Information Science
2: Functions

Naonori Kakimura
垣村尚徳
kakimura@global.c.u-tokyo.ac.jp

PEAK
Programs in English at Komaba

(Rev.) Goal of the Course

- Study basic concepts of information science
  - Introduction to computer science
    - Useful in various fields
  - More theoretical than previous course “information”

- Programming exercises using Ruby
  - easy-to-use language similar to Python or Perl
    - introduction to other advanced languages(C, Java)
    - Used in large-size system (Rakuten, Twitter)
  - We use “irb”: interactive Ruby
  - Made in Japan

Remarks1

- Lectures w./ simple demos + Exercises

- You need to do exercises on your own
  - A lot of exercises on ITC-LMS
  - Many online tutorials
    - Project Euler https://projecteuler.net/

- You can do at your own pace
    - It’s impossible to keep pace with everyone
  - some of you cannot finish exercises during the session
    - → take it to home and try it yourselves

Today’s Purpose

- Getting used to Ruby(using irb) 2
  - Functions
  - Loading files
  - Local variables

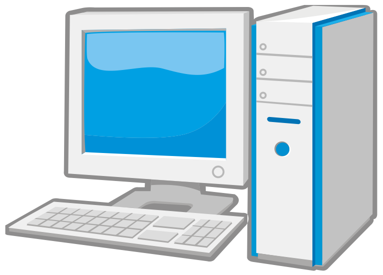
Today’s Contents

- Functions
  - Concept
  - How to use in Ruby
- Loading a file in irb
- Local variables

Difference btw Calc and Computers

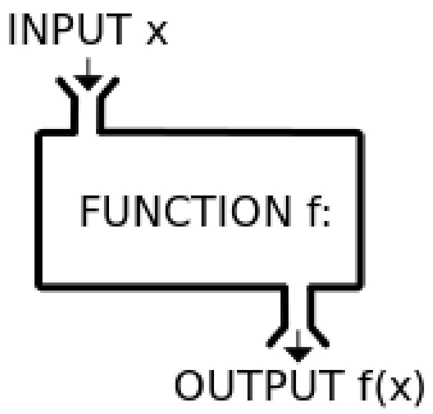
- A computer has a memory
  - Can calculate such as (10+2)*(100/40+31)
  - Can store a variable
  - Can store a program (stored-program concept)
    - IF, WHILE, ..
  - A program executes a sequence of commands
  - function: a principal component of programs

Last week



Function f in Mathematics

- Given an input x, return value $f(x)$
  - Like a black box



- Example
  - $f(x) = x^2$
    - if $x=2$, f returns $f(2)=4$
    - if $x=5$, f returns $f(5)=25$

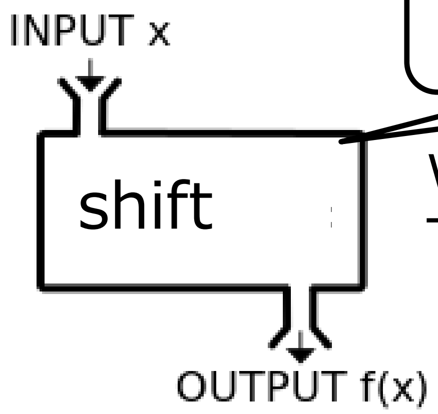
Rem. We do not need to see the inside when we write f

We separate the outside from the inside to understand
(capsulation)

Functions in Programs

- Give a name to some action, and do not see inside

ABC



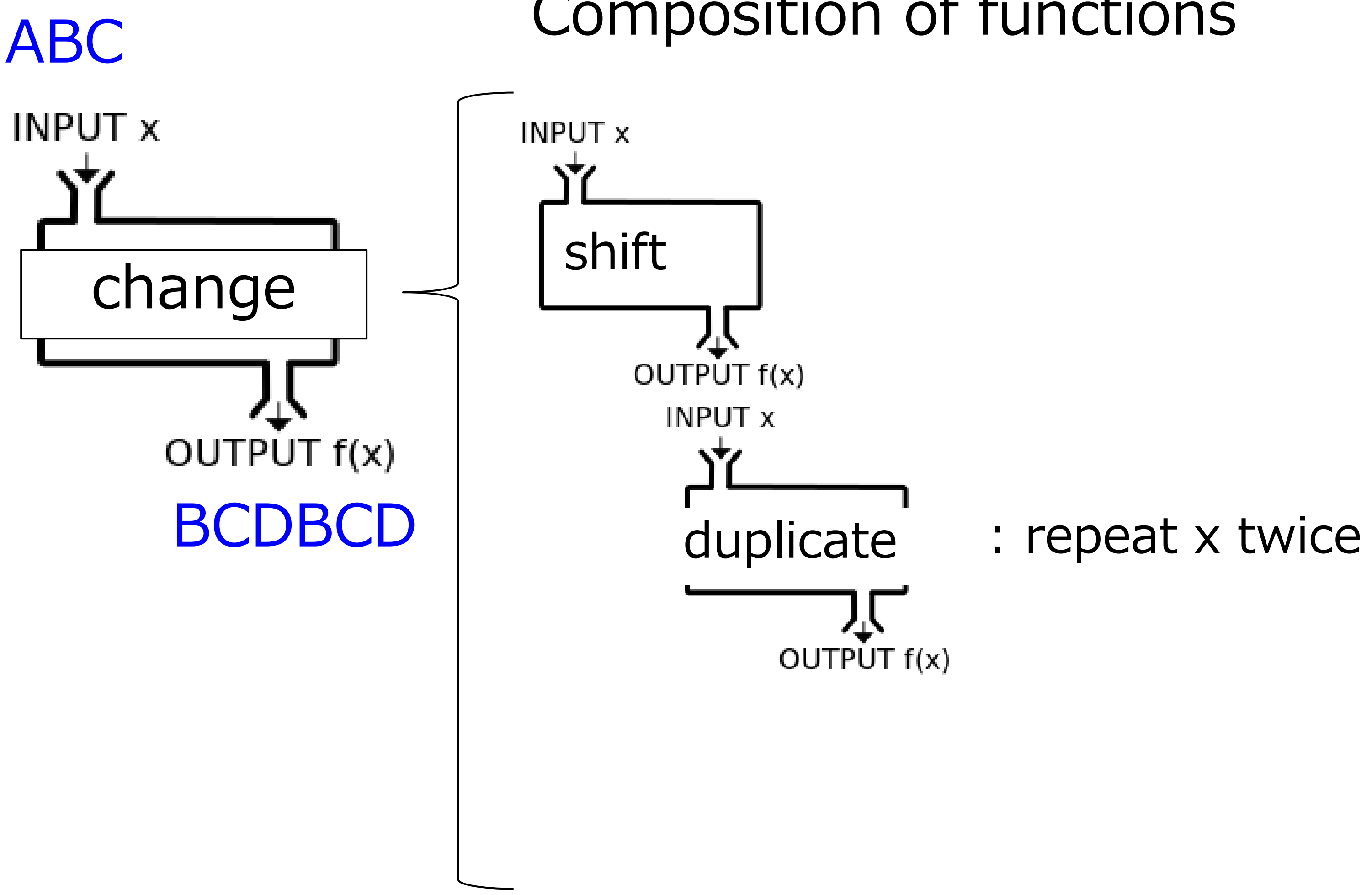
BCD

replace each letter by the letter after that

We do not care the inside
The inside is designed somewhere else

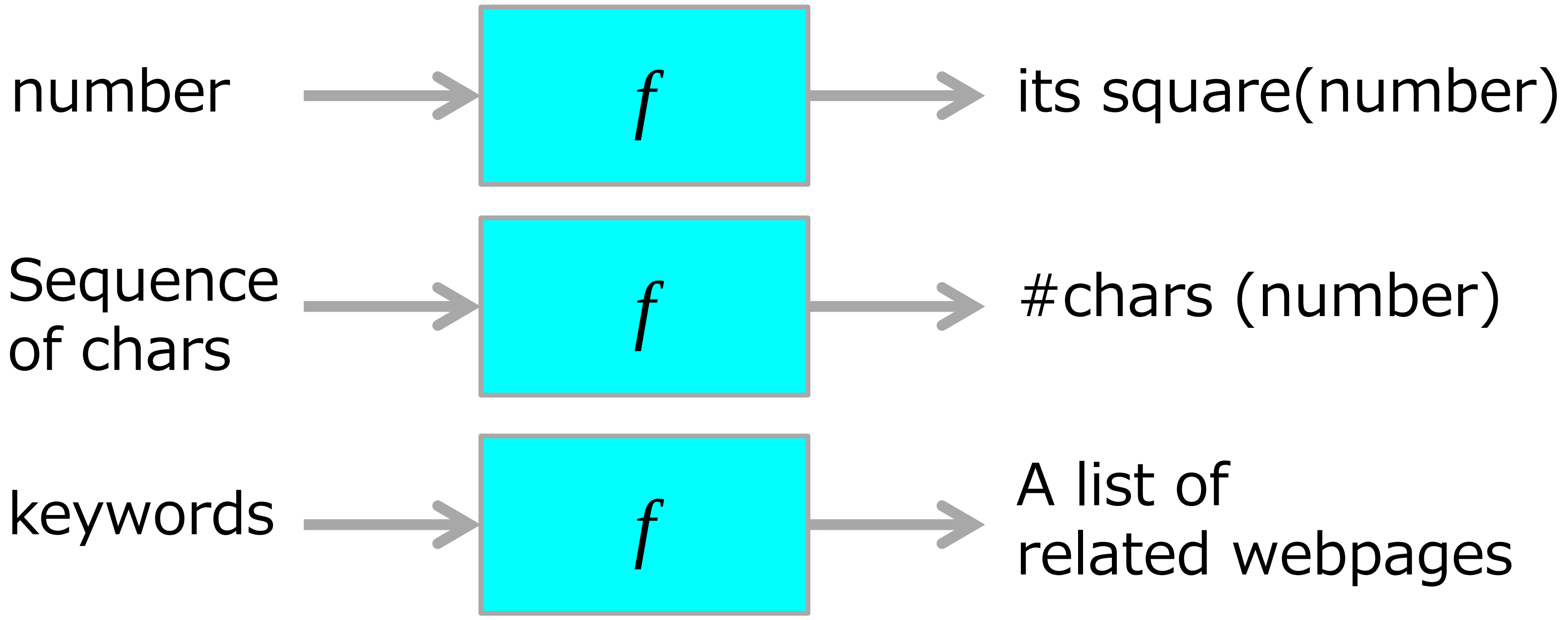
Functions in Programs

- We can group a sequence of commands



Functions in Programs

- ✓ A function receives a value (called an argument/parameter), and return a value (called the returned value)



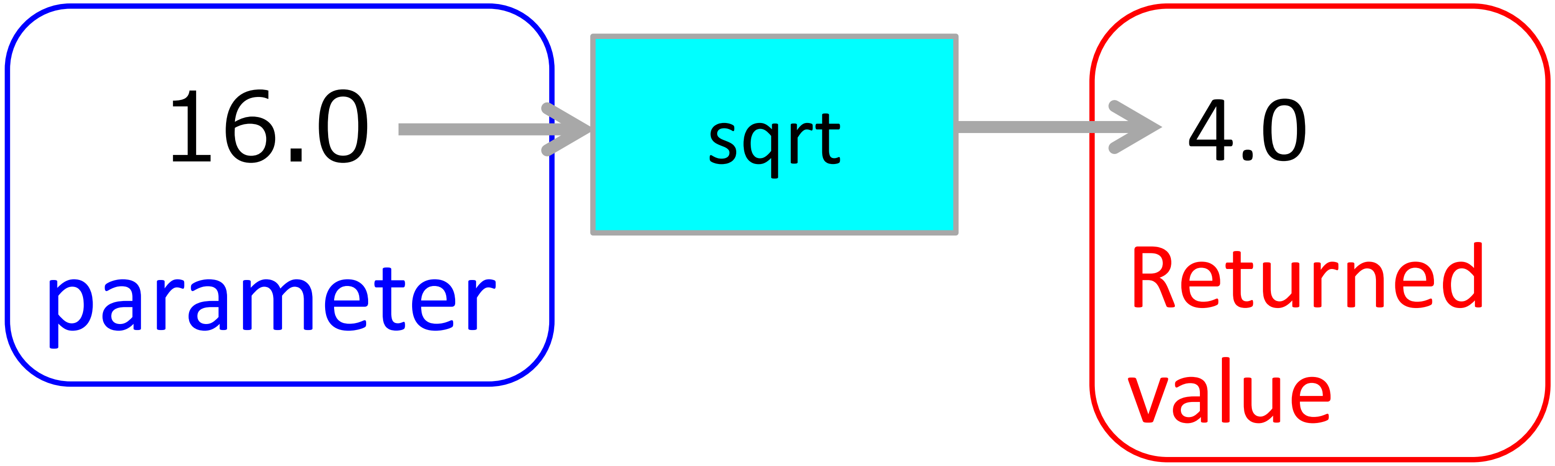
We Have Used Functions Last Week

✓sqrt(x);

・・・ Return the square root of x

Rem. Available even if you don’t know how to compute $\sqrt{x}$

Image



Examples

$$m = \text{sqrt}(16.0)$$

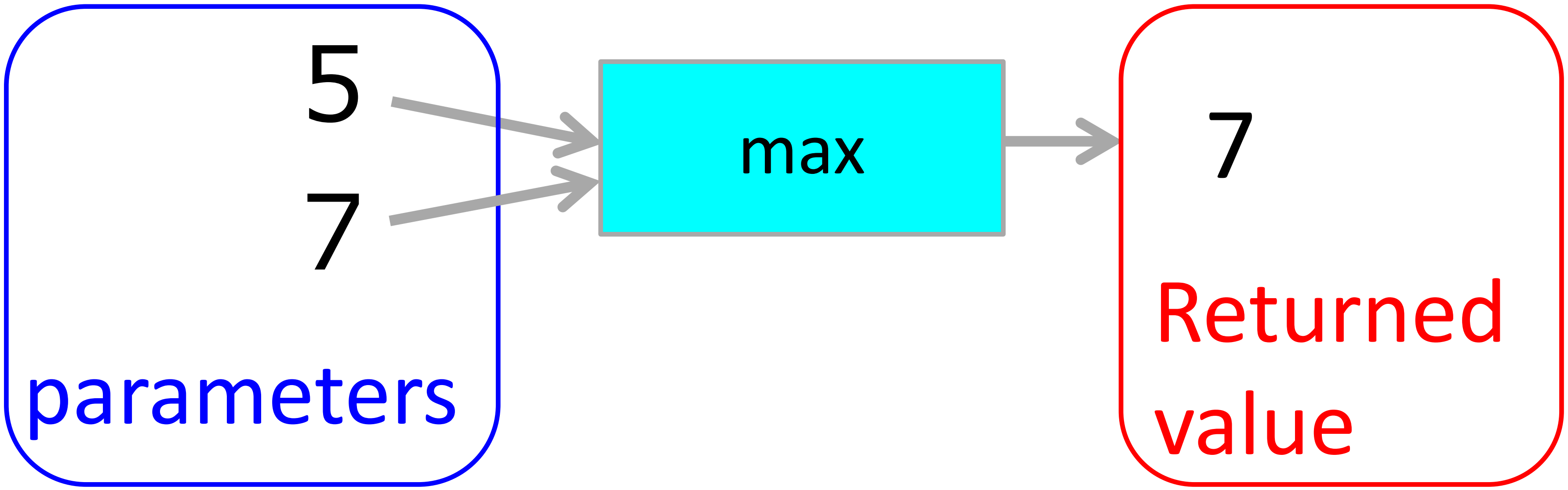
parameter/argument

m will have the returned value 4.0

How can we make an original function in Ruby?

Functions returning the max of two numbers

Suppose we want to make the function max



Parameter/argument: two numbers

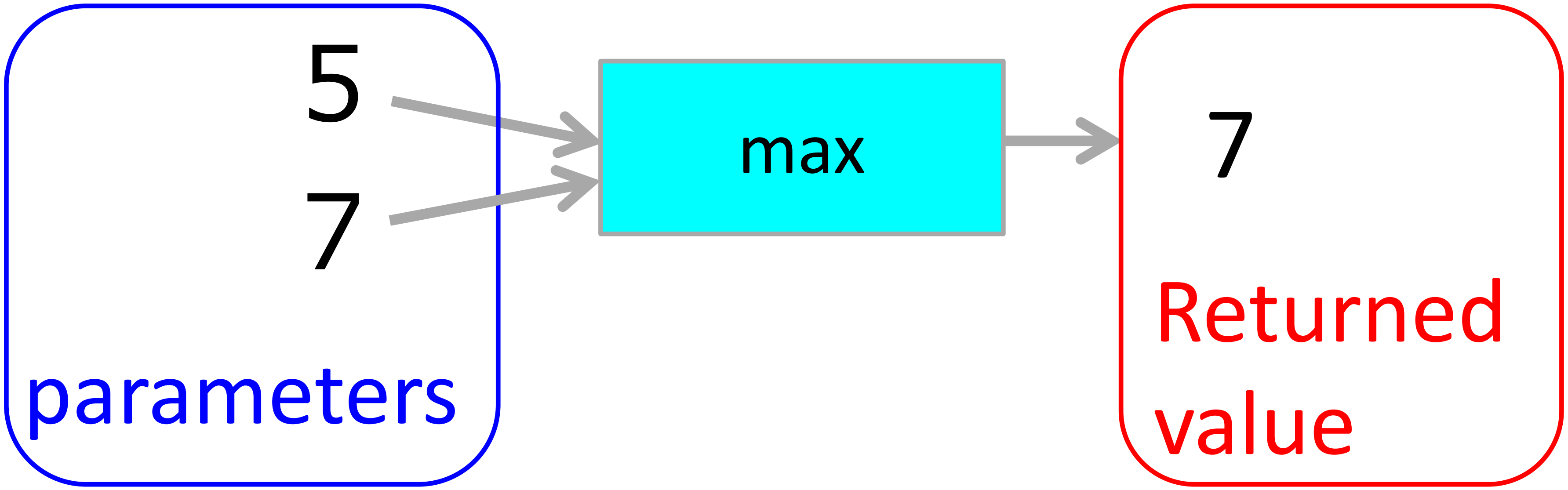
(allows multiple parameters)

Returned value: one number

(returns only one type of value)

Ruby Description

Suppose we want to make the function max



```
def  max(x,  y)
…                 Description to obtain the max value
end
```

After the above description, we can call “max(5,7)”, which returns 7

Ex. Function that returns the max of two

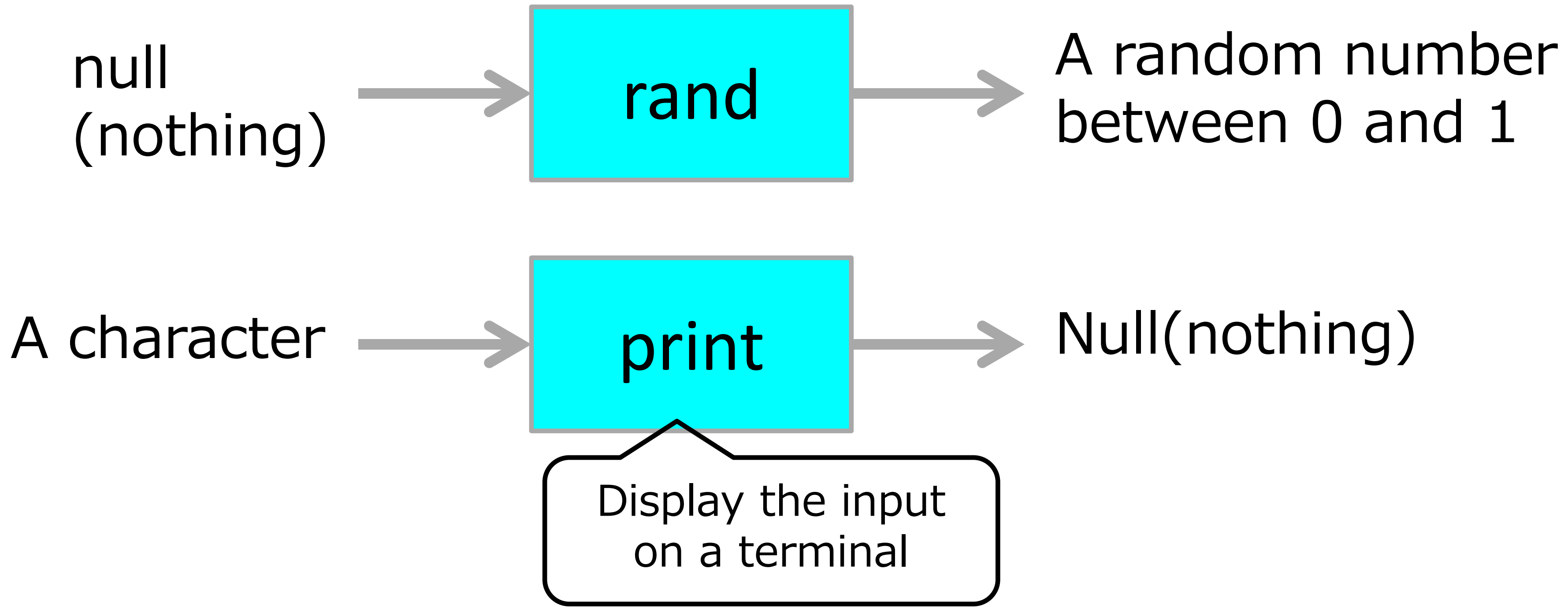
- Name・・・ max ← determined as you like
- parameters・・・ two numbers w/ any names
- returned value ・・・ a number(max of the two)

↓ Write it based on Ruby syntax

```
def  max(x,  y)
･･･    We here write the details(later)
end
```

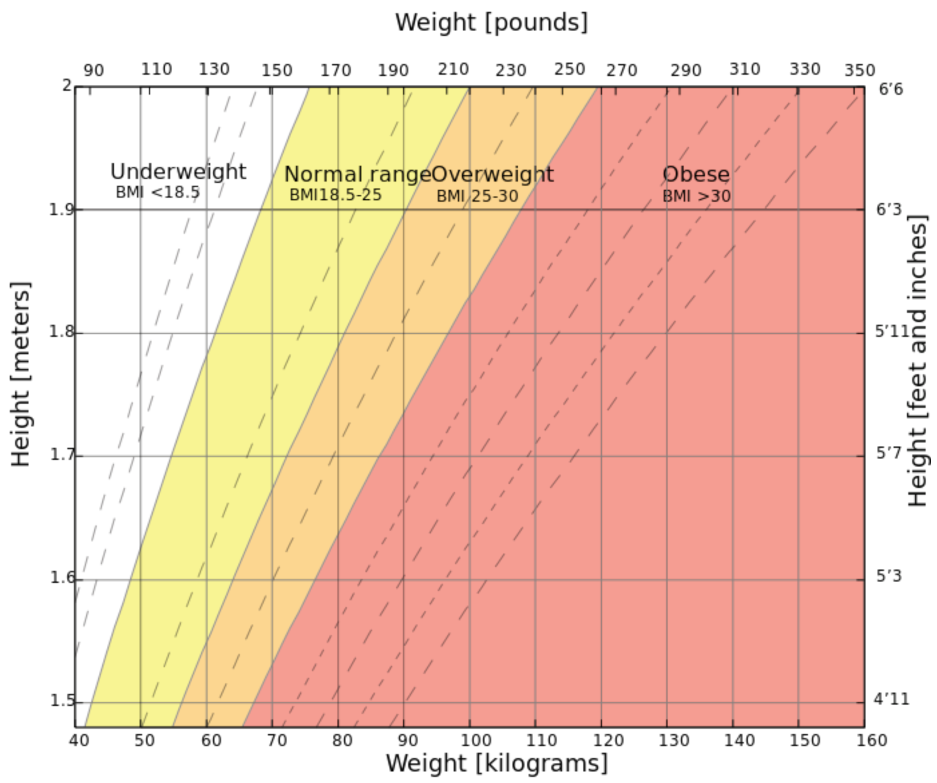
Remark: Other Functions

- ✓ The argument/retuned value of a function may be nothing



Ex. Defining Functions --- computing BMI

- body-mass index
  - a measure for human body shape
- Body mass[kg]÷ (height[m]* height[m])
  - ratio btw weight and height: implying fat or slim

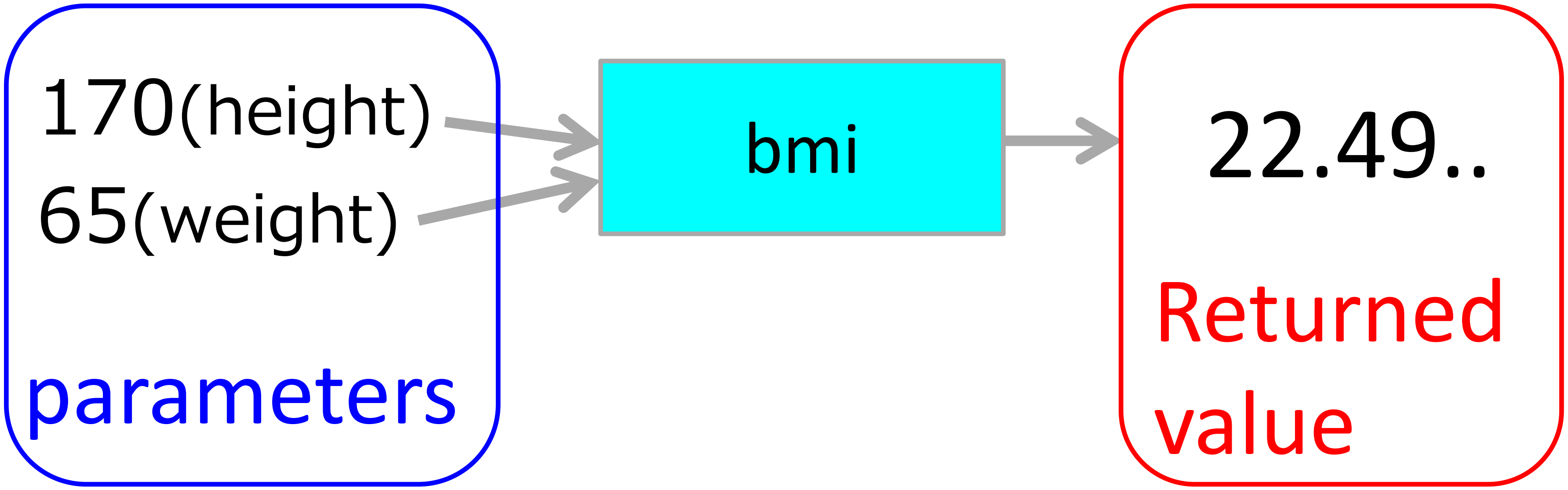


Normal: 18.5--25

Overweight: 25--30

Obese: 30--

Functions returning BMI



```
def  bmi(height,  weight)
…(expression to compute bmi)
end
```

Defining Functions --- computing BMI

```
irb(main):003:0> def bmi(height , weight)
irb(main):004:1>    weight / (height/100.0) ** 2
irb(main):005:1> end
=> nil
```

height & weight: variables

Defining Functions --- computing BMI

```
irb(main):003:0> def bmi(height , weight)
irb(main):004:1>     weight / (height/100.0) ** 2
irb(main):005:1> end
=> nil
```

height & weight: variables

```
irb(main):007:0> bmi(188.0, 104.0)
=> 29.4250792213671
irb(main):008:0> 1.1*bmi(174.0, 119.0 * 0.454)
=> 19.6289470207425
```

can calculate using the function bmi

Exercises during the class

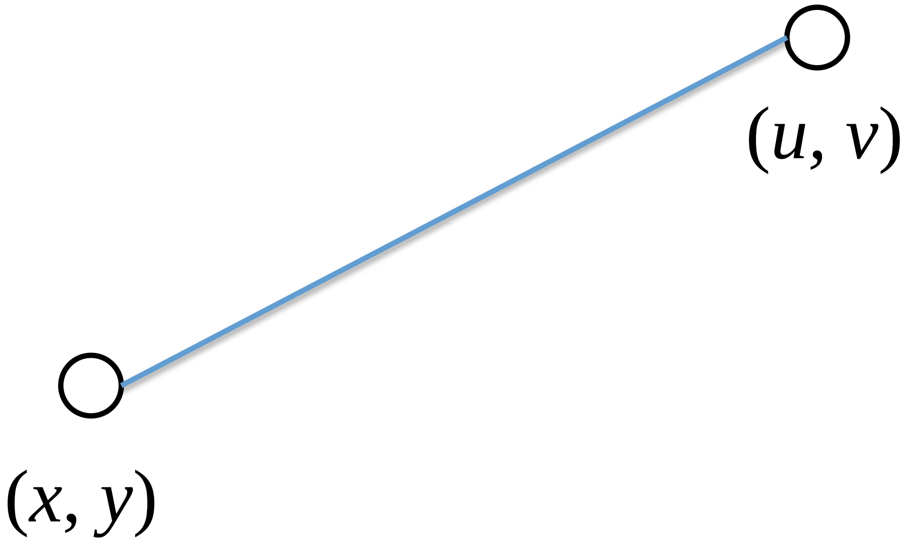
Make the following functions

1. feet_to_cm(f, i) that converts f feet i inch to cm. Note that 1 feet = 12 inch = 30.48 cm.
2. pound_to_kg(p, o) that converts p pound o ounce to kg. Note that 1 pound = 16 ounce = 0.4536 kg.

Exercises (Requiring Submission Later)

Make the following functions

3. distance(x, y, u, v) that computes the distance between two points (x, y) and (u, v) in plane.



Remark on Exercises

- When you have made a program,
  - Confirm that it returns no error
  - Confirm that it works well for some input

Answers of Exercises

```
def  feet_to_cm(f, i)
   30.48*f + 30.48/12*I        # (f + i/12.0)*30.48
end


def  pound_to_kg(p, o)
   0.4536*p + 0.4536/16*o
end                                # (p + o/16.0)*0.4536


def  distance(x, y, u, v)
   sqrt((x-u)*(x-u) + (y-v)*(y-v))
end                    # same as sqrt((x-u)**2+(y-v)**2)
```

Don’t forget
Include(Math) before

Today's Contents

- Functions
  - Concept
  - How to use in Ruby
- Loading a file in irb
  - Make a directory
  - Make a new file
- Local variables

Making a Ruby File

- It is annoying to enter commands each time you run irb
  - We want to store programs in a file separated from Ruby

So far: Type command one-by-one

```
irb(main):003:0> def bmi(height , weight)
irb(main):004:1>    weight / (height/100.0) ** 2
irb(main):005:1> end
```

→ Prepare a file in another place, and load it in irb

```
irb(main):003:0> load(“./bmi.rb”)
```

```
# BMI of a person with height (cm) and weight (kg)
def bmi(height , weight )
    weight / ( height /100.0) ** 2
end
```

Before Making a Ruby File

- This course will make a lot of Ruby files
  - Better to make a new directory(folder) in your Home directory, and to put Ruby files in it

Step 1: Exit “irb”,

Step 2: Prepare a new directory in Home

- By Finder: Right-click → New Folder
- By a terminal
  - > mkdir MyRubySource (you can change the name)

Step 3: Go to the directory with a terminal

- Ex: > cd MyRubySource

Step 4: Start “irb”

Define/Read a Function from a File

- Download bmi.rb from ITC-LMS in the directory
  - it is prepared / You can make it by yourself

```
irb(main):001:0> bmi(188.0, 104.0)
NoMethodError: undefined method
```

Because we quit irb,
all functions vanished

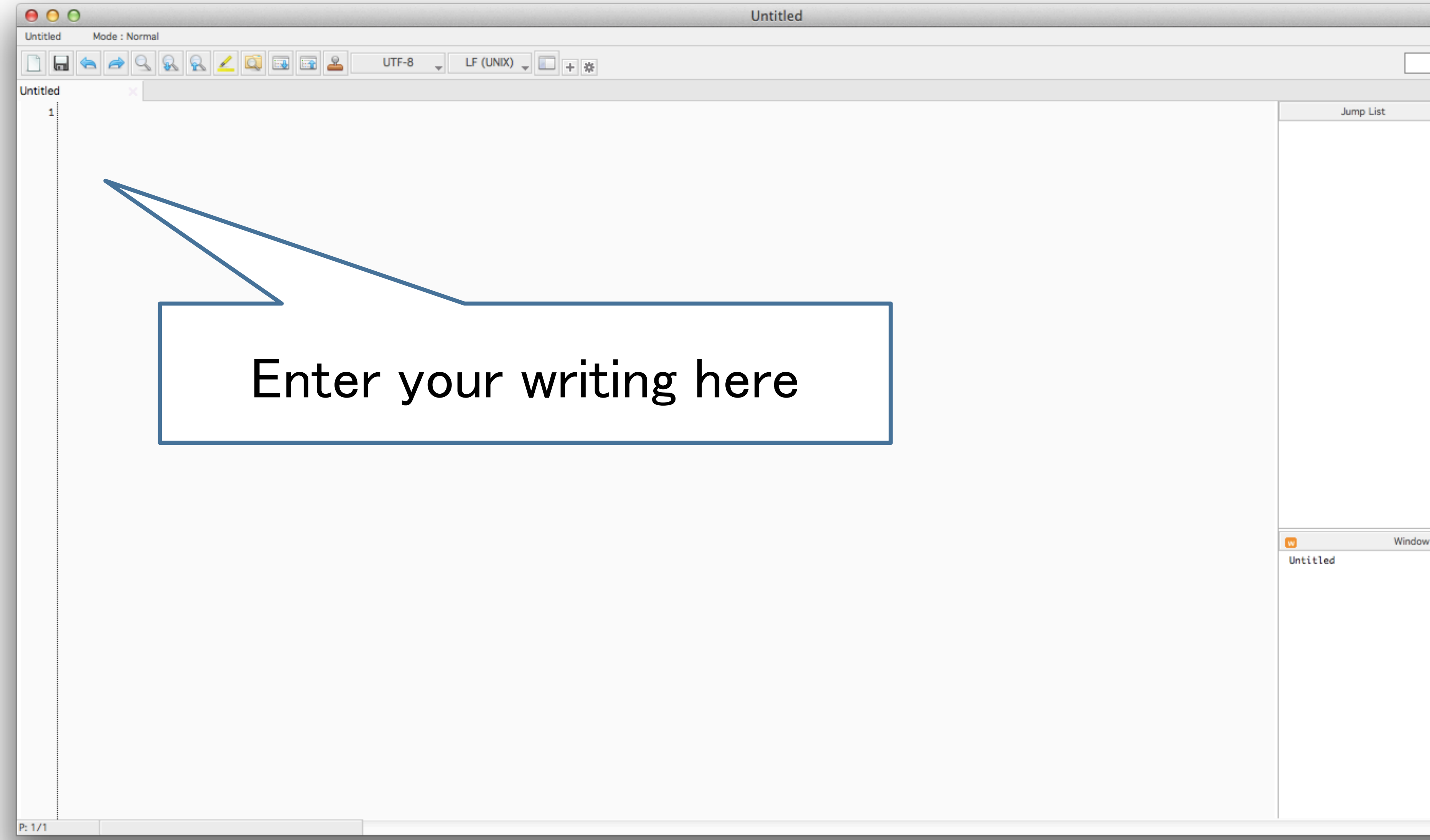
```
irb(main):001:0> load("./bmi.rb")
=> true
irb(main):002:0> bmi(188.0, 104.0)
=> 29.4250792213671
```

“./” means “current directory”

Make/Edit a Ruby file using a Text Editor

- Text editor: software to edit a text file
  - a text file: a file consisting of only characters
  - Open/Edit a Ruby file
- mi: pre-installed text editor
  - Click on the Dock
- Cf. Other text editors
  - Emacs ··· Advanced, useful editor for programmers
  - Notepad++, Memo pad, Sakura, etc ··· for windows

Window of mi



Enter your writing here

Open bmi.rb with mi

Comments for human being:
Sentences after # (until end of line) are not read by Ruby

```
# BMI of a person with height (cm) and weight (kg)
def bmi(height , weight)
    weight / ( height /100.0) ** 2
end
```

Better to have an indent for visibility

bmi.rb

Same as the description of a function
which you defined before in irb

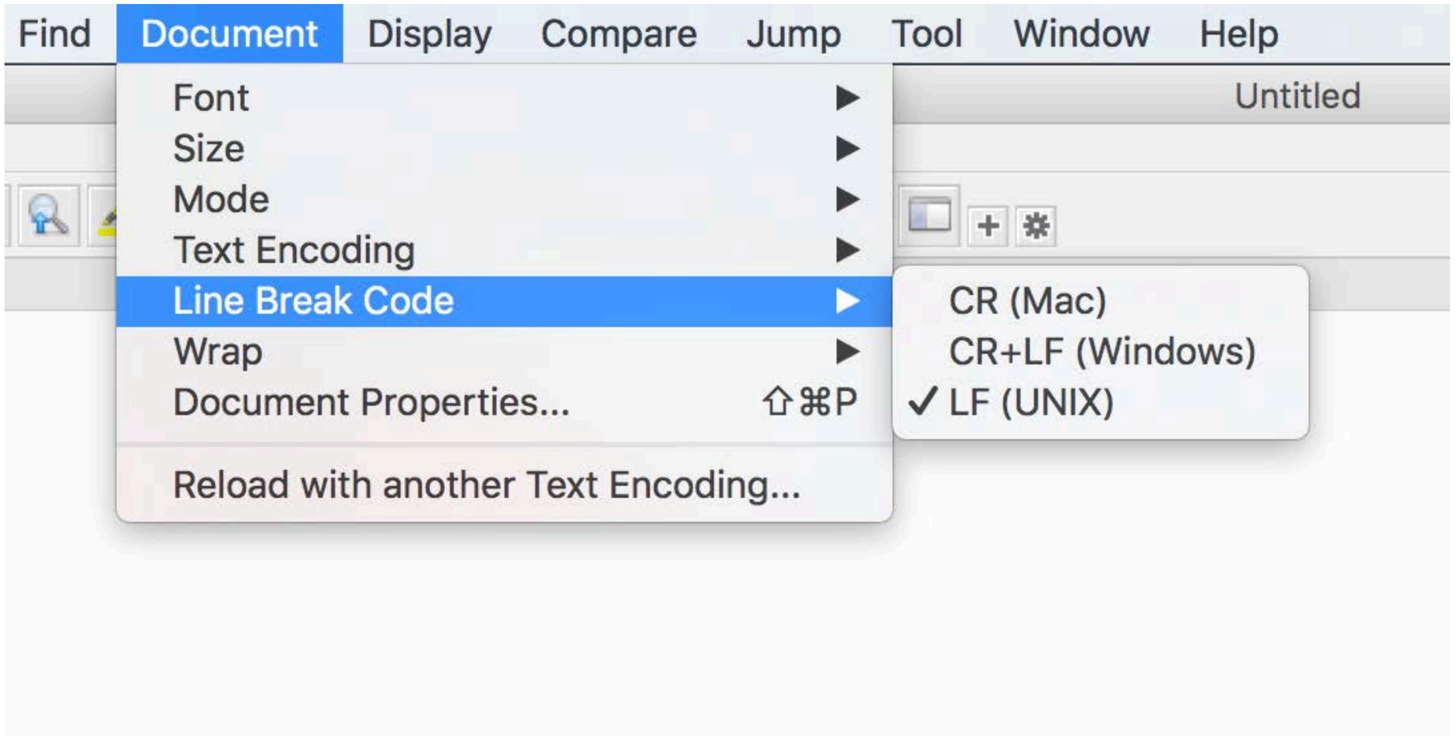
Exercises: make a Ruby File

1. Create yardpound.rb as in the next slides
    - Containing feet_to_cm(f, i) and pound_to_kg(p, o)
        - we can store several functions in one file

We can define multiple functions in one file

Common Trouble

- Try to change the line-break code
  - Code is different in environments



Can Define Multiple Functions in One File

```
def feet_to_cm(f, i)
    (f + i/12.0)*30.48
end
def pound_to_kg(p, o)
    (p + o/16.0)*0.4536
end
```

yardpound.rb

- Save a text file as a name with the postfix ".rb"
    - to recognize that it is a Ruby file
- Just add ".rb" at the end of the file name

Can Define Multiple Functions in One File

```
def feet_to_cm(f, i)
    (f + i/12.0)*30.48
end
def pound_to_kg(p, o)
    (p + o/16.0)*0.4536
end
                                        yardpound.rb
```

```
irb(main):003:0> load("./yardpound.rb")
=> true
irb(main):004:0> feet_to_cm(5, 11)
=> 180.34
```

Exercises: make a Ruby File

2. Create distance.rb containing the function distance (which we made)
   - Requiring submission later

3. Make a function bmi_yp(f,i,p,o) that returns BMI when we are given height of f feet i inch and weight of p pound o ounce
   - Using bmi.rb and yardpound.rb

Read a Ruby file in another Ruby file

Can use a defined function in another one if it is loaded

```ruby
load ("./bmi.rb")
load ("./yardpound.rb")

def bmi_yp(f,i,p,o)
    bmi(feet_to_cm(f,i), pound_to_kg(p,o))
end
```

bmi_yp.rb

Do not forget to “load” necessary files

to “include” Math if necessary

Tips: A Function as a Constant

When you want to define an invariant value

Cf. A variable may be changed with some reason

```
# BMI of a person with height (cm) and weight
  (kg)
def bmi(height , weight )
   weight / ( height/100.0) ** 2
end
def k_height()        #Height of Mr. K
   188.0
end
def k_weight()        #Weight of Mr. K
   104.0
end
```

bmi.rb

Tips: A Function as a constant

```
irb(main):005:0> load("./bmi.rb")
=> true
irb(main):006:0> k_weight()
=> 104.0
irb(main):007:0> bmi(k_height(), k_weight())
=> 29.4250792213671
```

Today’s Contents

- Functions
  - Concept
  - How to use in Ruby
- Loading a file in irb
- Local variables

Local Variable: Variables in a Function

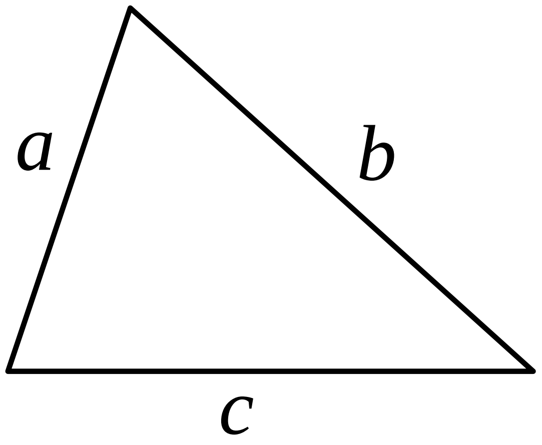
Ex. A function that computes the area of a triangle whose side have lengths a, b, and c.

- Use Heron's formula

$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$$

where

$$s = \frac{a+b+c}{2}$$



Useful to use the variable s to express computation

Local Variable

Ex. A function that computes the area of a triangle whose side have lengths a, b, and c.

Parameters a,b,c are also local variables.
→ We cannot invoke a,b,c outside

```
def heron(a,b,c)
    s = 0.5*(a+b+c)
   sqrt(s * (s-a) * (s-b) * (s-c))
end
```

s is called *a local variable*

heron.rb

A local variable is valid only inside the function

(See Exercises)

Increase readability of the expression

Remarks on Returned Values

- The returned value of a function is the value computed at the end
  - To specify the returned value, write “return”

```
def heron(a,b,c)
    s = 0.5*(a+b+c)
   return sqrt(s * (s-a) * (s-b) * (s-c))
end
```

Returned value is basically one computed at the last line

Exercises

- Make a file heron.rb, and confirm it works
  - Try heron(3,4,5), heron(3,4,7), heron(3,4,8), etc.
    - consider why we have sometimes an error
  - Check Problem 2-8 (Problem 8 in Section 2)

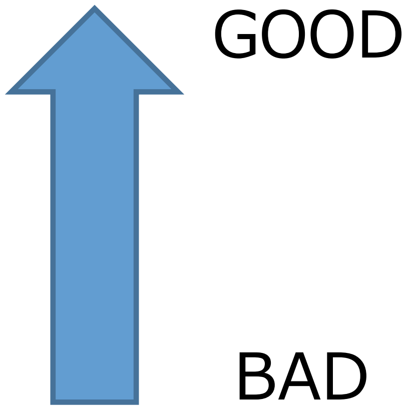
- **Solve for submission**
  - 2-2 logarithm
  - 2-4 computing the area of an equilateral triangle

  - If you have time
    - 2-3 (a)(b) and 2-7

Quizzes (**Deadline 10/19 23:59**)

- Copy&Paste programs which you have made during the session
  - distance.rb
  - heron.rb
  - solve 2-2, 2-4 etc.

- Evaluate your understanding on a scale of 1 to 5
  - 5: I have understood all
  - 4: very well
  - 3: well
  - 2: a bit
  - 1: I could not understood anything
    - Tell me if you have some specific comment



Next Lecture (10/17)

- Conditions, Iterations, and Arrays 1
  - Getting more functions...

- 10/10: public holidays