#

# Choices

# Running Ruby

Just like the other topics, we will be running our Ruby here

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

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

# Choices

How many choices have you made so far today? You decided when to get out of bed (or maybe your mum decided!), what clothes you put on, what to eat, you even decided to learn some Ruby. When you make a choice it always starts with a question – Do I have to go to school today? What’s the weather like? What cereal do I fancy? Shall I do some fun Ruby coding? In Ruby choices also start with a question. Let’s see how it works.

# You are the only cool person in the world!

I was 10 when I started coding. Here is one of the first programs I wrote. Type it in and run it.

puts "What is your name?"

name = gets.chomp

if name == "Spencer"

 puts "You are really cool"

else

 puts "You are not cool at all"

end

My name is Spencer you see, so when I ran the program and typed in my name it would tell me I was really cool, but everyone else who ran it wasn’t cool at all. The question it asked was your name, and the choice is whether it tells you you’re cool or not. Change it to be your name, then run the program and make sure it thinks you’re cool. Ask a tutor to type their name, the computer won’t think they’re cool, unless they happen to have the same name as you.

##

## *Challenge – Write a maths quiz*

Write a program that asks the question

What is 7 \* 8?

If they get the answer right, your program should say

Correct!

If they get it wrong your program should say

Wrong!

Does your program work? If you get stuck, look back at the Variables topic. If you’re still stuck, ask a tutor to give you a hint.

# True or False (again!)

Do you remember this program from the True or False topic?

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 in again and run it to remind yourself what it does. Now change it slightly…

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

raining = gets.chomp

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

hot = gets.chomp

if hot == "yes" and raining == "no"

 puts "You should wear a T shirt"

end

Now run it. If it is hot and not raining, it will say

You should wear a T shirt

Otherwise it will say nothing at all.

You see, True or False is about questions, and asking questions allows us to make choices. In Ruby, to make a choice you write **if**, then a True or False question, then the thing you will do if the answer is True. Oh, and then the end. Don’t forget the end.

# Don’t forget the End!

One of the mistakes that everyone makes, even people like me who have been coding for years and years, is missing the **end** off. If you miss the **end** Ruby will show you a scary error message. Try it now. Remove the **end** from your program, run it and look at the scary error message. Memorise it. You will make this mistake many times and you will see this error message every time you do. You will see it, know what it means and know how to fix it.

# Two important things

Now I will tell you the most important thing to learn in this whole topic. **Don’t skip this bit!**

**The stuff after an IF is just a True or False thing. If the answer is *True*, it does what it says between the IF and the END (or the ELSE if there is one).**

Now I will tell you the *other* really important thing to learn in this topic. **Don’t skip this bit either!**

**If the stuff after the IF is *False* and there is an ELSE, it does what it says between the ELSE and the END. If there is no ELSE, it does nothing at all.**

Let’s prove that second bit now. Add a couple more lines to your program…

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

raining = gets.chomp

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

hot = gets.chomp

if hot == "yes" and raining == "no"

 puts "You should wear a T shirt"

else

 puts "You should wear a coat"

end



Now run it and tell the computer it is hot and raining. It should tell you to wear your coat. The computer doesn’t want you to get wet.

Do you see how this works? If you tell the computer it is hot and raining, the bit after the **if** is False.

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

It is hot, that bit is True, but it is also raining, so raining == “no” is False, which makes the whole thing False. So Ruby jumps to the bit of code after the **else** and tells you to wear a coat.

If this is confusing, have a look back to the True or False topic, the bit about how **and** works.

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

Remember we did this challenge in the True or False topic? You had to write a program to ask the ages of you and your best friend, then tell you if you were younger than them. When you ran your program you saw something like this.

How old are you?

9

How old is your best friend?

10

You are younger than your best friend is

true

Well, now I want you to do the challenge again, but this time use **if**. If you are younger the program should print something like this…

How old are you?

9

How old is your best friend?

10

You are younger than your best friend

## Does it work?

## If you are older than your best friend you would like your program to say something like this…

How old are you?

9

How old is your best friend?

8

You are older than your best friend

## And if you and your best friend are the same age you would like your program to say this…

You and your best friend are the same age

But how do we make our program do all that? This isn’t a simple **if else** because there are three different things we might want our program to say. Well, that’s what we are going to learn about next.

# Three choices

# If and else are fine so long as you have only two choices to make, like wearing a T shirt or a coat depending on the weather. When you have more than two choices, you need a Ruby elsif. What on earth is elsif? It’s Ruby shorthand for “else if”. Let’s see it in action.

Remember my coolness program? Well, once I learnt about **elsif** I made it a bit cleverer. This is what I did…

puts "What is your name?"

name = gets.chomp

if name == "Spencer"

 puts "You are really cool"

elsif name == "Mitchell"

 puts "You are a complete idiot!"

else

 puts "You are not cool at all"

end

Can you see how this works? Mitchell is my brother’s name. When he ran the program it told him he was an idiot, which he didn’t find very funny.

## *Challenge – Finish the last challenge*

Knowing about **elsif** means you can finish off the last challenge, which tells you if you are older, younger or the same age as you best friend. Have a go. Run it a few times and put in different ages to make sure it all works.

## *Challenge – Nicknames*

Sometimes my brother calls himself Mitch, for short. I want my program to call him an idiot whether he types in his full name or his nickname. You can do it by changing just one line. Can you do it?

**Hint** – You need to use **or**. Look at the Questions topic if you don’t remember how **or** works.

# Multiple choices

With **elsif** you are not limited the three choices, you can have as many as you like. How does all this work? Well, you always start with an **if**, then you have as many **elsifs** as you like, then you might have an **else** (but you don’t have to), then after all that comes the **end**.

Let’s try it out. Can you guess what this code will do?

puts "What colour is the sky?"

colour = gets.chomp

if colour == "blue"

 puts "It must be sunny"

elsif colour == "black"

 puts "It must be night time"

elsif colour == "grey"

 puts "It might rain"

end

I’ve put in three colours for the sky – blue, black and grey. Can you think what other colours the sky might be?

## *Challenge – Red sky at night*

Sometimes, not often, but sometimes the sky will go red when the sun is setting. Make the program check for red skies. If the sky is red your program should say

It must be a sunset

## *Challenge – Crazy sky colours*

What happens if you run the program and type in some other colour, like pink? Try it. The program doesn’t print anything. It doesn’t print anything because there is no **else** in your code. In Ruby, **else** means that if none of the **if** or **elsifs** are True, run the code under the **else**.

Change your program so when you type in any colour other than blue, black, grey or red it says

What planet are you living on?

## *Bonus Challenge – Crazy sky colours II*

Do you remember how to print out variables? Check the Variable Topic sheets if you can’t. Then make your program print the name of the crazy colour, so if they type in “pink” it should say

pink! What planet are you living on?

#

# Third important thing

****So now we know about **elsif**, I can tell you the third really important thing to remember from this topic.

**Ruby moves down the IFs and ELSIFs looking for the first thing that is *True* so it can run the code underneath it. If nothing is *True* and there is an ELSE, it runs the code underneath that.**

#

# Nesting

Isn’t that what birds do? Well, yes, but it also has a meaning in Ruby. It means putting **ifs** inside other **ifs**. Why would you want to do that? I’ll show you. Let’s write an adventure game

## *Prison escape game*

In this game you find yourself in a prison cell. The aim of the game is to escape from the cell. There is a door, but it is guarded. In the cell are you, the guard, and a small elf sitting in the corner.

Here is the code for the game. Type this in and run it to play the game.

puts "You are in prison. An elf sits in the corner"

puts "To escape you have to get past the guard"

puts "Do you want to fight him? y is yes"

fight = gets.chomp

if fight == "y"

 puts "The guard is strong and wins. Game over!"

else

 puts "You talk to the elf who offers you a mars bar."

 puts "Do you eat it or give it to the guard? e or g"

 action = gets.chomp

 if action == "e"

 puts "Tasty, but you are still in prison. Game over!"

 else

 puts "You give the guard the mars bar and escape"

 puts "while he is eating it. You win!"

 end

end



 Do you see the way we have an **if** inside an **else** here? After you wisely decide not fight, Ruby starts running the stuff between the **else** and the **end,** which includes the question about the mars bar, then another **if**.

## *Challenge – Extend the prison game*

## At the moment if you eat the mars bar you lose. Change the game so when you eat the mars bar the elf then offers you a snickers, which you can either eat of give to the guard. If you eat it you lose, if you give it to the guard you win.

## Change the game again so you can only win if you give the guard the snickers. He doesn’t like mars bars, so if you give him the mars he gets angry and locks you to the bed!

Think up more bits to the prison game. Ask one of the other kids to play and see if they can win.

# The dangers of messy nesting

 So now we have **ifs** inside **ifs** inside **elses** it can all get very confusing. To make it less confusing it’s best if you line things up with spaces. If you have an **if**, **elsifs**, an **else** and an **end**, try to line them all up. Move the code between in a couple of spaces. It’s boring, but it will make your code easier to understand and you will make less mistakes.

If you don’t line it all up nicely the code will still work, but anyone looking at it will be scratching his or her head. To convince you I will write the prison game again without lining it up properly so you can see how confusing it looks.

puts "You are in prison. An elf sits in the corner"

puts "To escape you have to get past the guard"

puts "Do you want to fight him? y is yes"

fight = gets.chomp

if fight == "y"

puts "The guard is strong and wins. Game over!"

else

puts "You talk to the elf who offers you a mars bar."

puts "Do you eat it, or give it to the guard? e or g"

action = gets.chomp

if action == "e"

puts "Tasty, but you are still in prison. Game over!"

else

puts "You give the guard the mars bar and escape"

puts "while he is eating it. You win!"

end

end

What a mess!

And that’s it for the Choices topic. The next one will be about looping. We will be writing more games too.