Getting started with LaTeX

A laboratory manual edited by Duncan Hull

Last updated on 19 October, 2022

Contents

Welcome

Hello and welcome to the LaTeX lab manual, part of COMP101 at the University of Manchester.

Figure 1: The LaTeX project logo by Jonas Jacek CC BY 4.0 via Wikimedia Commons w.wiki/3Daw

Reading this LaTeX manual and doing the five exercises it contains will enable you to develop your written communication skills so that you can:

1. Create a simple document in pdf using LaTeX
2. Illustrate a document with figures and cross references
3. Typeset some mathematics
4. Share and collaborate on LaTeX documents using overleaf
5. Draft a CV using LaTeX templates

Acknowledgements

This manual is a substantially revised version of earlier LaTeX lab manuals created by Ulrike Sattler, Graham Gough, Paul Waring, Toby Howard and Steve Pettifer.

Improve this manual

The source of this manual is available on github so if you have any comments or suggestions on how to improve it, you can raise an issue or submit a pull request. We'll credit every contribution, however small, because they all make a difference.

Thanks to contributions from Hamza Latif (@ultrasockhead).

If you want to make suggestions for improvements that **don't** get attributed to your name, email me directly.

Chapter 1

What is LaTeX?

LaTeX is a document preparation system for high-quality typesetting. (Tex, 2020) It is part of a mature and established toolchain that has been around since 1980's. (Knuth, 1984) Originally written by Leslie Lamport, a computer scientist now working at Microsoft Research, its development has long been taken over by the open-source LaTeX community around the world. LaTeX is actually built on top of another system called TeX, a computer typesetting system designed by another influential Computer Scientist, Donald Knuth of Stanford University. Lamport and Knuth are shown in Figure 1.1.

LaTeX is typically used for technical documents but it can be used for almost any form of publishing, including writing CVs, letters, books, posters, presentations and much more. Whatever you create with LaTeX, one of its key strengths is making documents look professional in portable document format (pdf) using industrial-strength typesetting.

1.1 NOT a word processor

LaTeX is *not* a word processor! Instead, LaTeX encourages you to concentrate on the *content* of your documents, while it takes care of the details of its presentation. This is similar to the approach you've been using for creating web pages in COMP1010 where the style (in your cascading style-sheet: `*.css`) should be cleanly separated from the raw content (in your `*.html`). This is a classic abstraction technique in computing by separation of concerns (SoC). If you're reading this page in a web browser, view the source of this page `latex.html` as an example, the html only describes the content and says very little about its presentation, that is described separately in the cascading style sheet.

LaTeX is **not** a what you see is what you get (WYSIWYG) system either. The raw document you edit (a `*.tex` file input) is not your final result (usually a

Figure 1.1: Turing award winners Leslie Lamport and Donald Knuth created TeX and LaTeX during the 1980's. Lamport portrait by Leslie Lamport, GFDL via Wikimedia Commons w.wiki/3Daz, Knuth portrait by Jacob Appelbaum CC BY-SA via Wikimedia Commons w.wiki/3Day. Is Donald Knuth the The Yoda of Silicon Valley? (Roberts, 2018)

`*.pdf` file output). It doesn't even come with a spell-checker, though there are many plug-ins you can use for that.

1.2 So what is LaTeX then?

The best way to understand LaTeX is to create some documents which we'll do in the next chapter.

Chapter 2

Simple LaTeX documents

Let's start by creating a simple LaTeX document.

2.1 A very short document

Open up your favourite text editor[1], enter the following text, then save it as a file called `turing.tex`:

```
\documentclass[a4paper]{article}
\begin{document}
Computational excursions
\end{document}
```

To turn this into a pdf we need to use a LaTeX compiler, we're going to use `pdflatex`, though several other compilers are available.[2] On Linux the `pdflatex` compiler is already installed and can be used from the command line.[3] If you're not using Linux, you'll need to explore the options at latex-project.org/get.

```
# compile turing.tex
pdflatex turing.tex
# open the pdf output
xdg-open turing.pdf
# or just open turing.pdf on a mac
```

The first command outputs a pdf file from your `turing.tex` input using the

[1] it is worth configuring your editor so that it is LaTeX aware, and can do syntax highlighting, suggest auto-completions and spell-check for you. Tools like atom.io and sublimetext and many others will do this for you

[2] https://www.overleaf.com/learn/latex/Choosing_a_LaTeX_Compiler

[3] for example the Linux VM image for VirtualBox https://wiki.cs.manchester.ac.uk/index.php/CSImage_VM

`pdflatex` compiler. The second command opens the file you've created. If you list the directory contents, you'll see that the compiler has also created an auxillary `*.aux` file and a `*.log` file which can be helpful when you're debugging the compilation:

```
# files created on compilation
turing.tex
turing.pdf
turing.aux
turing.log
```

2.2 A longer LaTeX document

Our `turing.tex` file is a very simple document, so let's add some sections and fill them out a bit:

```
\documentclass[a4paper]{article}
\begin{document}


\section{Algorithms: Cooking Up Programs}
A program specifies in the \textbf{exact syntax} of some programming language the compu

\section{Finite automata: The Black Box}
It occasionally happens in industrial, military or educational settings that one is pre

\section{Systems of Logic: Boolean Bases}
In an age of computers and automation, almost every electronic device one can name inco

\section{Simulation: The Monte Carlo method}
In the quest to understand the many systems that comprise the modern world we turn incr

\section{Gödel's Theorem: Limits on Logic}
In the early 1930's, Kurt Gödel, a German mathematician, attempted to show that predica

\section{Can machines think?}
Turing addressed the question ``Can machines think?'' in his 1950 paper \textit{Comput

\section{But what is LaTeX good for?}
We're using this \LaTeX\ document to demonstrate some of its key strengths that you wil

\begin{enumerate}
\item LaTeX can quickly create pdf files
\item LaTeX uses professional typesetting
\item LaTeX documents can be more legible, clear, and visually appealing to the reader
\end{enumerate}
```

```
\end{document}
```

The text here is excerpted from *The New Turing Omnibus: 66 excursions in Computer Science* (Dewdney, 2001). The Omnibus is a lovely introduction to the fundamentals of Computer Science that you might enjoy. In his book review, the software engineer Jeff Atwood calls the omnibus an "incredibly fun little book". (Atwood, 2007)

2.3 Exercise one: documentum

In your file `turing.tex` either cut-and-paste this longer text into your document or make your own sections and text. You could use text from Lorem ipsum at lipsum.com to fill out the page.

Now, at the top of your document after the `\begin{document}` line and before first `\section`, add the following commands, each on their own line:

```
\title{The New Turing Omnibus}
\author{A. K. Dewdney}
\maketitle
\tableofcontents
\newpage
```

The `title`, `author`, `tableofcontents` and `newpage` commands are self-explanatory. The `maketitle` automatically inserts today's date. Your table of contents won't be created until you run pdflatex **twice** because on the first run, LaTeX gathers and stores information about what to put in the table of contents, and only creates it on the second run.

```
# remember to run pdflatex twice for the table of contents
pdlfatex turing.tex
pdflatex turing.tex
```

2.4 Bold, italic and lists

Here's a few points to note about the text above:

- Notice how **bold** and *italic* formatting are created using `\textbf{}` and `\textit{}`
- Notice how lists are created with `\item`s inside either `\begin{enumerate}` for numbered lists or `\begin{itemize}` for bulleted lists.

2.5 Quotation marks

Typographic or "curly" quotation marks are different characters and *not* the same as straight quotes. They look nicer, but you have to remember that unlike straight quotes, open and close quotation marks are different characters:

- open quote " which looks a bit like a miniature `66`
- close quote " which looks a bit like a miniature `99`

Look carefully at the quotation marks in the text below:

- Turing addressed the question "Can machines think"…
- Turing addressed the question "Can machines think"…

In your `*.tex` file the correct version looks like this

```
Turing addressed the question ``Can machines think''
```

not

```
Turing addressed the question ``Can machines think``
```

or

```
Turing addressed the question ''Can machines think''
```

The difference is subtle in most web browsers, but it's really noticeable in print and pdf so worth paying attention to. Historically, web browsers have had poor typography, although its getting better all the time, see fonts.google.com for example.

2.6 Summary

You've created a basic document in LaTeX and we've introduced some of its advantages:

- LaTeX can quickly create pdf files
- LaTeX uses professional typesetting
- LaTeX can create pdf files that look better than those created with conventional word processing software packages

Next we'll look at adding some cross-references, figures and citations. This is one thing that LaTeX does much better than your average word processing software which makes it great for editing larger and longer documents, such as your Bachelors, Masters or PhD thesis.

Chapter 3

Cross-referencing, illustrating and citing

LaTeX has simple but powerful tools to allow you to cross-reference, illustrate and cite sources in your documents.

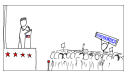


Figure 3.1: It has always been important to cite your sources and LaTeX gives you the tools to cite properly. *Wikipedian Protester* cartoon by Randall Munroe at xkcd.com/285 published under a Creative Commons Attribution-NonCommercial 2.5 License

3.1 Cross-referencing

LaTeX allows you to cross-reference almost anything in your document, including sections, sub-sections and figures. To use the cross-referencing feature you simply insert the command:

```
\label{mymarker}
```

at the point in the document you want to refer to, and then use the command:

```
\ref{mymarker}
```

when you want to refer to it. Obviously you replace the text `mymarker` with something more meaningful. An important tip here is to call the marker something that refers to the content of that part of the document, and to avoid the temptation to use numbers in case you re-order your document. For example, lets say we wanted to cross reference between sections:

```
\documentclass[a4paper]{article}
\begin{document}
\section{Turing Machines: The Simplest Computers}
\label{sec:simplest}
Turing machines are the simplest and most widely used theoretical models of computing.

\section{Alan Turing}
The Turing machines described in section \ref{sec:simplest} are named after Alan Turing
\end{document}
```

3.2 Illustrating your documents

Documents without figures, images, pictures and graphs are pretty dull, so you'll want to illustrate your document with figures such as the one in Figure 3.2, for example.

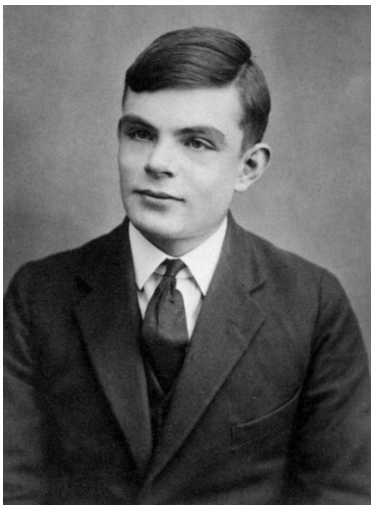


Figure 3.2: Alan Turing at the age of sixteen, portrait by unknown author, public domain, via Wikimedia Commons w.wiki/oZx

Including figures and pictures in your document is straightforward. You do it like this:

```
\begin{figure}
\includegraphics[width=10cm]{images/turing.jpg}
\caption{Turing machines are named after Alan Turing}
\label{figure:turing}
\end{figure}
```

The command `\includegraphics{}` gets your image, which can be PDF, PNG,

JPG, GIF or PostScript. The `\begin{figure}` and `\end{figure}` code wraps up whatever picture you're including, and allows LaTeX to treat it as an unbreakable floating thing that it will position for you as best it can in the document, while maintaining an overall nice typographical layout. This "floating" of figures can sometimes result in the figure ending up in a place you didn't expect, but in most cases LaTeX will make the most sensible choice. It's possible to employ finer control over figure placement, but that's beyond the scope of this guide.

The `\includegraphics` command is not built in to core LaTeX but is in an additional package, which needs to be explicitly loaded. You can load this package by using the command `usepackage` in the document preamble, in between the `\documentclass` and the `\begin{document}`.

```
\documentclass[a4paper]{article}
\usepackage{graphicx}
\begin{document}
```

3.3 Exercise two: picture this

Create a document `image.tex`, that contains some text (maybe from Lorem Ipsum), together with a figure containing an image of your choice. Create a cross reference to the figure in the text.

3.4 Citations and footnotes

Footnotes[1] can be added to a document with `\footnote{}`

```
\footnote{This is a footnote about footnotes}
```

You can cite sources such as websites, books or journal articles in your document using the `\cite` command.

```
\cite{alanturing}
```

The metadata for the citations can be stored in a separate `*.bib file` using a format called BibTeX, in this case we'll use `turing.bib`. BibTeX provides citation types so books[2] are described using the `@book` type like this:

```
@Book{turingomnibus,
  title = {The New Turing Omnibus: Sixty-six excursions in Computer Science},
  author = {A. K. Dewdney},
  publisher = {Henry Holt},
  address = {New York},
```

[1] This is a footnote about footnotes. Very meta.

[2] Using tools like https://www.ottobib.com can help you to quickly generate BibTeX entries from a given ISBN number

```
  year = {2001},
  isbn = {9780805071665},
  url = {https://en.wikipedia.org/wiki/Special:BookSources?isbn=978-0805071665}
}
```

and articles[3] use the `@article` type like this:

```
@article{alanturing,
  doi = {10.1112/plms/s2-42.1.230},
  url = {https://doi.org/10.1112/plms/s2-42.1.230},
  year = {1937},
  publisher = {Wiley},
  volume = {s2-42},
  number = {1},
  pages = {230--265},
  author = {Alan Turing},
  title = {On Computable Numbers, with an Application to the Entscheidungsproblem},
  journal = {Proceedings of the London Mathematical Society}
}
```

For everything else[4] you can use the `@misc` type:

```
@misc{googlescholar,
  author       = {Anon},
  title        = {Alan Turing Google Scholar page},
  url          = {https://scholar.google.co.uk/citations?user=VWCHlwkAAAAJ},
  year         = {2022},
}
```

So to create the text:

"*You can find Turing's publications in Google scholar. (Anon, 2020) His paper on the Entscheidungsproblem was published in 1937. (Turing, 1937) He wrote about thinking machines in 1950. (Turing, 1950)*"

You add this to your tex file:

```
You can find Turing's publications in Google scholar. \cite{googlescholar}
His paper on the Entscheidungsproblem was published in 1937. \cite{alanturing}
He wrote about thinking machines in 1950. \cite{turing50}
```

To generate your bibliography you'll need to specify what style of bibliography you're using with `\bibliographystyle{stylename}` and where to find the metadata with `\bibliography{bibfile}`. The example below uses a style

[3] BibTeX entries for journal articles can be automatically generated from persistent identifiers known as digital object identifiers or doi's, such as https://doi2bib.org for example. This saves you the unpleasant, time consuming and error prone task of typing them in by hand.

[4] There are many other types of BibTeX entries besides articles, books and misc. See https://en.wikibooks.org/wiki/LaTeX/Bibliography_Management#BibTeX

called `unsrt` and uses a bib file called `turing.bib` which might contain many articles and books. You don't need to specific the file extension `.bib`:

```
\bibliographystyle{unsrt}
\bibliography{turing}
```

When you compile you need to run `pdflatex` before and after running `bibtex` like this:

```
pdflatex turing.tex
bibtex turing
pdflatex turing.tex
pdflatex turing.tex
```

If you find all the typing at the command line tedious, you could write a little bash script to automate this simple workflow including opening the pdf when it is created.

3.5 Summary

You have cross-referenced, illustrated, added citations and footnotes to your document. Next we'll look at doing some maths.

Chapter 4

Doing the maths

LaTeX provides powerful tools for typesetting mathematics and this chapter looks at some of them.

If you can't see the equations and matrix below in your web browser, you'll need to use the pdf or epub version of this manual at latex4year1.pdf or latex4year1.epub. Otherwise, if you can see the equation below, it is safe to read on…

4.1 Equations

Consider this equation:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

To render the equation, you include this in your tex file:

```
\[ x = \frac{-b \pm \sqrt{b^2-4ac}}{2a} \]
```

You can probably work out how most of this creates the formula, but it won't be obvious that the `\[` and `\]` symbols that enclose the formula mean typeset this as a displayed formula, giving it some vertical space from the surrounding text. If we'd used `\(` and `\)` instead to enclose the formula it would appear in-line, like this: $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$. It still looks very nice, and observe how it's automatically been resized to fit, and that the lines of text have had their spacing changed a bit. This all looks simple, but the implementation inside LaTeX and TeX is complex. It involves parsing the description of the formula to create a corresponding tree data structure, which is then recursively walked to work out the horizontal and vertical typographical spacings needed. You'll meet these ideas in COMP11120 Mathematical Techniques for Computer Science in your

first year and COMP26120 Algorithms and Imperative Programming in your second year.

Here's another example, taken from computer graphics, it's a local illumination model incorporating ambient, diffuse and specular reflection by multiple lights:

$$I = k_a I_a + \sum_{i=1}^{M} \frac{I_{p_i}}{d'_i}[k_d(\hat{N} \cdot \hat{L_i}) + k_s(\hat{R_i} \cdot \hat{V})^n]$$

We write this in LaTeX as follows:

```
\[ I = k_a I_a + \sum_{i=1}^M { \frac{{I_p}_i}{d'_i} }
[ k_d(\hat{N} \cdot \hat{L_i})
+ k_s(\hat{R_i} \cdot \hat{V})^n] \]
```

Try to match the LaTeX commands with the formula displayed above. You'll see lots of curly brackets, and this example illustrates their two uses in LaTeX. The first is to provide an argument to a command; for example `\hat{N}` means apply the `\hat` command to N', which creates $\hat{N}$, the vector N with a little hat on.

The second use of curly brackets is to group things together to avoid ambiguities. In the example you can see $\sum_{i=1}^{M}$, which creates a summation sign and its lower and upper limits: $\sum_{i=1}^{M}$. We wrap the lower bound, `i=1` in curly brackets to group it into an indivisible unit. If we were to omit the brackets, writing `\sum_i=1^M`, LaTeX would then produce $\sum_i = 1^M$, which is not at all what we want (even LaTeX can't always know what we really want).

4.2 Matrices

As well as equations, LaTeX can display matrices. This one expresses a particular 3D geometrical transformation:

$$T_1 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & \delta \\ \sin\theta & \cos\theta & 0 & \epsilon \\ 0 & 0 & 1 & \eta \\ \alpha & \beta & \gamma & 1 \end{bmatrix}$$

The source for this matrix looks like this:

```
\[ T_1 = \left[
 \begin{array}{cccc}
 \cos \theta & -\sin \theta & 0 & \delta \\
 \sin \theta  & \cos \theta & 0 & \epsilon \\
 0 & 0 & 1 & \eta \\
 \alpha & \beta & \gamma & 1
```

```
    \end{array}
    \right] \]
```

In the LaTeX code, `\left[` means 'big opening square bracket please'; `{cccc}` means 'an array with 4 columns please, with the items in each column centred'; `&` means 'start a new column'; and `\\` means 'start a new row'. You'll notice that LaTeX knows about greek letters; it knows about most standard maths symbols too, and also the ways they're usually used.

There are some symbols that you might need (for example `\therefore`, which produces the usual three dots symbol) that are not part of standard LaTeX. Many such symbols are provided by the `amssymb` package of symbols compiled by the American Mathematical Society. Details of the many symbols provided by this package can be found online. To use these extra symbols, you need to have `\usepackage{amssymb}` in your document preamble.

LaTeX really shines at typesetting mathematics, we've really only scratched the surface here. For more have a look at en.wikibooks.org/wiki/LaTeX/Mathematics or books such as the *Guide to LaTeX* by Helmut Kopka and Patrick W. Daly. (Kopka and Daly, 2003)

4.3 Exercise three: you do the maths

Now for an exercise that involves some mathematics. Create a file `maths.tex` which typesets the following piece of text and mathematics:

There are many positive integer solutions to the equation

$$x^2 + y^2 = z^2$$

which can be rewritten as

$$z = \sqrt{x^2 + y^2}$$

For example $(3, 4, 5)$ or $(5, 12, 13)$. Such solutions are called *Pythagorean triples*.

However, for higher powers the situation is very different, and we have:-

Theorem: Fermat-Wiles For all natural numbers $n \geq 3$, there are no integers x, y, z satisfying the equation:

$$x^n + y^n = z^n$$

4.4 Summary

We've briefly introduced typesetting mathematics in LaTeX with some equations and a matrix. LaTeX can handle a lot more maths than that but this gives you a flavour of what it can do. In the next chapter we'll look at collaborative authoring and editing in the cloud with overleaf.

Chapter 5

Collaborative editing with overleaf

If you are the sole author of a document, then compiling files on your local machine is fine. However, if you need to collaboratively co-author a document with other people, you'll need to share your TeX somehow. Sharing `*.tex` files by email or dropbox or whatever would be cumbersome. What you need is something like Google Docs for `*.tex`. Overleaf (overleaf.com) is one of several web applications that allows you to do this, shown in the screenshot in Figure 5.1.

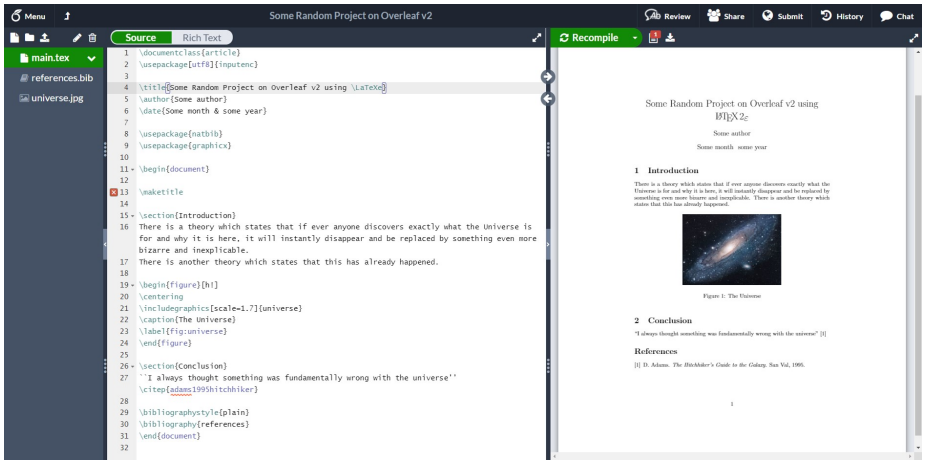


Figure 5.1: A screenshot of overleaf showing the source TeX on the left hand side and the corresponding pdf on the right hand side. Screenshot by Dan Cherniy via Wikimedia Commons at w.wiki/omo

Compared to LaTeX, overleaf is relatively new. Write Latex Limited, the company behind overleaf was founded fairly recently (in LaTeX terms) in 2013 by John Lees-Miller and John Hammersley. Overleaf has several features which you might find useful:

- Your documents are automatically saved to the cloud
- Packages, class files, compilers and other LaTeX components are also in the cloud, saving you time installing and managing them
- Templates are provided for common types of documents, although you don't need to use overleaf to get access to LaTeX templates[1]
- Overleaf publish lots of tutorials to help you learn LaTeX

5.1 Exercise four: overleaf

Login to overleaf.com and try the following:

1. Create and save a simple document using the overleaf tutorial *creating a document in overleaf*[2]

2. Note that overleaf allows you to store your TeX source in a git repository so you can use version control if you want to
3. Browse the overleaf tutorials at overleaf.com/learn if you want to take things further

5.2 Summary

Overleaf provides a modern and convenient cloud based interface to tried and tested LaTeX tools, which have been around for over thirty years. (Knuth, 1984) It also allows you to share the source of your documents while providing some handy templates for common document types. Overleaf and the command line are just two popular interfaces to LaTeX amongst many others. (Tex, 2020) Which interface is "best" for you will largely depend on what kind of documents you are writing and what your workflow is. In the next chapter we will create a curriculum vitae using CV templates provided by overleaf.

[1] see http://www.tug.org/texshowcase and https://www.latextemplates.com for example
[2] https://www.overleaf.com/learn/how-to/Creating_a_document_in_Overleaf

Chapter 6

Your curriculum vitae

You'll write many documents at University, but there is one document that is **really** important in a potentially life-changing way. Your curriculum vitae[1] or resume. An example CV is shown in Figure 6.1.

Creating a professional looking CV is particularly important, because it determines if you are invited to interview for *opportunities* you are applying for. The decision to interview is typically based on several factors:

- how your CV looks, the typesetting and style (typography)
- the content of your CV, what you've done
- the quality and clarity of your written communication, how you describe yourself and your experience
- the editing, what you've decided to leave in (and leave out) of your CV

6.1 Opportunity knocks

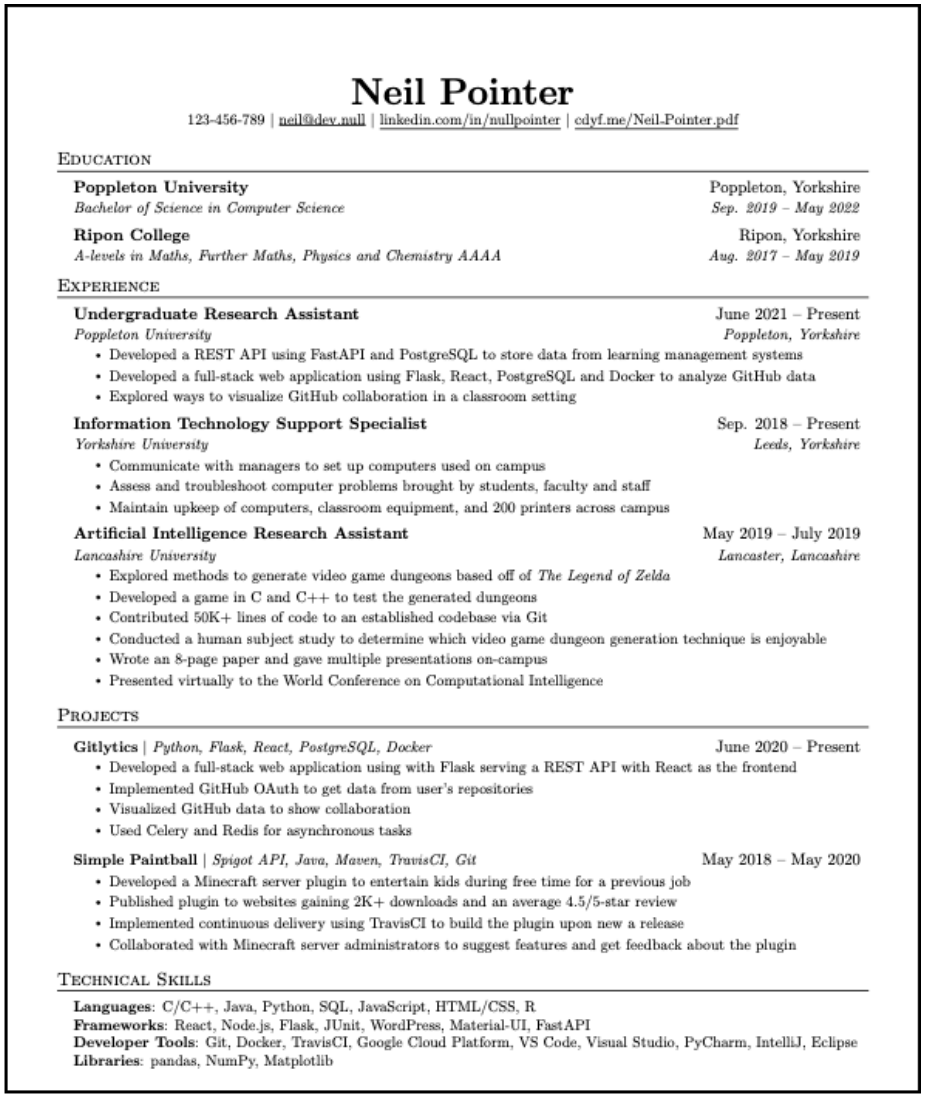
By *opportunities* we mean both immediate ones within the next 12 months as well as those further in the future.

6.1.1 First year opportunities

During your first year of study, *opportunities* include:

- Spring insights during easter next year
  - see ratemyplacement.co.uk/insights some of which have application deadlines before Christmas
  - see opportunities for first year students at careers.manchester.ac.uk/findjobs/internships/1stYearOpps
- Summer internships suitable for first years:

[1] strictly speaking it should be vitæ (not vitae) if you're being pedantic



Neil Pointer

123-456-789 | neil@dev.null | linkedin.com/in/nullpointer | cdyf.me/Neil-Pointer.pdf

Education

Poppleton University — Poppleton, Yorkshire
Bachelor of Science in Computer Science — *Sep. 2019 – May 2022*

Ripon College — Ripon, Yorkshire
A-levels in Maths, Further Maths, Physics and Chemistry AAAA — *Aug. 2017 – May 2019*

Experience

Undergraduate Research Assistant — June 2021 – Present
Poppleton University — *Poppleton, Yorkshire*
- Developed a REST API using FastAPI and PostgreSQL to store data from learning management systems
- Developed a full-stack web application using Flask, React, PostgreSQL and Docker to analyze GitHub data
- Explored ways to visualize GitHub collaboration in a classroom setting

Information Technology Support Specialist — Sep. 2018 – Present
Yorkshire University — *Leeds, Yorkshire*
- Communicate with managers to set up computers used on campus
- Assess and troubleshoot computer problems brought by students, faculty and staff
- Maintain upkeep of computers, classroom equipment, and 200 printers across campus

Artificial Intelligence Research Assistant — May 2019 – July 2019
Lancashire University — *Lancaster, Lancashire*
- Explored methods to generate video game dungeons based off of *The Legend of Zelda*
- Developed a game in C and C++ to test the generated dungeons
- Contributed 50K+ lines of code to an established codebase via Git
- Conducted a human subject study to determine which video game dungeon generation technique is enjoyable
- Wrote an 8-page paper and gave multiple presentations on-campus
- Presented virtually to the World Conference on Computational Intelligence

Projects

Gitlytics | *Python, Flask, React, PostgreSQL, Docker* — June 2020 – Present
- Developed a full-stack web application using with Flask serving a REST API with React as the frontend
- Implemented GitHub OAuth to get data from user's repositories
- Visualized GitHub data to show collaboration
- Used Celery and Redis for asynchronous tasks

Simple Paintball | *Spigot API, Java, Maven, TravisCI, Git* — May 2018 – May 2020
- Developed a Minecraft server plugin to entertain kids during free time for a previous job
- Published plugin to websites gaining 2K+ downloads and an average 4.5/5-star review
- Implemented continuous delivery using TravisCI to build the plugin upon new a release
- Collaborated with Minecraft server administrators to suggest features and get feedback about the plugin

Technical Skills

Languages: C/C++, Java, Python, SQL, JavaScript, HTML/CSS, R
Frameworks: React, Node.js, Flask, JUnit, WordPress, Material-UI, FastAPI
Developer Tools: Git, Docker, TravisCI, Google Cloud Platform, VS Code, Visual Studio, PyCharm, IntelliJ, Eclipse
Libraries: pandas, NumPy, Matplotlib

Figure 6.1: A fictitious example CV using an overleaf template. Would you invite Neil to interview based on his CV? This is what your CV needs to do, convince a decision maker they really need to contact you to find out more. This is just a screenshot, the pdf can be found at cdyf.me/Neil_Pointer.pdf. There are over 600 CV templates available to choose from at overleaf.com/gallery/tagged/cv for you to choose from.

  - see internships (open to first years) at gradcracker.com/search/computing-technology/work-placements-internships
  - see the chapter on *Finding your Future* in *Coding your Future* (Hull, 2021b)

6.1.2 Subsequent opportunities

After your first year *opportunities* include:

- Year long placements in your penultimate year, if you're considering doing industrial experience
- Summer internships (aimed at penultimate year students) + there are usually around ten summer internships in the Department of Computer Science for example, these *normally* don't get advertised around April/May time
- Graduate jobs or graduate schemes after graduation
- Postgraduate study or research via masters or PhD etc

Any time you invest in creating a convincing CV will pay off in the long run. Yes, you've only just started University, so might not have much to talk about just yet, but it's never too early to make a start.

6.2 Exercise five: your CV

Create a basic CV which tells your story, in particular:

- your *education*, including high school and University
- your *experience*, voluntary, paid, casual, technical and non-technical: *any* experience demonstrates your range of soft and hard skills
- your *projects*, personal, social, educational and entrepreneurial

You can do this using the ready-made templates at overleaf.com/gallery/tagged/cv. Have a good look around, there are over 600 templates to choose from.

6.3 Debugging your CV checklist

As you progress through University, continuously update your CV and solicit feedback from as many people as you can. Your fellow students, personal tutors, friends, family and anyone else you trust can all give you valuable feedback. There will be opportunities to debug your CV later, but you'll need a "beta release" (version 1.0) of your CV to get started. The best time to start debugging is now, so that you can squash any bugs before employers see them. Innocent bugs **can be fatal** because most employers typically have to deal with lots of applicants. Here's a check list of some common bugs we have seen in students CVs:

1. Is your year of graduation, degree program, University and expected (or achieved) degree classification clear?
2. Are there any spelling mistakes, typos and grammatical errors? Don't just rely on a spellchecker, they can't detect everything
3. Does it look good, decent layout, easy to scan?
4. Does it fit comfortably on one page (preferably) or two pages only? Not too cramped or gappy?
5. Is it in reverse chronological order? Are the most important (usually recent) things first?
6. Have you talked about what you have actually done using prominent verbs, rather than just what you know? See the *Actioning your Future* chapter of *Coding your Future* for more examples
7. Have you mentioned disciplines you are studying now and throughout the current academic year, not just courses you have finished?
8. Have you quantified and provided evidence for the claims you make?
9. Is your CV robot proof? Many large employers use automated applicant tracking systems that use software to screen CVs long before a human ever sees them. You can feed your CV through software like careerset.io/manchester and resume.io, what feedback do the robots give you? How can you make your CV more robot proof?
10. Find out more in the *debugging your future* checklist at cdyf.me/debugging#checklist (Hull, 2021a)

6.4 Summary

Your curriculum vitae is a *really* important document and it will most likely take many iterations to get it right. We recommend you start working on it sooner rather than later and get feedback from as many different people (and bots) as you reasonably can.

In the meantime, enjoy exploring and using LaTeX to create professional documents for:

- your individual COMP101 coursework (see blackboard)
- your CV / resume
- your third year project dissertation. It might seem a long way off now but it comes around very quickly!

…and more.

Appendix A

Typography

A mostly empty placeholder appendix which will cover some typographical details about kerning and ligatures etc.

Bibliography

Anon (2020). Alan Turing's Publications.

Atwood, J. (2007). Practicing the Fundamentals: The New Turing Omnibus.

Dewdney, A. K. (2001). *The New Turing Omnibus: 66 excursions in Computer Science*. Henry Holt, New York.

Hull, D. (2021a). Debugging your future. In *Coding Your Future: A Guidebook for Students*. University of Manchester, github.com.

Hull, D. (2021b). Finding your future. In *Coding Your Future: A Guidebook for Students*. University of Manchester, github.com.

Knuth, D. (1984). *The TeXbook*. Addison-Wesley, Boston, Massachusetts.

Kopka, H. and Daly, P. W. (2003). *Guide to LaTeX*. Addison-Wesley, Boston, Massachusetts, fourth edition edition.

Roberts, S. (2018). The yoda of silicon valley: Donald knuth, master of algorithms, reflects on 50 years of his opus-in-progress, "the art of computer programming.". *New York Times*.

Tex, L. (2020). LaTeX – a document preparation system.

Turing, A. (1937). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265.

Turing, A. (1950). I.—Computing Machinery and Intelligence. *Mind*, LIX(236):433–460.