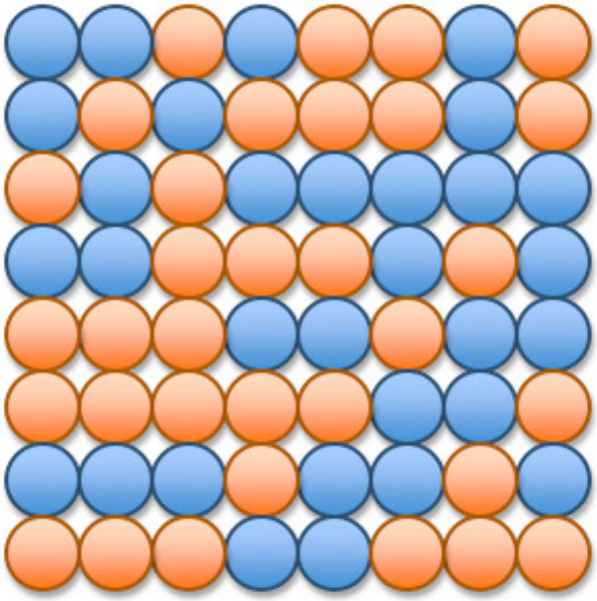


UNIT 3B
Implementing Algorithms

Announcements

- Check the grades for lab1, PA1, PS1 on autolab
- Hope you submitted PA2 last night (no exceptions)
- PS2 is due Friday Feb 1.
- If you cannot find the CA in 3rd floor during office hours, email them immediately
- Sign up for piazza to see Q&A

Algorithmic Thinking Review

- An algorithm is a precise set of Rules
- What are the properties of correct algorithms? Correct order, clear, valid output

- A program is an implementation of an algorithm
- How do you test an algorithm?

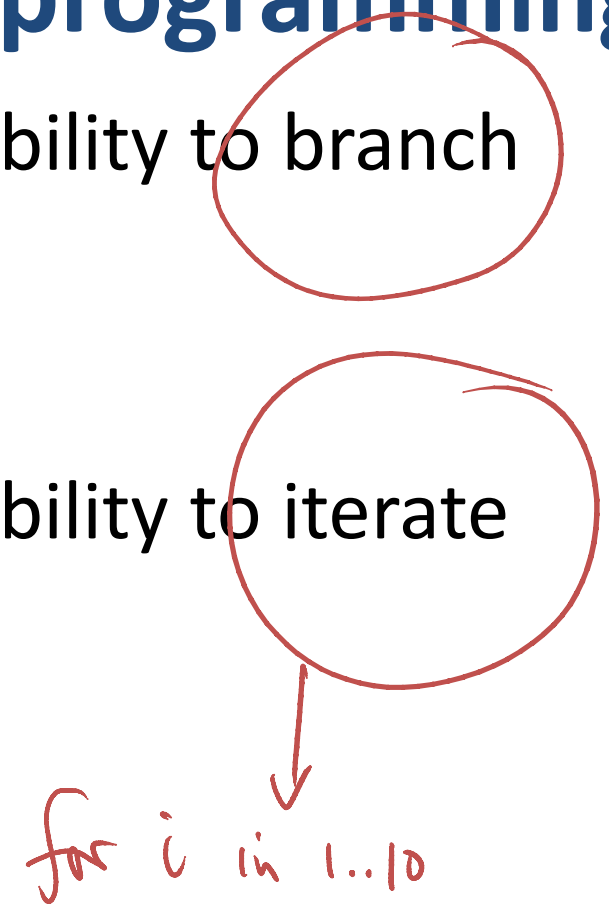
run it gcd(a,b)

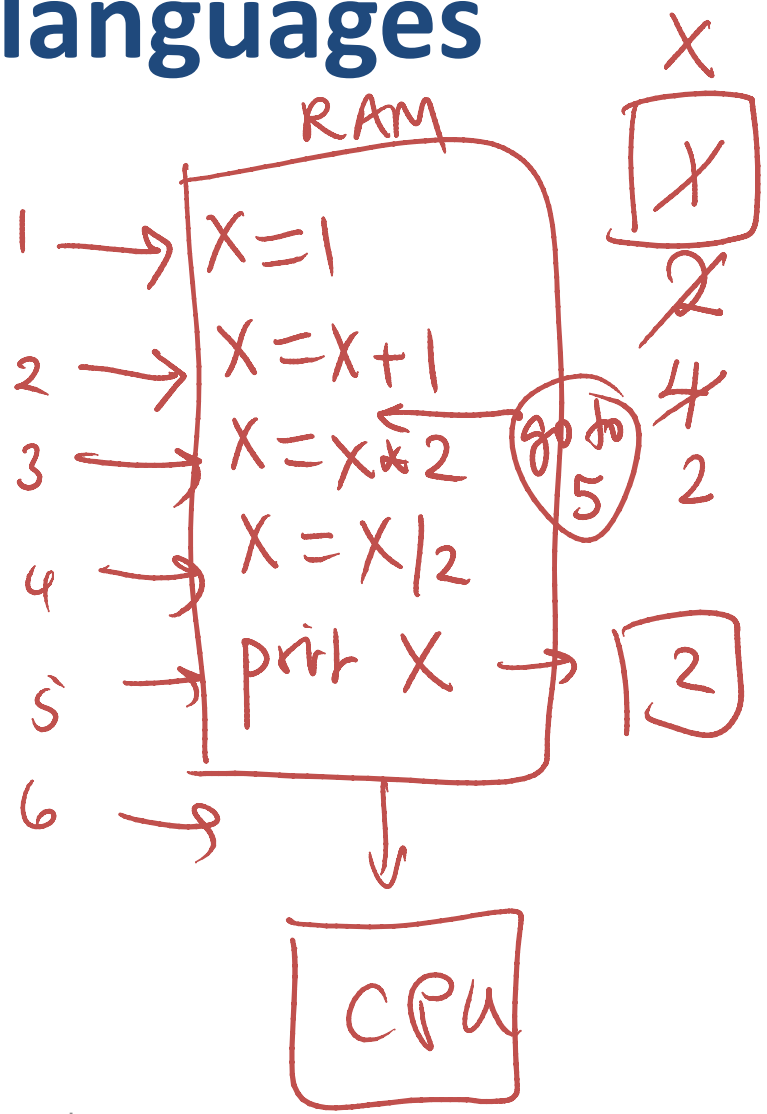
Classes of inputs
→ both positive
→ both negative
→ one is zero

Tools for Implementing algorithms

Two key constructs needed in all programming languages

- The ability to branch
- The ability to iterate





Branching

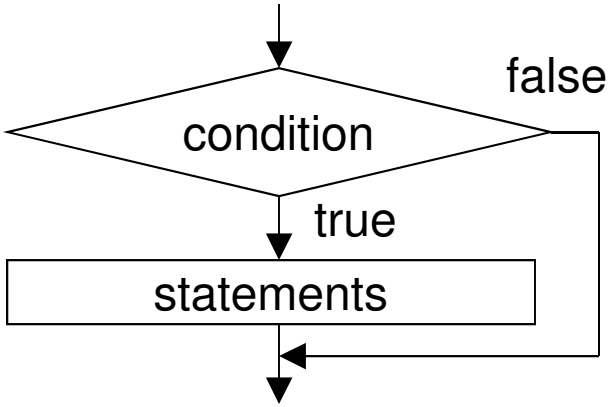
What is branching?

- Programs usually execute instructions in sequence
- But sometimes programs must jump to a different instruction based on a boolean condition (true | false)
- Programming languages have constructs that lets us jump on a condition

`if` statement

Format:

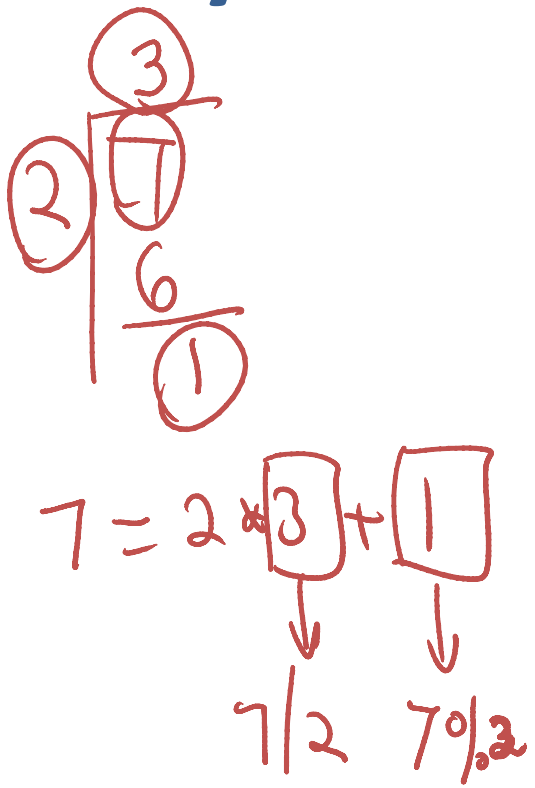
```
if bool_condition then
    statement_list
end
```



Write a function that determines if a number is divisible by 3

```
def div3(x)
    if x%3 == 0 then
        return "true"
    end
end
```

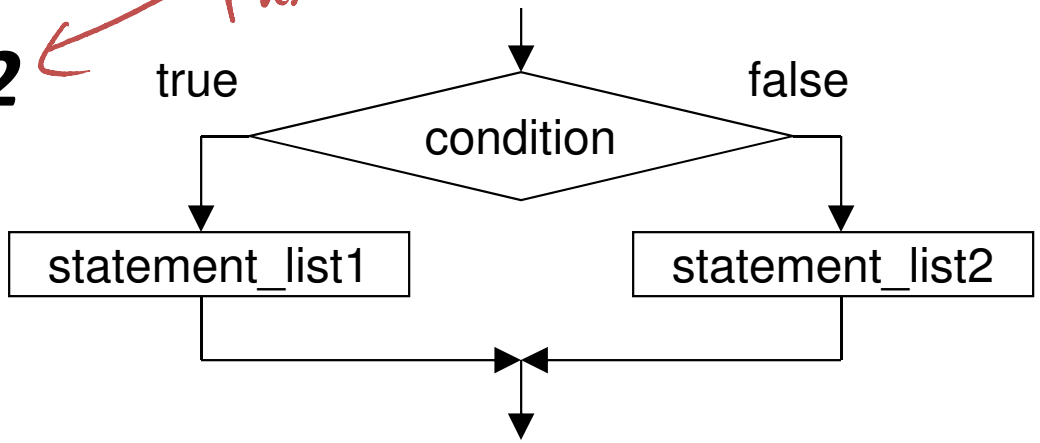
compare



`if/else` statement

Format:

```
if bool_condition then
    statement_list1
else
    statement_list2
end
```



Write a function to find the max of two numbers

```
def max(a,b)
    if (a>b) then
        return a
    else
        return b
    end
end
```

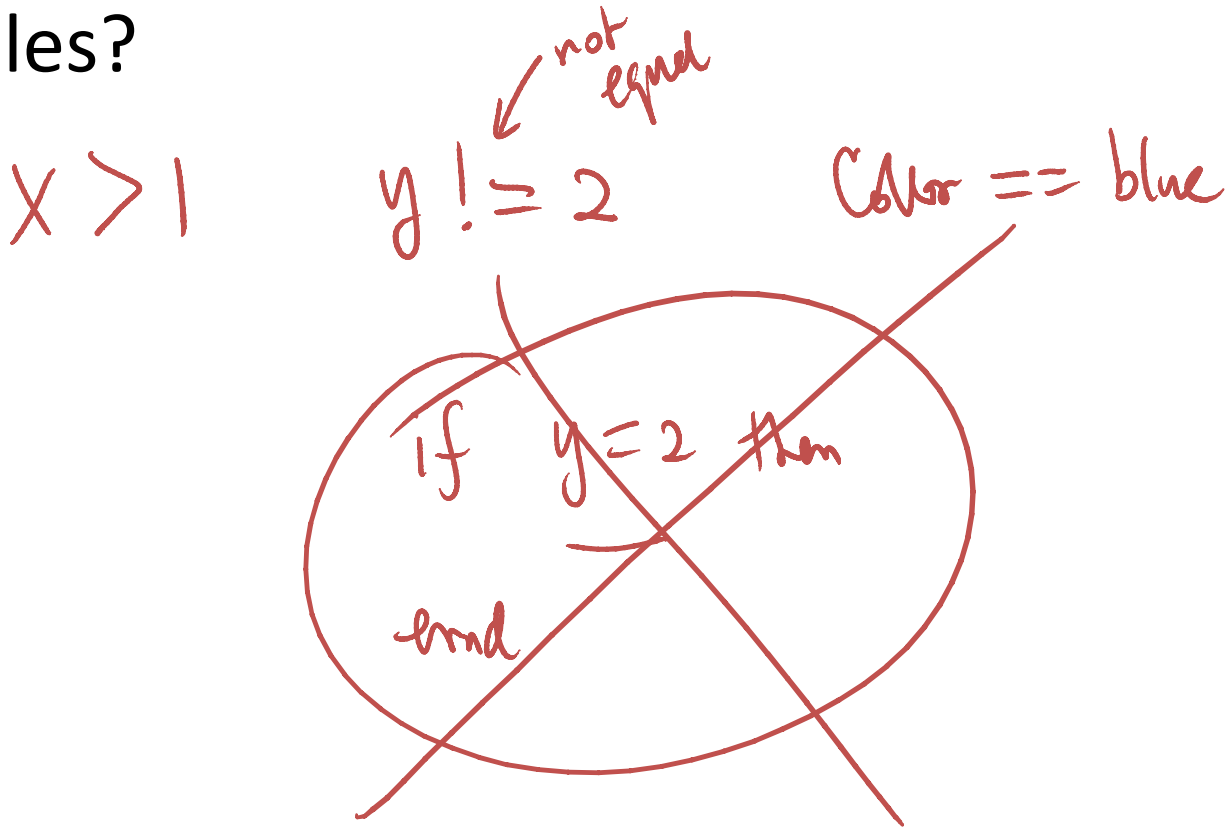
12, 5, 7

max(max(12,5) , 7)

12

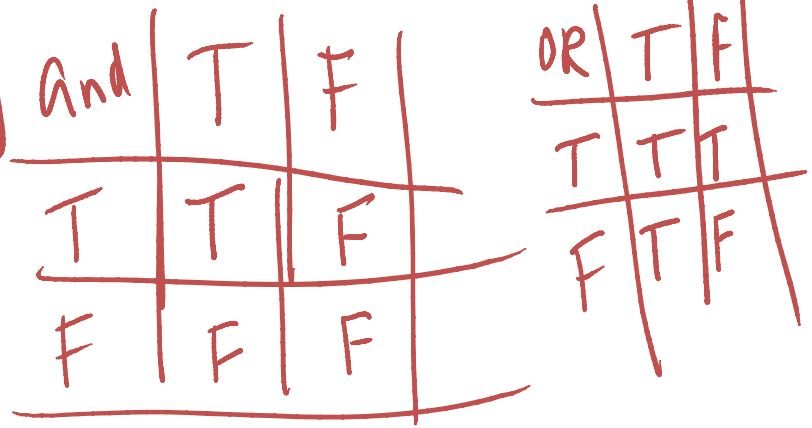
Boolean Statements

- A boolean statement is either TRUE or FALSE
- Examples?



Boolean Operators

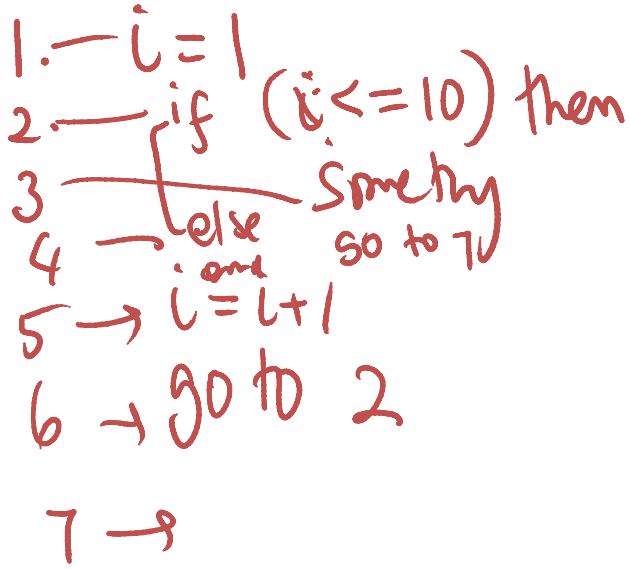
- Two or more bool statements can be combined using boolean operators
- Boolean operators can only be applied to boolean variables. i.e. variables that are true or false
- Ruby boolean operators
  - AND operator
  - OR operator
  - NOT operator



iteration

Iteration

- Iteration is a sort of branching
- for i in 1..10 do something end

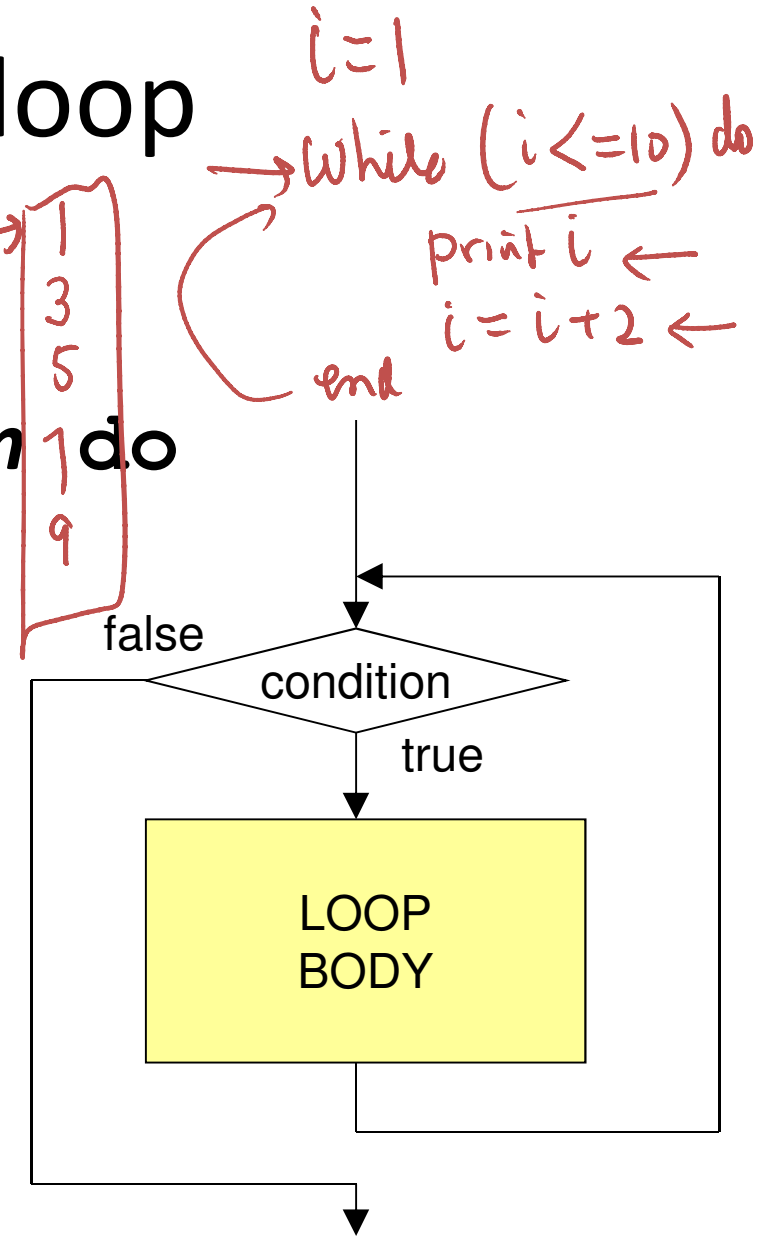


`while` loop

Format:

```
while bool_condition do
    loop body
end
```

one or more instructions to be repeated

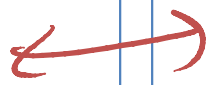


If the loop condition becomes false during the loop body, the loop body still runs to completion before we exit the loop and go on with the next step.

While vs. For Loops

#for loops

```
for i in 1..10 do
    Stat
end
```



while loop

```
i = 1
while (i<=10) do
    Stat
    i = i+1
end
```

Going backwards

10..1

#for loops

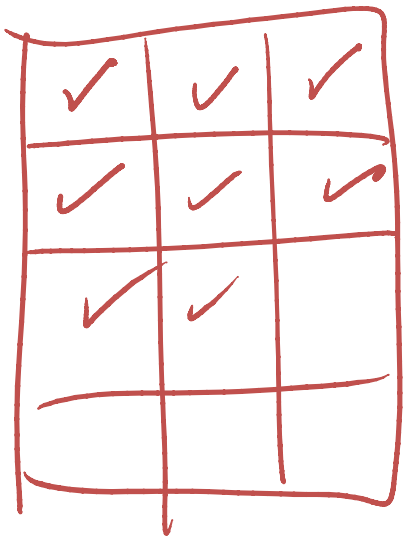
for i in 10..1 ?

while loop

```
i=10;
while (i>=1) do
    Start
    i=i-1
end
```

Nested Loops

- Table calculation



```
for i in 1..10 do
    for j in 1..5 do
        print i+j
    end
    print "\n"
end
```

outer

row

inner

col

```
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
  ⋮
11 12 13 14 15
```

i = 1 → j = 1..5

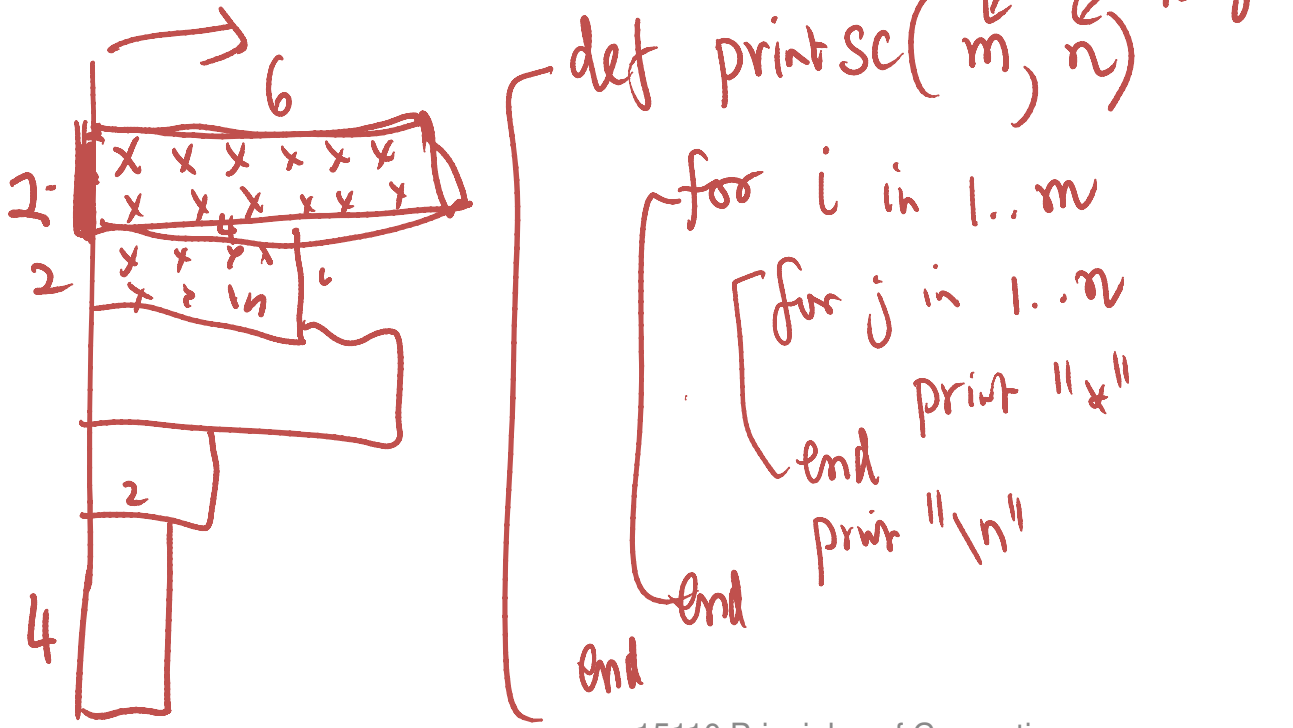
i = 2 → j = 1..5

⋮

i = 10 → j = 1..5

Creating Art

- How would you draw a skyscraper?
- How would you combine them to create a skyline?



Representing Lists as Arrays

Array types

- One dimensional arrays

- Two dimensional arrays

Arrays

- Arrays can hold any kind of object:

```
a = [8,  “strawberry”,  -5.062,  false]

a[0]    =>  8
a[1]    =>  “strawberry”

a.length  =>  4
```

Ruby numbers items from 0!

- The empty array is written as [ ]

Converting a Range to an Array

```
r = 3..8
r.to_a    =>  [3, 4, 5, 6, 7, 8]

(8..3).to_a  =>  [ ]

s = "gu" .. "he"
s.to_a    =>  ["gu", "gv", "gw", "gx", "gy",
                "gz", "ha", "hb", "hc", "hd", "he"]
```

The *to_a method uses succ to generate elements.*