CS61A

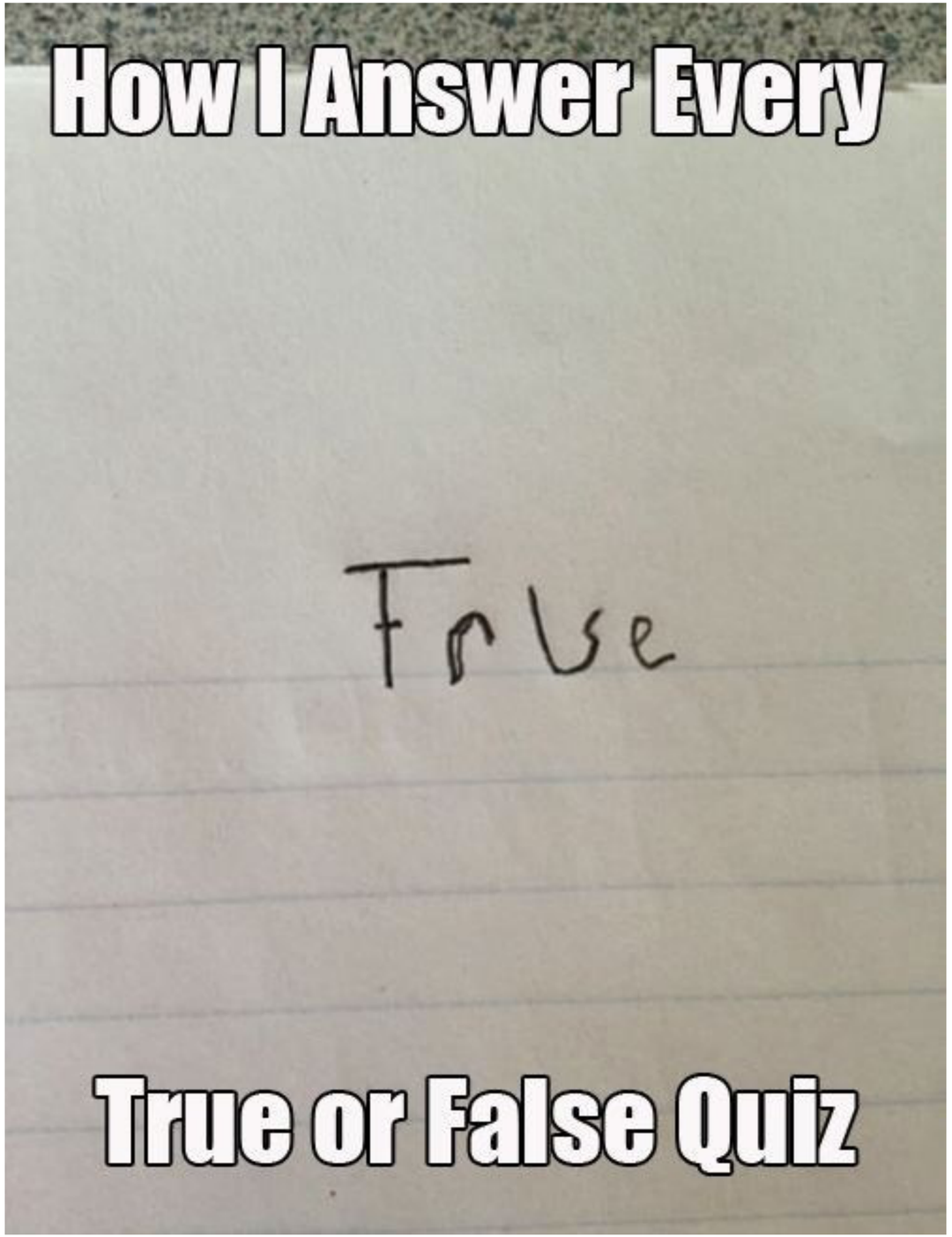
section 1

attendance

http://links.cs61a.org/jasonxu

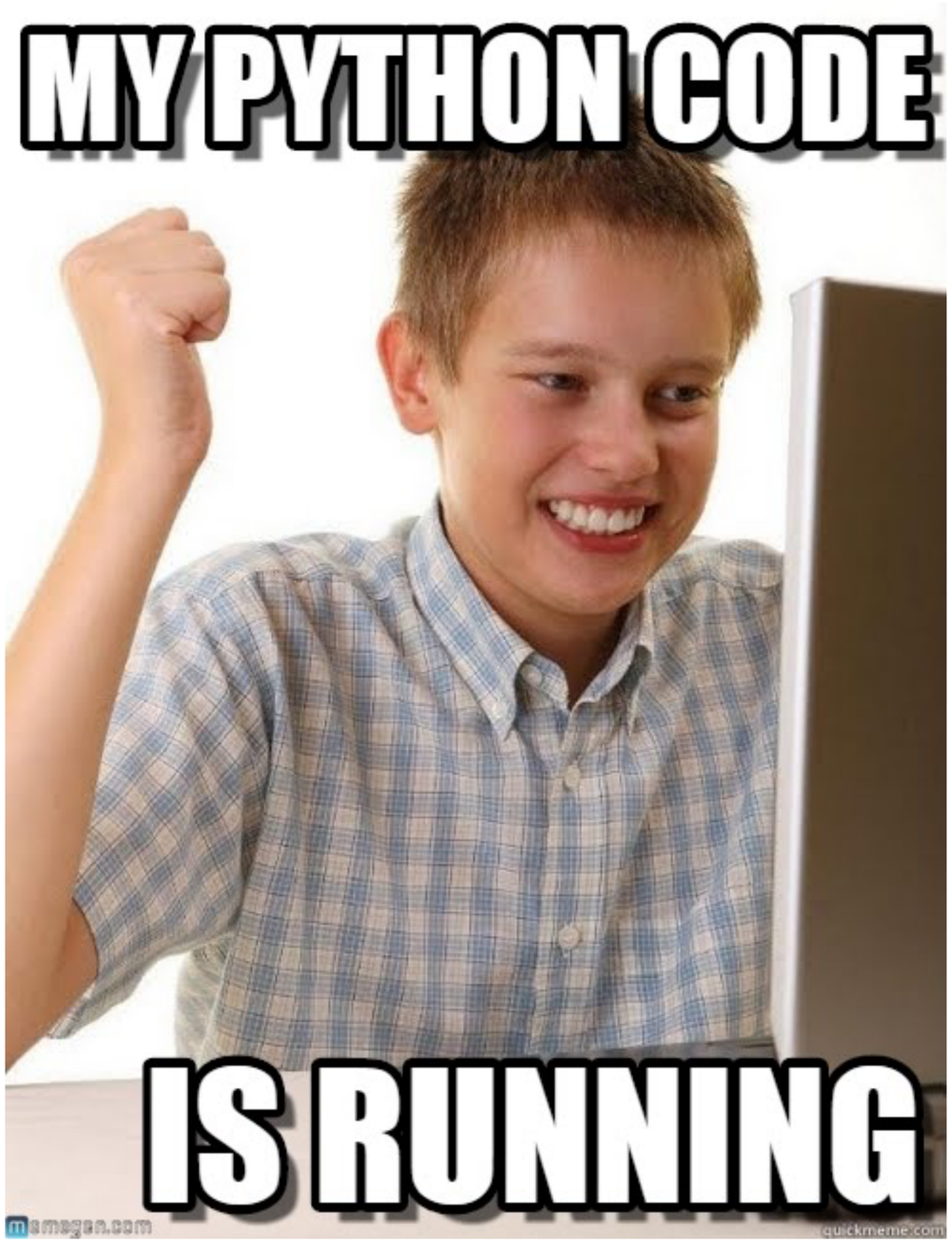
upcoming

lab 0
lab 1
hw 1
hog



CS61A

lab 0



CS61A

today

a lot of mechanical details

CS61A

today

a lot of mechanical details

not much why, just is today

CS61A

today

a lot of mechanical details

not much why, just is today

“setting up the rules of the game”

CS61A

functions

There are 2 things to consider for a function

1. Input/Output of function

2. Body of function

CS61A

functions

There are 2 things to consider for a function

1. Input/Output of function

2. Body of function

CS61A

function calls

$f(3 \times 2) =$

CS61A

function calls

$f(3 \times 2) =$ **?**

CS61A

function calls

$f(x) = 2x$ general formula, i can put in any x that is a number

CS61A

function calls

$f(x) = 2x$ general formula, i can put in any x that is a number

$x = 3 \times 2$

CS61A

function calls

$f(x) = 2x$ general formula, i can put in any x that is a number

$x = 3 \times 2$

$f(x) = 12$ i know f, x, can solve!

CS61A

function calls

$f(x) = 2x$ $\qquad$ $double(z) = 2z$

$x = 3 \times 2$ $\qquad$ $x = 3 \times 2$

$f(x) = 12$ $\qquad$ $double(x) = 12$

CS61A

function calls

$$f(x) = 2x \qquad double(z) = 2z \qquad double(z) = 2z$$

$$x = 3 \times 2 \qquad x = 3 \times 2 \qquad double(3 \times 2)$$

$$f(x) = 12 \qquad double(x) = 12 \qquad = 12$$

CS61A

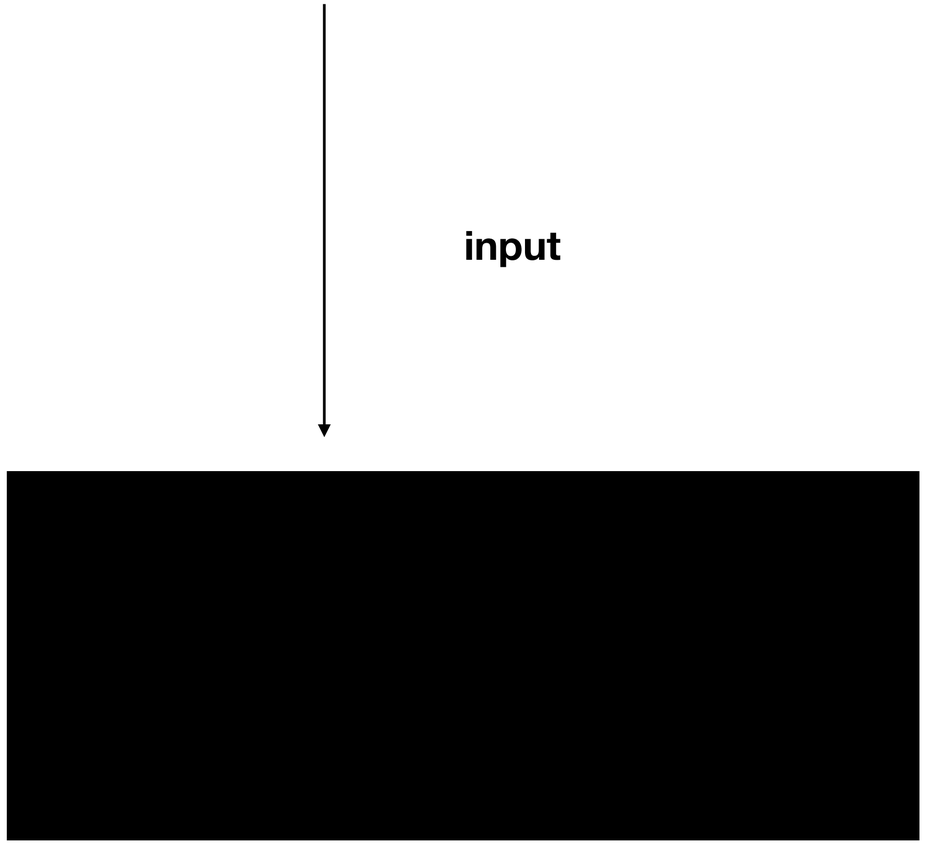
black boxes

we’re going to use these to see an abstract picture of functions

CS61A

black boxes

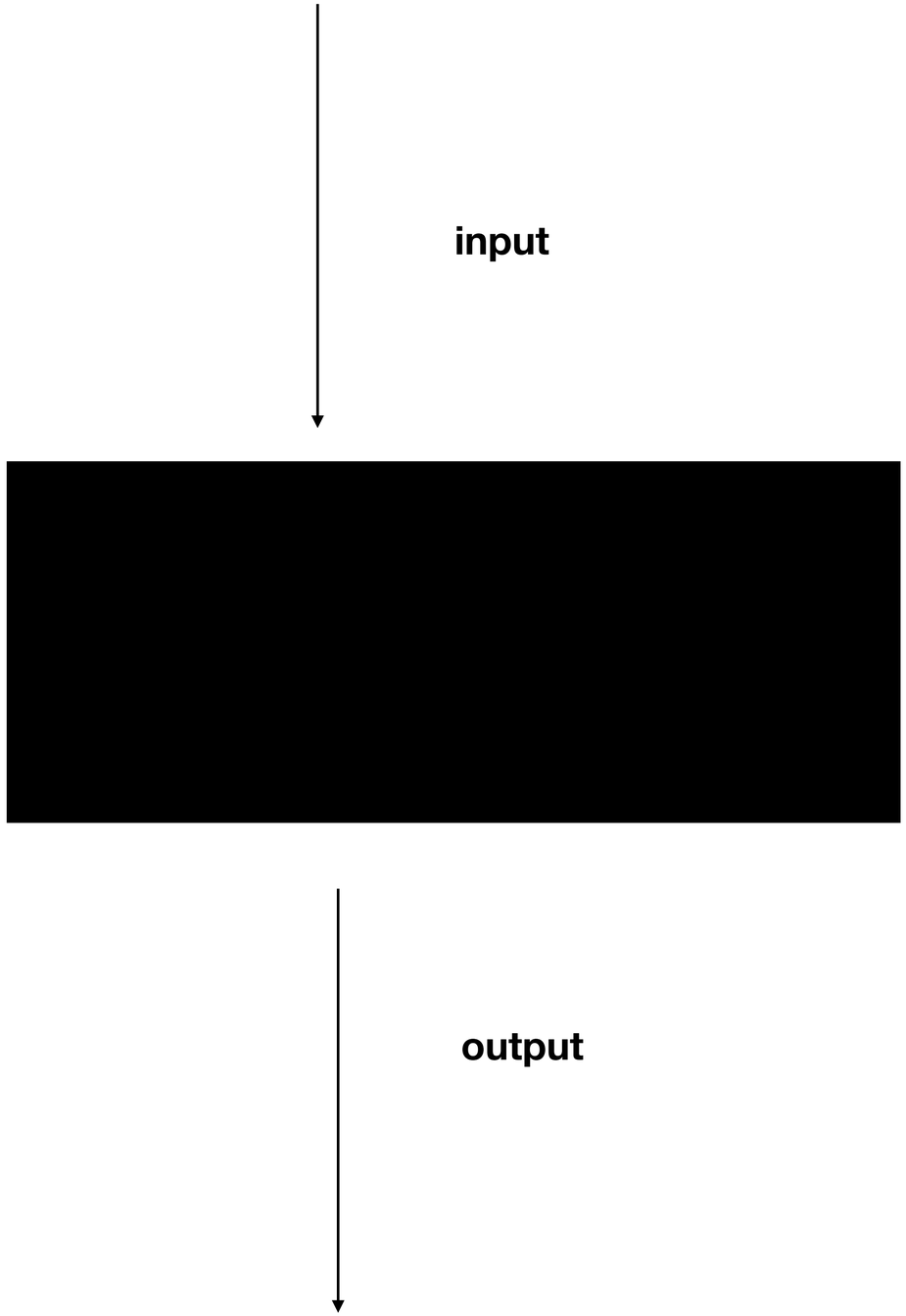
we’re going to use these to see an abstract picture of functions



CS61A

black boxes

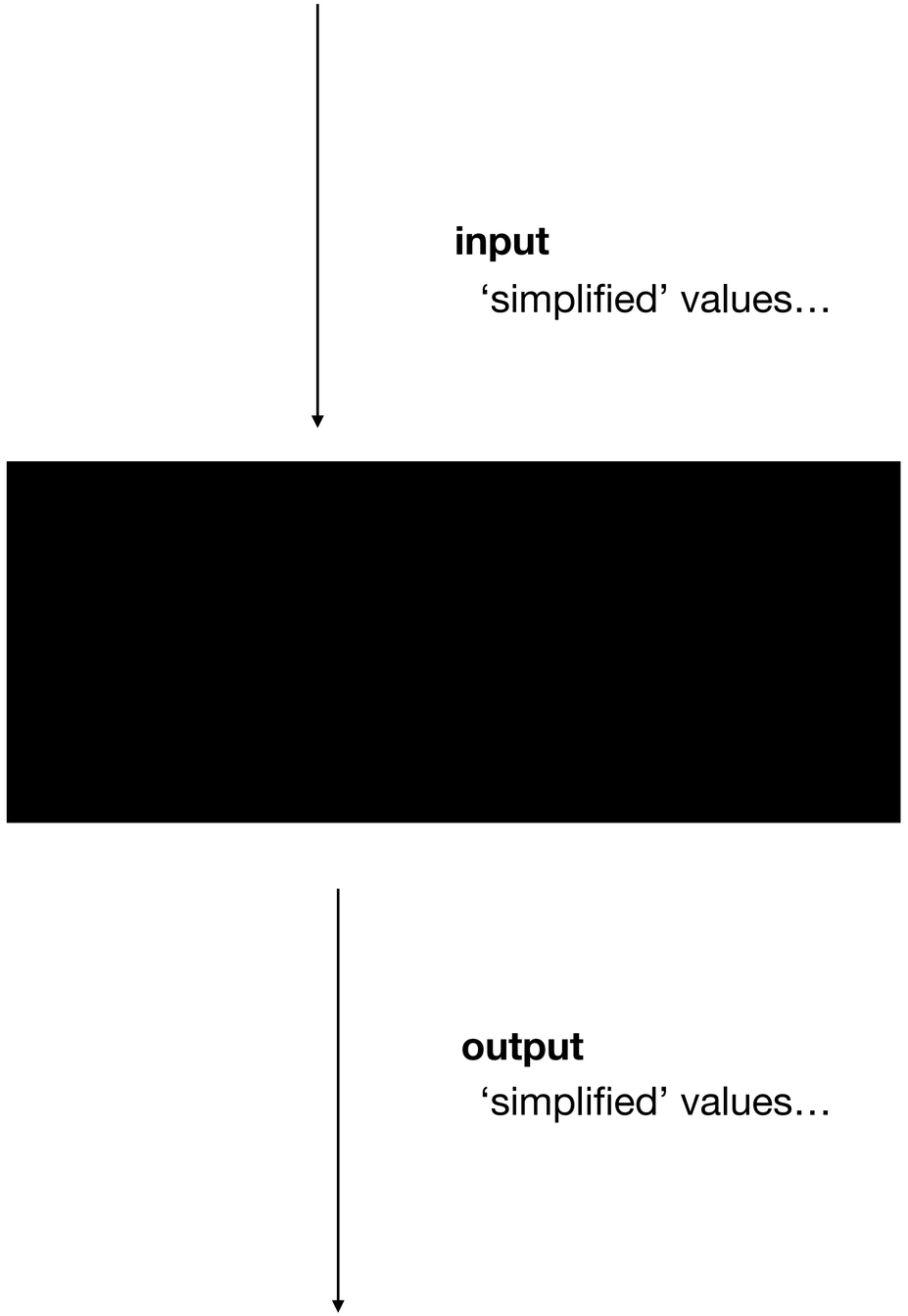
we’re going to use these to see an abstract picture of functions



CS61A

black boxes

we’re going to use these to see an abstract picture of functions



CS61A

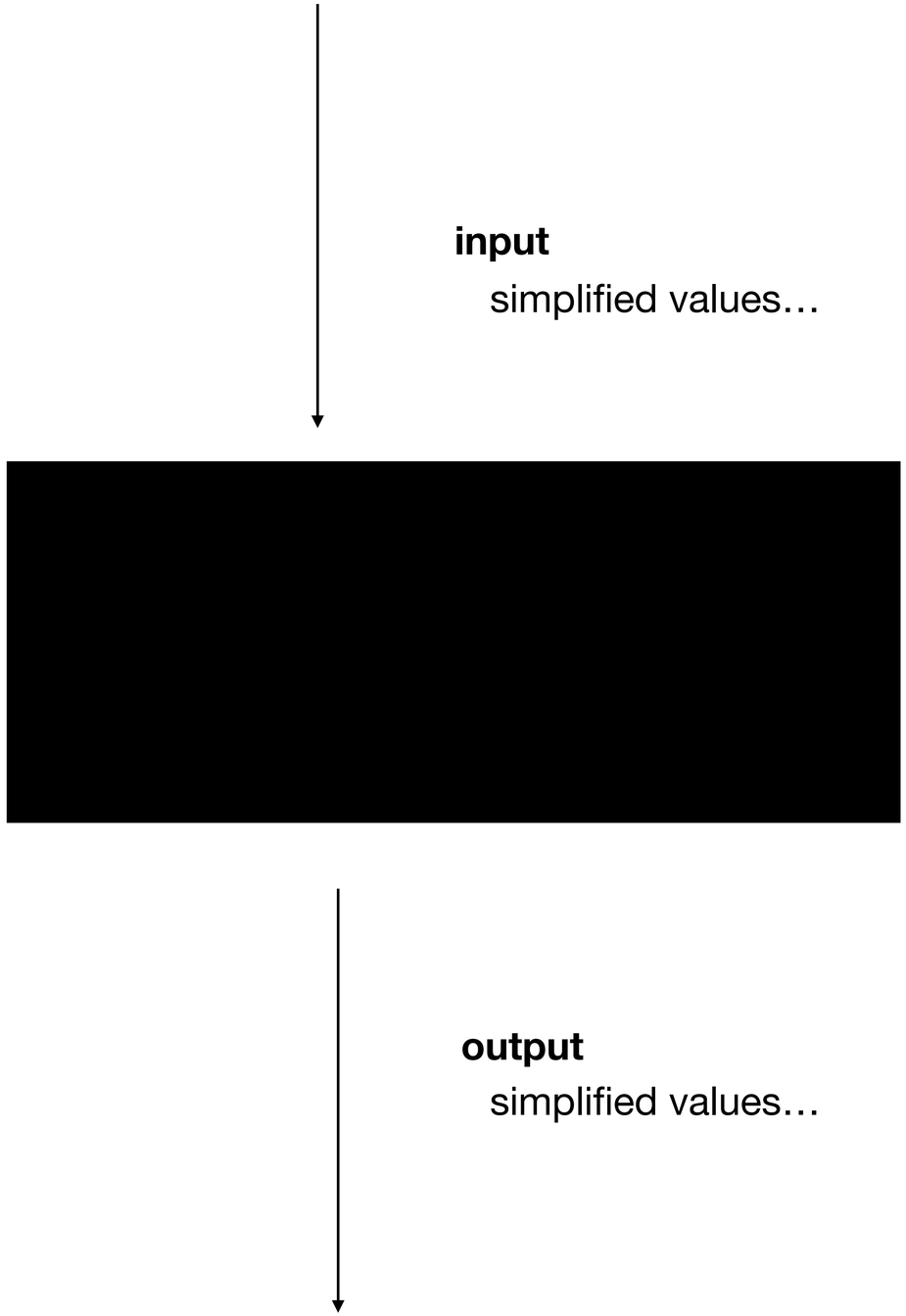
values

Type	Values	Literals (Denotations)
Integers	$0 \quad -1 \quad 16 \quad 13$	`0 -1 0o20 0b1101`
	36893488147419103232	`0x20000000000000000`
Boolean (truth) values	true, false	`True False`
"Null"		`None`
Functions		`operator.add, operator.mul, operator.lt, operator.eq`
Strings	Say "Hello"	`"Say \"Hello\""`

CS61A

black boxes

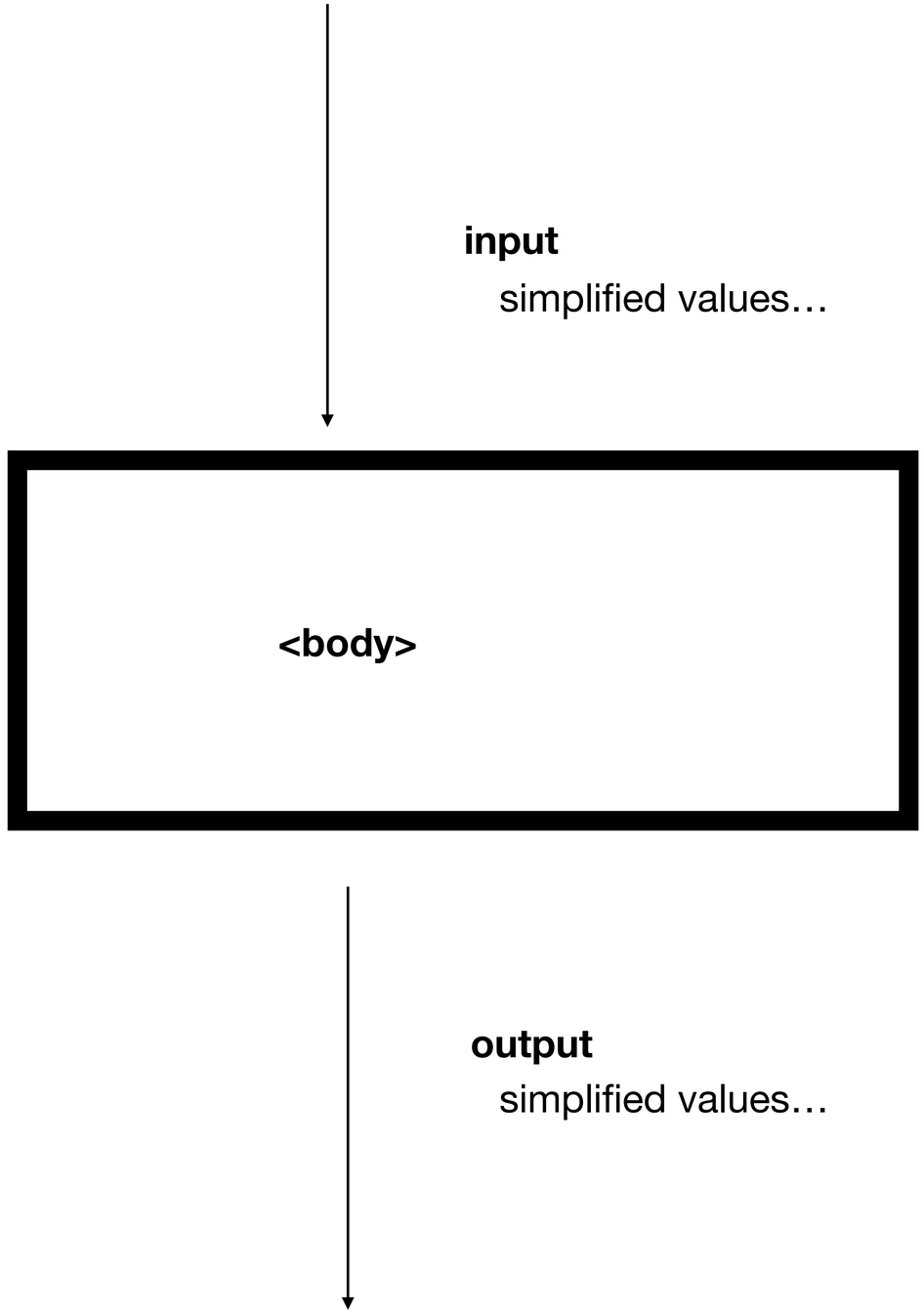
we’re going to use these to see an abstract picture of functions



CS61A

black boxes

we're going to use these to see an abstract picture of functions



CS61A

function calls

```
> max(10 + 5, 9, double(18))
36
```

CS61A

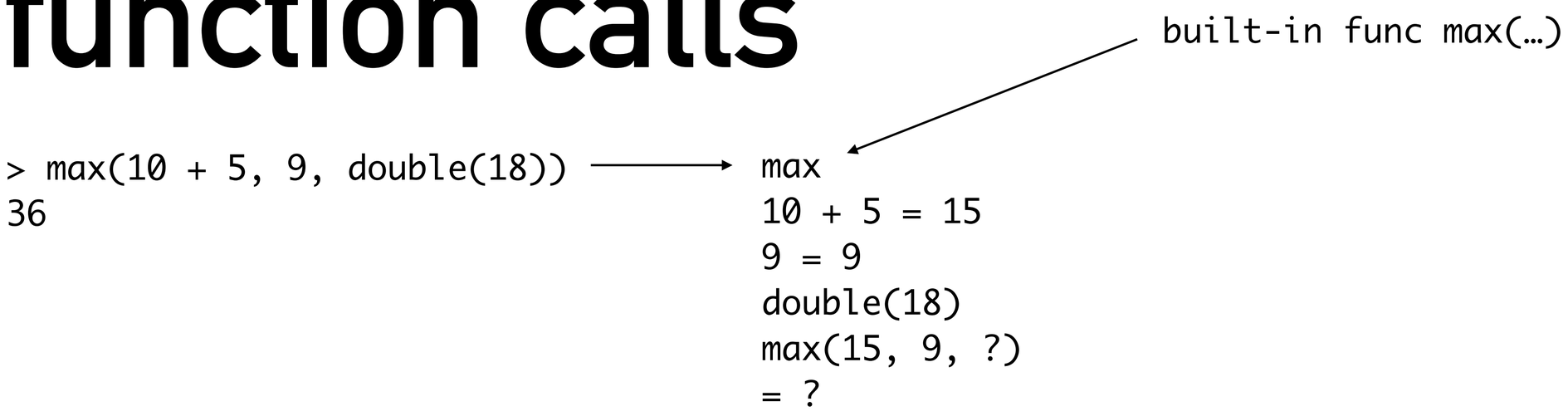
function calls

```
> max(10 + 5, 9, double(18))
36
```

```
max
10 + 5 = 15
9 = 9
double(18) = 36
max(15, 9, 36)
= 36
```

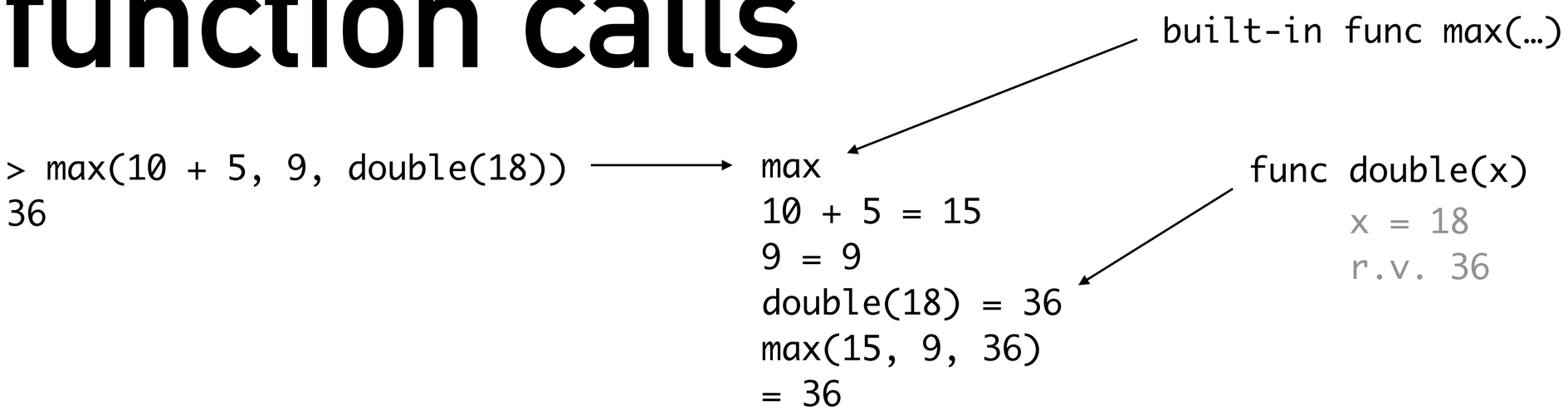
CS61A

function calls



CS61A

function calls



CS61A

function calls

```
> max(10 + 5, 9, double(18))
36
```



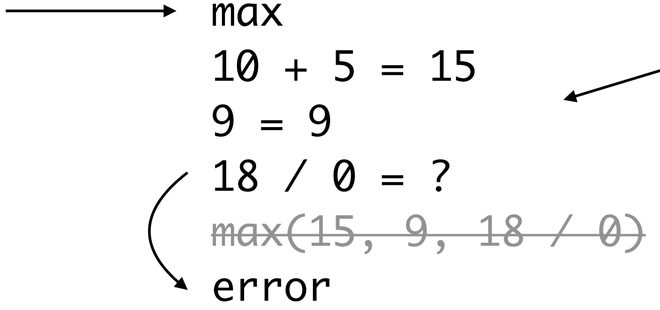
```
max
10 + 5 = 15
9 = 9
double(18) = 36
max(15, 9, 36)
= 36
```

if any part of this breaks,
you get an error and stops

CS61A

function calls

```
> max(10 + 5, 9, 18 / 0)
error
```



if any part of this breaks,
you get an error and stops

typing an error != will error

EVALULATE OPERATOR
EVALUATE OPERANDS
APPLY OPERATOR

```
double(2 + 3)
double(...)
2 + 3
5
double(5)
10
```

CS61A

returning

return stops procedure and outputs something

print is an action, function

CS61A

returning

return stops procedure and outputs something

print is an action, function

```
def showFivePrint():
    x = 2 + 3
    print(x)
```

```
def showFiveReturn():
    x = 2 + 3
    return x
```

CS61A

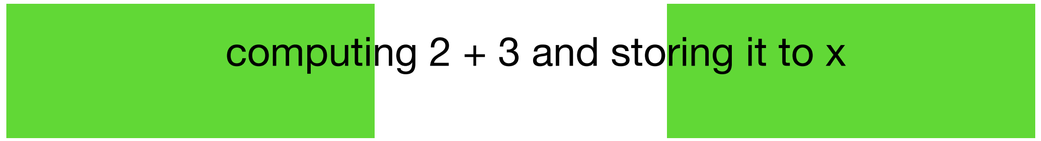
returning

return stops procedure and outputs something

print is an action, function

```
def showFivePrint():
    x = 2 + 3
    print(x)
```

```
def showFiveReturn():
    x = 2 + 3
    return x
```



CS61A

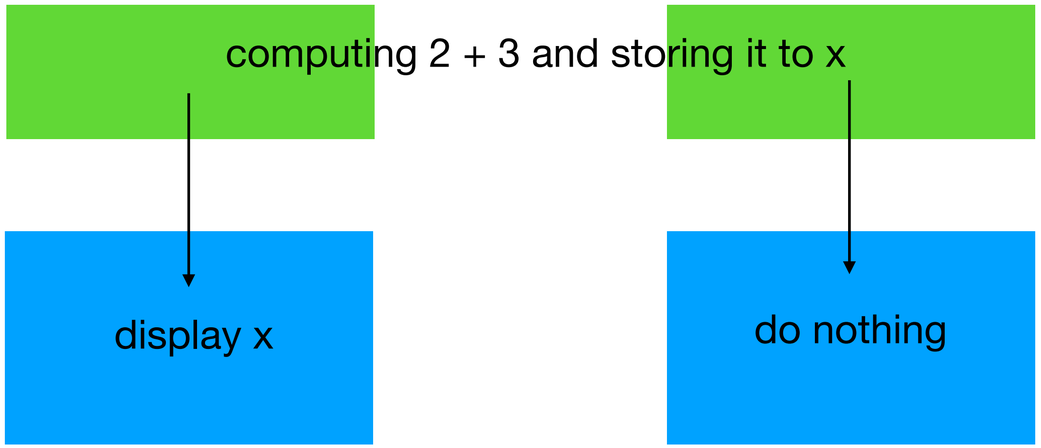
returning

return stops procedure and outputs something

print is an action, function

```
def showFivePrint():
    x = 2 + 3
    print(x)
```

```
def showFiveReturn():
    x = 2 + 3
    return x
```



CS61A

returning

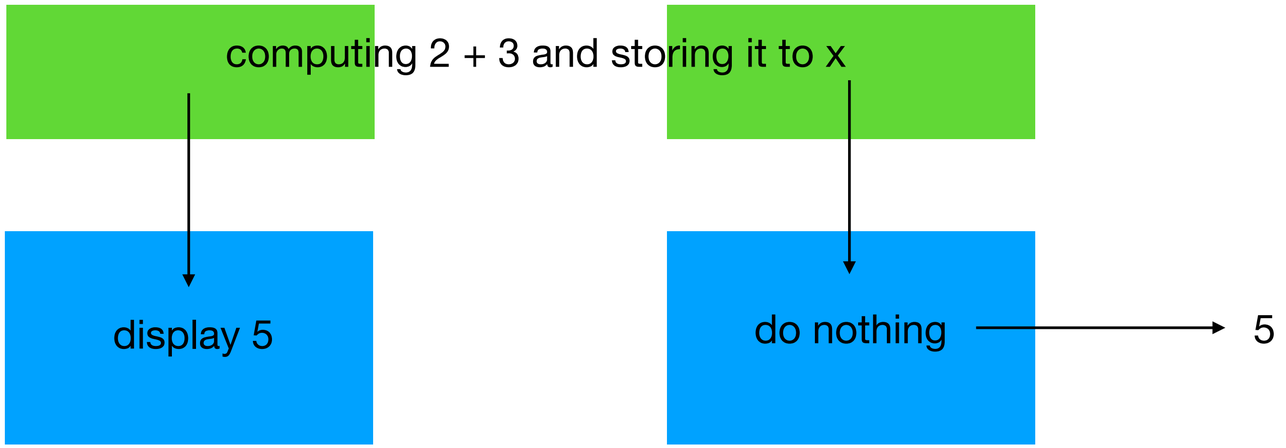
return stops procedure and outputs something

print is an action, function

every function has a return at end of None

```
def showFivePrint():
    x = 2 + 3
    print(x)
```

```
def showFiveReturn():
    x = 2 + 3
    return x
```



CS61A

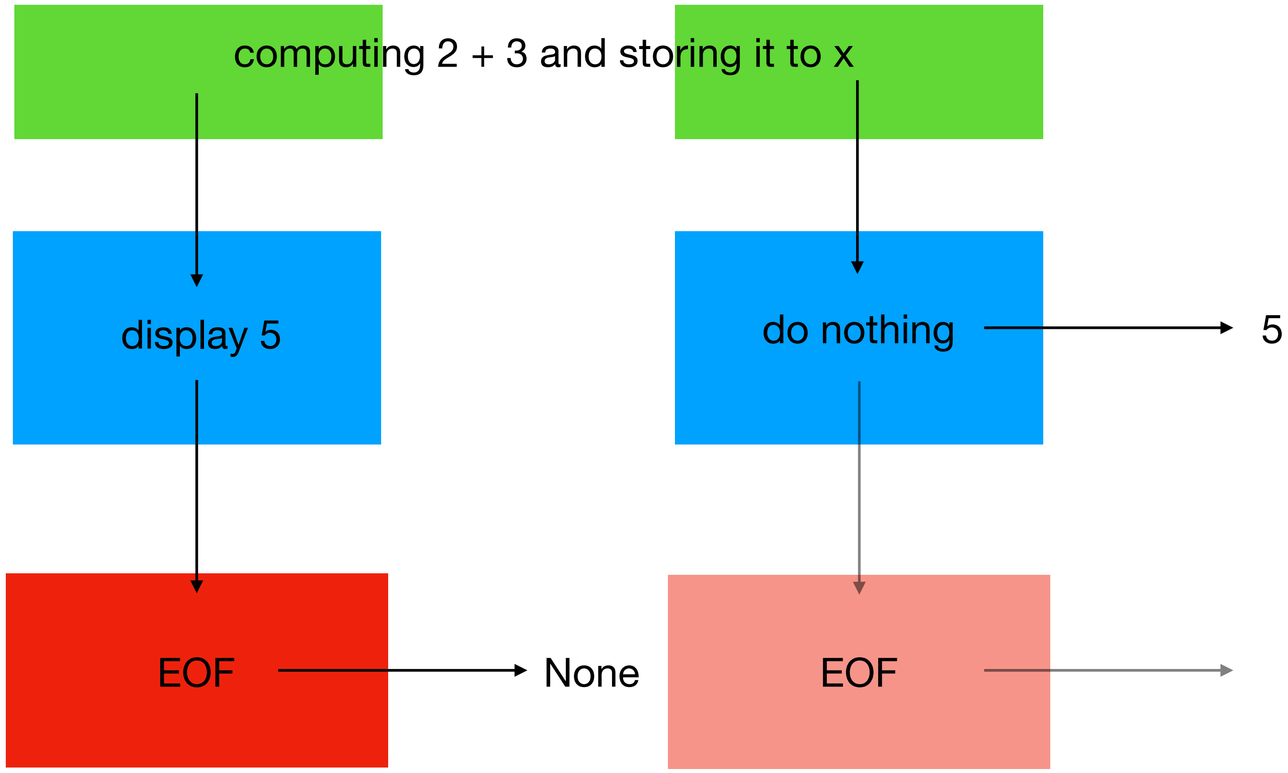
returning

return stops procedure and outputs something

print is an action, function

```
def showFivePrint():
    x = 2 + 3
    print(x)
    return
```

```
def showFiveReturn():
    x = 2 + 3
    return x
```



CS61A

returning

return stops procedure and outputs something

print is an action, function

return only 1 thing but can also return tuples (pair structures)

```
def showFivePrint():
    x = 2 + 3
    print(x)
```

```
> val = showFivePrint()
5
> val
> val is None
True
```

```
def showFiveReturn():
    x = 2 + 3
    return x
```

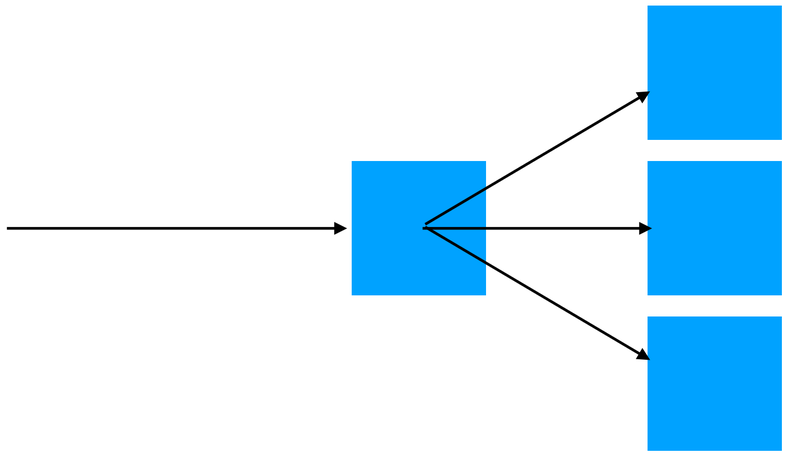
```
> val = showFiveReturn()
> val
5
```

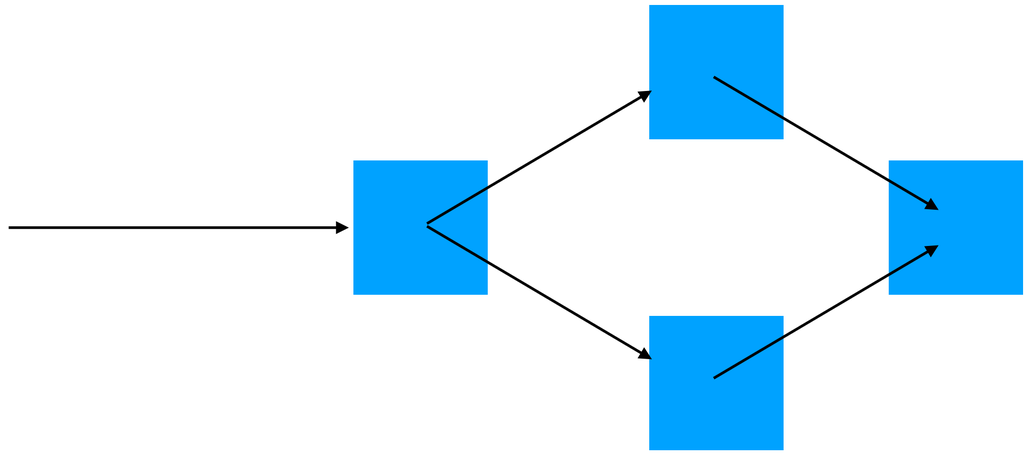
CS61A

controls

```
if <predicate>:
  <do this>
elif <predicate>:
  <do this>
else:
  <do this>
```

```
if <predicate>:
  <do this>
elif <predicate>:
  <do this>
<do this>
```





CS61A

booleans

bools have rules

TRUE	FALSE
1	0
‘non-empty’ values	‘empty’ values
	None

CS61A

boolean operators

and	<a> and <b> …
or	<a> or <b> …
not	not <a>

a	b
1	0
1	1
0	0

CS61A

boolean operators

`and`	`<a> and <b> …`
`or`	`<a> or <b> …`
`not`	`not <a>`

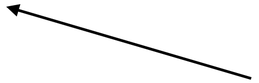
a	b	
1	0	
1	1	**which 1?**
0	0	

CS61A

boolean operators another approach

if it’s sunny and not hot

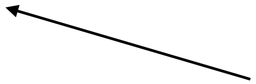
i will go for a run



only will do so when ‘<True> and <True>’

if it’s sunny or not hot

i will go for a run



will do so when either condition is true

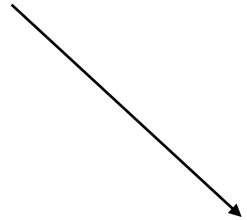
CS61A

booleans

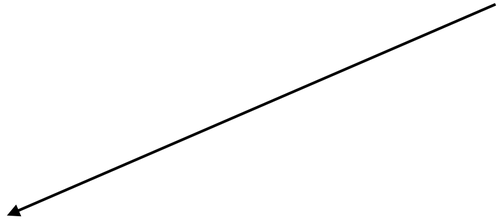
bools have rules

`and` looks for `False`

`or` looks for `True`



short circuit!

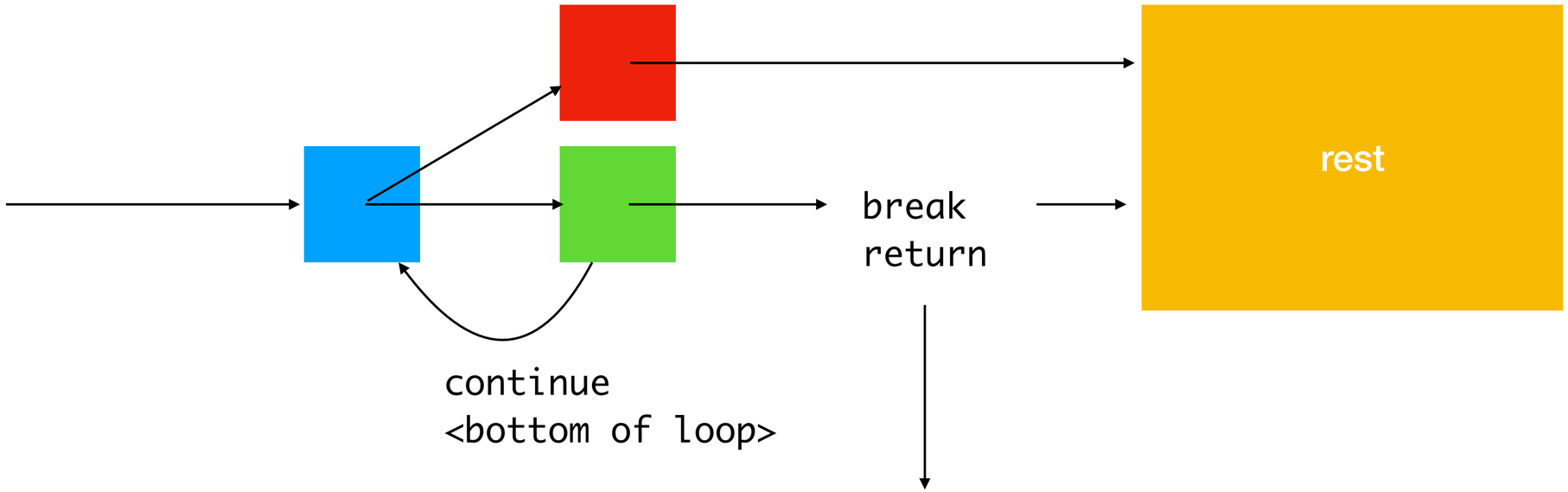


we can process faster

CS61A

controls

```
while <this>:
  <do this>
```



CS61A

environment diagrams

worthwhile to learn

~ > 30% of MT1

all these slides… maybe too much but describe how to think in cs

which is what env diagrams do!

a lot of rules… but internalize them so they become intuitive