Basic Programming

Ordering a Pizza

:{) Codaisseur

Installing Ruby

Codaisseur Reader:

https://read.codaisseur.com/topics/day-2-basic-programming

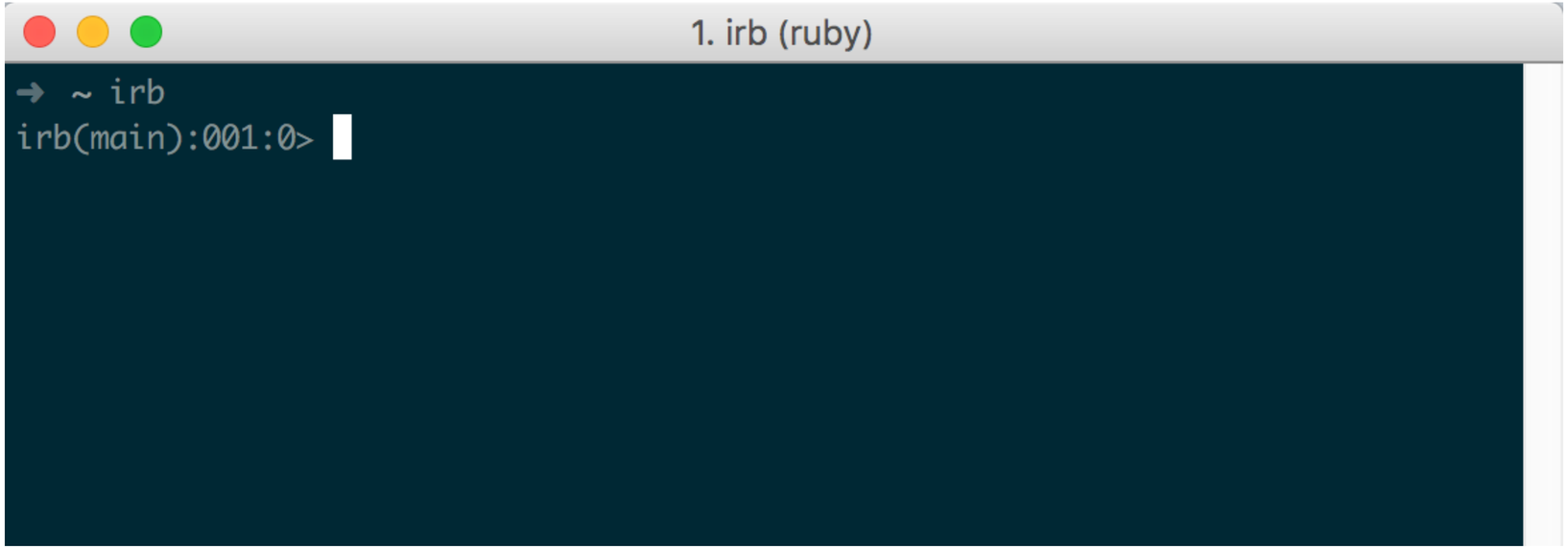
:{) Codaisseur

Ruby

Let’s start programming

IRB

- Allows you to experiment with ruby in real-time with immediate response

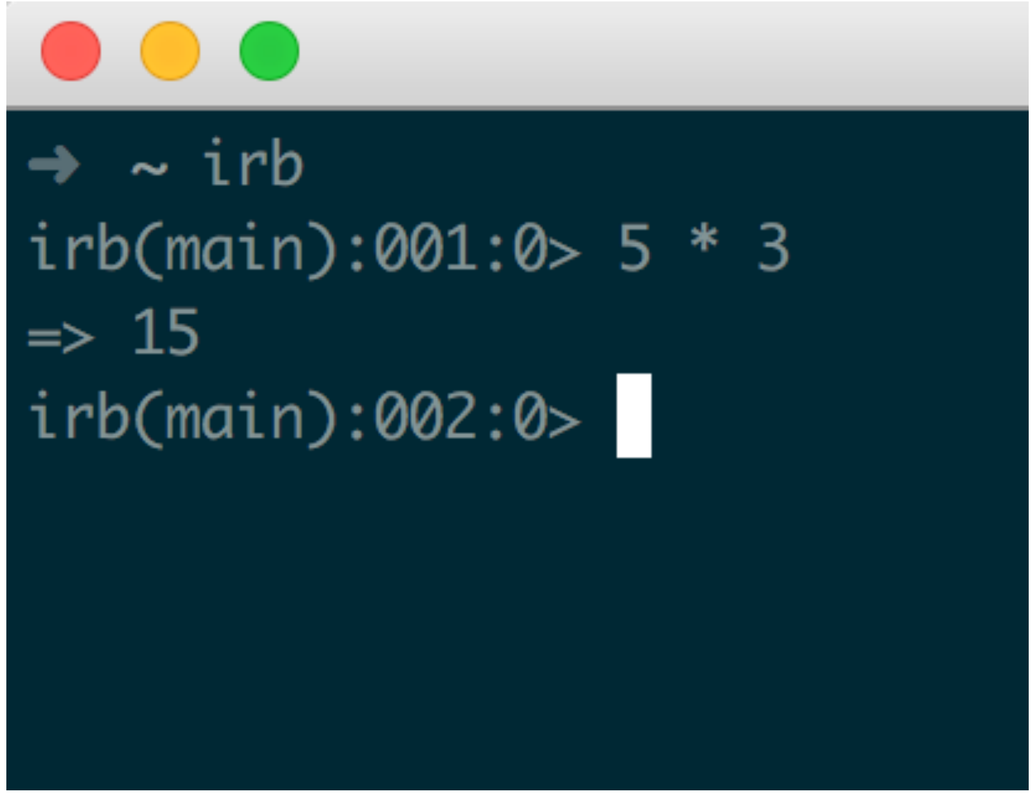


:{) Codaisseur

Variables, Expressions & Numbers

Calculate

```
5 * 3
```



:{) Codaisseur

Variables

- Variables are used to keep track of objects
- A single variable holds a reference to an object
- Start with an underscore or lowercase letter

:{) Codaisseur

Variables

```
radius = 5
diameter = 2 * radius
```

:{) Codaisseur

Variables

Variables

Objects

```
radius = 5
diameter = 2 * radius
```

:{) Codaisseur

...Variables

```
radius = 5
diameter = 2 * radius
pi = 3.14159
circumference = pi * diameter
area = pi * radius ** 2
```

:{) Codaisseur

Expressions

Expressions can be built from:

- **(+) addition**
- **(-) subtraction**
- **(*) multiplication**
- **(/) division**
- **(**) exponent**
- **parentheses and other operators.**

:{) Codaisseur

Comments

- Comments begin with a hash (#) and serve as documentation/notes

- Comments are not visible when your code is rendered

:{) Codaisseur

Comments

```
# Properties of a circle
radius = 5
diameter = 2 * radius
```

:{) Codaisseur

More Examples

```
# Farm inventory
cows = 5
sheep = 10
horses = 3
animals = cows + sheep + horses
average = animals / 3
```

:{) Codaisseur

More Examples

```
# Purchase cost
price = 5.99
quantity = 8
cost = price * quantity
```

:{) Codaisseur

More Examples

```
# Standard deviation
v1 = 2.3
v2 = 3.9
v3 = -1.2

mean = ( v1 + v2 + v3 ) / 3
variance = ( (v1-mean)**2 + (v2-mean)**2 + (v3-mean)**2 ) / 3
std_dev = variance ** 0.5
```

:{) Codaisseur

Variable names

```
# Use underscores to divide words
grazing_cows = 120
milk_per_cow = 38
milk_total = milk_per_cow * grazing_cows
```

:{) Codaisseur

Integer vs Decimal

```
number_of_children = 4       # integer
average_age        = 3.45    # decimal
```

:{) Codaisseur

Integer vs Decimal

> Integer with integer = no decimals

> Decimal with integer/decimal = decimals

```
iscale  = 7   / 3   # is 2
dscale = 7.0 / 3   # is 2.3333
```

:{) Codaisseur

Modulo/Modulus

- **Modulo (%) returns the remainder**

```
scale = 7 / 3   # is 2
extra = 7 % 3   # is 1
```

:{) Codaisseur

Reader

Variables, Expressions & Numbers - Exercises

Week 1 - Beginner Bootcamp » Day 2 - Basic Programming

Strings

text

Strings hold Text

```
person = "Alan Turing"
interest = "automata"
quote = "Sometimes it is the people no one can imagine
anything of who do the things no one can imagine."
```

:{) Codaisseur

Join Text

```
person = "Alan Turing"
interest = "automata"
quote = "Sometimes it is the people no one can imagine anything
of who do the things no one can imagine."

description = person + " likes " + interest +
                    " and said " + quote
```

:{) Codaisseur

Interpolation

- Utilising placeholders within a string
- Placeholders are replaced with their corresponding values
- Only exists between double quotations (“”)

:{) Codaisseur

Interpolation

```
person = "Alan Turing"
year = 1950
quote = "Sometimes it is the people no one can imagine anything
of who do the things no one can imagine."
```

```
msg = "#{person} said #{quote} in #{year}"
```

:{) Codaisseur

Ruby between {}

```
pi = 3.14159
r = 3
lesson = "A circle of radius #{r} has area #{pi * r * r}"
```

:{) Codaisseur

Single Quotes

```
pi = 3.14159
r = 3
lesson = "A circle of radius #{r} has area #{pi * r * r}"


#no interpolation
‘A circle of radius #{r} has area #{pi * r * r}’
```

:{) Codaisseur

Reader

Strings - Exercises

Week 1 - Beginner Bootcamp » Day 2 - Basic Programming

Input/Output

I/O

Display text with print

```
print "The rain"
print " in Spain.\n"

print "#{person} said #{message}\n"
```

:{) Codaisseur

Newlines with puts

```
print "The rain in Spain\n"
puts "The rain in Spain"     # same
```

Read what user typed with gets

```
puts "Tell me something…"
response = gets
puts "You said #{response}"
```

:{) Codaisseur

Get rid of the \n

```
puts "What is your name?"
name = gets.chomp
puts "Hi #{name}, nice to meet you!"
```

:{) Codaisseur

Convert input to number

```
puts "Enter a number:"
v = gets.to_i              # convert to integer
puts "Square is: #{v * v}"
```

Use **to_f** for decimal, also called float or floating point.

:{) Codaisseur

Reader

Input/Output - Exercises

Week 1 - Beginner Bootcamp » Day 2 - Basic Programming

If/Else

Conditions

Test Condition with if

```
puts "How old are you?"
age = gets.chomp.to_i

if age < 18
    puts "No beer for you."
end
```

:{) Codaisseur

Optionally, add else

```
if age < 18
    puts "No beer for you."
else
    puts "Enjoy."
end
```

:{) Codaisseur

Or even elsif

```
if age < 18
    puts "No beer for you."
elsif age > 45
    puts "Wouldn’t you rather have a cocktail?"
else
    puts "Enjoy."
end
```

:{) Codaisseur

Equality with ==

```
if animal == "cat"
    puts "Meow."
end
```

:{) Codaisseur

Smaller or equal <=

```
if stories <= 5
    puts "I'll take the stairs."
end
```

:{) Codaisseur

Both true &&

```
if color == "red" && type == "gemstone"
    puts "It’s a ruby."
end
```

:{) Codaisseur

Either true ||

```
if month < 5 || month > 8
    puts "I'll wear a jacket."
end
```

:{) Codaisseur

Reader

If/Else - Exercises

Week 1 - Beginner Bootcamp » Day 2 - Basic Programming

Methods

Execute a function

Methods

- Tiny program
- A couple of lines of code that carry out one specific task
- AKA “function”

:{) Codaisseur

Method examples

```
‘capitalize me’.capitalize  # Capitalize me
‘NO SHOUTING’.downcase      # no shouting
‘stop, hammertime’.upcase   # STOP, HAMMERTIME
‘codaisseur’.reverse        # ruessiadoc
```

:{) Codaisseur

Make method with def

```
def hello_world
    puts "Hello world!"
end
```

:{) Codaisseur

And call the method

```
def hello_world
    puts "Hello world!"
end

hello_world # Hello world!
```

:{) Codaisseur

Return something

```
def hello_world
    "Hello world!"
end

puts hello_world # Hello world!
```

:{) Codaisseur

Parameters

```
def age_in_days(age)
    (age + 1) * 365.25
end

puts age_in_days(24) # 9131.25
```

:{) Codaisseur

Reader

Methods - Exercises

Week 1 - Beginner Bootcamp » Day 2 - Basic Programming

Arrays

Lists of things

Arrays

- A list of objects
- Indexed using integer values
- Placed between square brackets [ ]

:{) Codaisseur

Arrays

```
fruits = ["apple", "pear", "banana"]
even_numbers = [2, 4, 6, 8, 10]
```

:{) Codaisseur

Arrays

- **The items in the array can be mixed**

```
fruits_and_numbers = ["apple", 2, "banana"]
```

:{) Codaisseur

Get first or last item

```
fruits = ["apple", "pear", "banana"]
fruits.first # "apple"

even_numbers = [2, 4, 6, 8, 10]
even_numbers.last # 10
```

:{) Codaisseur

Zero-based index

```
fruits = ["apple", "pear", "banana"]
fruits[0] # "apple"
fruits[1] # "pear"

even_numbers = [2, 4, 6, 8, 10]
even_numbers[3] # 8
```

:{) Codaisseur

Reader

Arrays - Exercises

Week 1 - Beginner Bootcamp » Day 2 - Basic Programming

Hashes

Things with properties

Symbols

- Simplified string
- Used to name or identify values
- Starts with a colon (:)
- Used when the value doesn’t need to change

:{) Codaisseur

Symbols

```
:name
:first_name
:city
:age
:price
```

:{) Codaisseur

Hashes

- Similar to an array
- Indexed as pairs using a key and value
- Placed between braces { }

:{) Codaisseur

Hashes

```
person = { :first_name => "Slim", :last_name =>
"Shady", :profession => "Rapper" }
```

:{) Codaisseur

Hashes

```
person = {
            :first_name => "Slim",
            :last_name => "Shady",
            :profession => "Rapper"
        }
```

:{) Codaisseur

Hashes

```
person = {
            first_name: "Slim",
            last_name: "Shady",
            profession: "Rapper"
        }
```

:{) Codaisseur

Read hash values

```
puts person
    # {:first_name=>"Slim", :last_name=>"Shady",
        :profession=>"Rapper"}

puts "Hi #{person[:first_name]} #{person[:last_name]}!"
    # Hi Slim Shady!
```

:{) Codaisseur

Write hash values

```
person[:first_name] = "Fat"

puts person
    # {:first_name=>"Fat", :last_name=>"Shady",
      :profession=>"Rapper"}
```

:{) Codaisseur

Combine all types

```
cars = [
          {make: "Ford", type: "Focus", color: "grey", topspeed: 202.5},
          {make: "Mercedes", type: "E", color: "black", topspeed: 246.9},
          {make: "Ferrari", type: "F430", color: "red", topspeed: 315.3}
       ]

puts cars[2][:make] # Ferrari
```

:{) Codaisseur

Reader

Hashes - Exercises

Week 1 - Beginner Bootcamp » Day 2 - Basic Programming

Assignment

Order a Pizza

Show Pizzas • Ask what to buy • Show price