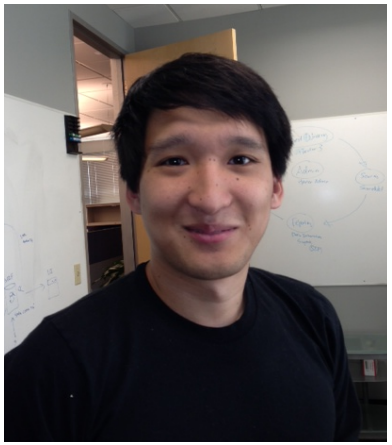
61A LECTURE 1 – FUNCTIONS, VALUES

Steven Tang and Eric Tzeng
June 24, 2013

Welcome to CS61A!

The Course Staff - Lecturers



Steven Tang

Graduated L&S CS from Cal

Back for a PhD in Education

Eric Tzeng

Graduated EECS from Cal

Back for a PhD in Computer Science

The Course Staff

Teaching Assistants

Rohan Chitnis

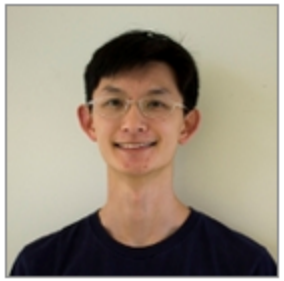
Email: cs61a-tb

Leonard Truong

Email: cs61a-tc

Robert Huang

Email: cs61a-td



Albert Wu

Email: cs61a-te

Andrew Huang

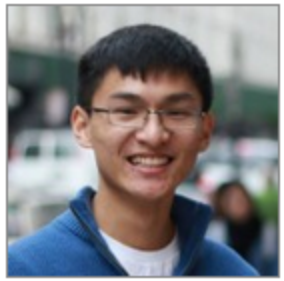
Email: cs61a-tf

Mark Miyashita

Email: cs61a-tg

Jeffrey Lu

Email: cs61a-th



Brian Hou

Email: cs61a-ti

Acknowledgement

Thanks to:

- Amir Kamil, who we are borrowing many of the lecture slides from
- John DeNero, who has developed much of the course material, including the fantastic online readings

What *is* Computer Science?

"Computer science deals with the theoretical foundations of information and computation, together with practical techniques for the implementation and application of these foundations"

- Wikipedia

"Computer science uses computers to make cool stuff."

- Steven Tang

What is CS61A?

- An introduction to the **"big ideas"** in Computer Science
  - Functions, recursion, data structures, interpretation, parallelism...
- Although the course uses Python, the ideas apply to any language
- General focus: Using *abstraction* to manage complexity

What is Abstraction?

- Abstraction is exposing how to use something while hiding how it works

- Many layers of abstraction in a typical system

Application
Libraries (Graphics, Physics)
Operating System
Hardware (CPU, RAM, etc.)
Logic Gates

- This course will teach you how to build and use abstractions

Some applications...

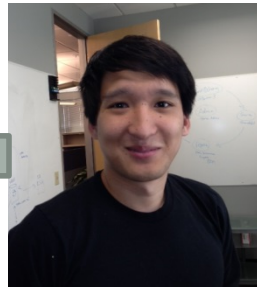
Phones

Cars

Politics

Games

Education



Movies

Music

Sports

Anything connected to the Internet

…

Systems

Programming Languages

Graphics

Artificial Intelligence

Databases

Theory

Security

Parallel Computing

Quantum Computing

On to logistics….

Course Structure

- **Readings** cover the material; read before lecture
- **Lectures** summarize material, present in new way
- **Labs** introduce new topics or practical skills
- **Discussions** provide practice on the material
- **Homeworks** are deeper exercises that require more thought than labs
- **Projects** are larger assignments designed to teach you how to use and combine ideas from the course in interesting ways

Assignments and Grading

- ~2 homeworks per week, due on Mondays and Thursdays
  - Homework 1 released later today, due Thursday
- 4 projects, one every 2 weeks
  - Project 1 released tomorrow, due in ~2 weeks
- 2 midterms, 1 final
  - Midterm 1 on Thursday, July 11 at 7PM
- Grading is on an absolute scale, rather than a curve
  - See course website http://www-inst.eecs.berkeley.edu/~cs61a

Seems fast...

- CS61A in the summer moves roughly twice as quickly as the regular semester
- Start assignments *early*, and get help quickly
- Staff is here to help
  - 8 teaching assistants
  - 30+ (!!!) academic interns
- Use office hours, use Piazza

Piazza

- We are using an online discussion form:

  https://piazza.com/class#summer2013/cs61a/

- Place to ask questions
- Both instructors and fellow students can post replies
- **Official announcements** will be posted to Piazza, so it is a requirement to use Piazza

Collaboration

- Remember: Grading is on a flat scale!
- Talk to each other
- EPA: Effort, participation, and altruism
- Homework may be completed with a partner
- Projects **should** be completed with a partner
- Find a project partner in your section!

Limits of collaboration:

- Never share code (don't e-mail, copy paste, etc.)
- Copying projects is a serious offense. We have of ways of detecting duplicate work.

FAQ

- Midterms on 7/11 and 8/01
- Final on 8/15
  - Let us know ASAP if you have any conflicts
- To waitlisted: In the summer , 61A is generally able to admit all students on the waitlist. Continue to complete and turn in assignments

Announcements

- Make sure you have an account form and register
  - All assignments (homeworks and projects) are submitted through your account
  - Account forms handed out in lab and discussion this week
- Office hours start Wednesday
  - See website for schedule
- Homework 1 due Thurs. at 11:59PM

Break

Data, Functions, and Interpreters

Data: the things that programs fiddle with

"UC Berkeley"

2

(5, 3, 2)

Functions: rules for manipulating data

Count the words in a line of text

Add up numbers

Pronounce someone's name

Interpreter: an implementation of the procedure for evaluation

Primitive Values and Expressions

- An **expression** is something that produces a data value.
- The simplest types of expressions produce a value directly. We call them **primitive expressions**.
  - Integers: 42, -9001, 8417765
  - Floating point (decimal) values: 8.3, -39.2
  - Strings: “It was a dark and stormy night”
  - Booleans: True, False

- A **compound expression** combines primitive expressions to produce a value.
  - 2 + 3
  - sqrt(3004)
  - abs(50 – 100 * 5)

Examples in the interpreter

Anatomy of a Call Expression

add	(	2	,	3	)
Operator		Operand 0		Operand 1	

Operators and operands are expressions, so they evaluate to values

Evaluation procedure for call expressions:

1. Evaluate the operator and operand subexpressions in order from left to right.
2. Apply the function that is the value of the operator subexpression to the arguments that are the values of the operand subexpressions

Infix Expressions in Python

- Infix expressions can use function call notation

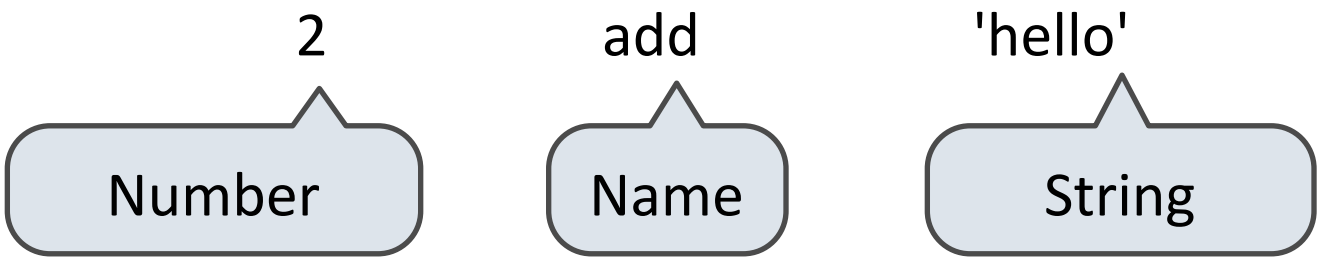
2 + 3	add(2, 3)
abs(-128 + 42 * 3)	abs(add(-128, mul(42, 3)))

- Infix operator notation is *syntactic sugar* for function calls

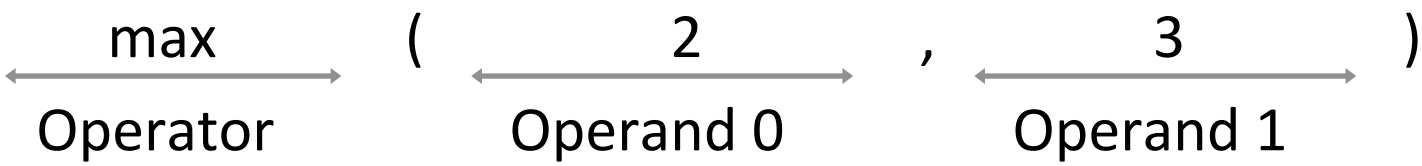
- Mathematical operators obey usual precedence rules

Summary of expressions

Primitive expressions:



Call expressions:



One big nested call expression

max(min(pow(3, 5), -4), min(1, -2))

Infix operators represent implicit call expressions

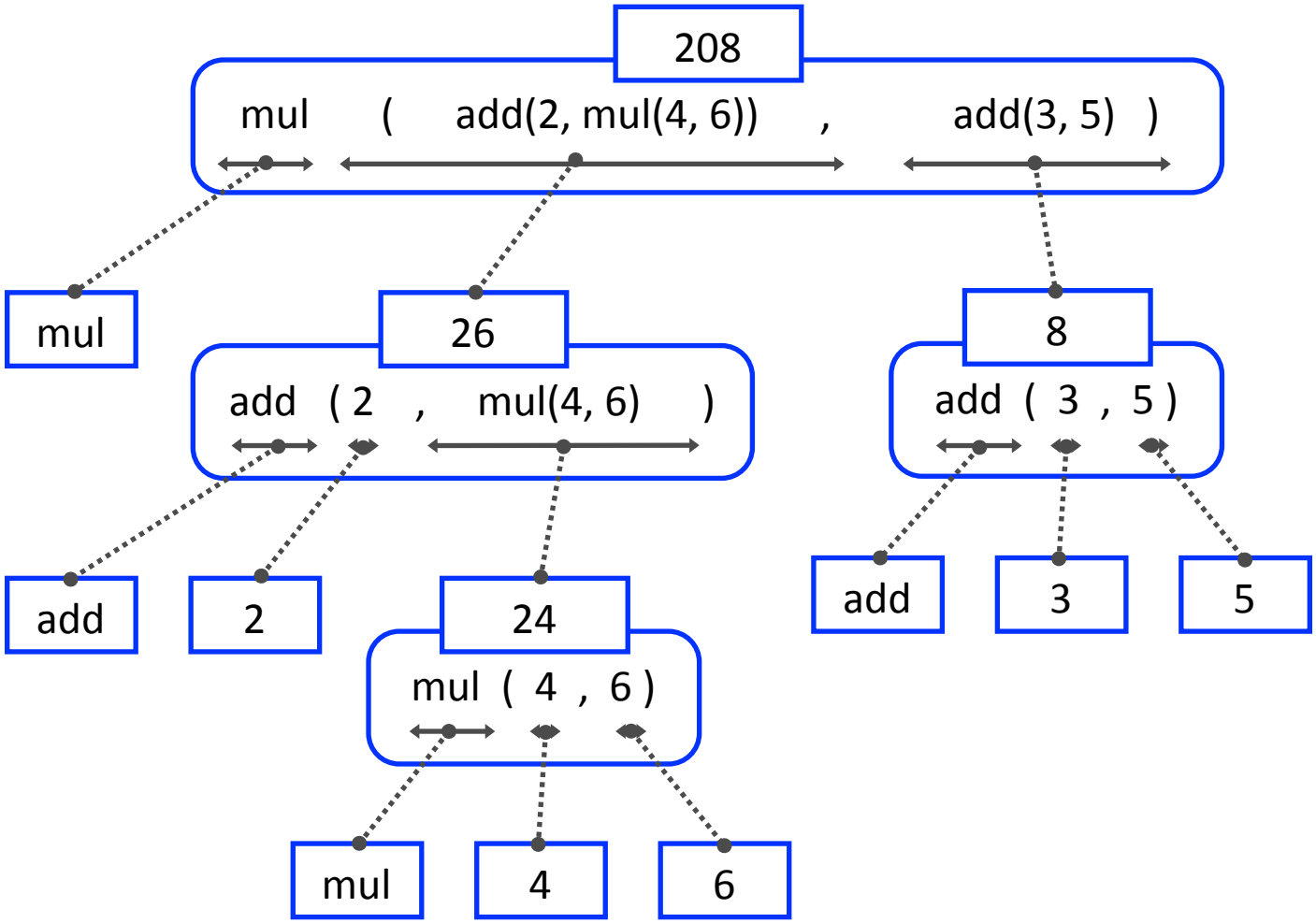
2 + 3 → add(2, 3)

Remember the rules…

Evaluation procedure for call expressions:

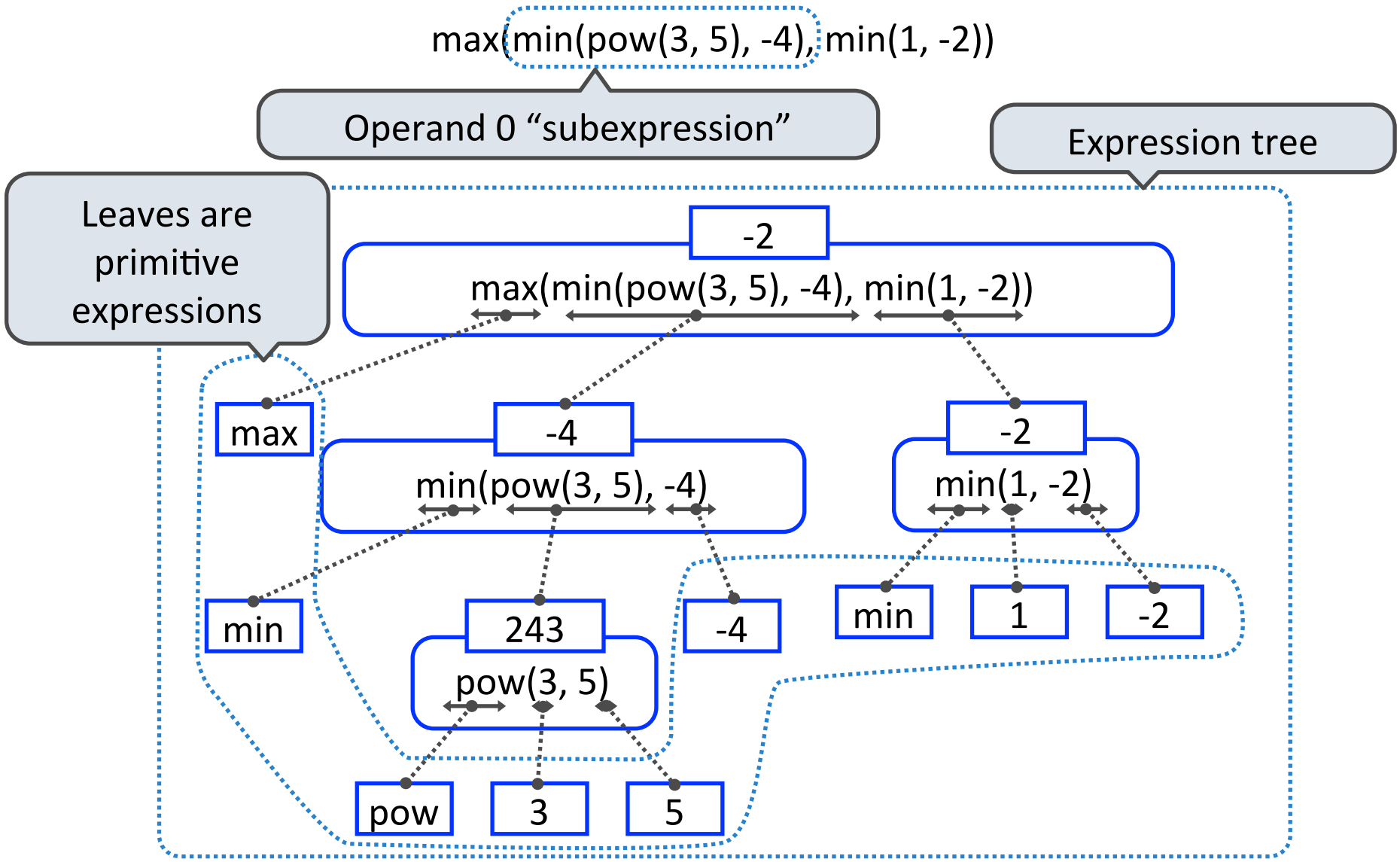
1. Evaluate the operator and operand subexpressions in order from left to right.
2. Apply the function that is the value of the operator subexpression to the arguments that are the values of the operand subexpressions

Evaluating Nested Expressions



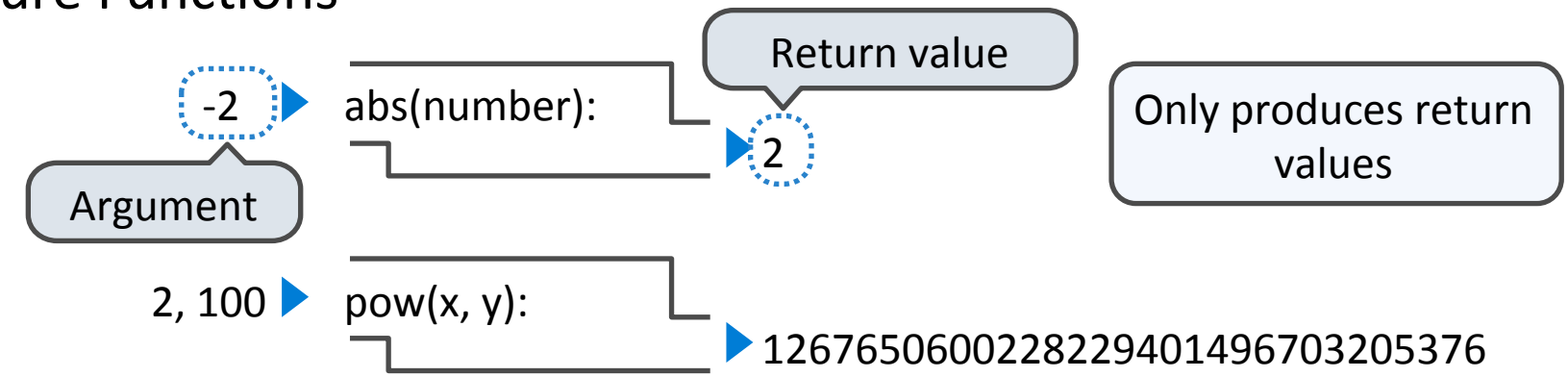
Break

Recap of Expression Trees

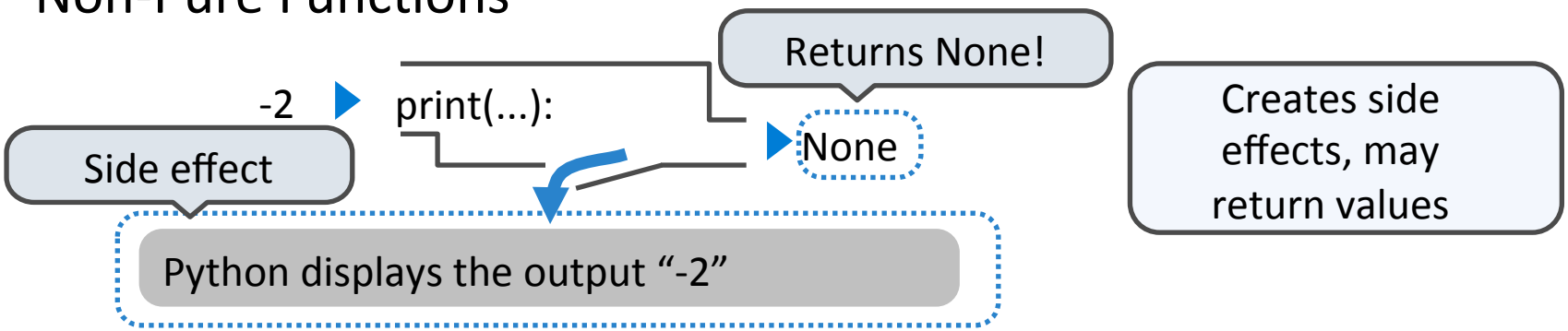


Types of Functions

Pure Functions



Non-Pure Functions



The interactive interpreter displays all return values except None.

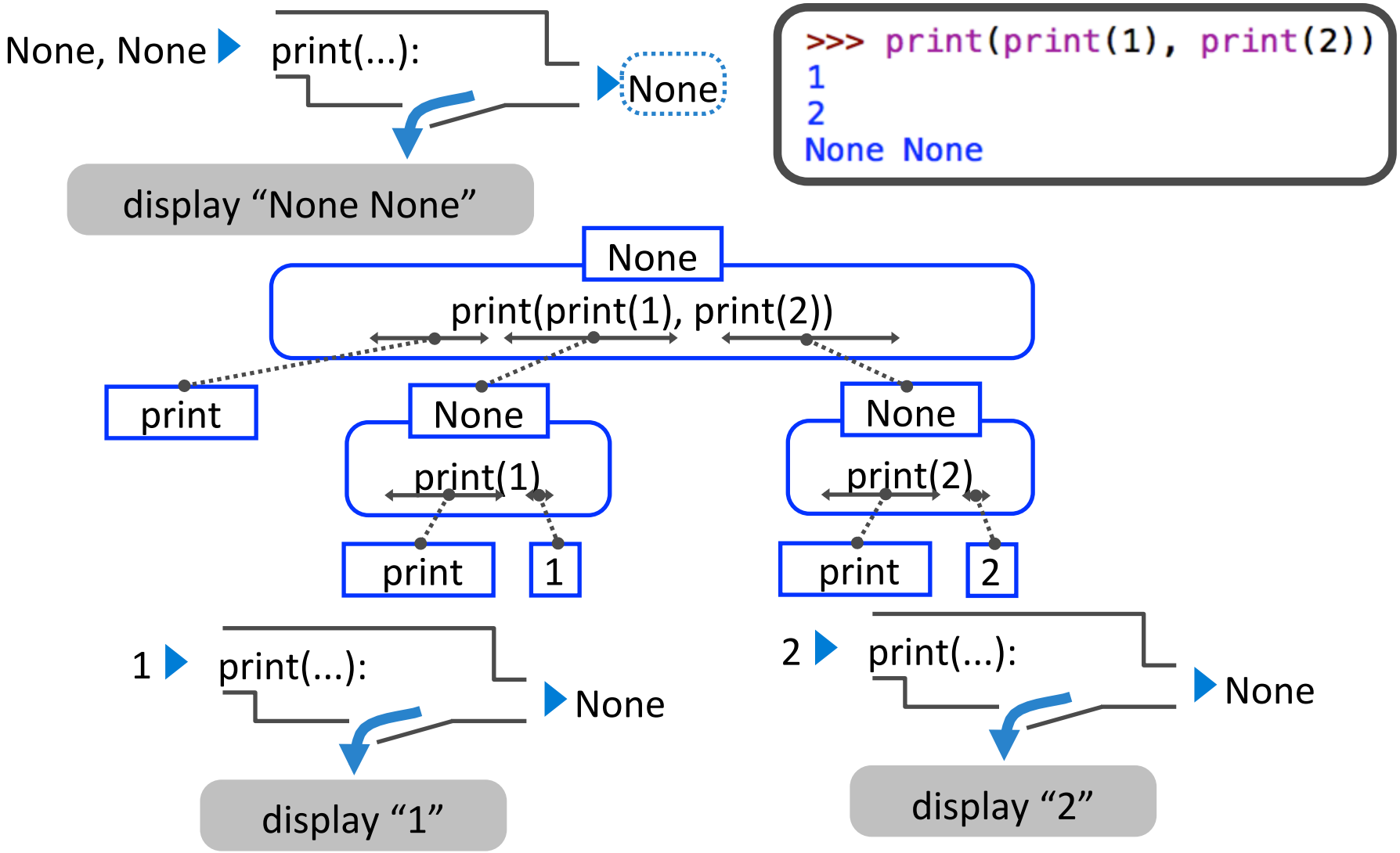
Back to the interpreter

- What do you think is printed by Python when you input:

```
print(print(1), print(2))
```

Draw an expression tree.

Nested Print Expressions



The Elements of Programming

- Primitive Expressions and Statements
  - The simplest building blocks of a language
- Means of Combination
  - Compound elements built from simpler ones
- Means of Abstraction
  - Elements can be named and manipulated as units

Reminders

- Account forms handed out in lab today
  - Go to your section!
- Homework 1 is due Thursday
- Project 1 released tomorrow, due July 5 at 11:59PM
- Sign up for Piazza ASAP
- No office hours today; they start tomorrow