

Iteration

While Statements

(Demo)



George Boole

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

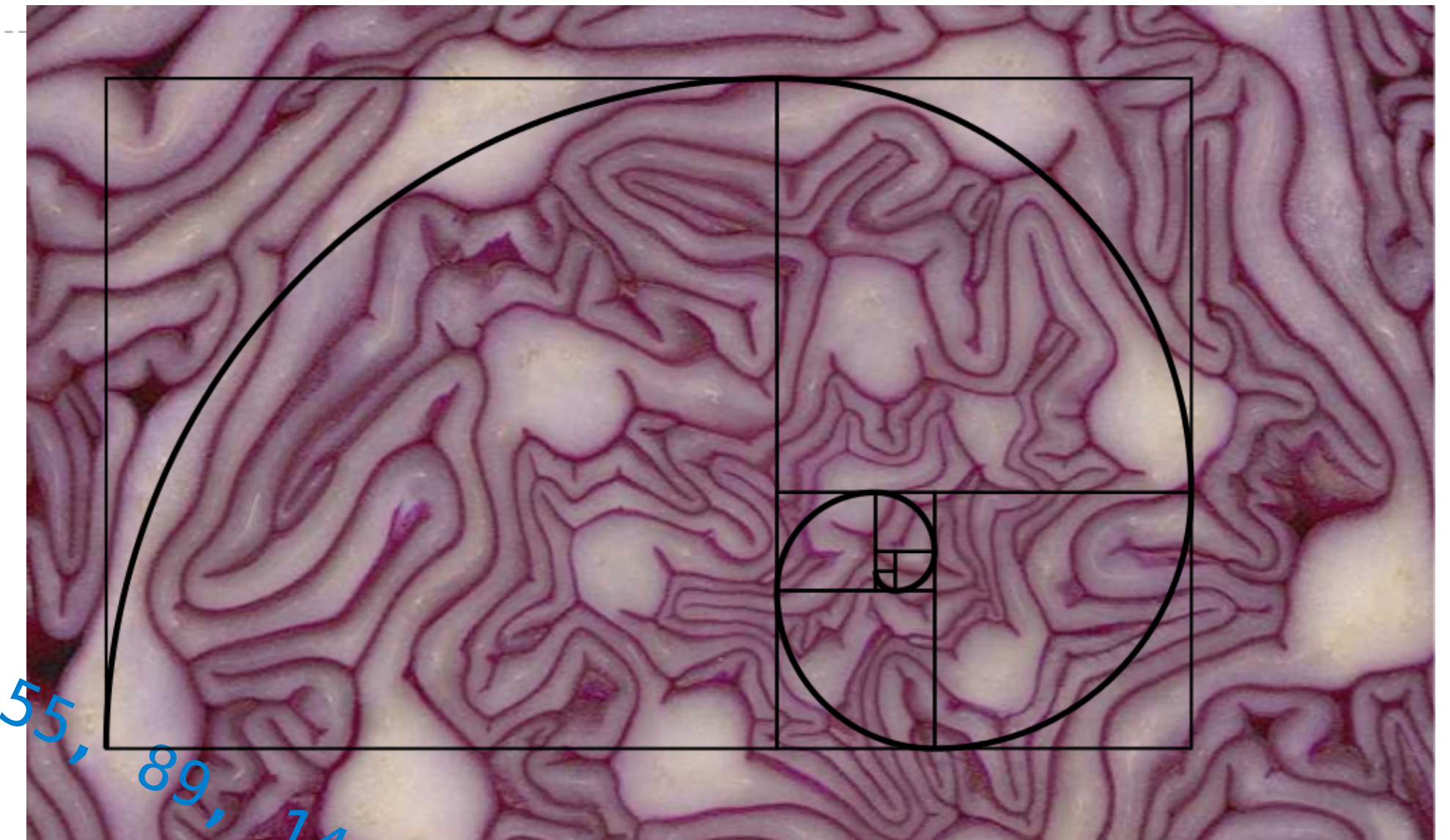
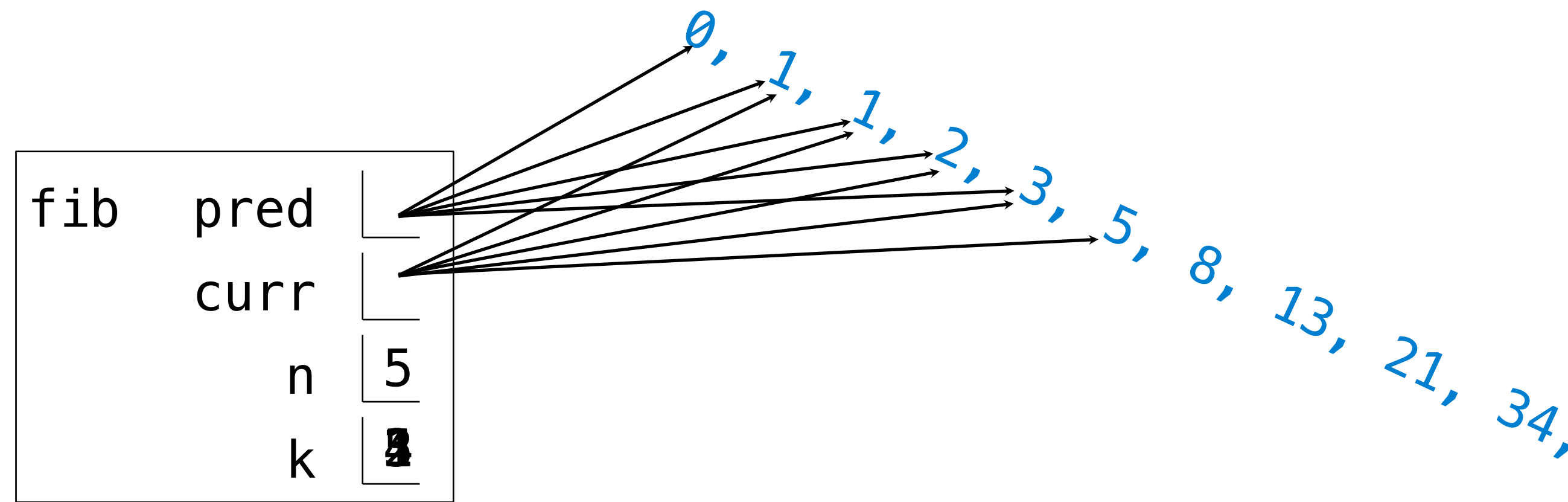
Global frame

i	0	1	2	3
total	0	1	3	6

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



```
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)      >>> 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.

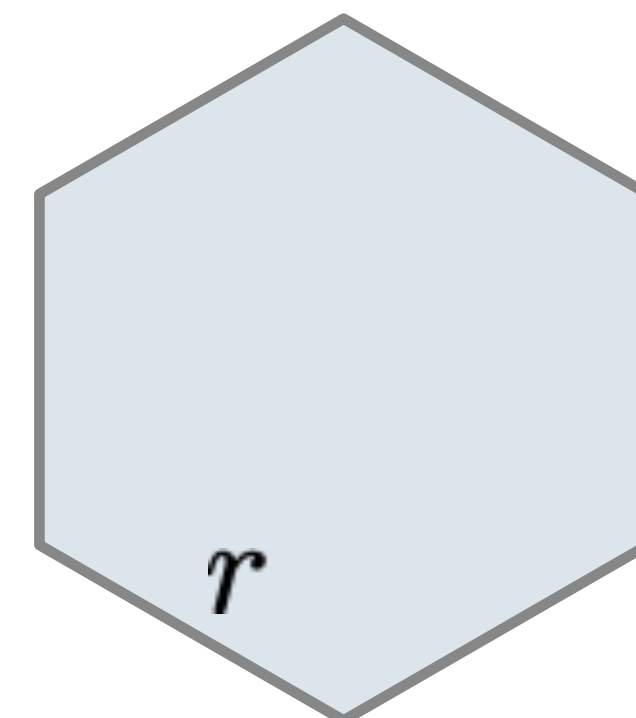
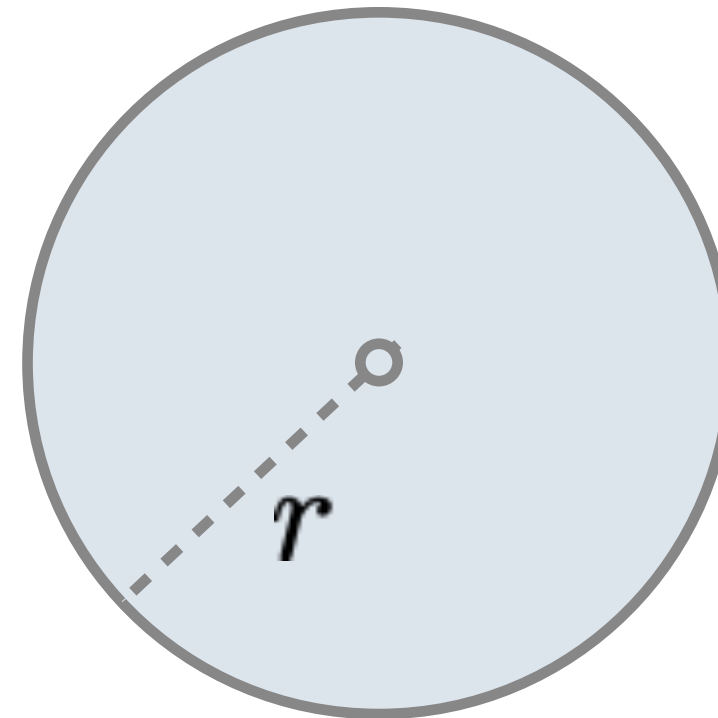
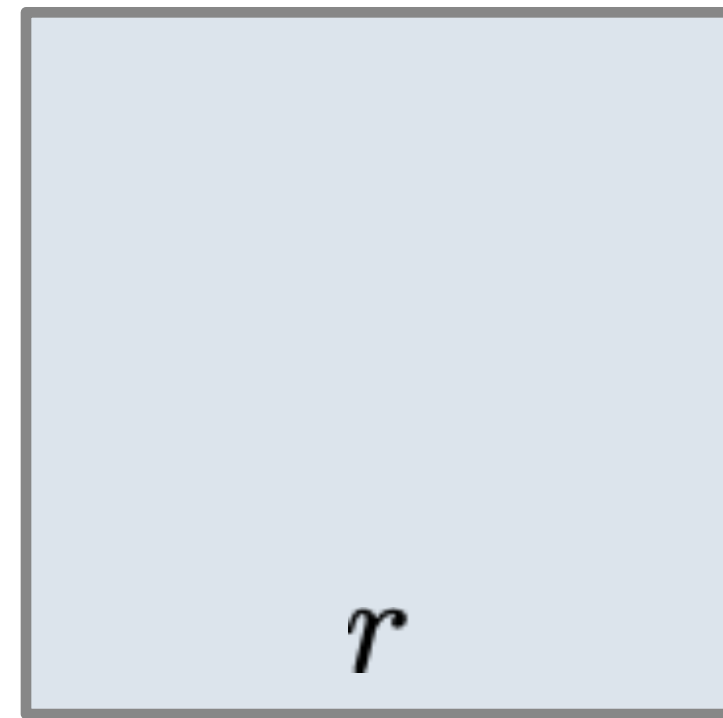


Generalization

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)
```

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

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

A formal parameter that will
be bound to a function

```
>>> summation(5, cube)
```

```
225
```

The cube function is passed
as an argument value

```
    total, k = 0, 1
```

```
    while k <= n:
```

```
        total, k = total + term(k), k + 1
```

```
    return total
```

The function bound to term
gets called here

0 + 1 + 8 + 27 + 64 + 125

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  
"""
```

The name `add_three` is bound to a function

```
def adder(k):  
    return k + n  
return adder
```

A `def` statement within another `def` statement

Can refer to names in the enclosing function

Call Expressions as Operator Expressions

