Higher-Order Functions

Announcements

Office Hours: You Should Go!
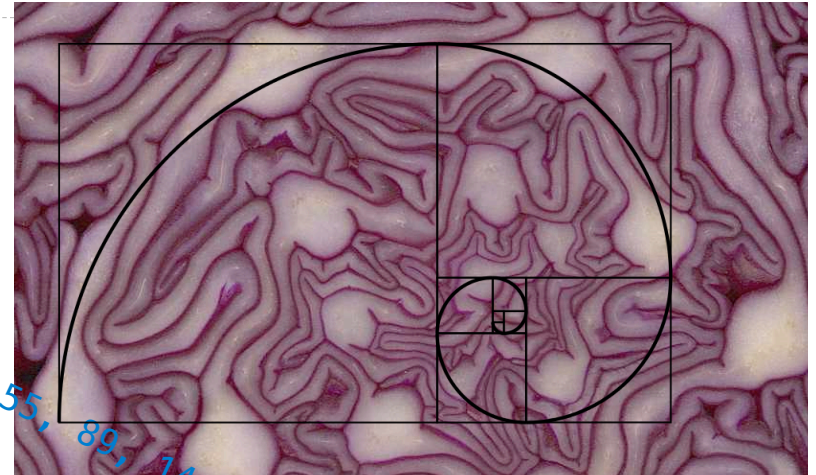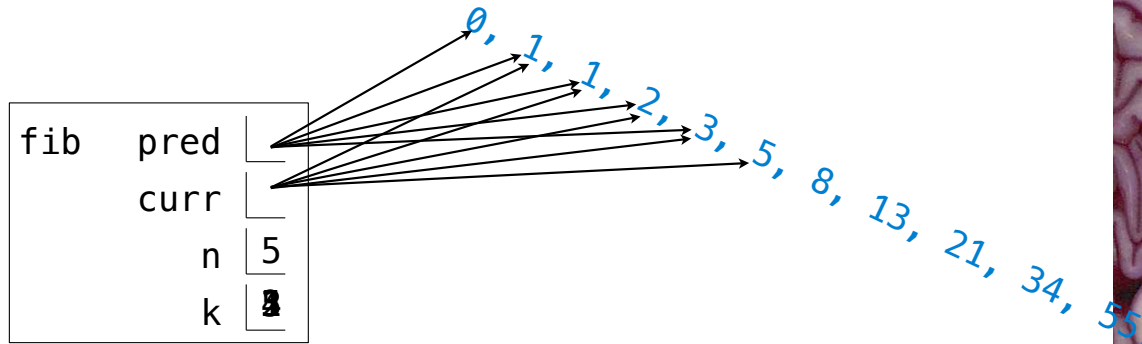
You are not alone!

http://cs61a.org/office-hours.html
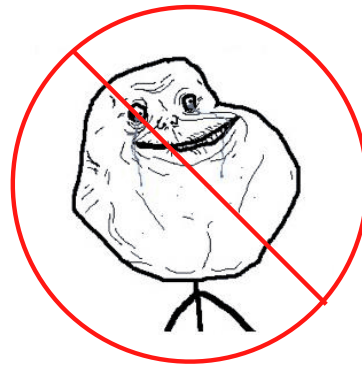
Iteration Example

The Fibonacci Sequence

fib   pred
       curr
       n | 5 |
       k | 5 |

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987
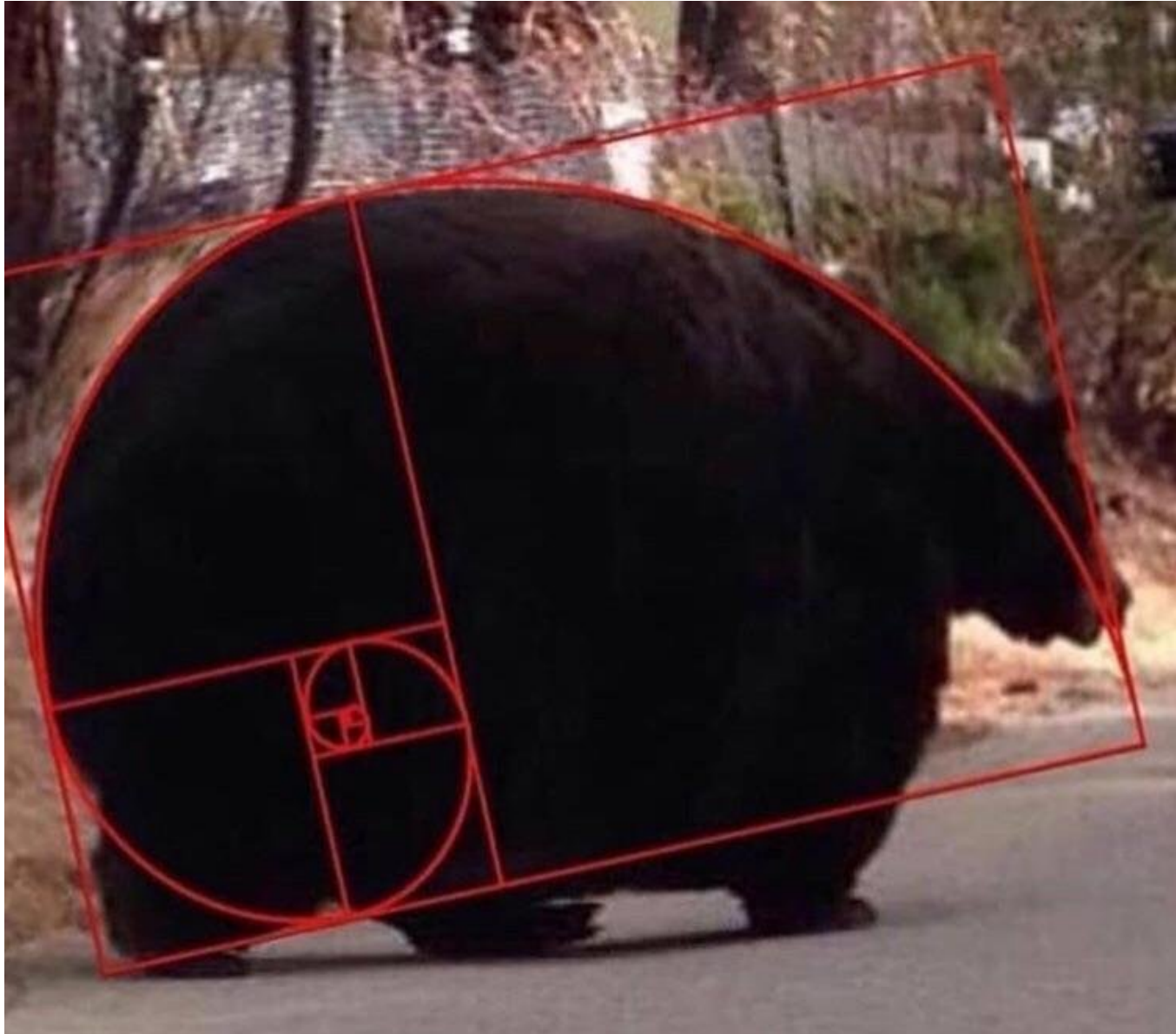
```python
def fib(n):
    """Compute the nth Fibonacci number, for N >= 1."""
    pred, curr = 0, 1   # 0th and 1st Fibonacci numbers
    k = 1               # curr is the kth Fibonacci number
    while k < n:
        pred, curr = curr, pred + curr
        k = k + 1
    return curr
```

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

Go Bears!

Designing Functions

Describing Functions

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

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

x is a number

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

square returns a non-negative real number

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

square returns the square of x

A Guide to Designing Function

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

```
>>> round(1.23)       >>> round(1.23, 1)      >>> round(1.23, 0)      >>> round(1.23, 5)
1                     1.2                     1                      1.23
```

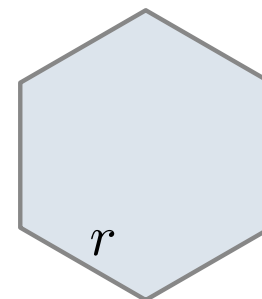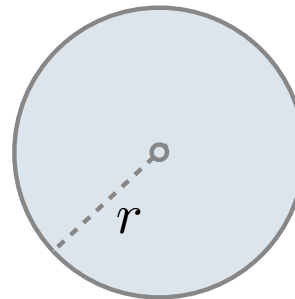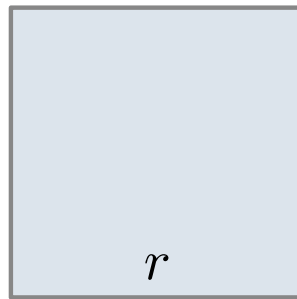Don't repeat yourself (DRY): Implement a process just once, but execute it many times

(Demo)

Generalization

Generalizing Patterns with Arguments

Regular geometric shapes relate length and area.

Shape:

Area: $\boxed{1} \cdot r^2$ $\qquad$ $\boxed{\pi} \cdot r^2$ $\qquad$ $\boxed{\dfrac{3\sqrt{3}}{2}} \cdot r^2$

Finding common structure allows for shared implementation

(Demo)

Higher-Order Functions

Generalizing Over Computational Processes

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

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

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

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

(Demo)

Summation Example

```python
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

The function bound to term gets called here

0 + 1 + 8 + 27 + 64 + 125

Functions as Return Values

(Demo)

Locally Defined Functions

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

> A function that
> returns a function

```python
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

An expression that evaluates to a function

An expression that evaluates to its argument

Operator

Operand

```
         3
make_adder(1)   (      2      )
```

```
func adder(k)                    2
make_adder(1)
```

```
func make_adder(n)      1  ▶ make_adder( n ):
                               def adder(k):
                                   return k + n
                               return adder   ▶ func adder(k)
```