#

# Questions

# Running Ruby

Just like when we learnt about variables, we will run our Ruby here

<http://repl.it/languages/Ruby>.

We’ll be using some of the stuff we learnt about variables, so unless you have a really, really good memory, it’s a good idea to keep the variable bits of paper handy.

#

# Is it bigger?

Is 10 bigger than 8? We know it is, and so does Ruby. In Ruby we write ‘bigger than’ as **>,** so to ask Ruby if 10 is bigger than 8 we would type

puts 10 > 8

Ruby, as you can see, agrees that this is true

true

Is 8 bigger than 8? Ask Ruby

puts 8 > 8

false

Nope, it’s the same size.

# Is it smaller?

To ask Ruby if one thing is smaller than another you use **<**.

## *Challenge – Ask Ruby a Smaller Than question*

Write a program to ask Ruby if 5 is smaller than 9. You should, of course, see

true

## *Bonus Challenge – Are you younger than your best friend?*

Write a program to ask the ages of you and your best friend, then tell you if you are younger than them. When you run your program you should see something like this.

How old are you?

9

How old is your best friend?

10

You are younger than your best friend is

true

**Hint –** Look back at the variables stuff to see how to read in a number.

# Is it the same?

You can also ask Ruby if two things are the same. For that you use ==.

puts 3 == 3

true

## *Bonus Challenge continued – Ask Ruby all three questions*

Now we know three questions we can ask Ruby, can you make your Ruby program answer all of them. If you get it right and run it, you should see something like this

How old are you?

9

How old is your best friend?

10

You are younger than your best friend is

true

You are older than your best friend is

false

You and your best friend are the same age is

false

Ask someone else to run the program, someone sitting near you, a tutor, anyone. Does it always give the right answers? It should.

# Not!

Run this Ruby program

puts 99 > 9

true

Now run this one

puts !(99 > 9)

false

The answer changes. What is going on here?

Well, if you put your question in brackets and stick a ! in front of it, that is like Ruby saying “not”, so you get the opposite answer. Here Ruby is answering the question “Is 99 NOT bigger than 9”.

But “not bigger” means the same as smaller, doesn’t it? So this is just like asking “Is 99 smaller than 9”, which it isn’t, so Ruby says “false”. Well, that’s almost right, but not quite. If I’m not taller than you, that doesn’t mean I’m smaller, I might be the same height.

## *Challenge – Not bigger*

Try to guess what this bit of Ruby will do

puts !(2 > 2)

Now run it and see if you were right. Grab a tutor and explain to them what is happening. If you can explain it to someone else, that means you understand it, which seems a bit backwards. Normally when your learning stuff, it gets explained to you!

Not also works with equals.

puts !(4 == 8)

true

4 is NOT equal to 8 is right, so Ruby says “true”. If fact there is a short way of writing Not with ==. It means exactly the same thing

puts 4 != 8

true

!= means that lazy programmers to have less to type, which is why they like it.

## *Challenge – Guess the answers*

Try to guess what Ruby will print out for all these.

puts 1 == 1

puts 1 != 1

puts 3 > 4

puts 5 < 5

puts !(7 < 7)

puts "hello" == "world"

# Comparing strings

Notice I snuck in a string at the end of that last challenge? You can ask Ruby if two strings are equal, just the same as you can for numbers. You can even use **<** and **>** with strings, which tells you which is first in the alphabet

## *Challenge – Whose name comes first in the alphabet?*

Write a program than asks your name and your friend’s name, then tells you who comes first in the alphabet. When you run it, it should look something like this

What is your name?

Lisa

What is your friend’s name?

Karen

Lisa comes before Karen in the alphabet is

false

# Asking two questions at once

You can ask Ruby to answer two questions at the same time. Run this program

puts 6 > 4 and 5 == 5

true

We are asking Ruby if 6 is bigger than 4 AND 5 is the same as 5, and since both of these are true, Ruby says “true” for the whole question.

But all this maths is getting a bit dull. Let’s do something useful…

# What to wear?

When I’m leaving the house I need to decide whether to wear a T-shirt or a jumper. How do I decide?

Well, if it’s hot and not raining, then I go for the T-shirt. So I could write some Ruby to tell me whether to wear my T-shirt

puts "Is it raining (yes or no)?"

raining = gets.chomp

puts "Is it hot (yes or no)?"

hot = gets.chomp

puts "You should wear a T shirt is"

puts hot == "yes" and raining == "no"

Type this is and run it. When it’s hot and not raining, it should tell you to wear the T-shirt.

## *Challenge – How many different answers are there?*

How many different ways can the weather be? Hot and not raining is one, can you spot the others? Run your program with the other answers. What does it say about wearing a T-shirt? Tell your answer to a tutor. They’ll probably get excited and start rabbiting on about truth tables. Ignore them!

# Either...or

We have used the Ruby word “and” to ask if it is hot *and* raining, but Ruby also knows the word “or”, which does exactly what you’d expect.

## *Challenge – Should I put my coat on?*

Change the T-shirt example so it tells you whether to put your coat on. For me, I wear my coat if it is raining OR it is cold.

When I run the program it should go something like this

Is it raining (yes or no)?

no

Is it cold (yes or no)?

yes

You should wear a coat is

true

It this case it tells me to wear a coat because it is cold outside. Try all the different answers and see what happens. The tutor is probably still with you from the last challenge, but if they’ve wandered off, shout at them to come back and look at your coat program.

## *Challenge – Think of your own decisions*

Deciding what clothes to wear is just one of many things we have to think about every day. What other decisions do you make? How do you decide? Are there “ands” and “ors” in your thinking? Write some Ruby programs to make decisions for you. Show them to the tutors and see how excited they get.

#

# Lots and lots of questions

We’ve been using “and” and “or” to ask two questions at once, but you can ask Ruby as many as you want.

puts 1 == 1 and 5 > 4 and 7 < 10

true

You can do the same with “or”s.

But this is boring maths again. Time for a big challenge.

# Lots and lots of challenges

I can read your mind. You’re getting bored. This is easy, you understand it all, so you’re going to skip the challenges. There are three more pages of the things, three whole pages! It’ll take ages to do them. You can’t be bothered with boring challenges, you want to learn something new, skip on to the next topic.

I agree, it’s boring, I’d be tempted to skip it too.

Well you can skip them if you want. This isn’t school. You do what *you* want to do.

The problem is that if you skip the challenges, you won’t really learn about Ruby questions. You’ll think you know it, but you’ll wake up tomorrow and remember nothing! Unfortunately that’s the way our brains work. Basically we’re all stupid – me, you, your teachers at school, all of us! We have to learn the same thing again and again and again before it sticks. The again and again bit is boring, but it’s the only way to learn since we’re all so stupid.

Hopefully one day someone will invent a pill that makes us really clever and means we only have to learn things once. But they haven’t invented the Clever Pill yet.

So if you want to remember all this tomorrow you should really do the challenges. Then you remember tomorrow, you will remember next week, you will remember next year. You will wake up as a 70 year old man or woman and *still* remember. That’s pretty cool.

## *First Challenge – Remember Ruby Warrior?*

**Have you played Ruby Warrior? In Ruby Warrior you have to make decisions, whether to rest or walk or attack, based on the things you know.

Write a program that asks questions then decides whether your warrior should rest. The questions you can ask are

* Question - Are you tired?
* Question - Is there an enemy right in front of you?

Your warrior should rest if they are tired, unless there’s an enemy right in front of them. If you rested with an enemy in front of you, they’d kill you!

Depending on which answers you give, you should see something like this

Are you tired (yes or no)?

yes

Is there an enemy right in front of you (yes or no)?

yes

You should rest is

false

Are you tired (yes or no)?

no

Is there an enemy right in front of you (yes or no)?

yes

You should rest is

false

Are you tired (yes or no)?

yes

Is there an enemy right in front of you (yes or no)?

no

You should rest is

true

Are you tired (yes or no)?

no

Is there an enemy right in front of you (yes or no)?

no

You should rest is

false

Did it work?

##

## *Bonus Challenge – Ruby Warrior continued…*

Add some more code to your Ruby Warrior program that decides whether to attack. How many questions do you need to make this decision? When you run your new program it might look something like this

Are you tired (yes or no)?

yes

Is there an enemy right in front of you (yes or no)?

yes

You should rest is

false

You should attack is

true

Finally, write more code to decide whether to walk. What questions do you need to ask? Your program should look something like this

Are you tired (yes or no)?

no

Is there an enemy right in front of you (yes or no)?

no

You should rest is

false

You should attack is

false

You should walk is

true

Don’t forgot the important Ruby Warrior rule – you can only do one thing per go, so whatever the answers, there should only be one **true** in your decisions! Answer the questions in all the different ways you can and make sure you always see two **falses** and one **true** in your answers.

##

## *Super Special Bonus Challenge – Advanced Ruby Warrior players only!*

Do you remember the archers? They can hurt you from far away by firing arrows. You could ask another question about them

* Question – Are you taking damage from an archer?

If you were, you had to walk forward and attack them, even if you were tired. Does this change your decision about walking, resting or attacking? Add the question to your program and change your walk, rest and attack decisions if you have to.

Are you tired (yes or no)?

yes

Is there an enemy right in front of you (yes or no)?

no

Are you taking damage from an archer (yes or no)?

yes

You should rest is

false

You should attack is

false

You should walk is

true

Remember, no matter how you answer the questions, only one **true** answer is allowed.



Do you remember the captives? You could ask another question about them

* Question - Is there a captive right in front of you?

And make another decision

* Action - Rescue captive

Add captives to your program and decide whether to rescue them. Don’t forget the rule – only **true** per turn allowed. You might need to change your code to get this right.

Are you tired (yes or no)?

no

Is there an enemy right in front of you (yes or no)?

no

Are you taking damage from an archer (yes or no)?

no

Is there a captive right in front of you (yes or no)?

yes

You should rest is

false

You should attack is

false

You should walk is

false

You should rescue captive is

true

Do you remember any other questions you could ask and actions your warrior could take? Put in as many as you can remember.

Keep going until you’re bored.