boot camp

/ˈbo͞ot ˌkamp/

noun NORTH AMERICAN

noun: **bootcamp**

a military training camp for new recruits, with strict discipline.

- a prison for youthful offenders, run on military lines.
- a short, intensive, and rigorous course of training.
  "a grueling, late-summer boot camp for would-be football players"

You can use this space to make a list of your fellow students in class,

A. Preface

▶ Exercise A1

Welcome to your first Exercise. Your task is to setup your very own public **Blog**, which will be accessible to the entire world! **Do** the following (where applicable),

a. Choose a Blogging platform you are going to use
   E.g. **wordpress.com**, **medium.com**, etc
b. Signup for your blog
c. Pick a **Theme** you love
d. Create a few **Pages** (e.g. 'About', 'Contact', etc)
e. Create a new "Hello World' blog **Post**
f. Upload some **Images**
g. Configure your site to either use a '**Static**' homepage or a '**Blog**' view (make sure you know how to switch between the two)
h. Create a **Menu** to assist in navigation of your site

Blogs are great platforms for communication & collaboration and also provide an effective platform to showcase you & your skills, talents and portfolio.

What is the web address (URL) of your blog?

Throughout your time at Bootcamp you will nurture & build your blog. Followings are some Do's and Dont's to follow,

Do's

- Create tutorials blog posts on how to do stuff (e.g. setting up Ruby, etc)
- Blog on interesting & relevant articles and news in the current media
- Provide your opinion, analysis & research
- Blog on meetups & industry events that you attend
- Build a list of references and links of interest
- Chronicle research on topics that interest you
- Setup an online Resume
- Update your blog regularly (at least a few times a week)

Dont's

- Do NOT publish or provide any Coder Academy content on your site, including this worbook, handouts, assignment, answers to your Exercises or projects, etc
- Do NOT neglect your blog and try and update it all at once at the end of the term

Have fun Blogging!

▶ Exercise A2

Over the course of this workbook you will be creating a lot of (source code) files. It is vitally important you are organized from the start. So before we dive in, we need to setup a place for you to store all your files,

Do

1. On your computer navigate to where ever you store your documents (i.e. your home directory)
2. Create a folder (directory) called `ca_workbook`
3. As you work through this workbook create a new folder in `ca_workbook` for each exercise that requires you to create a file (e.g. source code file)
4. Store your exercise solution file(s) in your associated exercise folder

Exercise A3 ◀

Irrespective of whether you use *Mac*, *Linux* or *Windows* having an operating system (OS) in which you can playaround and experiment is vital. A **Virtual machine** is a software computer that, like a physical computer, runs an operating system and applications. You can install it on your current OS and use it as a sandbox to play and experiment in. Your mission (should you choose to accept it) is to,

Do

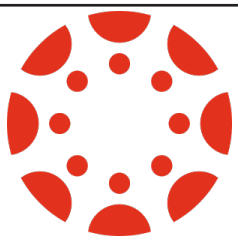
1. Download & install the popular & freely available **Virtual Box** (https://www.virtualbox.org/)
2. Once installed download a Linux distribution, for e.g. **Ubuntu** (https://www.ubuntu.com/) and install it in your Virtual Box

This might be a bit challenging if you've never done this before. But don't fear you will learn so much along the way. Do this as a group with your peers. Use google, youtube, etc as a good resource

B. Tools

▸ Exercise B1

Following is a list of common tools & services that will be used throughout the term(s). Make a note about the purpose of each,

Canvas	https://coderacademy.instructure.com
	☐ Tick this box if local installation is required

Chrome / Firefox	https://www.google.com/chrome/ https://www.mozilla.org/en-US/firefox/
	☐

Text Editor	https://code.visualstudio.com/
	E.g.: Visual Studio Code, Sublime Text, Atom, TextMate, Vim, Nano, Pico ☐

Slack	https://slack.com/
	☐

Python	https://www.python.org
	☐

Use this area to list some of the other tools that may not be presented on these pages,

Git	https://git-scm.com/
	☐

GitHub	https://github.com/
	☐

Trello	https://trello.com/
	☐

PyCharm	
	☐

Cloud Services	
	☐

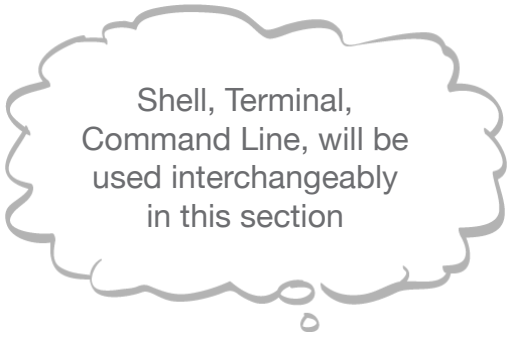
VMs	
☐	☐

C. Terminal (Command Line)

"In the Beginning was the Command Line."

► Exercise C1

This is an example Terminal session,



Annotate the picture of the terminal with your answers

For each line in the Figure above identify the following,

- Prompt
- Command
- Options
- Arguments
- Cursor
- Output/Results

► Exercise C2

As you dive into coding you will find that it is really useful to have more than one terminal open. Many modern terminal programs provide the ability to create multiple tabs.

Examine your terminal program and work out how to create a new tab. List the steps in the box below,

What is the keyboard shortcut for creating a new tab,

► Exercise C3

A list of common terminal commands is listed below. Provide descriptions for each,

Commands	Description
cd <dir>	
cd ~/<dir>	
pwd	
mkdir <name>	
rmdir <dir>	
rm -rf <dir>	
ls	
ls -l	
ls -a	
touch <file>	
mv <old> <new>	
cp <old> <new>	
echo <string>	
man <command>	
^C	
^A	
^E	
^U	
clear	
exit	
cat <file>	
diff <f1> <f2>	
> [file]	
>> [file]	
which	
curl	
head <file>	
tail <file>	
wc <file>	
grep <string> <file>	
ps	
ping <url>	
sudo	
chmod	
env	
find	

▸ Exercise C4

Carry out the following instructions and document the commands used,

Question	Commands
Open up Terminal and change to the `Desktop` directory.	
Create a directory called `flavours`. Change into this directory.	
Print the current directory the shell is in.	
Make the file `vanilla.txt`	
List all the files that are in the `flavours` directory.	
Add the text "`My favourite flavour`" to the `vanilla.txt` file	
Print the contents of the `vanilla.txt` file in the command line.	
Change back to the `Desktop` directory.	
Create a directory called `recipe`	
Move the `~/Desktop/flavours/vanilla.txt` file to the `~/Desktop/recipe` directory.	
Create a file called `chocolate` in the `recipe` directory.	
Copy `~/Desktop/recipe/chocolate` file to the `~/Desktop` directory.	
Delete the `~/Desktop/chocolate` file.	
Delete the `~/Desktop/flavours/` directory.	
Delete the `~/Desktop/recipe/` directory.	

▸ Exercise C5

`CmdChallenge.com` is an interactive online tutorial for learning and practicing the terminal by incrementally completing small (increasing complex) challenges. Visit the website and mark of (☑) the challenges you complete below,

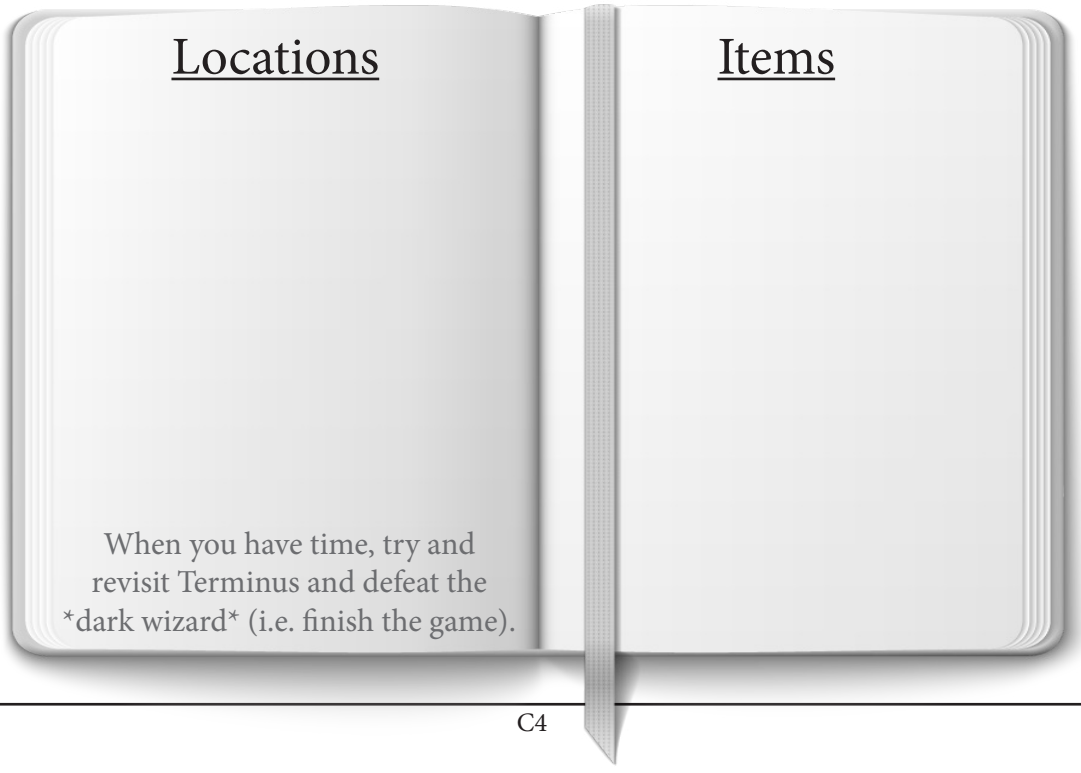
- ☐ hello_world/
- ☐ current_working_directory/
- ☐ list_files/
- ☐ print_file_contents/
- ☐ last_lines/
- ☐ find_string_in_a_file/
- ☐ search_for_files_containing_string/
- ☐ search_for_files_by_extension/
- ☐ search_for_string_in_files_recursive/
- ☐ extract_ip_addresses/
- ☐ delete_files/
- ☐ count_files/
- ☐ simple_sort/
- ☐ count_string_in_line/
- ☐ split_on_a_char/
- print_number_sequence/ ☐
- remove_files_with_extension/ ☐
- replace_text_in_files/ ☐
- sum_all_numbers/ ☐
- just_the_files/ ☐
- remove_extensions_from_files/ ☐
- replace_spaces_in_filenames/ ☐
- dirs_containing_files_with_extension/ ☐
- files_starting_with_a_number/ ☐
- print_nth_line/ ☐
- reverse_readme/ ☐
- remove_duplicate_lines/ ☐
- disp_table/ ☐
- find_primes/ ☐
- corrupted_text/ ☐

For fun, once you have completed the 'standard' challenges try some of the 'User Contributed' challenges

▸ Exercise C6

You are player dropped into the mysterious land of *Terminus*! By exploring your surroundings and acquiring magic spells, you are destined to defeat the dark wizard plaguing the local citizens. Visit the website (`http://www.mprat.org/Terminus/`) and explore at least **5** *Locations* and interact with at least **5** *items*. Remember to keep a log of your activities in your diary below,

`http://www.mprat.org/Terminus/`



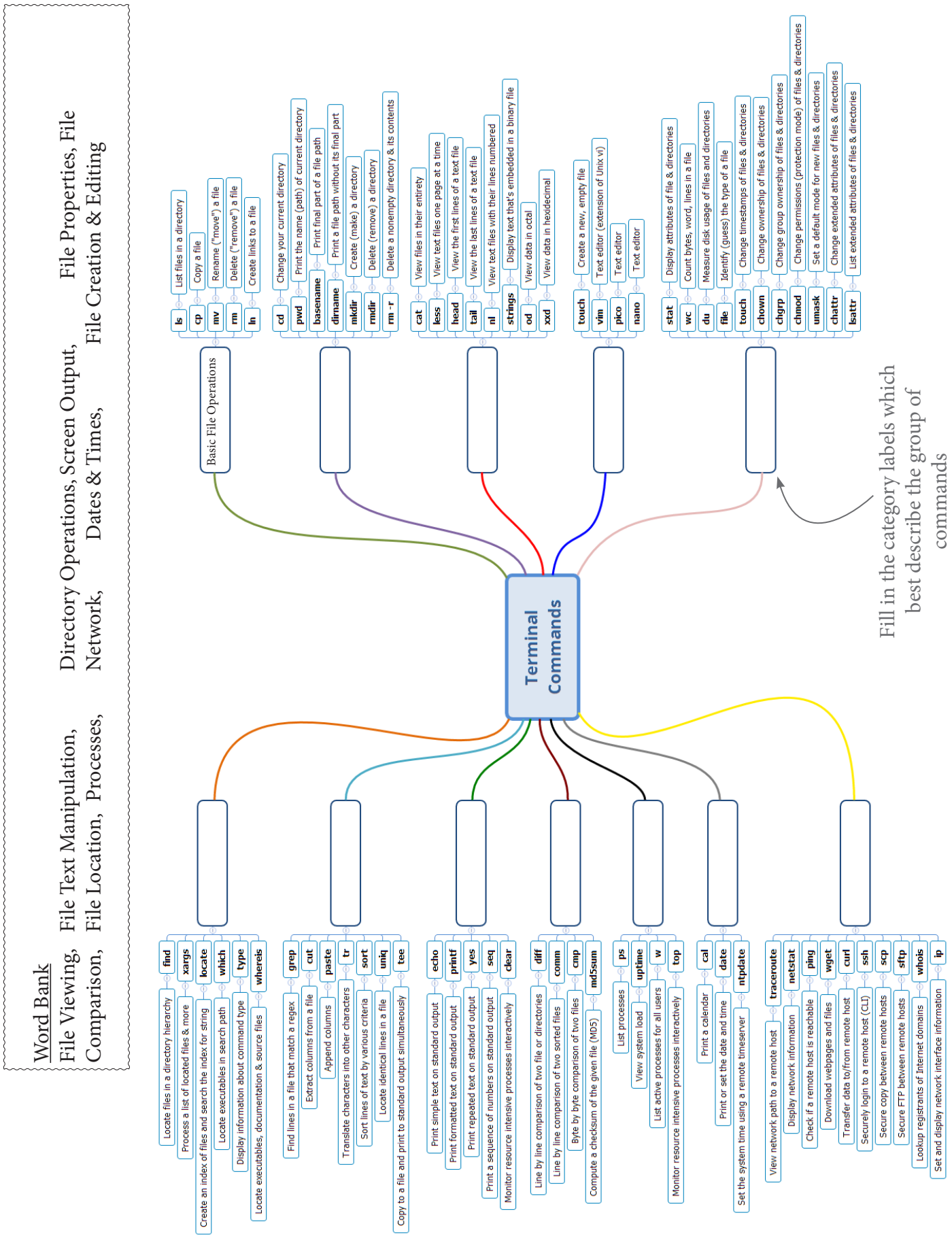
► Exercise C7

Carry out the following instructions and document the commands used,

Question	Commands
Create a folder `myLearnings`	
Enter (change) into the folder `myLearnings`	
Create a file `Terminal_Story`	
Create another file `test_file`	
Verify the files were created	
Add the text "`The Command Line! - Your window into the computer`" to the `Terminal_Story` file.	
Print the contents of the `Terminal_Story` file in the command line.	
Add the text "`Basic commands I learnt today`" to the `Terminal_Story` file.	
Display the contents of the `Terminal_Story` file in the command line.	
Add some more of your own lines to the `Terminal_Story` (Think of this is like your terminal diary where you write down the cool terminal stuff you learn everyday)	
Rename the file `Terminal_Story` to `my_Terminal_Story.txt`	
Verify the rename was successful	
Make a copy of `my_Terminal_Story.txt` to `backup.txt`	
Make sure `backup.txt` was created successfully and print out it's contents	
Delete `test_file`	
Compare the files `my_Terminal_Story.txt` to `backup.txt`	

▸ Exercise C8

Following is an overview of terminal commands generally available on most *nix OS's. Fill in the missing group/category labels by choosing from word bank below,



► Exercise C9

Carry out the following instructions and document the commands used,

Question	Commands
Create the directory `~/Desktop/ab`	
Create the files `f1`, `f2`, and `f3` in the `~/Desktop/ab` directory.	
Create the `~/Desktop/ab/yz` directory and move `f1` and `f3` into the `yz` directory.	
Append the text "`I am file 2`" to the `f2` file. Then move `f2` into the `~/Desktop/ab/yz` directory and rename it to be `really_cool` Check that the `really_cool` file still has the content "`I am file 2`".	
Copy `f1`, `really_cool`, and `f3` from the `~/Desktop/ab/yz` directory to the `~/Desktop/ab/www` directory.	
Create the file `~/Desktop/ab/f1` and add the text "`I am :)`" to the file. Copy `f1` to the `www` directory and explain what happened to the original `~/Desktop/ab/www/f1`	
Create the file `~/Desktop/ab/bob` . Then rename the file to be `~/Desktop/ab/fred` .	
Use the `-l` option to provide a long listing of all the files in the `~/Desktop/ab/` directory.	
Use the `-l` and `-F` options to provide a long listing of all files in the `~/Desktop/ab/` directory and distinguish between regular files and folders.	
Create the `~/Desktop/ab/sky` file. Use the `-i` (interactive) option and delete the `~/Desktop/ab/sky` file.	
Delete the `~/Desktop/ab` directory and all of its contents.	

► Exercise C10

Carry out the following instructions and document the commands used,

Question	Commands
What directory does the following code switch to? `$ cd /`	
Switch to the `Desktop` directory and make the folder `hiya`	
Change from the `~/Desktop/hiya` directory back to the `~/Desktop` directory.	
Identify the program and arguments of the following command line: `$ ls ~/Desktop/hiya`	

Run the following commands. Then list all the files in the `we_care` directory with the `.txt` extension.
`cd ~/Desktop`
`mkdir we_care`
`touch we_care/one.txt we_care/woo we_care/foo.txt`	

Delete the `~/Desktop/hiya` and the `~/Desktop/we_care` directories.	
Explain what the following command prints to the shell. `$ echo $HOME`	
Explain what the following command prints to the shell. What is the significance of this output? `$ echo $PATH`	
What file is the `ls` program defined in?	
What file is the `top` program defined in?	
Create the file `$HOME/Desktop/moo`. Then delete it.	
The command `ls -l` provides a long listing of the files in a directory. Alias `longls` so it can be used in place of `ls -l`	
Show a history of commands you've entered in the shell.	
Is `$HOME/Desktop/` an absolute or relative path? Explain.	

Hint: man pages

► Exercise C11

For the **`ls`** command try each of the options below and then match it to the description of what it does,

► Exercise C12

Carry out the following instructions and document the commands used,

Question	Commands
Write the text "`job`" to the file `~/Desktop/motivation`	
Now write the text "`lolly`" to the file `~/Desktop/motivation`. Describe what happened to the original content of the `~/Desktop/motivation` file.	
Append the text "`that was funny`" to the `~/Desktop/motivation` file	
Write the output from the history command to the `~/Desktop/last_500` file.	
Find all instances of the substring bin in the `~/Desktop/last_500` file.	
By default, the output of Shell programs is written to the shell. Use a pipe to redirect the output of the `history` command to the `grep` command and find all the commands in the history that contain the substring `ls`	
Return the number of results that are returned by the `history` command.	
The directories in the `$PATH` environment variable are typically separated with colons. List all the directories in the `$PATH` environment variable, but separate them with spaces.	
List all the files that are contained in the directories in the path.	

► Exercise C13

Carry out the following instructions and document the commands used,

Question	Commands
Create the directory `~/Desktop/lnk/` and add the file `food` to the directory.	
Create a hard link called `food-link` to the `~/Desktop/lnk/food` file.	
Append the text "`I like veggies`" to the `food-link` file and observe that this text is added to the `food` file too.	
Append the text "`Hungry`" to the `food` file and observe that this text is appended to the `food-link` file too.	
Delete the `food` file and verify that the text is still readable in the `food-link` file.	
Create a hard link called `desktop-link` to the `~/Desktop` directory.	
Create the file `cool` and the symbolic link `cool-link` to the file.	
View a long listing of all files in the directory and explain how this view demonstrates the symbolic link.	
Append the text '`star wars`' to the `cool` file and then view the contents of the file with `cool-link`.	
Append the text '`star trek`' to the `cool-link` file and then view the contents of the file with `cool`.	
Delete the `cool` file and view the contents of the `cool-link` file.	


10 Growth Mindset Statements
What can I say to myself?
FIXED MINDSET
GROWTH MINDSET
INSTEAD OF:
TRY THINKING:
I'm not good at this.
What am I missing?
I'm awesome at this.
I'm on the right track.
I give up.
I'll use some of the strategies we've learned.
This is too hard.
This may take some time and effort.
I can't make this any better.
I can always improve so I'll keep trying.
I just can't do Math.
I'm going to train my brain in Math.
I made a mistake.
Mistakes help me to learn better.
She's so smart. I will never be that smart.
I'm going to figure out how she does it.
It's good enough.
Is it really my best work?
Plan "A" didn't work.
Good thing the alphabet has 25 more letters!
(Original source unknown)
@sylviaduckworth

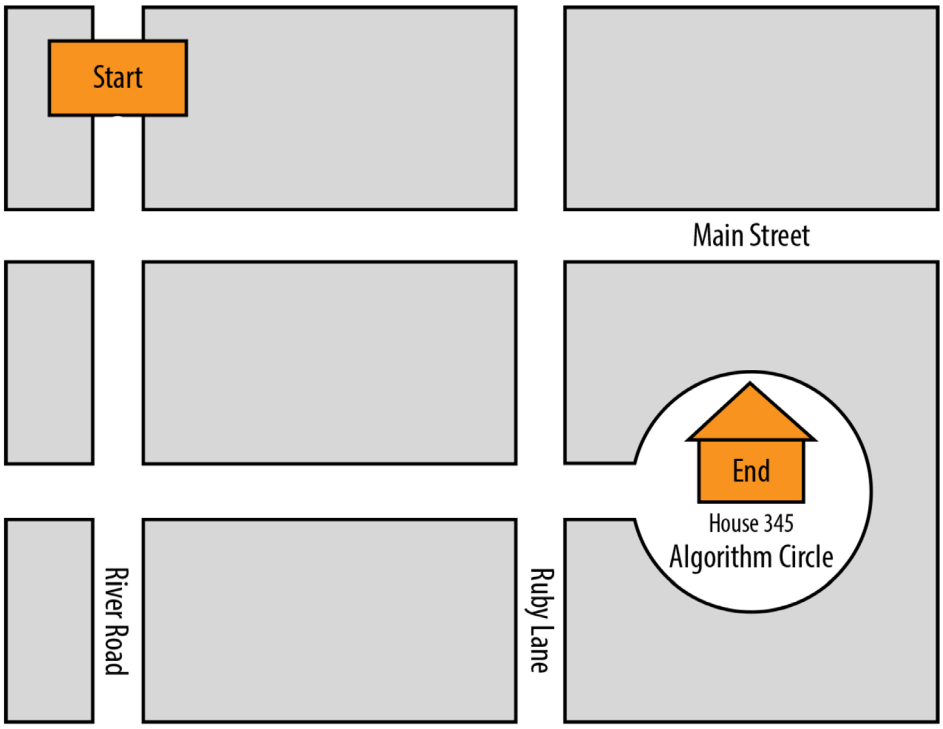

The Iceberg Illusion
Success is an iceberg
WHAT PEOPLE SEE
SUCCESS!
Persistence
WHAT PEOPLE DON'T SEE
Dedication
GOAL
Failure
FAIL
Hard work
Sacrifice
THINGS I HAVE TO GIVE UP
1.
2.
3.
Good habits
Disappointment
@sylviaduckworth

D. Code Design (Pseudocode, Algorithms & Flowcharts)

Computer programs are simply a series of instructions (i.e. algorithm). For us as programmers/coders/hackers we need to know what do to solve a particular problem before we begin to code. This is where code design in the form of *pseudocode* and *flowcharts* comes in. Start getting into the habit of planning before you dive into coding. In this section we won't do coding per se but start getting used to planning to code.

► Exercise D1

In the space provided below, write an algorithm in plain English (i.e. directions) to get from 'Start' to your destination, 'End'

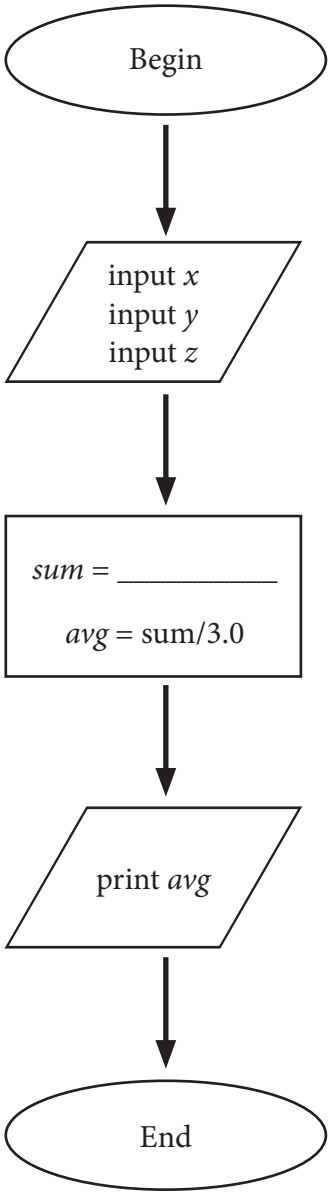


Is there only 1 way to get from 'Start' to 'End'?

Tip: The following are basic steps for solving a computer science problem:
Step 1: Understand the problem.
Step 2: Write out a solution in plain language.
Step 3: Translate the language into code.
Step 4: Test the code in the computer.

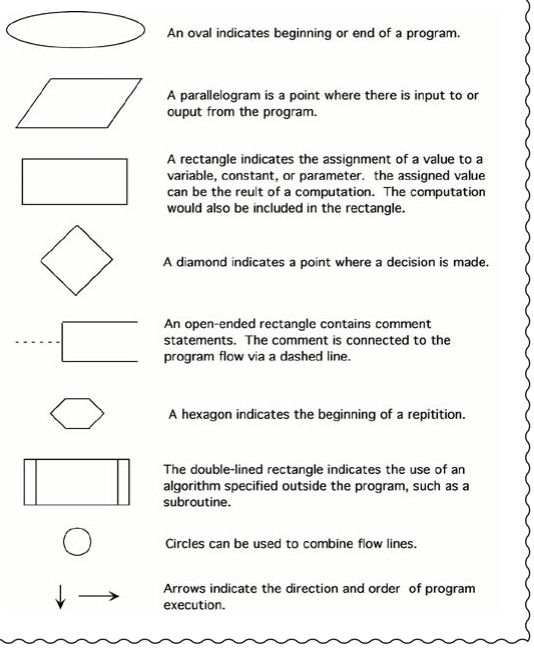
► Exercise D2

Following is a *flowchart* and *pseudocode* for **Averaging 3 numbers**. Fill in the blanks,



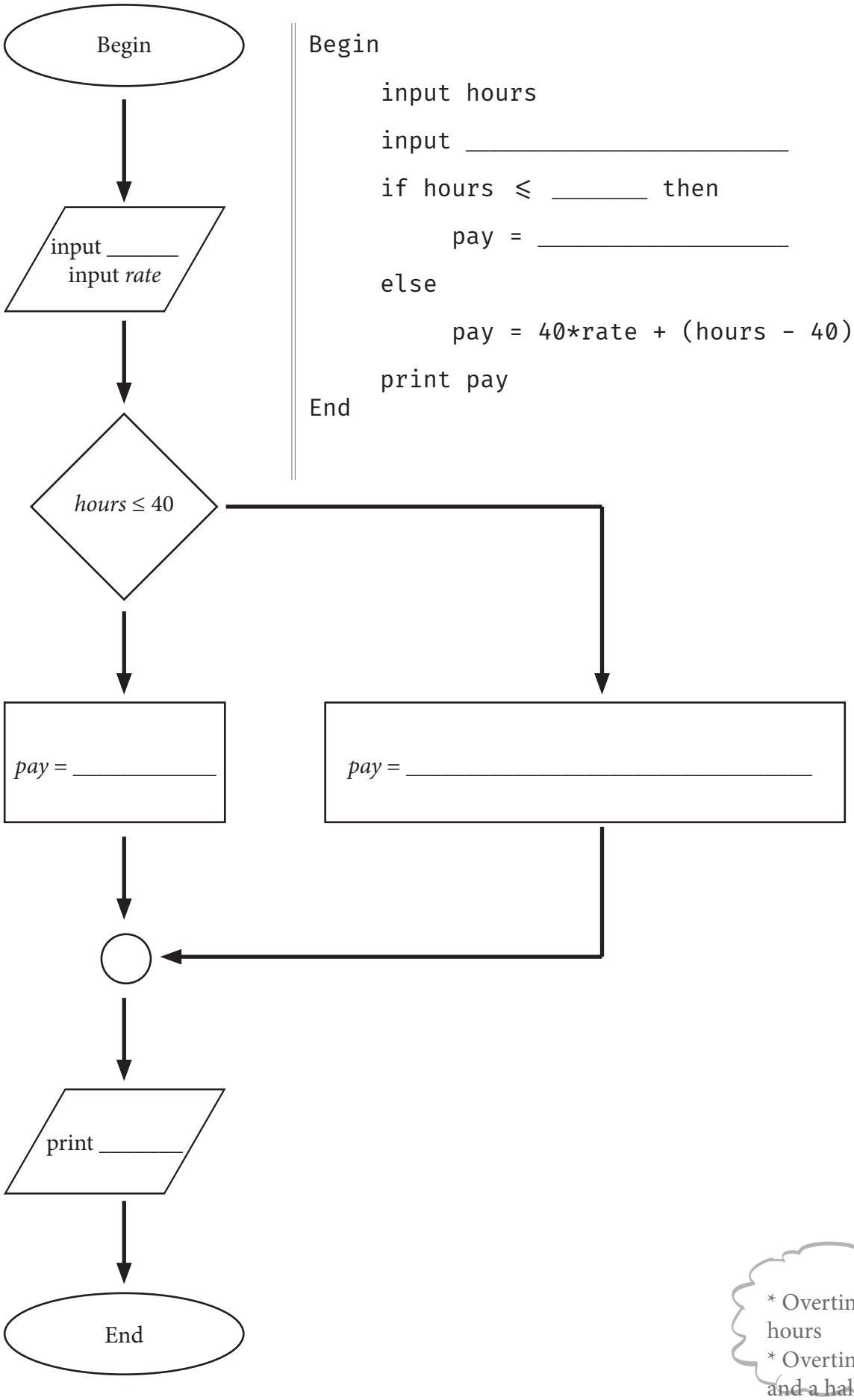

```
Begin
     input x
     input _____________________
     input _____________________
     sum = x + y + z
     avg = _____________________
     print avg
End
```

Tip: Following are some flowcharting conventions



► Exercise D3

Following is a *flowchart* and *pseudocode* for **Calculating Pay with Overtime**. Fill in the blanks,




```
Begin
     input hours
     input _____________________
     if hours ⩽ _______ then
          pay = ____________________
     else
          pay = 40*rate + (hours - 40)*rate*1.5
     print pay
End
```

* Overtime after 40 hours
* Overtime is at time and a half (i.e. x 1.5)

► Exercise D4

Write an algorithm (*pseudocode*) and *flowchart* to solve the following challenge,

List ALL the prime numbers between 1 to 100

Prime number - a number that is divisible only by itself and 1

► Exercise D5

Write an algorithm (*pseudocode*) and *flowchart* to solve the following challenge,

Convert numbers to words

E.g.:

Input = 342

Expected Result = Three and forty two

► Exercise D6

Write an algorithm (*pseudocode*) and *flowchart* to solve the following challenge,

Convert a number to digits

E.g.:

Input = 6789112

Expected Result = 6,7,8,9,1,1,2

► Exercise D7

Write an algorithm (*pseudocode*) and *flowchart* to solve the following challenge,

Reorder the digits in a number in descending order

E.g.:

Input = 473281

Expected Result = 874321

► Exercise D8

Write an algorithm (*pseudocode*) and *flowchart* to solve the following challenge,

Find the position of a digit in an array

E.g.:

Input array = [12, 4, 10, 6, 7, 9, 11, 23, 33]
Input = Find position of 9
Expected Result = 5

► Exercise D9

Write an algorithm (*pseudocode*) and *flowchart* to solve the following challenge,

Calculate the row sums of this triangle from the row index(starting at index 1)

```
1
3  5
7  9  11
13  15  17  19
21  23  25  27  29
```

E.g.:

row_sum(1) = 1

row_sum(2) = 3 + 5 = 8

► Exercise D10

Write an algorithm (*pseudocode*) and *flowchart* to solve the following challenge,

Once upon a time there was a series of 5* books about a very English hero called Harry. Children all over the world thought he was fantastic, and, of course, so did the publisher. So in a gesture of immense generosity to mankind, (and to increase sales) they set up the following pricing model to take advantage of Harry's magical powers.

One copy of any of the five books costs 8 EUR. If, however, you buy two different books from the series, you get a 5% discount on those two books. If you buy 3 different books, you get a 10% discount. With 4 different books, you get a 20% discount. If you go the whole hog, and buy all 5, you get a huge 25% discount.

Note that if you buy, say, four books, of which 3 are different titles, you get a 10% discount on the 3 that form part of a set, but the fourth book still costs 8 EUR.

Potter mania is sweeping the country and parents of teenagers everywhere are queueing up with shopping baskets overflowing with Potter books. Your mission is to write a piece of code to calculate the price of any conceivable shopping basket, giving as big a discount as possible.

(* Yes we know there are more now ;)

For example, how much does this basket of books cost?

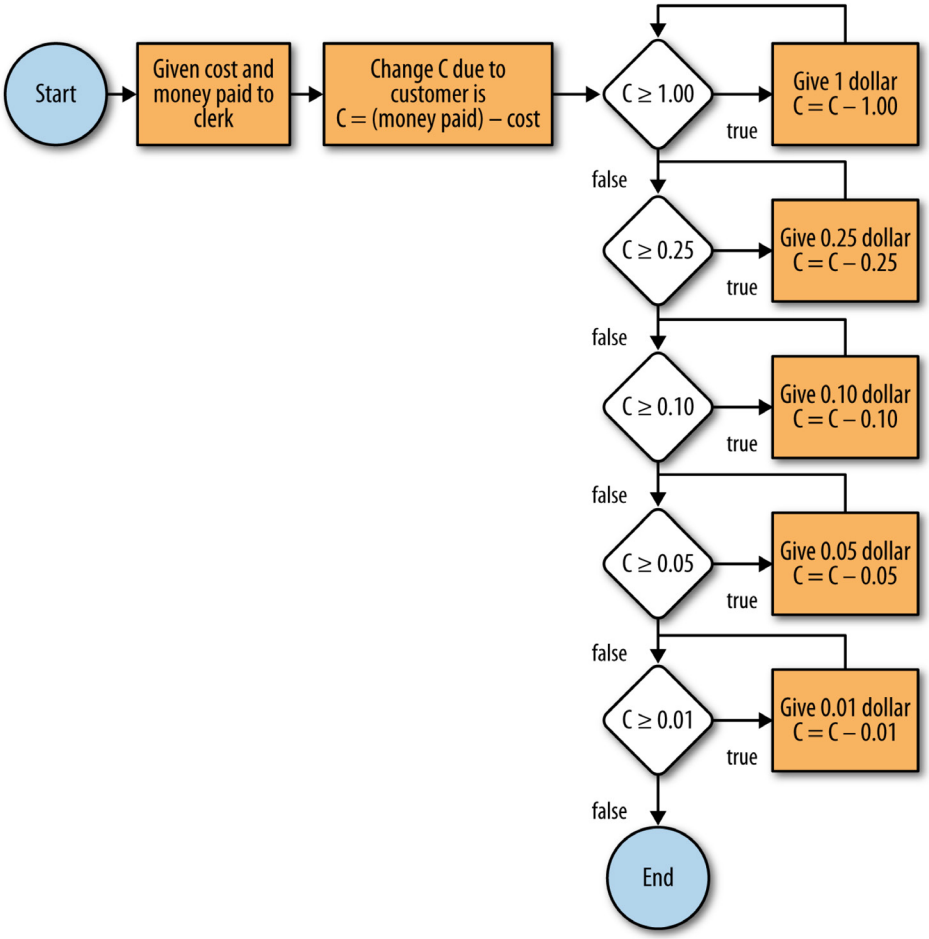
- 2 copies of the first book
- 2 copies of the second book
- 2 copies of the third book
- 1 copy of the fourth book
- 1 copy of the fifth book

Answer :
(4 * 8) - 20% [1st book, 2nd book, 3rd book, 4th book]
+ (4 * 8) - 20% [1st book, 2nd book, 3rd book, 5th book]
= 25.6 * 2
= 51.20

► Exercise D10: Continued

► Exercise D11

The following image shows an alternate (i.e. flowchart) representation of an algorithm.
(* We haven't covered some of the constructs included, but give it a go)



In the space provided below, answer the following questions,

a. Starting at the circle labeled "Start" work your way through the figure. What is the purpose of this algorithm?

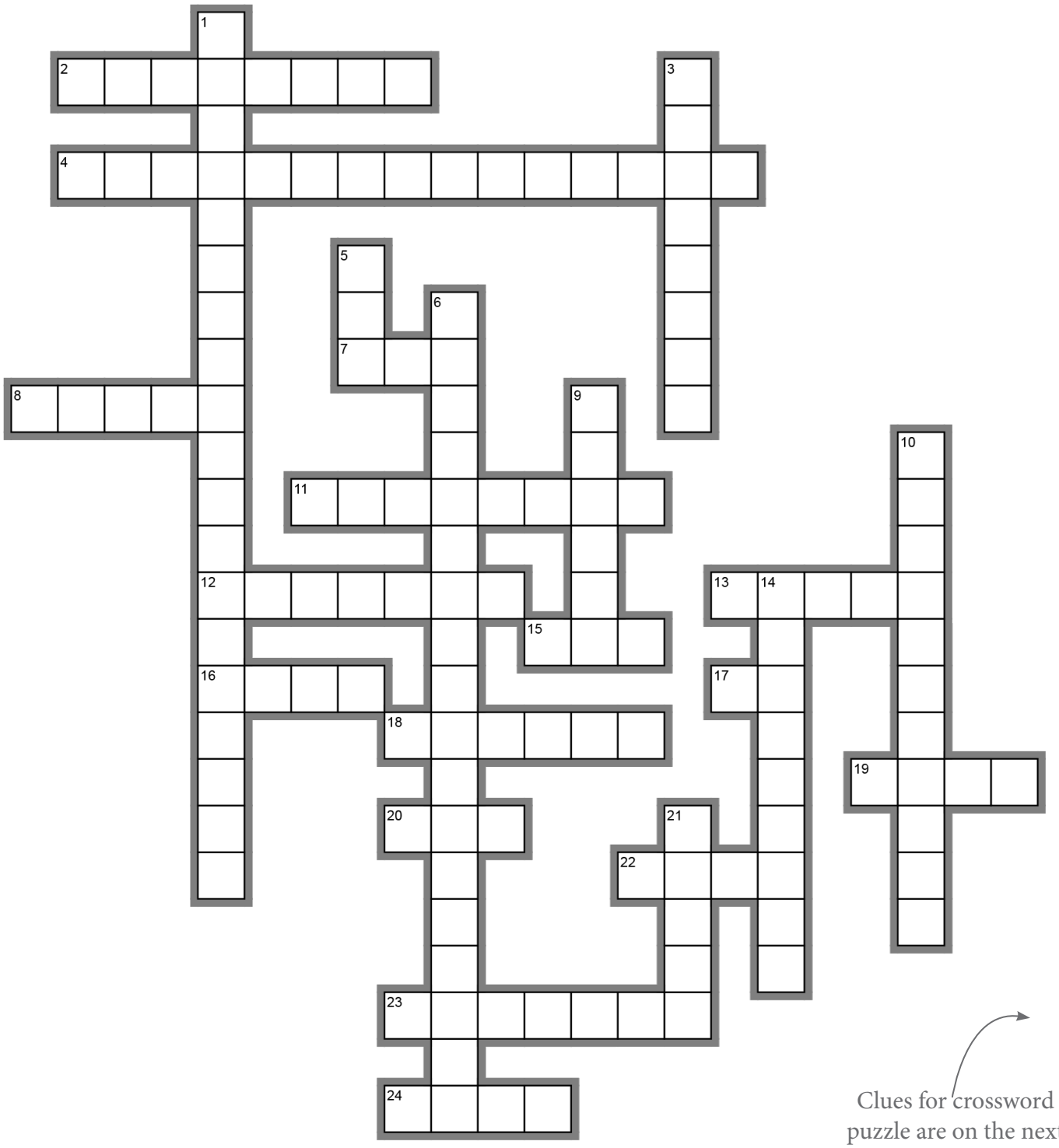
b. Translate the figure into simple language. Note that a diamond in the figure represents a condition that may be true or false.

E. Ruby

Introduction

► **Exercise E1**

Test your Ruby general knowledge by completing the following Crossword puzzle,



Clues for crossword puzzle are on the next page

Hint

- Multiple word answers do not have spaces in them (i.e. in the crossword words are entered without spaces between them - where applicable)
- All numbers are in word format (e.g. 9 = nine)

Across

2. Name an implementation of Ruby written in Ruby? (8)
4. What does the abbreviation IRB stand for? (15)
7. What is the reference implementation of Ruby called? (3)
8. Ruby originated from which country? (5)
11. Who many reserved words does Ruby have? (8)
12. Which major online service was originally written in Ruby? (7)
13. Name the lightweight implementation of the Ruby language that can be linked and embedded within an application (it's development is led by Ruby's creator) (5)
15. Which standards organization provides an internationally recognized specification for the Ruby language? (3)
16. The reference implementation of Ruby is written in C. True or False? (4)
17. What is the file extension for Ruby source code files? (2)
18. The source code for the reference implementation of Ruby can be found on which version control repository? (6)
19. What is the nickname of the inventor of Ruby? (4)
20. Can MacOS desktop applications be written in Ruby? (3)
22. What other programming language was released to the public in the same year as Ruby? (4)
23. The book commonly referred to as the definitive reference to Ruby is known as the "_____book" (7)
24. Ruby is a language created by blending parts of Smalltalk, Eiffel, Ada, Lisp and which other language? (4)

Down

1. In what year did the development of Ruby start? (19)
3. Ruby documentation can be viewed at this website www._____.org (8)
5. Ruby programs and libraries are commonly distributed in what self contained format? (3)
6. In what year was the first public release of Ruby? (18)
9. Which major Learning Management System (LMS) was written in Ruby? (Hint: It is the same LMS that is used by Coder Academy :) (6)
10. What App made Ruby famous? (Hint: Framework) (11)
14. The official Ruby language website can be found at www._____.org? (9)
21. Ruby is case insensitive. True or False? (5

Installation

► Exercise E2

Which version of **Ruby** did you install?

► Exercise E3

Which **OS** (Mac, Linux or Windows) did you install **Ruby** on?

► Exercise E4

Describe the steps required to install **Ruby** on your **OS**,

This could make a good tutorial for your blog

Remember to mention websites visited for downloads, etc

► Exercise E5

Ruby Version Manager (RVM) is a command-line tool which allows you to easily install, manage, and work with multiple ruby environments. Late one night a talented Ruby programmer sat and started experimenting with RVM. Below is a snapshot of their terminal session. Describe what is happening in each command and the expected output (if any),

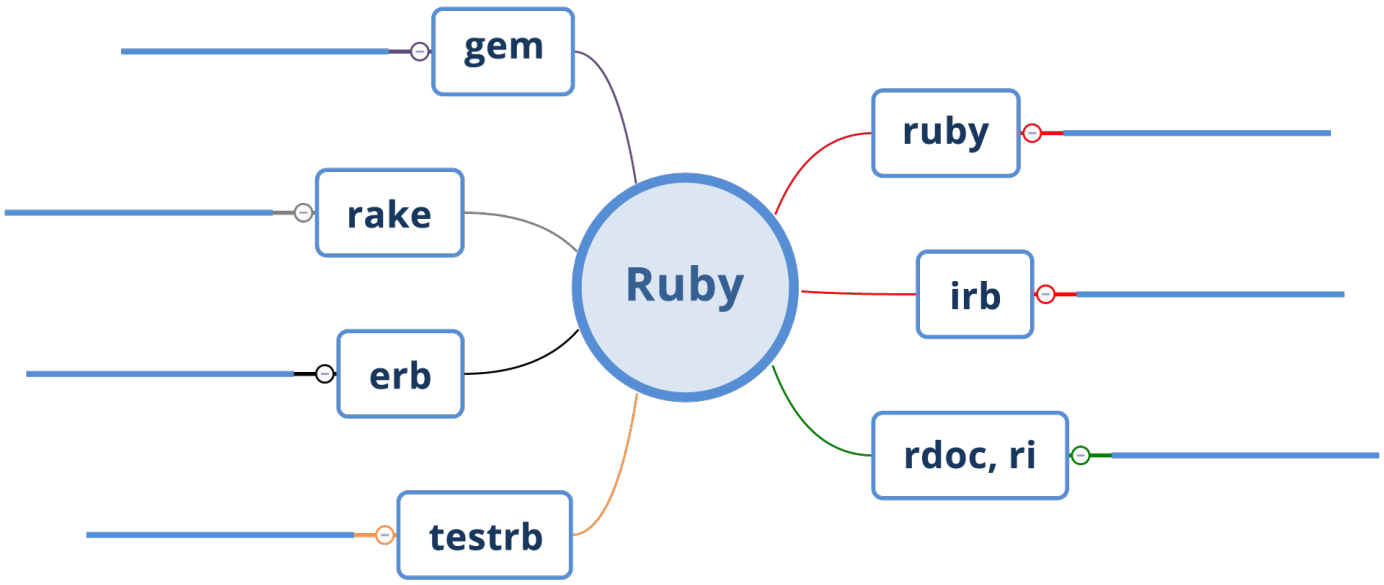
'rbenv' is another popular Ruby package manager. If you are using rbenv (instaed of RVM), no problems, find out the commands that match what is being done in the table. Have fun :)

Command	Description
`rvm list rubies`	
`rvm list default`	
`rvm install 2.1.1`	
`rvm install 2.5.0`	
`rvm --default use 2.5.0`	
`rvm use 2.1.1`	
`rvm list default`	
`rvm use system`	
`rvm use 2.5.0`	
`rvm uninstall 2.1.1`	

Point to ponder: Where does RVM get the different versions of Ruby ('rubies') from?

▸ Exercise E6

When you install **Ruby**, you get a handful of important **command line tools** by default (shown in the figure below). Match the descriptions to each of the tools,



i. The interpreter
ii. A templating system
iii. Ruby make, a task-management utility
iv. The interactive Ruby interpreter
v. A Ruby library and application package-management utility
vi. Ruby documentation tools
vii. A high-level tool for use with the Ruby test framework

▸ Exercise E7

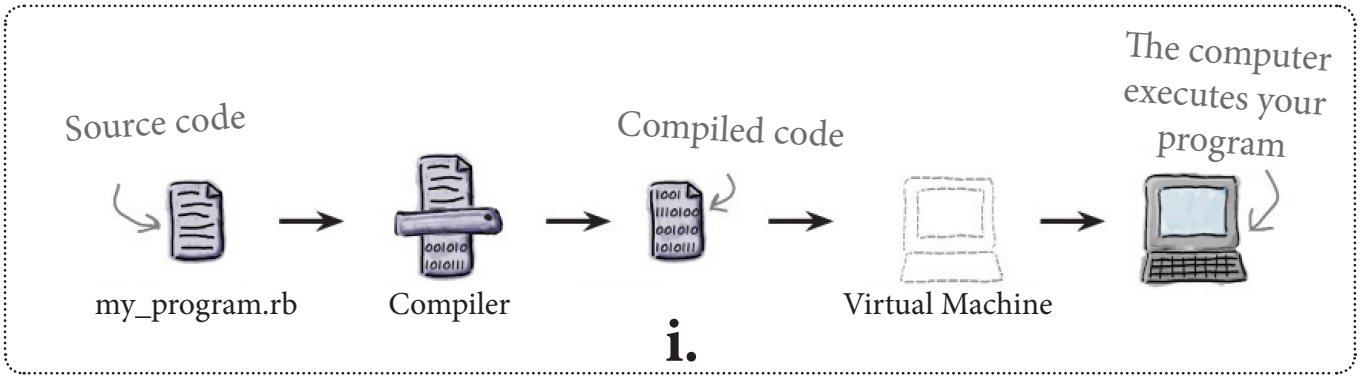
Having **Ruby** installed on your system means having several disk directories worth of Ruby *language libraries* and *support files*. In this exercise we will explore the anatomy of a Ruby installation. Find and document the location of the following folders (which were automatically created during your installation) for the default version of Ruby on your system,

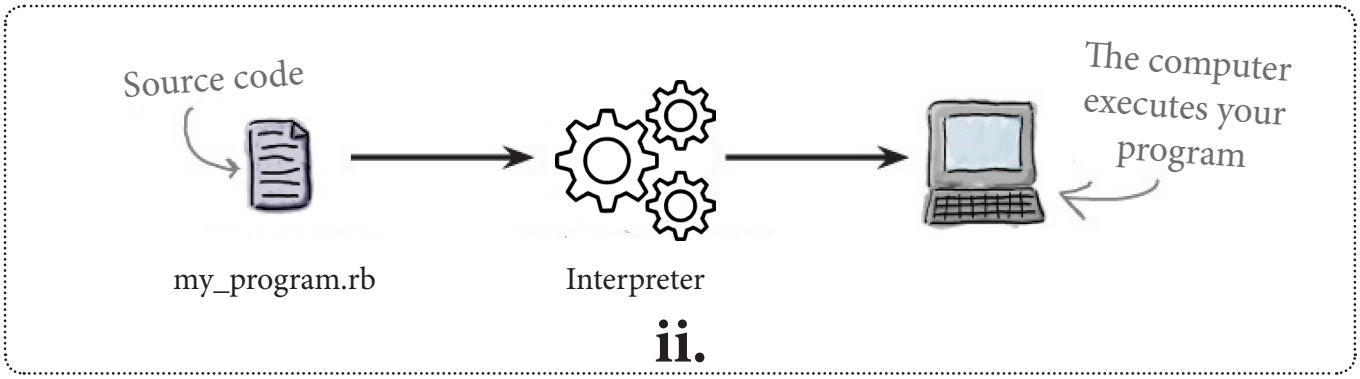
Folder	Location	Description
`bin/`		Ruby command line tools
`lib/ruby/<version>/`		Ruby standard library (rb files)
`gems/`		When you install gems, the unbundled library files land here
`site_ruby/`		Third-party extensions & libraries
`vendor_ruby/`		Some third-party extensions install themselves here (only fot Ruby versions 1.9+)

Running Ruby

▸ Exercise E8

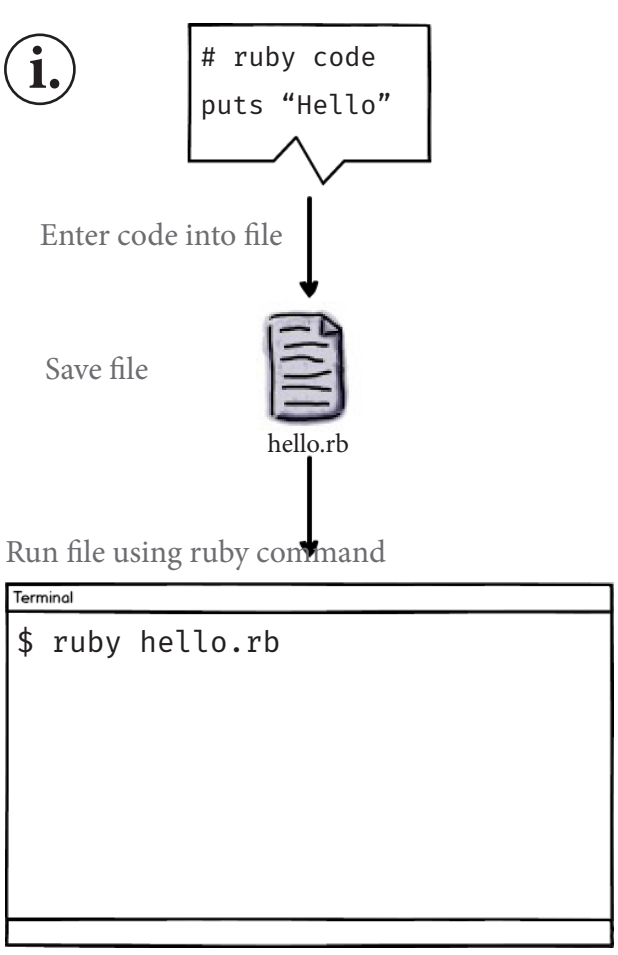
Which of the following figures (i. or ii.) best represents how **Ruby** programs are run on a computer,

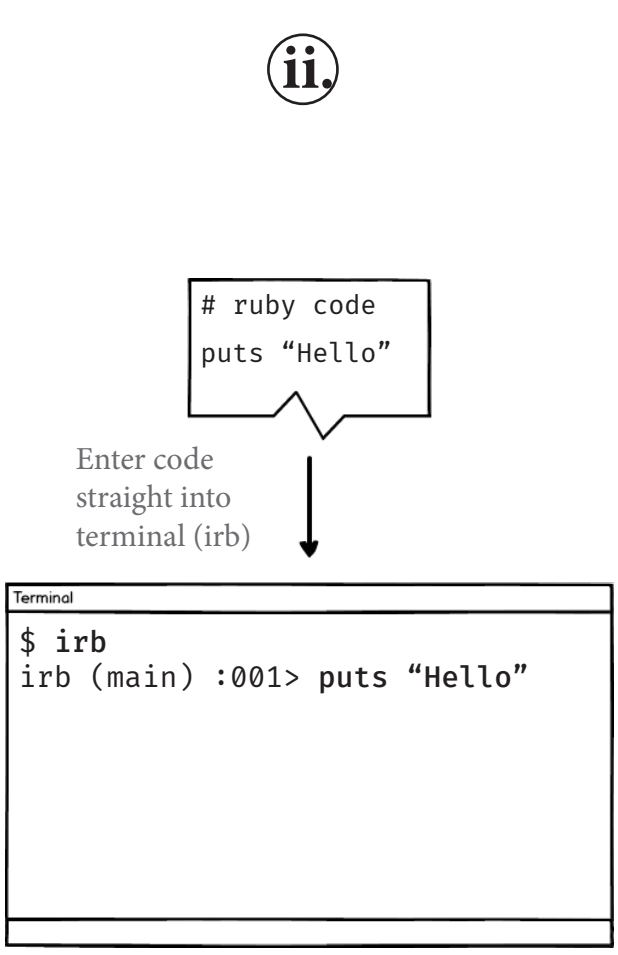




▸ Exercise E9

Now to actually running **Ruby** code *yay*. Which method below (i, ii) is the correct way to run Ruby code,





► Exercise E10

When you run the command **`irb`** in the terminal, what does the resultant **`irb`** prompt look like,

► Exercise E11

Run the command **`irb --simple-prompt`** in the terminal. What does the resultant **`irb`** prompt look like,

► Exercise E12

Open a new terminal window. Start **`irb`**. For each Ruby expression below, first write your guess for what the result will be (on the line below it), then try typing the expression into **`irb`**. See if you guess matches what **`irb`** returns,

```
49 / 7
```

_____

```
4.4 - 1.2
```

_____

```
2 * 7
```

_____

```
7 / 3.5
```

_____

```
7 / 2
```

_____

```
7 / 1000
```

_____

```
10 + 20 * 10
```

_____

```
100 - 5 * (2 - 1)
```

_____

```
3 > 5
```

_____

```
4 < 9
```

_____

```
5 == 5
```

_____

```
3 + 2 == 5
```

_____

```
2.0 + 2
```

_____

```
3 ** 2
```

_____

```
______ * ______
```

Type in an expression to compute the number of hours in a calendar year

_____

```
42.even?
```

_____

▸ Exercise E13

Open a new terminal window. Start **irb**. For each Ruby expression below, first write your guess for what the result will be (on the line below it), then try typing the expression into **irb**. See if you guess matches what **irb** returns,

```
"hello"
```

_____

```
'Darth'
```

_____

```
Darth
```

_____

```
"a" + "b" + "c"
```

_____

```
"xyz" - "y"
```

_____

```
w + z
```

_____

```
"Hello" * 3
```

_____

```
"Hello".upcase
```

_____

```
"Hello".downcase
```

_____

```
"CoolCool" / 2
```

_____

```
"Darth " * 2 + " Vader" * 3.0
```

_____

```
puts "Hello World"
```

_____

```
puts "Hello\nWorld"
```

_____

```
"nil"
```

_____

```
nil
```

_____

```
"Olympics".reverse
```

_____

```
"Olympics".upcase.reverse
```

_____

```
"Olympics".class
```

_____

Why does this work? Is there anything special about 'nil' ?

Tip: You can use the up and down arrow keys on your keyboard to navigate through the history of commands you've typed into irb. For example, use the up arrow to re-run the expression that prints 'Hello' in uppercase letters.

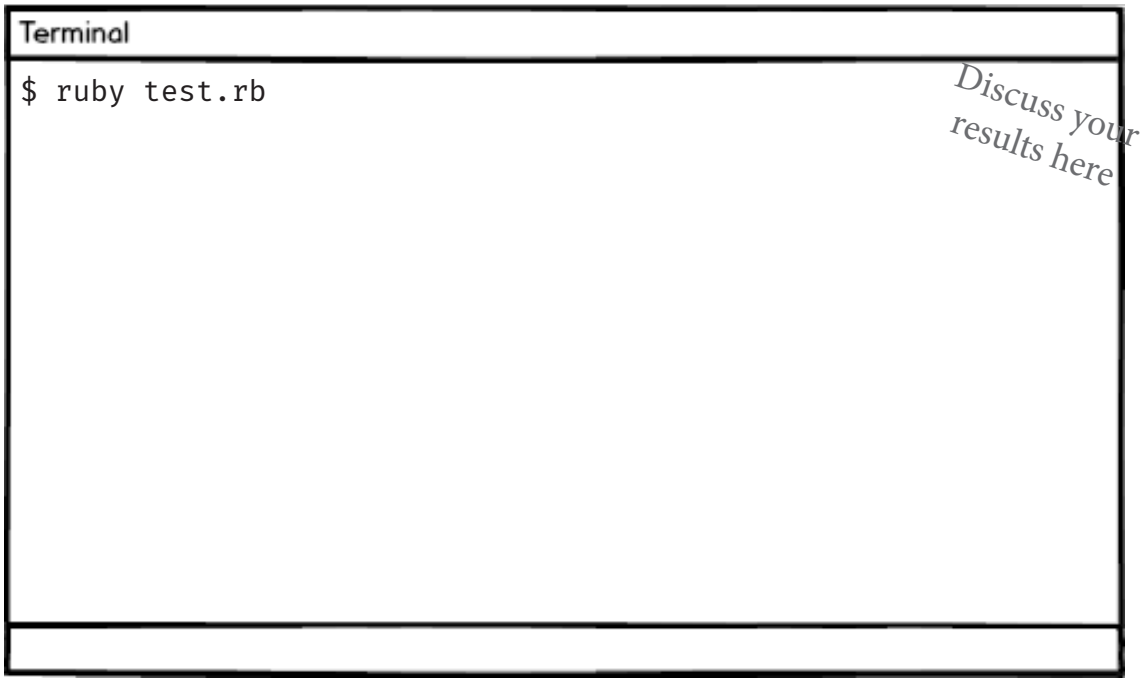
▸ Exercise E14

Save the statements below into a file called **`test.rb`**

```
"hello"
'Darth'
"a" + "b" + "c"
"Hello" * 3
"Darth " * 2 + " Vader" * 3.0
puts "Hello World"
"Olympics".reverse
"Olympics".upcase.reverse
"Olympics".class
```

`test.rb`

Start a new terminal and use the **`ruby`** command to run `test.rb`. Discuss what the results are,



How would you update the code in `test.rb` to display (print out) all the results (similar to IRB).

Update the code above

▸ Exercise E15

Apart from running code you will be refering to **Ruby documentation** alot. Mostly you will just use your web browser to look at documentation online. But Ruby ships with the `ri` command which provides easy access to help documentation. There's a catch - documentation may not get installed by default.

Your mission (should you choose to accept it) is work out out how to install Ruby documentation on your local machine and use the `ri` command to access it. For example `ri String`. Document your steps below,

Hint: If you used RVM to install Ruby you can use the command 'rvm docs generate-ri'

Language / Syntax

► Exercise E16

Find all the **Ruby *reserved*** words in the *WordSearch* below,

Every programming language has its own list of keywords which are reserved for it's own purpose (i.e. 'reserved words'). These 'reserved words' form the basis of instructions which are used to tell the computer what to do. 'Reserved words' should not be used as variable, method, class, or module names.

R	R	U	B	Y	F	A	P	E	S	S	E	L	N	U	R
O	E	G	_	D	R	I	S	B	A	E	L	U	D	O	M
M	P	L	_	N	M	A	S	E	E	S	L	E	D	N	A
R	U	I	G	E	C	T	S	L	R	G	B	E	S	T	F
R	S	N	N	F	E	T	X	R	E	E	I	I	E	N	R
E	D	D	I	N	O	L	T	E	E	G	D	N	D	E	M
S	E	R	D	N	I	R	I	R	N	N	K	O	T	_	A
C	F	J	O	P	F	G	U	H	E	G	J	R	_	L	I
U	I	V	C	X	X	S	E	H	W	D	Y	E	I	N	F
E	N	K	N	U	N	E	T	B	L	C	L	A	S	S	W
V	E	A	E	E	N	N	F	E	L	I	S	W	P	H	M
U	D	E	_	P	Z	D	I	L	F	M	R	D	E	F	Z
N	?	R	_	D	K	Y	E	_	E	T	N	N	E	O	R
T	H	B	O	N	I	K	_	F	K	S	N	H	R	F	L
I	J	P	K	H	F	T	R	U	E	F	A	L	S	E	M
L	M	N	R	U	T	E	R	_	_	L	I	N	E	_	_

Do,

- How many 'reserved words' are there in Ruby? _____
- List all the 'reserved words' in this box first. Then find them in the WordSearch above.

_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____

- Once you have found all the 'reserved words' in the WordSearch, what is the secret message ? _____

► Exercise E17

Do: Match the **Ruby** *reserved* words you found in the previous exercise to their correct descriptions in the adjacent table.

Reserved Words	Description
	Logical or Boolean true, instance of TrueClass.
	Logical operator; same as || except it has lower precedence.
	Calls method of the same name in the superclass. The superclass is the parent of this class.
	Current object (invoked by a method).
	Terminates a while or until loop or a method inside a block. Leaves a block early.
	Defines a class; closes with end.
	A special operator that determines if a variable, method, super method, or block exists.
	Begins a block and executes code in that block; closes with end.
	Logical or Boolean false, instance of FalseClass.
	Always executes at block termination; use after last rescue.
	Number of current line in the current source file.
	Ends a code block (group of statements) starting with begin, def, do, if, etc.
	Defines a method; closes with end.
	Code, enclosed in { and }, to run before the program runs; Runs before any other code in the current file.
	Executes code block if conditional statement is true. Closes with end.
	Repeats a method call outside of rescue; jumps to top of block (begin) if inside rescue. Retries an exception block.
	Starts a clause (one or more) under case.
	The script encoding of the current file.
	Executes the block passed to the method. Starts execution of the block sent to the current method.
	Begins a for loop; used with in.
	Begins a code block or group of statements; closes with end.
	Returns a value from a method or block. May be omitted; Exits a method.
	Jumps before a loop's conditional; Skips the rest of the block.
	A continuation for if, unless, and when. May be omitted.
	Name of current source file.
	Empty, uninitialized variable, or invalid, but not the same as zero; object of NilClass.
	Jumps after a loop's conditional; Restarts execution in the current block.
	Compares an expression with a matching when clause; closes with end.
	Logical operator; same as !. Inverts the following boolean expression. Has a lower precedence than !
	Creates an alias for an existing method, operator, or global variable.
	Makes a method in current class undefined.
	Defines a module; closes with end.
	Evaluates an expression after an exception is raised; used before ensure; Starts an exception section of code in a begin block.
	Used with for loop.
	Code, enclosed in { and }, to run when the program ends; Runs after any other code in the current file.
	Executes following code if previous conditional, in if, elsif, unless, or when, is not true.
	Executes code while the conditional statement is true. Creates a loop that executes while the condition is true.
	Logical operator; same as && except and has lower precedence.
	Executes following code if previous conditional, in if or elsif, is not true.
	Executes code block if conditional statement is false. (Compare with if, until.)
	Executes code block while conditional statement is false. (Compare with if, unless.)

► Exercise E18

Ruby supports a rich set of ***operators*** (including *arithmetic*, *comparison*, *assignment*, *logical*, etc). The following Table lists all of Ruby's operators in descending order of precedence. Fill in the blanks,

Operator	Description
`::`	Scope resolution
`[] []=`	Reference, set
_____	Raise to power (exponentiation)
`+ - ! ~`	Positive (unary), negative (unary), logical negation, complement
_____, _____, `%`	Multiplication, division, modulo (remainder)
_____, _____	Addition, subtraction
`<< >>`	Shift left, shift right
_____	Bitwise *and*
`| ^`	Bitwise *or*, bitwise exclusive *or*
____, ____, ____, ____	Greater than, greater than or equal to, less than, less than or equal to
`<=>`, _____, `===`, _____, `=~`, `!~`	Equality comparison (spaceship), equality, equality, not equal to, match, not match
_____	Logical *and*
_____	Logical *or*
`..` `...`	Range inclusive, range exclusive
`= += -= *= /= %=` `**= <<= >>= &= |=` `^= &&= ||=`	Assignment, abbreviated assignment
`not`	Logical negation
`and or`	Logical composition
`defined?`	Special operator (no precedence)

▸ Exercise E19

Comments are text ignored by the *interpreter*, but are vital in good software documentation. Which of the following are valid (☑) comments in Ruby,

Valid?

```
# I am a comment. Just ignore me
```
☒

```
// I am a comment
```
☒

```
puts “Hello world”  # A comment
```
☒

```
# This is a comment
# This is a comment, too
# This is a comment, too
# I said that already
```
☒

```
#
This is a comment
#
```
☐

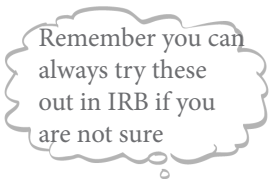
```
/*
This is a comment
*/
```
☐

```
=begin
This is a comment
This is a comment, too
This is a comment, too
I said that already
puts “Comments”
=end
```
☐

```
=BEGIN
This is a comment
This is a comment, too
=END
```
☐

```
puts “Hello” # A comment # “world”
```
☒

```
     =begin
This is a comment
This is a comment, too
This is a comment, too
I said that already
puts “Comments”
     =end
```
☐



Data Types, Variables ...

► Exercise E20

Identify which of the ***variable*** names below is **Valid** or **Invalid**

If it is invalid provide a reason why?

Variable Name	Valid or Invalid
`x`	
`y2`	
`_x`	
`7x`	
`this_is_a_test`	
`this is a test`	
`this'is@'test!`	
`this-is-a-test`	

► Exercise E21

Assign the variable `x` to the value `7`

► Exercise E22

What is the datatype of `"Stars"`?

► Exercise E23

`x = 9`. Is `x` a string?

► Exercise E24

What does the expression `"HIP" + "HOP"` evaluate to?

► Exercise E25

What does the expression `"Choco".+("late")` evaluate to?

▸ Exercise E26

What does the expression `"Choco" + 15` evaluate to?

▸ Exercise E27

```
a = "Ice"
b = "Cream"
a + b
```

What does this expression evaluate to?

▸ Exercise E28

```
flavour = "Vanilla"
flavour = "Chocolate"
puts flavour
```

What does this expression print?

▸ Exercise E29

Get the first letter from the string `"Earth"`

```
planet = "Earth"
```

```
planet = "Earth".char.first
```

▸ Exercise E30

Get the last letter from the string `"Earth"`

```
planet = "Earth"
```

```
planet = "Earth".char.last
```

▸ Exercise E31

Replace the letter `"P"` in the following string with letter `"P"`

```
word = 'Cure'
```

```
word = 'Cure'.gsub('C','P')
```

▸ Exercise E32

Assign the variables `my_name`, `my_full_name`, `my_legal_name` to the value , `"bart"`

Do this in 1 line

```
my_name = "bart", my_full_name = m_name, my_legal_name = my_full_name
```

▸ Exercise E33

What does the following expression evaluate to?

```
'Strawberry' = fav_flavour
```

```
Error
```

▸ Exercise E34
What does the following expression print?

```
something = “math”
subject = something
puts subject
```

"math"

▸ Exercise E35
What does the following evaluate to?

```
“I am whispering”.upcase()
```

I AM WHISPERING

▸ Exercise E36
Convert every letter of “Coder AcademY” to lower case

```
school = “Coder AcademY”
```

school = "Coder AcademY".downcase

▸ Exercise E37
Join the string

```
first = “Chicken ”
second = “or “
third = “the egg”
```

first + second + third

▸ Exercise E38
Convert the number `7` to the string `“7”`

number.to_s

▸ Exercise E39
What is the problem with the following code,

```
my variable = “Cool”
```

There should not be an space between 'my' and 'variable'

▸ Exercise E40
Update the following code so that it runs successfully,

```
service = “Cloud ” + 9
```

There could be 2 answers to this. Try to find both

Ans # 1:
```
service = "Cloud " + 9.to_s
puts service
```

Ans # 2:
```
service = "Cloud"
service = service + "9"
puts service
```

► Exercise E41

Open a new terminal window. Start **irb**. For each Ruby expression below, first write your guess for what the result will be (on the line below it), then try typing the expression into **irb**. See if you guess matches what **irb** returns,

Work sequentially down the column

```
name = “Bart”
```

"Bart"

_____

```
name.upcase
```

"BART"

_____

```
“Bart”.upcase
```

"BART"

_____

```
name.upcase.reverse
```

"TRAB"

_____

```
puts name
```

"TRAB"

_____

```
name.upcase!
```

BART

_____

```
puts name
```

Bart

_____

```
name.class
```

"string"

_____

```
name * 3
```

Bart Bart Bart

_____

```
x = 10
y = 20
puts “#{x} + #{y} = #{x + y}”
```

10 + 20 = 30

_____

```
x = 2
y = 3
z = 4
puts x + y + z
```

9

_____

```
x = 2
y = 3
z = x = y
puts x + y + z
```

8

_____

```
month = 12;
year = 2017;
puts month.to_s + “-” + year.
to_s;
```

ERROR

_____

```
p, q, r = 100, 200, 300
puts p, q, r
```

100
200
300

_____

```
a1, b1, c1 = “cash”, 1.99. 100
puts a1, b2, c1
```

cash
1.99.
100

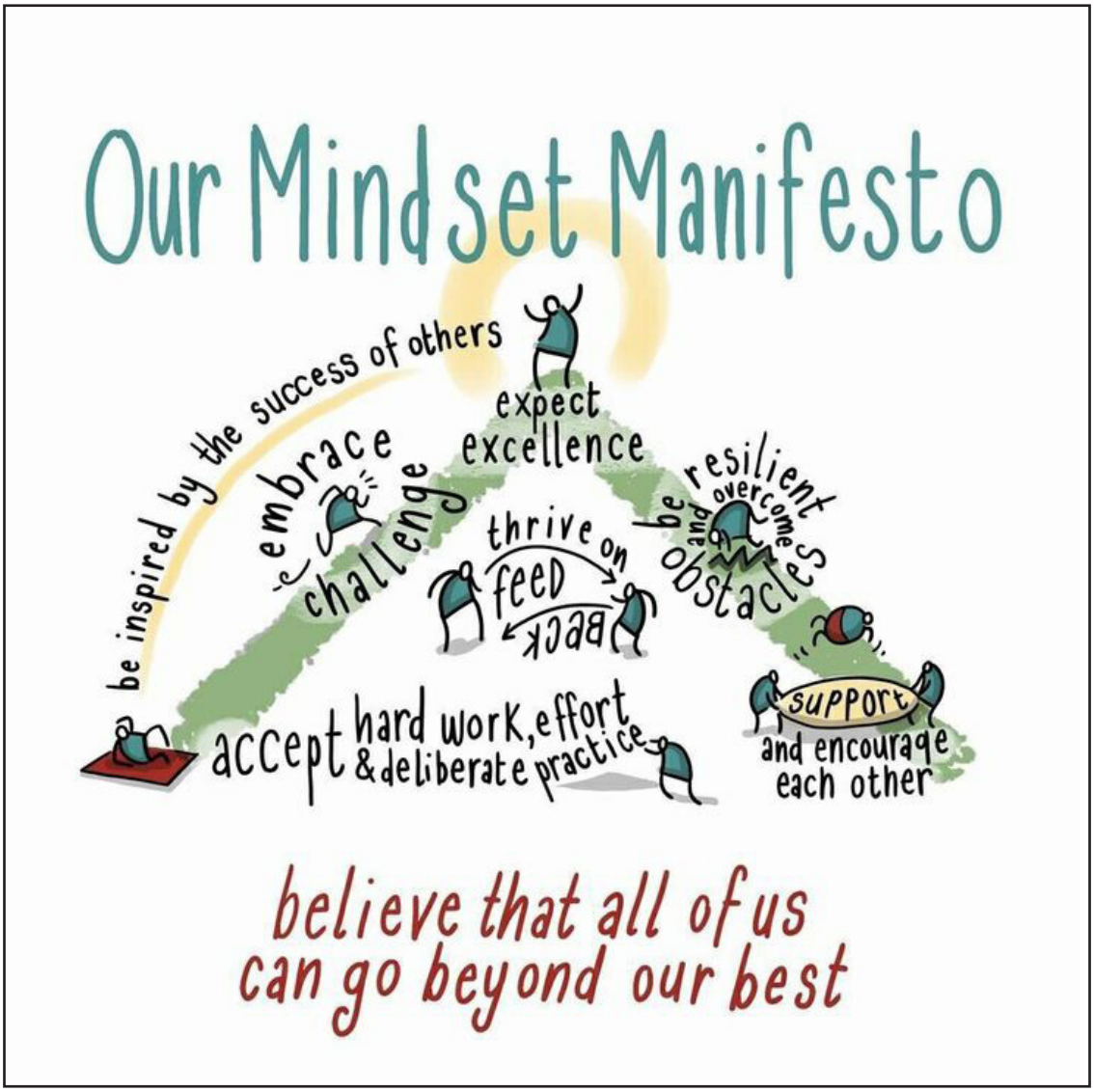
_____

```
i = 9
i += 1
j = 2
j -= i
```

-8

_____


Growth mindset vs fixed mindset
SUCCESS
FRUSTRATION
1. I can learn anything I want to.
2. When I'm frustrated, I persevere.
3. I like to challenge myself.
4. When I fail, I learn.
5. I like being told that I try hard.
6. If my classmates succeed, I'm inspired.
7. My effort and attitude determine everything.
1. I'm either good at it, or I'm not.
2. When I'm frustrated, I give up.
3. I don't like to be challenged.
4. When I fail, I'm no good.
5. I like being told that I'm smart.
6. If my classmates succeed, I feel threatened.
7. My abilities determine everything.
@sylviaduckworth
From Reid Wilson @wayfarepath
GRIT
Learned helplessness


Our Mindset Manifesto
be inspired by the success of others
expect excellence
embrace challenge
thrive on feedback
be resilient and overcomes obstacles
accept hard work, effort & deliberate practice
support and encourage each other
believe that all of us can go beyond our best

▸ Exercise E42

Ruby has several *datatypes*. Using the results of the following code snippet as a guide list the *datatypes* in Ruby (in the space provided below),

```
h = { :name => “Jane”, :age => 17 }

p true.class, false.class
p “Ruby”.class
p 1.class
p 4.5.class
p 3_463_456_457.class
p :age.class
p [1, 2, 3].class
p h.class
```

Ruby *datatypes*,

hash
Boolean
Integer
String
Float
Symbol
Array

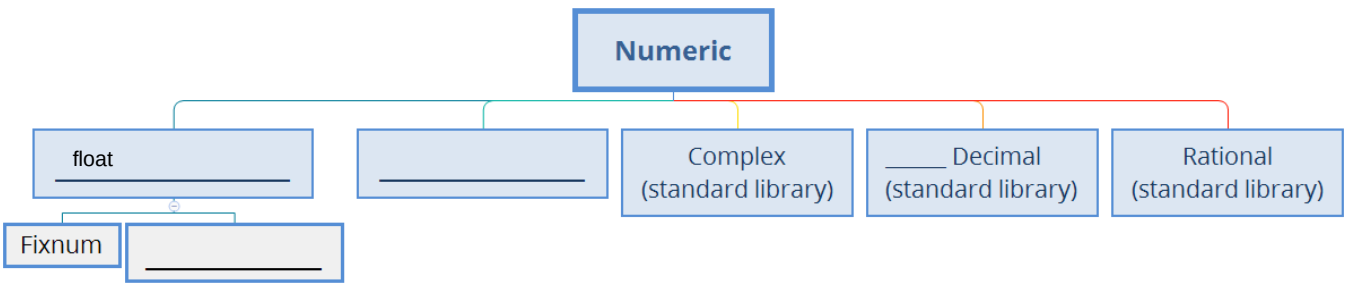
Not all of Ruby's *datatypes* were covered in code snippet above. Can you find any others?

ENV

If you haven't covered 'classes' yet don't worry it will make sense soon. Try and fill in the blanks with your best guess of datatypes

▸ Exercise E43

Ruby includes five built-in *classes* for representing numbers, and the standard library includes three more numeric *classes* that are sometimes useful. The following figure shows the *class* hierarchy. Fill in the blanks,



▸ Exercise E44

There are different ways to write numbers in Ruby. Guess the output of the following,

```
puts 122
puts 0x7a
puts 0172
puts 0b1111010
```

Output

Does this result look strange to you? If so, have a think about why?

Fill in the following blanks,

_____ numbers are preceded with the `0x` characters

_____ numbers are preceded with the `0` character

_____ numbers are preceded with the `0b` characters

▸ Exercise E45

What is the output of the following code,

```
baskets = 16
apples_in_basket = 24
total = baskets * apples_in_basket
puts “There are total of #{total} apples”
```

Do you need double quotes “ in this example or could you replace these with single quotes ‘

Output,

▸ Exercise E46

Guess the output of the following code,

```
p 23482345629
p 23_482_345_629

p 23482345629 == 23_482_345_629
```

Output,

~~Now run it. Did your guess~~ match?

▸ Exercise E47

Open a new terminal window. Start **`irb`**. Calculate the following,

Document both your commands/ code & results

* How many *hours* in a *year*?

* How many *minutes* in a *decade*?

* How many *seconds* old are *you*?

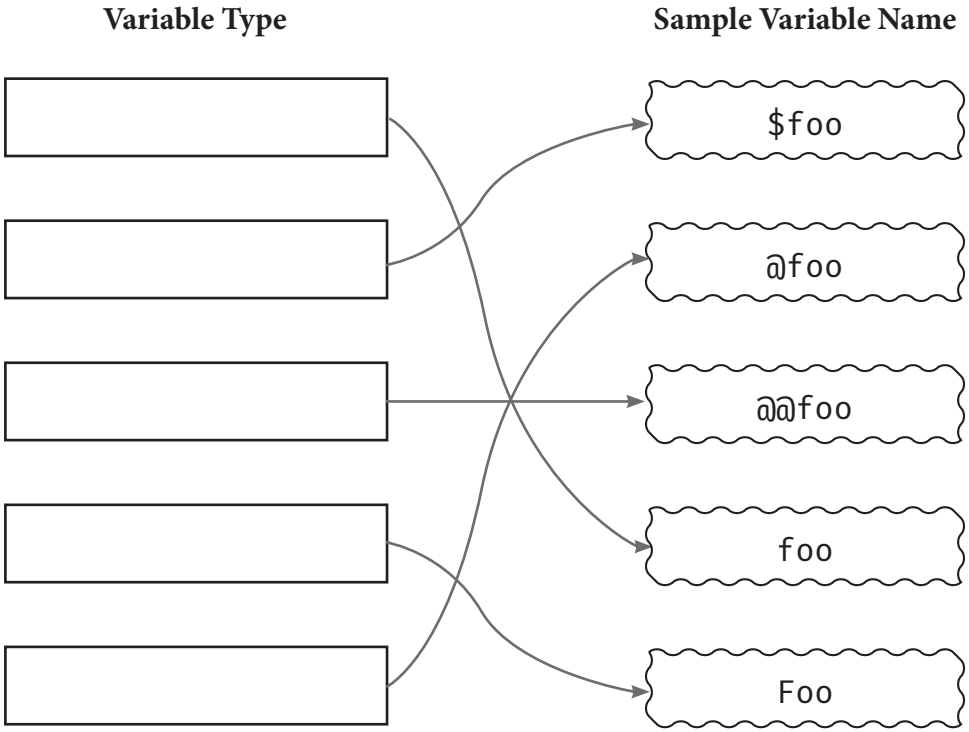
► Exercise E48

Try computing these in **`irb`**. Is the result a *float* or an *integer*? Tick (☑) the appropriate box,

	Integer	Float
`3.0 / 2`	☐	☐
`3 / 2.0`	☐	☐
`4 ** 2.0`	☐	☐
`4.1 % 2`	☐	☐

► Exercise E49

There are five types of *variables* in Ruby. List them below and match the sample *variable* name with its correct type,



► Exercise E50

Open a new terminal window. Start **`irb`**. For each Ruby expression below, first write your guess for what the result will be (on the line below it), then try typing the expression into **`irb`**. See if you guess matches what **`irb`** returns,

Work sequentially down the column

```
x = 5
```

_____

```
x += 5
```

_____

```
x /= 5
```

_____

```
x, y +=2
```

_____

```
x = 20
```

_____

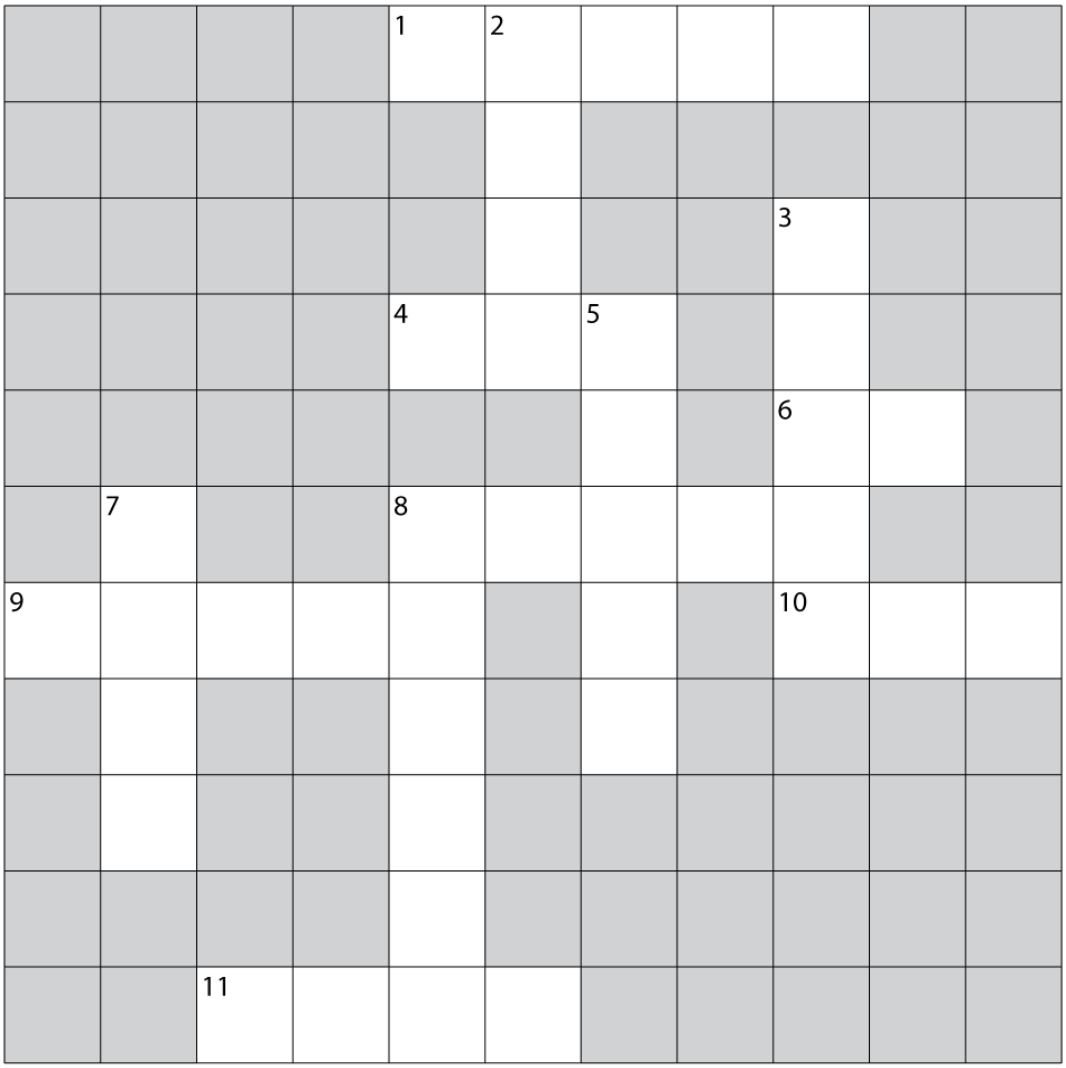
```
x /= 7
```

_____

Control Structures - Conditionals & Flow ...

► **Exercise E51**

Ruby has a number of *control structures* (conditionals & flow/loops). Identify each in the crossword below,



Across

1. Terminates the most internal loop. Terminates a method with an associated block if called within the block (with the method returning nil).
4. Executes code once for each element in expression.
6. Executes code if the conditional is true.
8. Executes code while conditional is false.
9. Declares code to be called before the program is run.
10. Declares code to be called at the end of the program (when the interpreter quits).
11. Compares the expression specified and executes the code of the when clause that matches.

Down

2. Jumps to the point immediately after the evaluation of the loop's conditional.
3. Executes code while conditional is true.
5. Repeats a call to a method with an associated block when called from outside a rescue clause.
7. Jumps to the next iteration of the most internal loop. Terminates execution of a block if called within a block (with yield or call returning nil).
8. Executes code if conditional is false.

Word Bank

BEGIN break case END for if
next redo retry unless until while

There are a few control structures missing in the crossword above including the 'begin' $ 'rescue' statements and 'raise' method. Do take a moment to find out what these do.

▸ Exercise E52

Fill in the blanks (in the comments) below to explain what is happening,

Hint - These are all either True or False

```
if condition
     # code executed if condition is ____________
else
     # code executed if condition is ____________
end
```

```
if condition1
     # code executed if condition1 is ___________
elsif condition2
     # code executed if condition1 is ___________
     # and condition2 is ___________
elsif condition3
     # code executed if neither condition1
     # nor condition2 is _______, but condition3 is
end
```

▸ Exercise E53

Conditional control structures depend on whether expressions evaluate to *true* or *false* (boolean) to decide on which code to run. Fill in the following truth table for *and*, *or* cases

A	B	A and B (A && B)	A or B (A || B)
True	True		
True	False		
False	True		
False	False		

▸ Exercise E54

For each Ruby expression below, first write your guess for what the result will be, then try typing the expression into **irb**. See if you guess matches what **irb** returns,

```
!false
!(true or false)
first = true
second = false
(first and second) or !(first and second)
```

▸ Exercise E55

One way to negate a condition is to use the `not` keyword. For example,

```
if not (x == 1)
```

Which of the following statements is equivalent to the statement above,

	Equivalent
`if !(x == 1)`	☐
`if not x == 1`	☐
`if !x == 1`	☐
`if (!x) == 1`	☐
`unless (x == 1)`	☐
`unless x == 1`	☐

▸ Exercise E56

Convert the provided `if-elsif-else` code snippet to equivalent code using the `case` statement,

```
fruit = “orange”
if fruit == “orange”
color = “orange”
elsif fruit == “apple”
color = “green”
elsif fruit == “banana”
color = “yellow”
else
color = “unknown”
end
```

As you can see this code snippet hasn't been formatted very well - which makes it quite difficult to read. Make sure to correctly format your code.

▸ Exercise E57

Explain what the following line of code does,

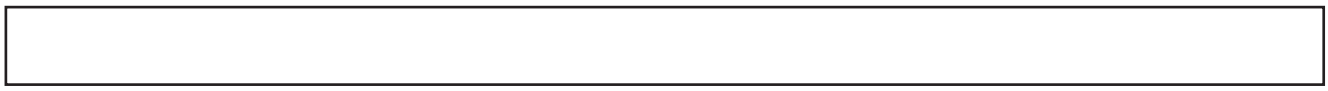
```
puts “done” && return (x > y && a < b) unless c == 0
```

▸ Exercise E58

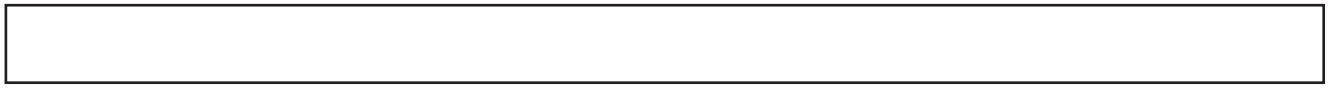
What do the following evaluate to?

```
age = 10
puts "You're too young to use this system" if age < 18
```

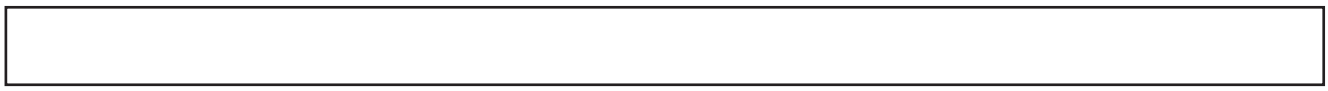
```
age = 24
puts "You're a teenager" if age > 12 && age < 20
```



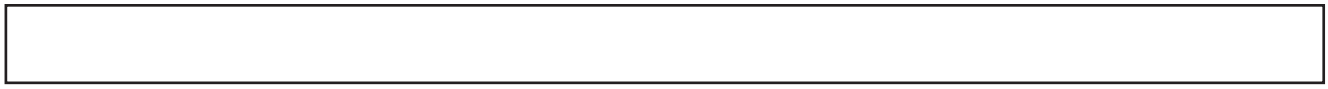
```
age = 24
puts "You're NOT a teenager" unless age > 12 && age < 20
```



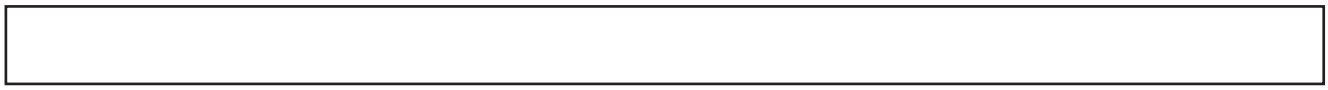
```
age = 24
puts "You're 24!" if age == 24
```



```
puts "You're either very young or old" if age > 80 || age < 10
```



```
puts "You're a working age man" if gender == "male" && (age >= 18 &&
age <= 65)
```



▸ Exercise E59

Prompt the user for an integer and test if it is *even* or *odd*. Save this program in a file called **`integer_compare.rb`**

▸ Exercise E60

Modify **`integer_compare.rb`** so that it,

- Prompts the user for an integer between 5 and 10 (inclusive) and displays whether or not the input was correct. Implement this using an if/else statement.

▸ Exercise E61

Modify **`integer_compare.rb`** so that it,

- Prompts the user for an integer between 5 and 10 and then informs the user if the integer was below the range, in the range, or above the range. Implement this using a case statement.

▸ Exercise E62

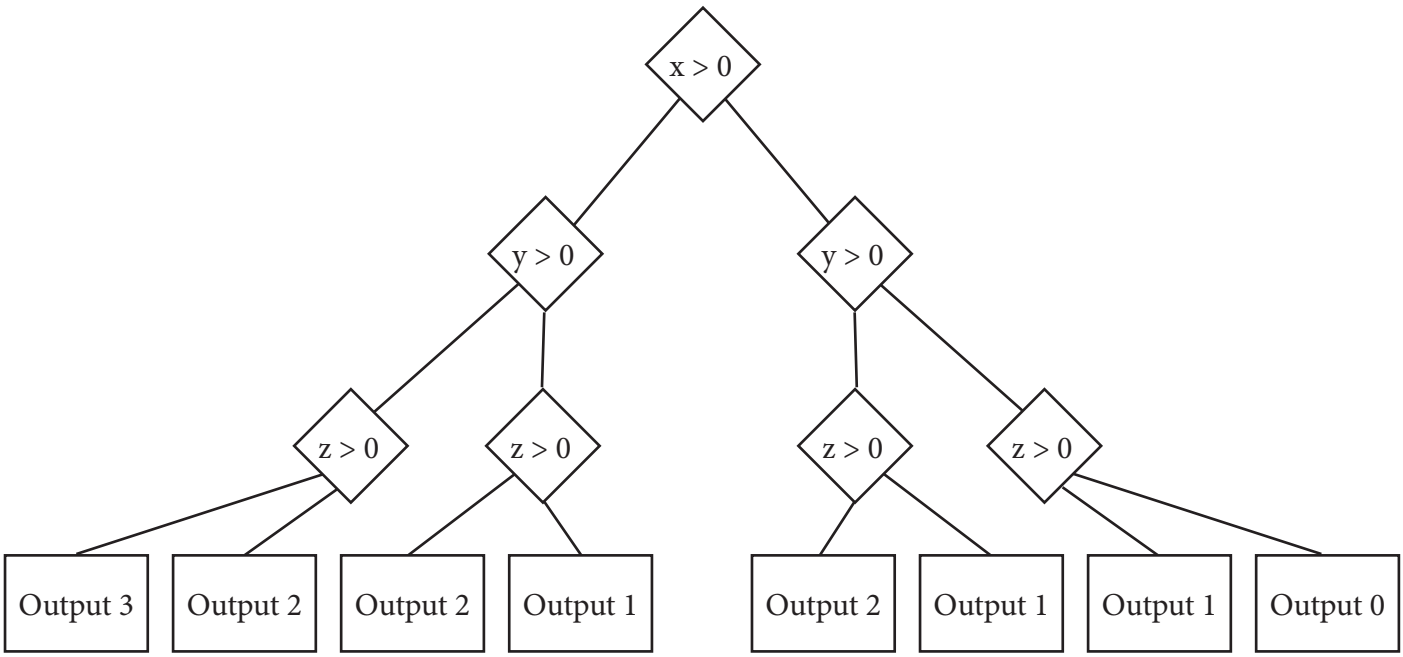
Write a program that, given two points on a two-dimensional graph, outputs a message (string) if the line that connects them is horizontal or vertical, or if the slope is positive or negative.

▸ Exercise E63

The unit pulse function, $d[\mathtt{n}]$, is very important in digital signal processing. This function is defined for integers only. It is equal to 1 when `n` is 0, and it is equal to 0 when `n` is any other integer. Write a program that prompts the user for an integer `n`, and returns the value $d[\mathtt{n}]$.

► Exercise E64

Prompt the user for integers *x*, *y* and *z*. Implement the decision tree shown in the following diagram based on the values using `if` statements. Output how many of the variables are greater than 0.
Note: each right flow option represents false, and each left flow option represents true.



► Exercise E65

Answer the following questions,

Question	Answer
What does the following expression evaluate to? `7 == 7`	
What does the following expression evaluate to? `true = 9`	
Demonstrate that "`hello`" is the same as "`hello`".	
What does the following expression evaluate to? `5 != 5`	
What does the following expression evaluate to? `7 > 6`	

What does the following expression print?
 	
`if 7 > 6`
`  puts "7 is greater than 6"`
`end`	

▸ Exercise E66

Answer the following questions,

Question	Answers

What does the following expression print?
 	
`if 7 < 6`
`  puts “7 is less than 6”`
`else`
`  puts “7 is not less than 6”`
`end`	

What does the following expression print?
 	
`if “tiger” == “cat”`
`  puts “tiger equals cat”`
`elsif “hello” == “hello”`
`  puts “hello equals hello”`
`else`
`  puts “whatever”`
`end`	

What does the following expression print?
 	
`if 7`
`  puts “Hello there”`
`end`	

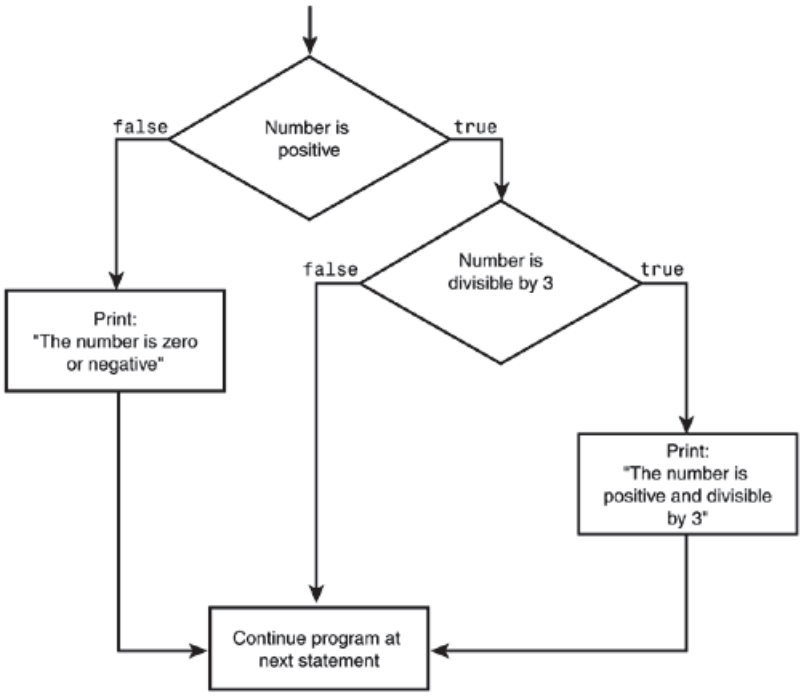
What does the following expression print?
 	
`if “cool”`
`  puts “cool is truthy”`
`else`
`  puts “cool is falsey”`
`end`	

What does the following expression print?
 	
`if nil`
`  puts “nil is truthy”`
`else`
`  puts “nil is falsey”`
`end`	

What does the following expression print? `puts “This syntax is cool” if true`	
What does the following expression print? `puts “Tall buildings” if false`	

▸ Exercise E67

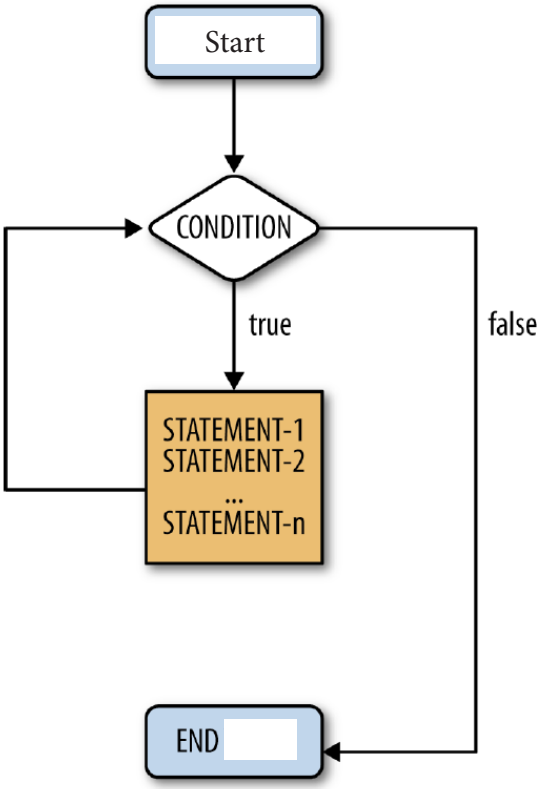
Convert the following flowchart into a ruby program.



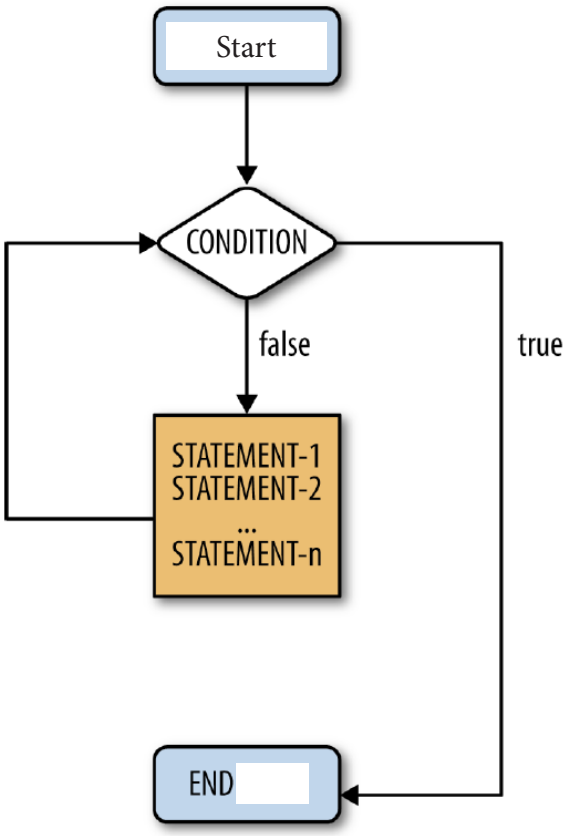
▸ Exercise E68

Identify which type of *loop* is represented by the following flowcharts,

Type of *loop* = Repeats if condition is true



Type of *loop* = Repeats if condition is false



► Exercise E69

What is the result (output) of the following code,

```
x = 1
while x < 100
  puts x
  x = x * 2
end
```

```
1
2
3
4
8
26
32
64
```

Rewrite the above code snippet using an `until` loop,

```
x = 1
until x > 100
  puts x
  x = x * 2
end
```

► Exercise E70

Answer the following questions,

Question	Answers

What does the following code print?
 	
`i = 1`
`i = i * 2 until i > 1000`
`puts i`
	2

What does the following code print?
 	
`counter = 0`
`while counter < 3`
`  puts “Went through loop”`
`  counter = counter + 1`
`end`
	Went through loop
	Went through loop
	Went through loop

What does the following code print?
 	
`while 3 > 17`
`  puts “This is not the end”`
`end`
	Nothing prints

What does the following code print?
 	
`while true`
`  puts “This is the song that never ends”`
`end`
	It runs infinite loop and prints the message This is the song that never ends

► Exercise E71

For each of the following, convert the given loop type (while, until) into its opposite,

Write the opposites in the respective box

`while (x == 7)`

until (x != 7)

`until (x < 7)`

while (x < 7)

`until ((x!=0) and (y>2))`

while ((x !=0) && (y > 2))

► Exercise E72

Walk through the following program & explain what it does,

```
puts “Enter a number >= 0: ”
n = gets.to_i
a = 1
while (n > 1)
  a = (n * (n - 1)) * a
  n = n - 2
end
puts a
```

Walkthrough

Line 1: prints "Enter a number >= 0:" message on screen
Line 2: gets user input, converts into integer, and assigns it into 'n' variable
Line 3: assigns integer value '1' to a variable 'a'
Line 4: starts a while loop with a condition if value of variable 'n' is greater than '1'
Line 5: calculates the value in the given formula and assigns the result into variable 'a'
Line 6: subtracts integer value 2 from variable 'n' and assigns the result into 'n'
Line 7: marks end of code
Line 8: prints the value in variable a

► Exercise E73

Write a program to calculate compounded interest using a `while` loop. The user inputs the amount deposited, the interest rate (as a percentage) per period, and the number of periods the deposit accumulates interest. Compound interest means that every period, your new balance is calculated using the last period's balance times the interest rate.

► Exercise E74

Implement the mod operator without using the mod operator but using a loop. (Assume the numerator is always greater than the denominator and both are greater than 0.)

► Exercise E75

Make a simple calculator. It should read in two numbers, apply an operator (`+`, `-` , `*` , `/`), and display the result. It should continue to do this until a condition of your choosing stops it.

► Exercise E76

Write a program that outputs the first 20 numbers in the Fibonacci sequence. In the Fibonacci sequence, the current number is the sum of the previous two numbers. The first two numbers in the sequence are 1 and 1.

► Exercise E77

Check if the following *code* works & explain what it does,

Walkthrough

```
loop do
  print “Type something: ”
  line = gets
  break if line =~ /q|Q/
  puts line
end
```

► Exercise E78

Are the following *loops* identical in their output (results),

```
for i in 1..5 do
  print i, “ ”
end
```

X

```
for i in 1..5
  print i, “ ”
end
```

X

```
for i in 1..5 do print i, “ ”
```

X

► Exercise E79

What does the following code print,

```
10.times { |i| print i, “ ” }
```

0 1 2 3 4 5 6 7 8 9 => 10

► Exercise E80

What does the following code print,

```
1.upto(10) { |i| print i, “ ” }
```

1 2 3 4 5 6 7 8 9 10 => 1

► Exercise E81

Write a **`for`** and **`while`** loop that does what the loops in the previous two exercises are doing,

for loop

```
for i in1..10 do
 p i
end
```

while loop

```
i = 0
while i < 10
  p i
  i += 1
end
```

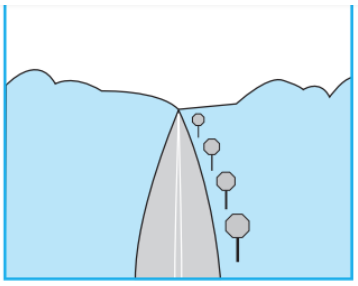
► Exercise E81

What does the following code print,

```
5.downto(1) { |i| print i, “ ” }
```

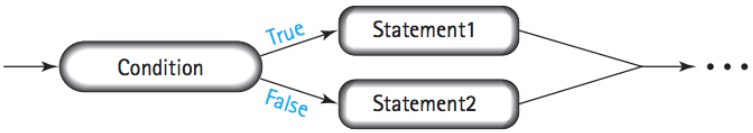
5 4 3 2 1

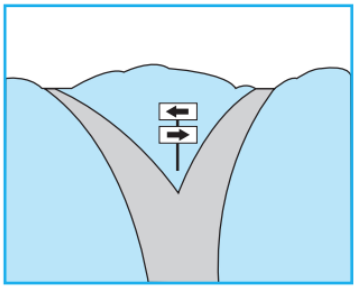
SEQUENCE



SELECTION (also called *branch* or *decision*)

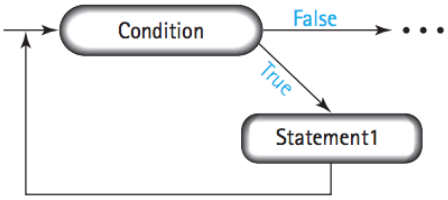
IF condition THEN statement1 ELSE statement2

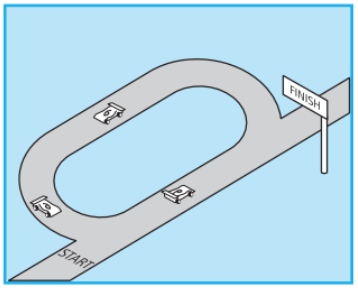




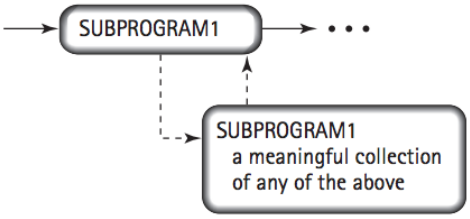
LOOP (also called *repetition* or *iteration*)

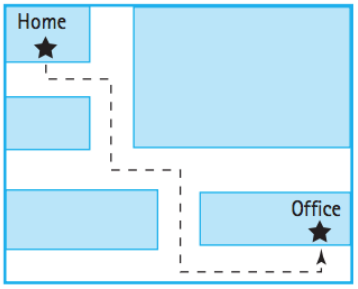
WHILE condition DO statement1

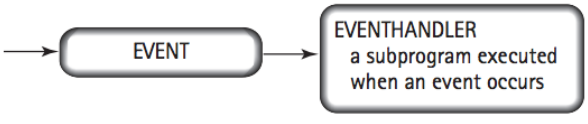


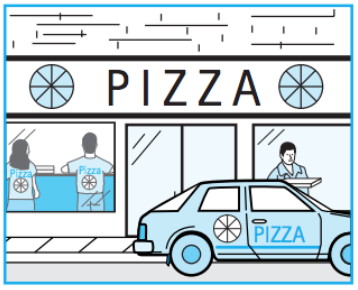


SUBPROGRAM (also called *procedure*, *function*, *method*, or *subroutine*)





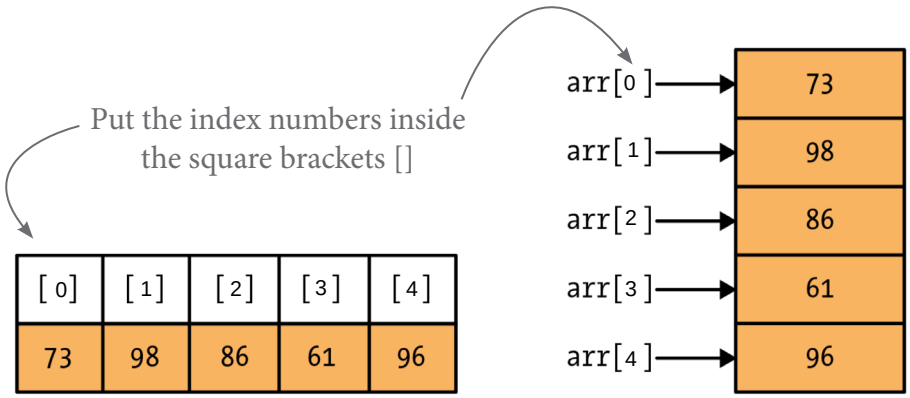




Arrays

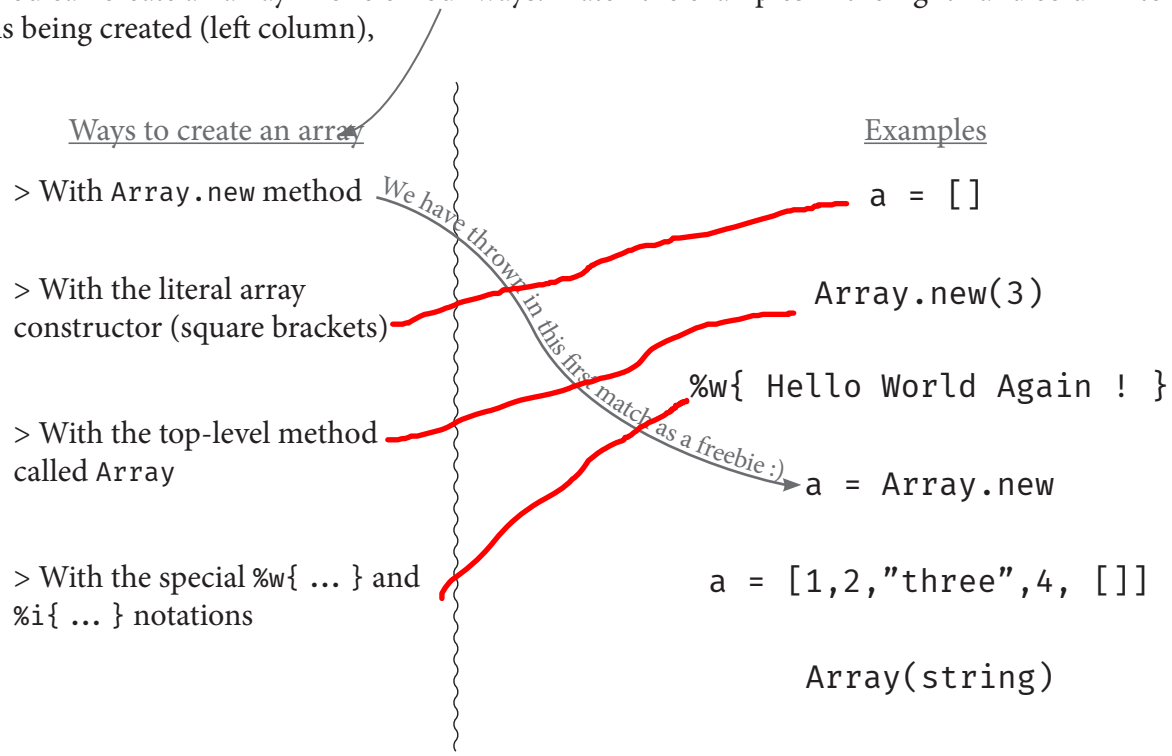
▸ Exercise E82

An array is an ordered list of variables. To get the idea, imagine a row of compartments, each of the compartments can contain something or be empty, and they are numbered sequentially. In the array below fill in the index number of each array compartment (i.e. element),



▸ Exercise E83

You can create an array in one of four ways. Match the examples in the right hand column to the way the array is being created (left column),



▸ Exercise E84

Explain the following code,

```
[1,2,3][1]
```

one array contains three elements, and an another array contains only one element

▸ Exercise E85

Write code to create a multidimensional array (called `arr`)from the following table,

	[0]	[1]	[2]	[3]	[4]
[0]	73	98	86	61	96
[1]	60	90	96	92	77
[2]	44	50	99	65	10

```
arr = [0,1,2,3,4, [0,1,2]]
```

▸ Exercise E86

Summary of common array query methods are summarized in the following table. Complete the 'Meaning' column explaining what each method does,

Method name/Sample call	Meaning
`a.size (synonym: length)`	Number of elements in the array
`a.empty?`	True if a is an empty array; false if it has any elements
`a.include?(item)`	checks for any particular element in an array
`a.count(item)`	total numbers of elements in an array
`a.first(n=1)`	checks the first element of an array
`a.last(n=1)`	Last *n* elements of array
`a.sample(n=1)`	

▸ Exercise E87

Using the array `arr` with value `a[0] = 9`, `a[1] = 2`, `a[2] = 5`, `a[3] = 4`, `a[4] = 3`, determine the output of the following code,

```
i = 0

while (i < a.size)
  puts a[i]
  i = i + 1
end
```

```
9
2
5
4
3
```

▸ Exercise E88

The following code looks for the first two elements that are out of order and swaps them; however, it is not producing the correct results. Fix the code so that it works correctly.

```
arr = [5, 22, 29, 39, 19, 51, 78, 96, 84]
i = 0
while (i < arr.size - 1 and arr[i] < arr[i + 1])
  i = i + 1
end
puts i

arr[i] = arr[i + 1]
arr[i + 1] = arr[i]
```

```
?
```

▸ Exercise E89

Create an array that contains the values "`cool`", `100`, `3.1`

```
arr ["cool",  100, 3.1]
```

▸ Exercise E90

Answer the following questions,

Question	Answers
Create an array with nothing inside it.	arr[]
What does the following expression return: `["uno", "dos", "tres"].length()`	3

`name = "clem"`
`age = 32`
Is this array valid?
`[name, age]`
	Yes it is valid

Get the last element of the lyric Array (hint: there are two answers) `lyric = ["laughter", "best", "free"]`	p lyric[-1] p lyric.last
`arr = ["stop", "up", "down"]` Add the string "`go go go`" to the end of the `arr` Array.	arr.push "go go go"
What does the following code return? `["roof", "top"].include?("bar")`	false
What does the following code return? `["One", "Two", "Three", "Four,", "Five", "Six", "Seven", "Eight", "Nine"].join(" ")`	"One" "Two" "Three" "Four" "Five" "Six" "Seven" "Eight" "Nine"
What does the following code return? `[2, 4, 6, 8].map { |number| number ** 2 }`	square roots the values in the array
What does the following code do? `a =`	nil

I have a **GROWTH MINDSET.**

I am in charge of how smart I am because I can **GROW** my **BRAIN** like a muscle by learning hard things.

I can achieve **ANYTHING** with **EFFORT** and **RIGHT STRATEGIES.**

And when I fail or make a mistake, it is a **GREAT** thing, because I can **LEARN** from them and **I GET BETTER!**