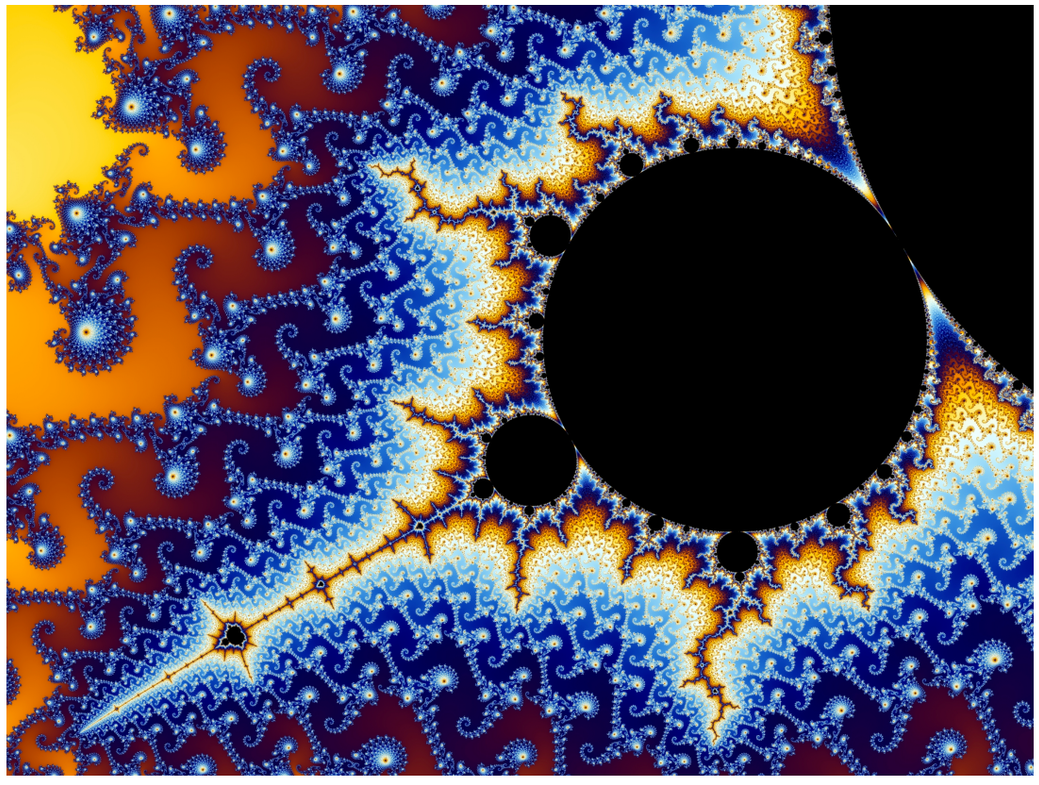
```
(lambda x = 4, y = 5: x + y * y)
```

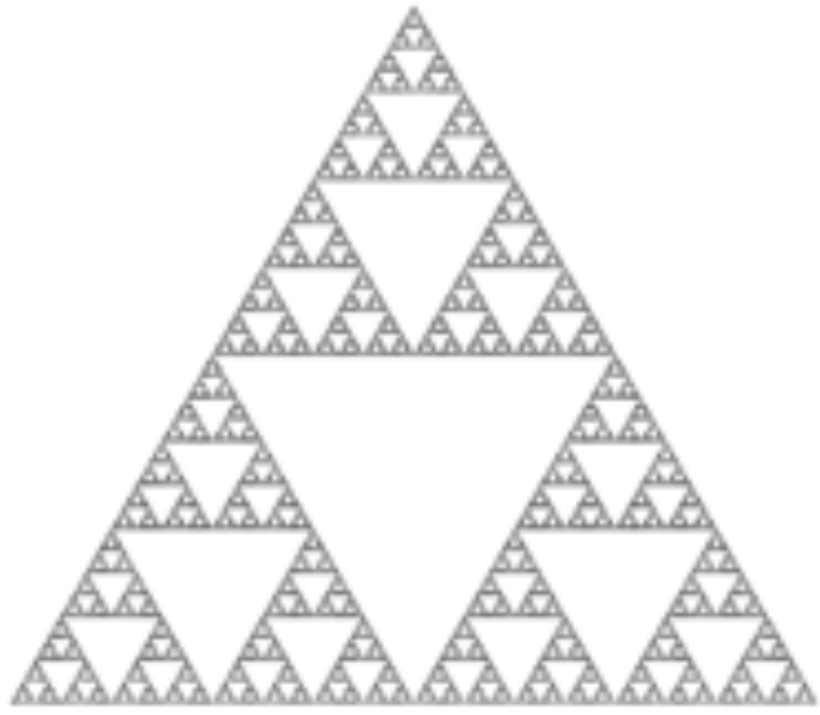
Recursion



Drawing Hands by M. C. Escher

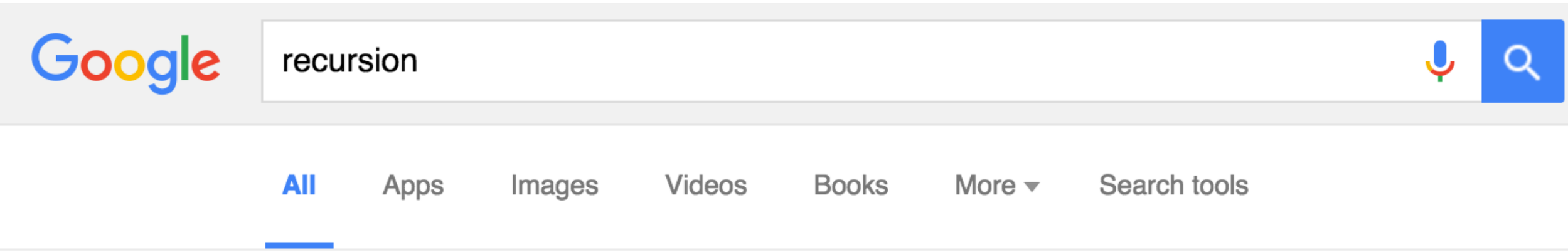
Recursion





Fractals: Mandelbrot Set and Sierpinski Triangle

Recursion


Google
recursion
All
Apps
Images
Videos
Books
More
Search tools
About 9,030,000 results (0.34 seconds)
Did you mean: recursion

Recursion

A recursive function can call itself,

which can call itself,

...

Recursion

Components of a recursive function

- **Base case:** some simple stopping condition
- **Recursive calls:** call ourself

  Must be **simpler** than the original problem

  Leap of faith: assume our recursive function solves any simpler version of the problem

Recursion

Exponentiation example from lab:

```
def exp(b, n):
    if n == 0:
        return 1
    if n % 2 == 0:
        return exp(b ** 2, n / 2)
    else:
        return b * exp(b, n - 1)
```

Tree Recursion

Recursive functions can sometimes require more than one call!

```
Fib(n) = Fib(n - 1) + Fib(n - 2)
```

Very powerful, but also potentially very slow (why?)