UC Berkeley's CS61A – Lecture 04 – Higher-Order Functions

YouTube says it will crack down on recommending conspiracy videos

www.cnn.com/2019/01/25/tech/youtube-conspiracy-video-recommendations

"YouTube would start reducing its recommendations of 'borderline content' (e.g., videos featuring fake miracle cures for serious diseases, claiming the earth is flat and making blatantly false claims about historic events such as 9/11) and videos that may misinform users in 'harmful ways.' Such content doesn't violate YouTube's community guidelines, but the company says it comes close. (It would apply to less than 1% of content on its platform)

YouTube has long faced criticism for allowing misinformation, conspiracy theories and extremist views to spread on its platform, and for recommending such content to users. People who came to the site to watch videos on innocuous subjects, or to see mainstream news, have been pushed toward increasingly fringe and conspiracist content.

For example, after a mass shooting at a high school in Parkland, Florida last year, a top trending video on YouTube suggested survivor David Hogg was an actor."

PolitiFact's Lie of the Year: Online smear machine tries to take down Parkland students

By Angie Drobnic Holan, Amy Sherman on Tuesday, December 11th, 2018 at 9:30 a.m.

Announcements

Lab 00 is a take-home lab. Try to complete it before your lab this week.

Homework 1 is released and is due this Thursday 1/31 @ 11:59pm.

Drop-in office hours start this week.

Extra lectures (optional, on Additional Topics) start this week, Wednesdays 1–2pm in Soda 310: https://cs61a.org/extra.html

Hog has been released! Entire project due Thursday 2/7

- Solve Phase 1 individually; Work with a partner on Phases 2 & 3.
- Phase 1 checkpoint due Tuesday 2/5.
- Submit everything by Wednesday 2/6 for an early submission bonus point.
- Project Party next Monday 2/4.

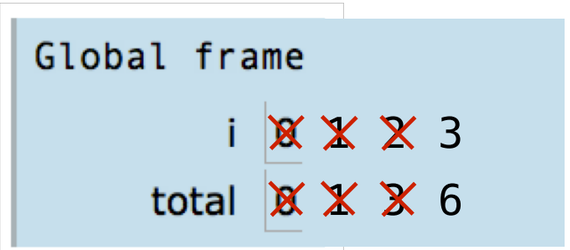
Iteration

While Statements

George Boole

(Demo)

```
1  i, total = 0, 0
2  while i < 3:
3      i = i + 1
4      total = total + i
```

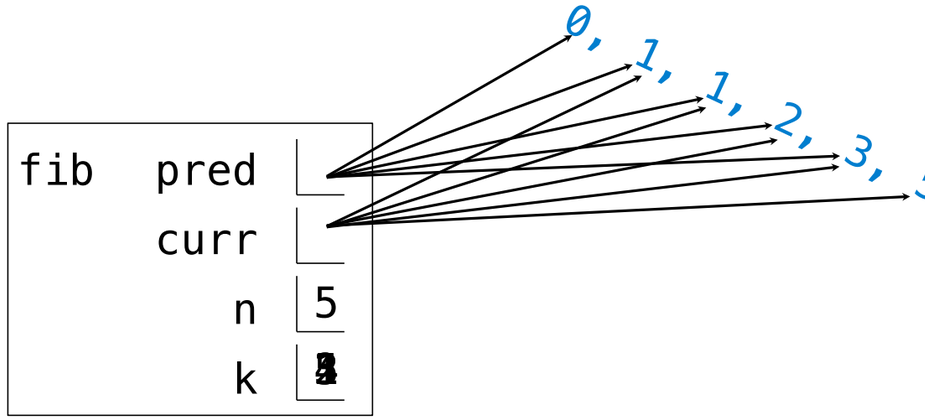


Execution Rule for While Statements:

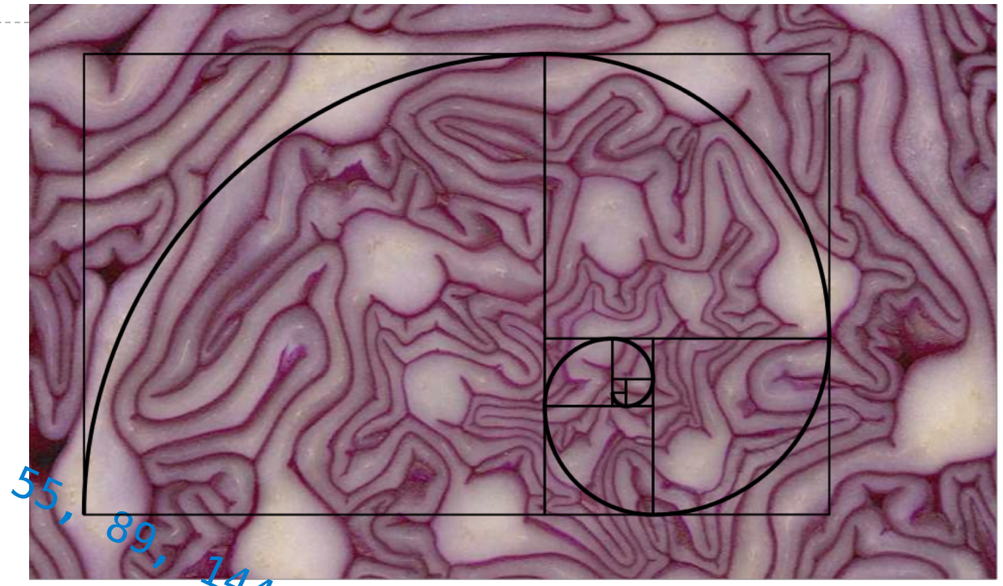
1. Evaluate the header's expression.

2. If it is a true value,
   execute the (whole) suite,
   then return to step 1.

The Fibonacci Sequence



0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987



```
def fib(n):
    """Compute the nth Fibonacci number, for N >= 1."""
    pred, curr = 0, 1   # 0th and 1st Fibonacci numbers
    k = 1               # curr is the kth Fibonacci number
    while k < n:
        pred, curr = curr, pred + curr
        k = k + 1
    return curr
```

The next Fibonacci number is the sum of the current one and its predecessor

Designing Functions

Describing Functions

A function's *domain* is the set of all inputs it might possibly take as arguments.

A function's *range* is the set of output values it might possibly return.

A pure function's *behavior* is the relationship it creates between input and output.

```
def square(x):
    """Return X * X."""
```

x is a number

square returns a non-negative real number

square returns the square of x

A Guide to Designing Function… Generalization!

Give each function exactly one job, but make it apply to many related situations

```
>>> round(1.23)
1
```

```
>>> round(1.23, 1)
1.2
```

```
>>> round(1.23, 0)
1
```

```
>>> round(1.23, 5)
1.23
```

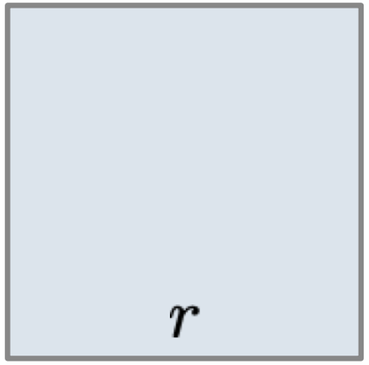
Don’t repeat yourself (DRY). Implement a process just once, but execute it many times.

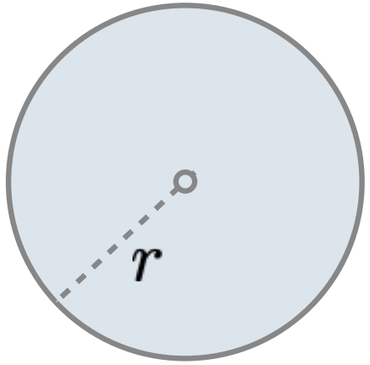
Generalization

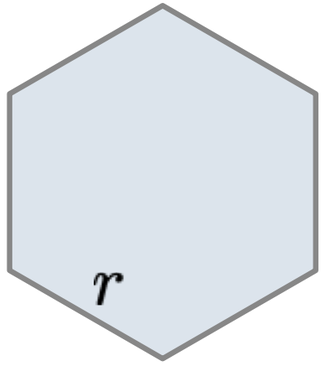
Generalizing Patterns with Arguments

Regular geometric shapes relate length and area.

Shape:







Area:

$1 \cdot r^2$ $\pi \cdot r^2$ $\frac{3\sqrt{3}}{2} \cdot r^2$

Finding common structure allows for shared implementation

(Demo1)

Higher-Order Functions

Generalizing Over Computational Processes

The common structure among functions may be a computational process, rather than a number.

$$\sum_{k=1}^{5} k = 1 + 2 + 3 + 4 + 5 = 15$$

$$\sum_{k=1}^{5} k^3 = 1^3 + 2^3 + 3^3 + 4^3 + 5^3 = 225$$

$$\sum_{k=1}^{5} \frac{8}{(4k-3)\cdot(4k-1)} = \frac{8}{3} + \frac{8}{35} + \frac{8}{99} + \frac{8}{195} + \frac{8}{323} = 3.04$$

(Demo2)

Summation Example

```
def cube(k):
    return pow(k, 3)

def summation(n, term):
    """Sum the first n terms of a sequence.

    >>> summation(5, cube)
    225
    """
    total, k = 0, 1
    while k <= n:
        total, k = total + term(k), k + 1
    return total
```

Function of a single argument (*not called "term"*)

A formal parameter that will be bound to a function

The cube function is passed as an argument value

0 + 1 + 8 + 27 + 64 + 125

The function bound to term gets called here

Functions as Return Values

(Demo3)

Locally Defined Functions

Functions defined within other function bodies are bound to names in a local frame

A function that returns a function

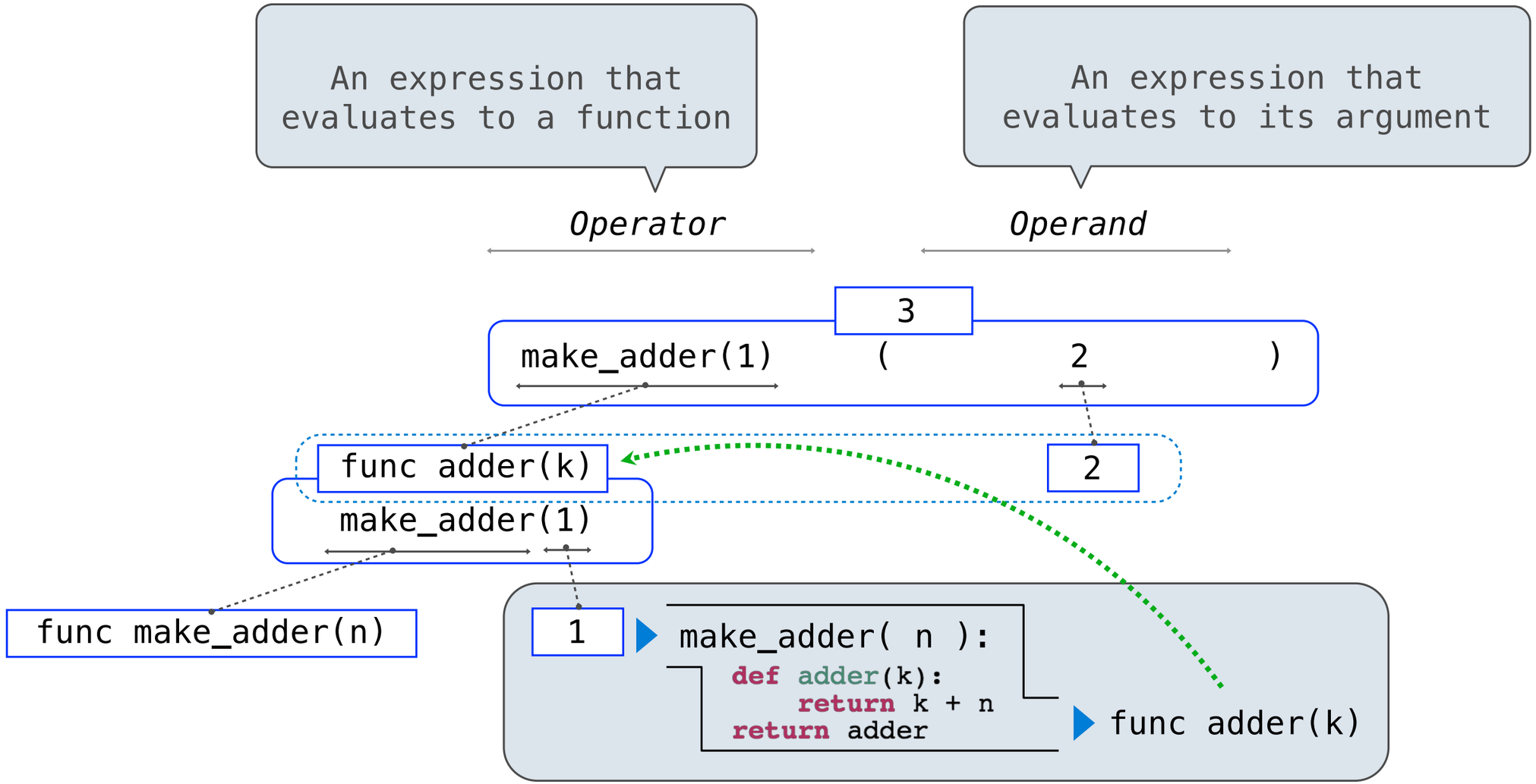
```
def make_adder(n):
    """Return a function that takes one argument k and returns k + n.

    >>> add_three = make_adder(3)
    >>> add_three(4)
    7
    """
    def adder(k):
        return k + n
    return adder
```

The name add_three is bound to a function

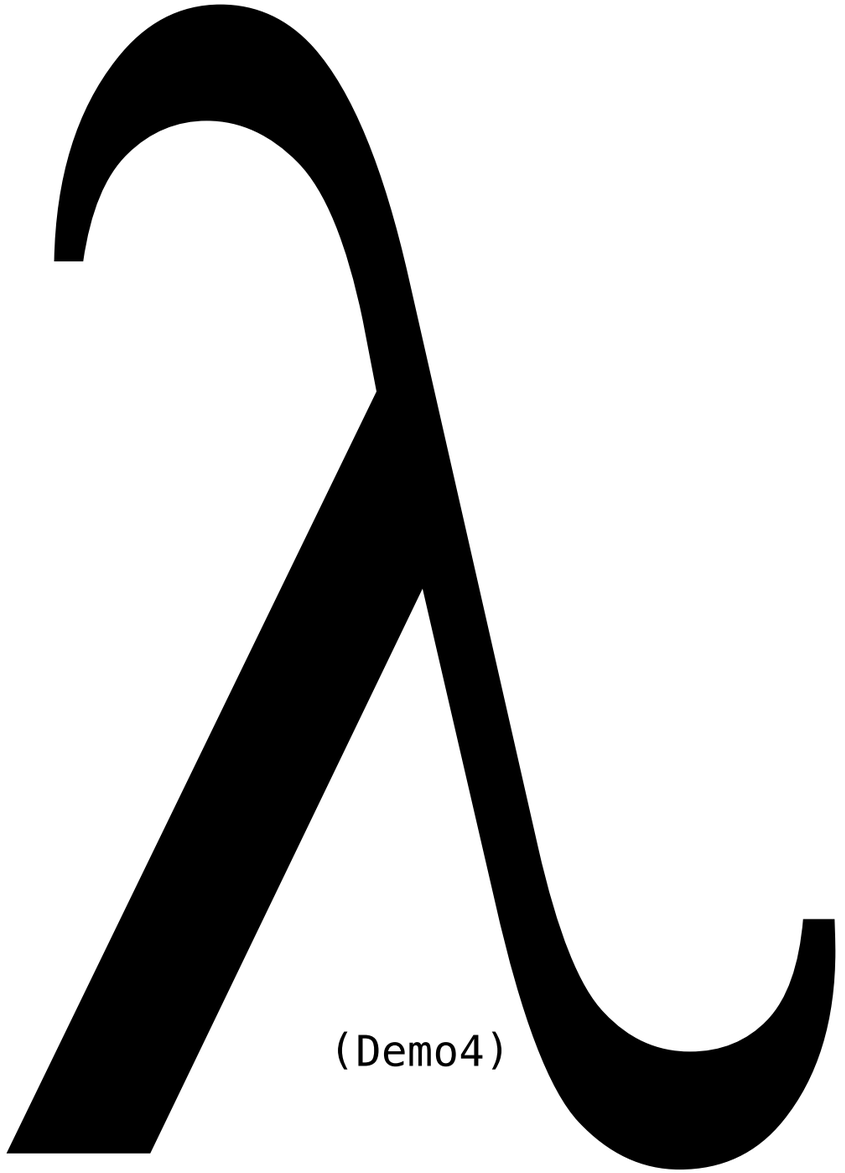
A def statement within another def statement

Can refer to names in the enclosing function

Call Expressions as Operator Expressions



Lambda Expressions



(Demo4)

Lambda Expressions

```
>>> x = 10
>>> square = x * x
>>> square = lambda x: x * x
>>> square(4)
16
```

An expression: this one evaluates to a number

Also an expression: evaluates to a function

Important: No "return" keyword!

A function with formal parameter x that returns the value of "x * x"

Must be a single expression

Lambda expressions are not common in Python, but important in general

Lambda expressions in Python cannot contain statements at all!

Lambda Expressions Versus Def Statements

```
square = lambda x: x * x
```

VS

```
def square(x):
    return x * x
```

- Both create a function with the same domain, range, and behavior.
- Both bind that function to the name square.
- Only the def statement gives the function an intrinsic name, which shows up in environment diagrams but doesn't affect execution (unless the function is printed).