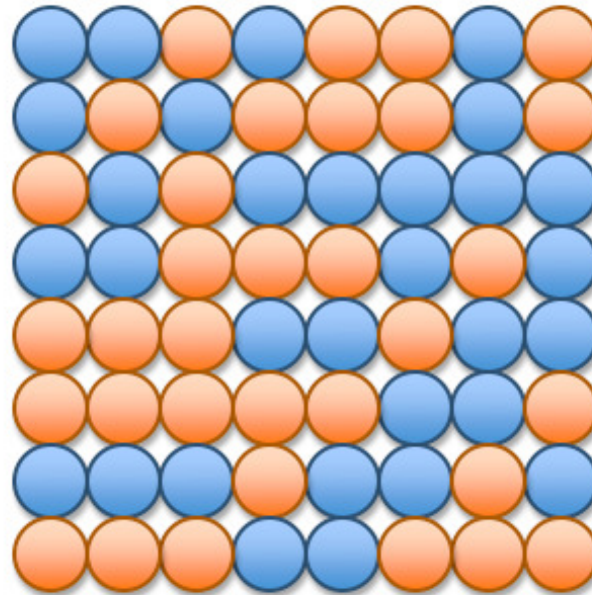




UNIT 3B

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? specify, expected output
- A program is an implementation of an algorithm
- How do you test an algorithm?
 $\text{gcd}(a,b)$ both positive ← test cases
 both negative ←

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
- Programming languages have constructs that lets us jump on a condition

1. start
2.
3. if $x > 1$ goto 5
4.
5.
6. end



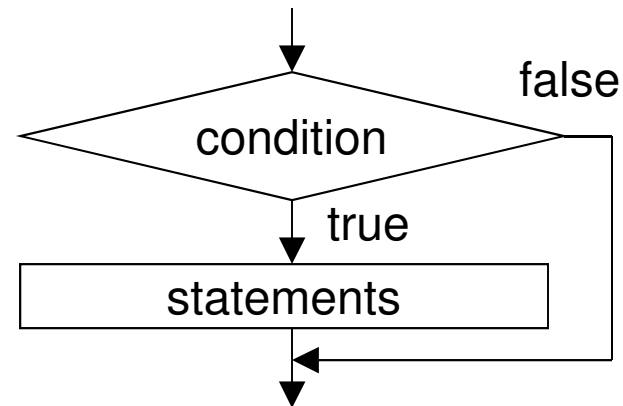
if statement

Format:

if *bool_condition* **then**
statement_list

end

if $x > y$ then
 $max = x$
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
```

% - remainder

/ - quotient

$$7 = 2 \times 3 + 1$$

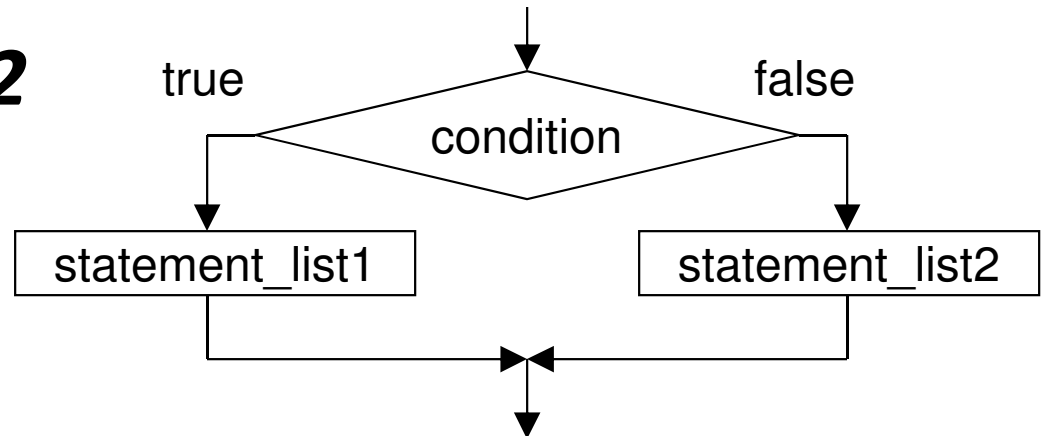
$$7/2 = 3$$

$$7\%2 = 1$$

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(x, y)
  if (x > y) then
    return x
  else
    return y
  end
end
```

$\max(\max(x, y), z) \rightarrow 8$

↓

8

Boolean Statements

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

$x > y$

$x \neq y$ *not equal*

$\text{Color} == \text{"Blue"}$ *equal*

if $x == y$ then
end

?
Don't do this

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

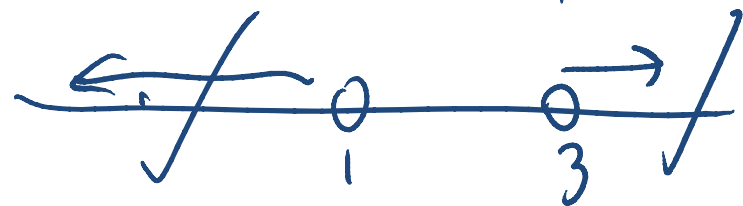
— **OR** operator

— **NOT** operator

$$\neg(x > 3) \Rightarrow x \leq 3$$

$$x > 3 \text{ and } x < 1 \rightarrow \text{false}$$

$$x > 3 \text{ or } x < 1$$



iteration

Iteration

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

1. $i = 1$
2. if ($i > 10$) go to 5
3. something
4. $i = i + 1$
5. end go to 2

2 4 6 8 10

while loop

Format:

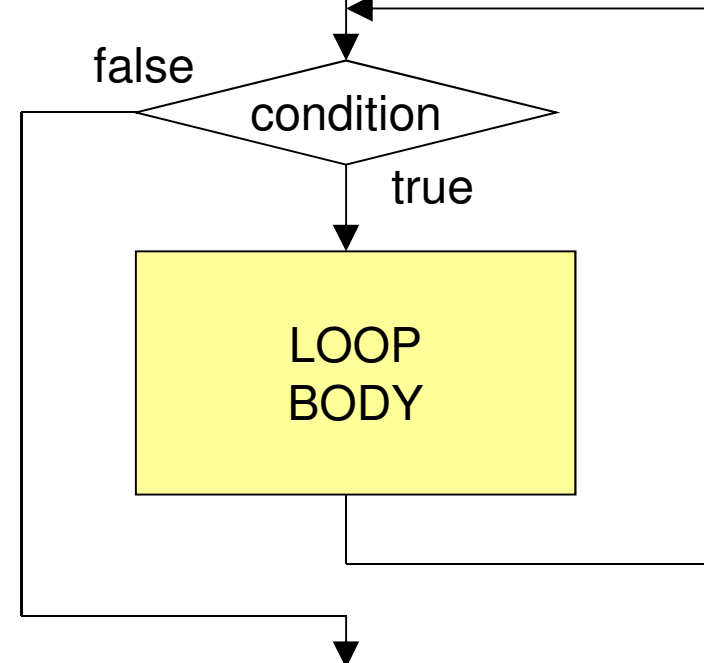
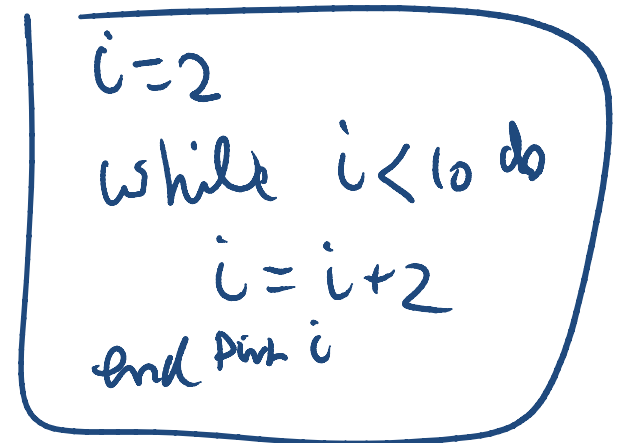
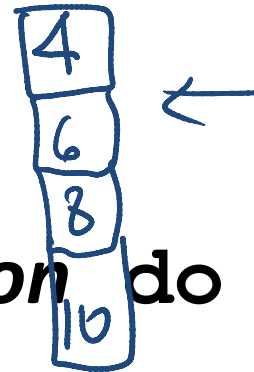
```
while bool_condition do
```

```
  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
```

~~XXXX~~
<=

Going backwards

10, 9, ..., 1

#for loops

Challenge

while loop

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



7

Nested Loops

- Table calculation

2	3	4	5	6
3	4	5	6	7
4	5	6	7	8
5	6	7	8	9

```
for i in 1..10 do
  for j in 1..5 do
    Print(i+j).to_s()+"\n"
  end
end
```

outer

inner

Convert int to string

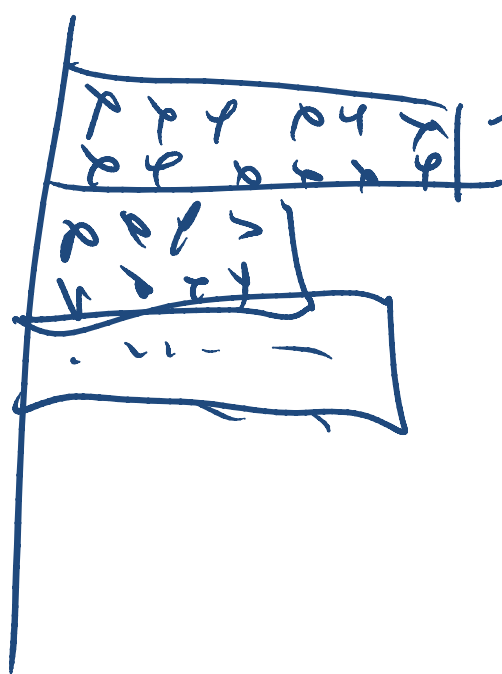
$i = 1$
 $i = 2$
 $\vdots$
 $i = 10$

$j = 1..5$
 $j = 1..5$
 $j = 1..5$

15

Creating Art

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



```
for i in 1..m do
  for j in 1..n do
    print "x"
  end
  print "\n"
end
```

row

col

n

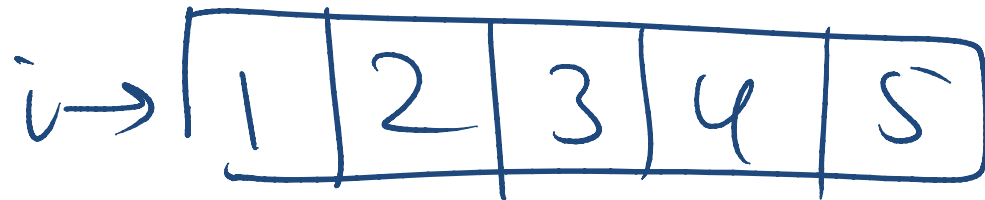
m

A hand-drawn sketch of a skyline. It consists of four rows of 'x' characters. The first row has 5 'x's, the second row has 4 'x's, the third row has 3 'x's, and the fourth row has 4 'x's. The sketch is enclosed in a rectangular frame with a vertical line on the left and a horizontal line at the bottom. A line from the text 'm' points to the vertical line.

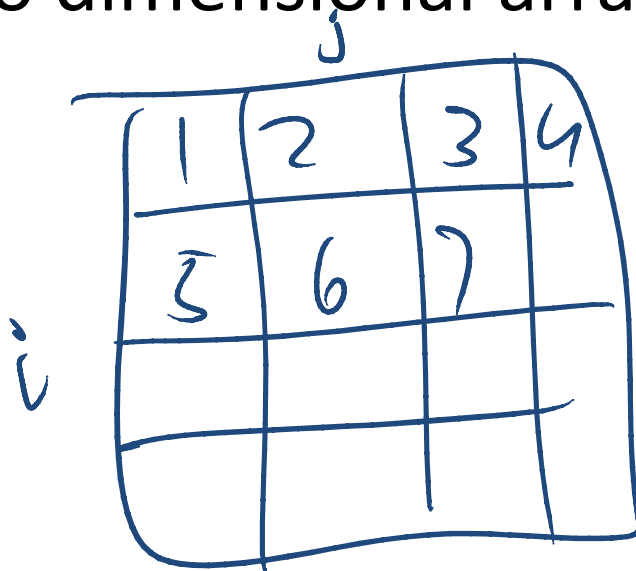
Representing Lists as Arrays

Array types

- One dimensional arrays

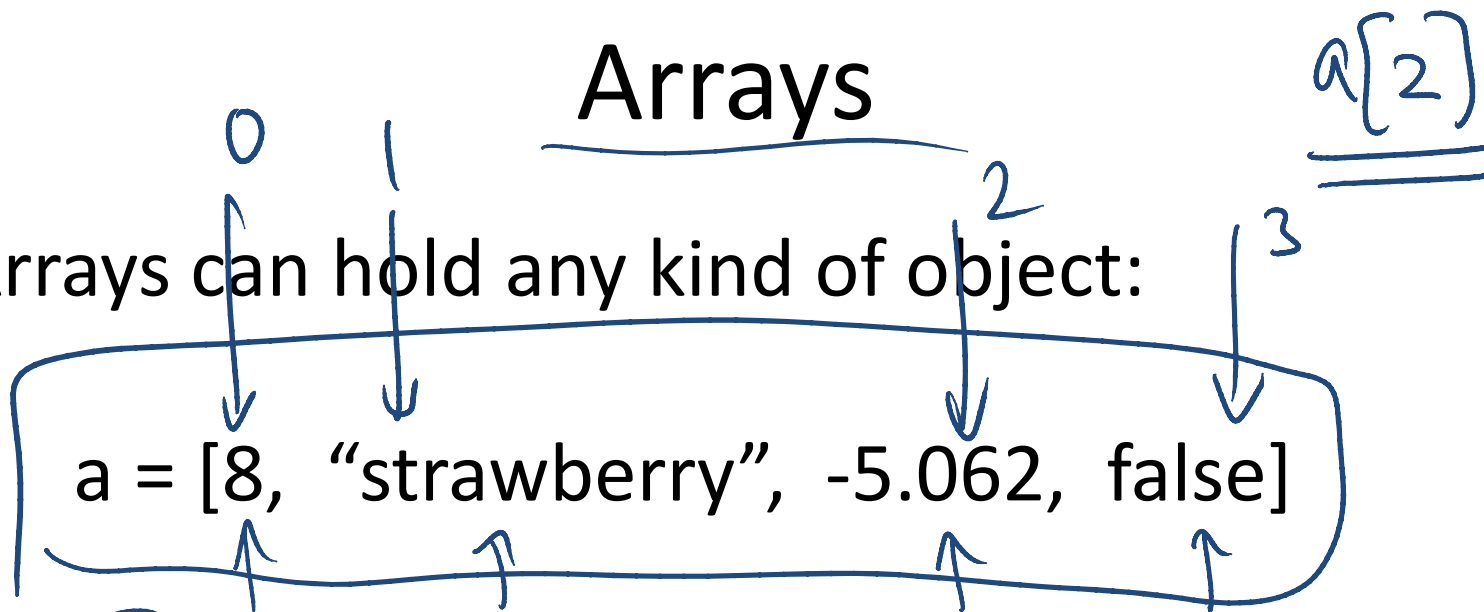


- Two dimensional arrays



Arrays

- Arrays can hold any kind of object:



a[0] => 8 *Ruby numbers items from 0!*

a[1] => "strawberry"

A = []

a.length => 4

- The empty array is written as []

Converting a Range to an Array

`r = 3..8`

`r.to_a` $\Rightarrow$ `[3, 4, 5, 6, 7, 8]`

`(8..3).to_a` $\Rightarrow$ `[]`

`s = "gu" .. "he"`

`s.to_a` $\Rightarrow$ `["gu", "gv", "gw", "gx", "gy",
"gz", "ha", "hb", "hc", "hd", "he"]`

The `to_a` method uses `succ` to generate elements.