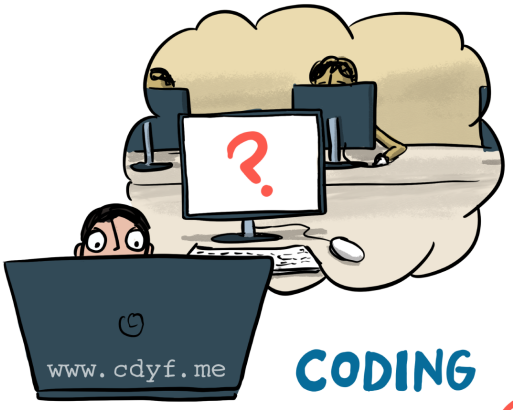
Coding Your Future: A Guidebook for Students

Duncan Hull and Bryan Mathers

Contents

Welcome to your future



CODING your future

Hello and welcome to *Coding Your Future* (cdyf.me) the guidebook that will help you to design, engineer, test and `code` your future in computing. Also published at cdyf.pdf & cdyf.epub, this book is aimed at ALL students in higher education. While the guide supports undergraduate teaching at the University of Manchester, it doesn't actually matter:

- what *stage* of your degree you are at, from first year through to final year
- what *level* you are studying at, foundation, undergraduate or postgraduate
- what *subject* you are studying, as long as you are **computationally curious**
- what *institution* you are studying at, this book is University and institution agnostic
- *where* in the world you are studying

There is something in this guidebook for *any* student of computing, both those inside and outside of Computer Science departments.

0.1 Imagining your future

A lot of self-help literature can be dry, dull, textbooky, generic and boring with few illustrations and conversations. In the novel *Alice's Adventures in Wonderland* (Caroll, 1865) shown in figure 1, the protagonist Alice wonders why her sister is reading a book without pictures.

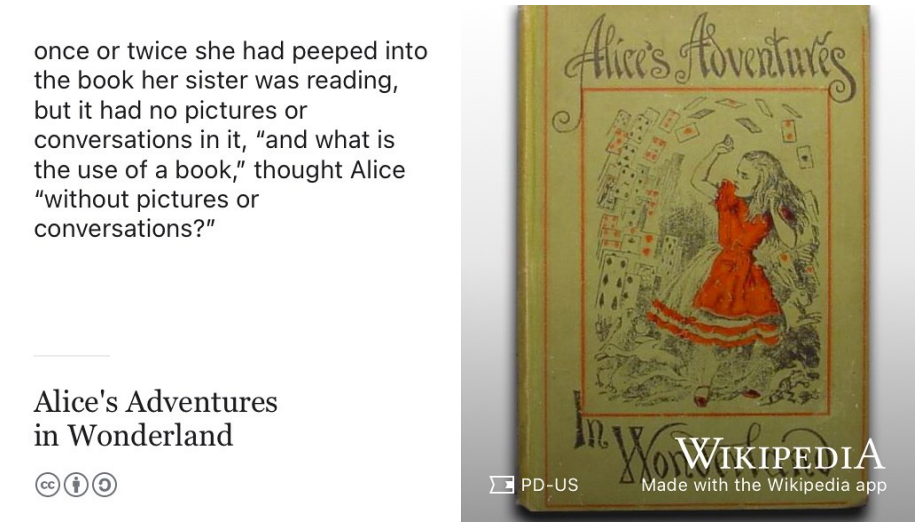


Figure 1: Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice "without pictures or conversations? (Caroll, 1865) Public domain image of the cover of the 1898 edition of the novel *Alice's Adventures in Wonderland* via Wikimedia Commons w.wiki/3S4C adapted using the Wikipedia app

Pictures explain. Pictures help you understand. Pictures help you imagine. So this book uses pictures (and conversations) to help you imagine and visualise your future.

0.2 Your future aims

This guidebook aims to help you develop stronger habits of mind, body and soul using five key ingredients:

1. **Code:** Instructions, algorithms, recipes and strategies contained in this guidebook. This `code` is for your consumption, not for a machine.

2. **Data:** Facts, statistics, graphs and pictures collected together for your analysis
3. **You:** Activities for you to do in addition to reading
4. **Futures:** Possible futures for you to think about. Try not to dwell on the past. Think about the future. Think about *your* future. (Ryder, 1988, 2019)
5. **Me:** Hello, my name is Duncan. I'm your tour guide here. If you're feeling a bit lost, follow me.

Figure 2: Hello my name is Duncan. If you're feeling a bit lost, follow me. Image adapted from *Hello my name is … sticker* by Eviatar Bach, public domain w.wiki/32RV

Coding your future explores techniques for making career decisions, job searching, submitting applications and competing successfully in interviews and the workplace.

Alongside these practical engineering issues, this guidebook also encourages you to *design your future* by taking a step back and reflecting on the bigger picture. You will apply computational thinking techniques, to reflect on who you are, what your story is, how you communicate with other people and what your experience is. As there is a computational theme, you will also need to reflect on what your inputs and outputs (I/O) are, both now and in the future. You'll also need to think about what recipes (or algorithms) you might start experimenting with

This guidebook tackles professional issues in computing, for those with and without Computer Science degrees in the early stage of their careers.

0.3 What you won't learn

This guidebook will NOT teach you how to write code, there's already lots of fantastic resources to help you do that. We discuss some of them in chapter 6 on *computing your future*.

0.4 Learning your future

So what *will* you learn from this guidebook? After reading this guidebook, watching the videos and doing the exercises you will be able to:

1. Improve your self-awareness by describing who you are, what motivates you and your strengths and weaknesses
2. Decide on a job search strategy and identify employers, sectors and roles that are of interest to you

3. Improve your written communication skills both for job applications and communicating with other people
4. Plan and prepare competitive written applications using standard techniques including CVs, covering letters, application forms and digital profiles
5. Compete successfully in interviews and assessment centres by preparing for technical and non-technical questions
6. Plan further steps in your career such as promotion, postgraduate study & research, alternative employment and longer term goals
7. Search and navigate a large "wordbase" (this guidebook and the work it cites). A wordbase is like a `codebase`, only written predominantly in natural language.

0.4.1 Your future requirements

As the title of this guidebook implies, there is a computational flavour here, but you do not have to be studying Computer Science to benefit. There are two main target audiences for this guidebook:

1. Undergraduate and postgraduate students studying Computer Science as a major or minor part of their degree. This includes software engineering, artificial intelligence, human-computer interaction (HCI), information systems, health informatics, data science, gaming, cybersecurity and all the other myriad flavours of Computer Science
2. Undergraduate and postgraduate students studying *any* subject, with little or no Computer Science at all. You are curious to know about what

> role computing could play in your future career because computing is too important to be left to Computer Scientists, see chapter 6 on *Computing your Future*

So the prerequisites for this book are that you are studying (or have studied) at University where English is one of the main spoken languages. You *may* have some experience already, either casual, voluntary or otherwise, but this book does **not** assume that you have already been employed in some capacity.

0.4.2 Gutting your future

Reading this book from cover to cover like a novel is not recommended. That would be foolish.

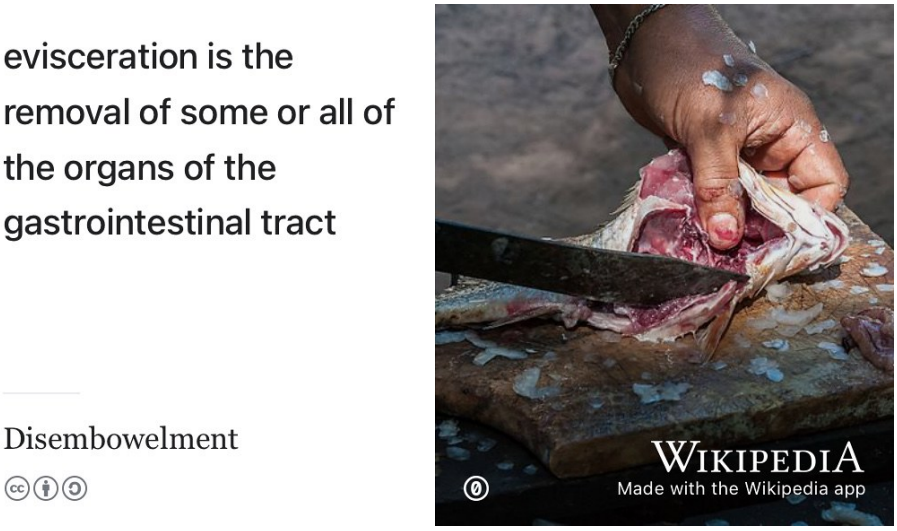


Figure 3: Don't read this book! Disembowel it. Eviscerate it. Gut it like a fish. Enjoy the nourishing flesh and discard the less appetising organs of its gastrointestinal tract. You'll need to decide which is which, depending on your tastes and appetite. CC0 Public domain image of fish gutting by Wilfredor via Wikimedia commons w.wiki/_23m adapted using the Wikipedia app

Instead of reading this book, I suggest you follow the advice given to historian William Woodruff about reading books when he was at University:

> "You don't READ books, you GUT them!" (Woodruff, 2003)

So, gut this book like the fish in figure 3. Identify the chapters that are most useful to you (the flesh), and skip the rest (the guts). Which chapters are flesh

and which are guts will depend on what stage of the journey you are at. This guidebook is designed to be as "guttable" as possible. To aid gutting, the version published at cdyf.me has a built in search and tables of contents. Before you can gut the fish, you'll need an anatomical map shown in figure 4.

0.5 Mapping your future

This guidebook is split into three parts. The first part (Chapters 1 to 6) is on design while the second part (chapters 7 to 13) is on building and testing your future shown in the map in figure 4. The final part is a help section for supporting your future (chapters 14 to 20). Let's look in a bit more detail at the content of each of the three parts of this guidebook:

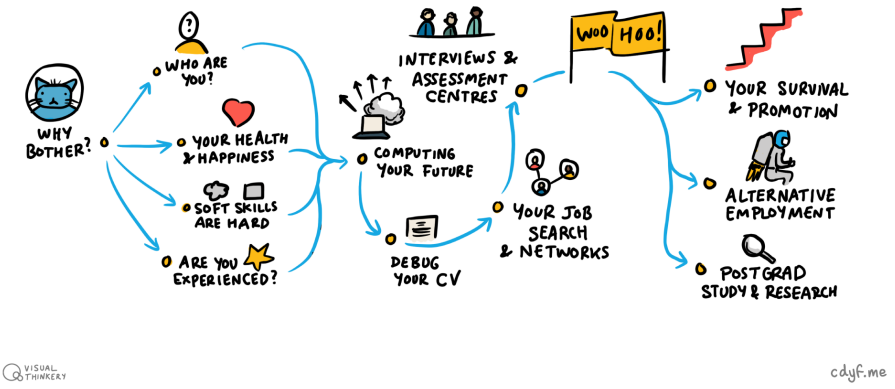


Figure 4: Mapping your future: Each yellow dot on this diagram is a chapter in *Coding Your Future*. The chapters on the left tackle design issues like *who are you*? Chapters on the right tackle the practicalities of executing and testing your career choices, such as *debugging your CV*. Mapping your Future artwork by Visual Thinkery is licenced under CC-BY-ND

0.5.1 Designing your future

The first six chapters of this guidebook look at what engineers call *design*. When you build anything, a bridge, a piece of software, a car or a plane you'll need to do some design like the blueprint in figure 5

Building a career isn't that different to building anything else, you'll need to do some design work and it will probably be iterative. Designing things often

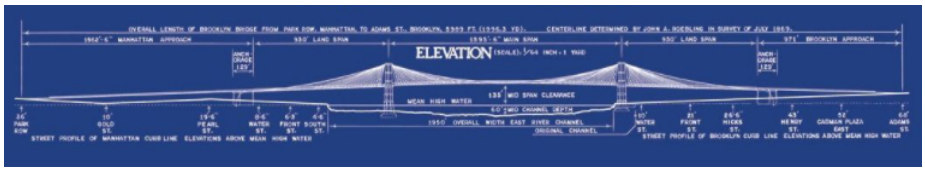


Figure 5: Designing your future is about drawing up a blueprint, like this one for the elevation of the Brooklyn Bridge in New York. What does your blueprint look like? Chapter's 1 through to 6 will help you design your future.

involves asking tricky questions. So when you're designing your future you'll need to cover the following:

- Chapter 1: *Rebooting your future* discusses why you should bother reading this guidebook
- Chapter 2: *Knowing your future* challenges you to reflect on who you are, what makes you unique and why you are here
- Chapter 3: *Nurturing your future* encourages you to take care of your mental and physical health
- Chapter 4: *Writing your future* explores your soft skills, and how they complement your hard skills and why employers value them so much
- Chapter 5: *Experiencing your future* asks you to reflect on your work experience and help identify where you can improve it
- Chapter 6: *Computing your future* looks at the role computing can play in your career, especially if Computer Science is not a major part of your degree

0.5.2 Building your future

The next seven chapters look at building (and testing) your future, what engineers like to call *implementation* or *execution* shown in figure 6.

Once you've started to answer the design questions in the first part, you can start to implement (or build) your career and think about what the next steps will be.

- Chapter 7: *Debugging your future* looks at debugging your written communication such as covering letters, application forms and digital portfolios.
- Chapter 8: *Finding your future* looks at where and how can you look for interesting opportunities
- Chapter 9: *Broadening your future* encourages you to broaden your horizons. What are the possibilities beyond the obvious?
- Chapter 10: *Speaking your future* looks how can you turn interviews to your advantage and negotiate any offers you receive

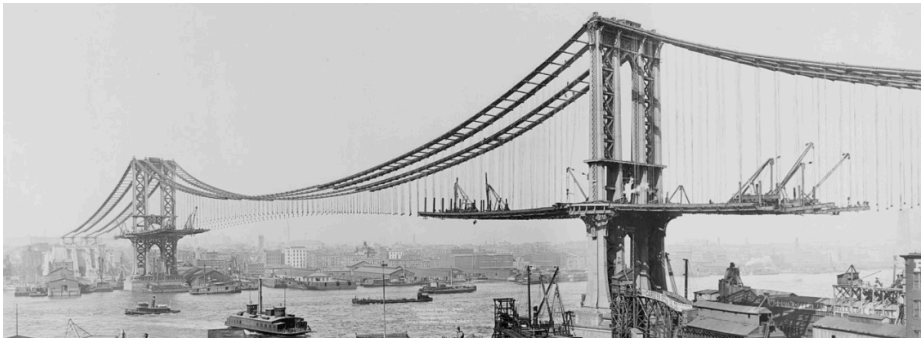


Figure 6: Just like the Manhattan Bridge, your future will be easier to build once you've done some design. You don't need a grand design with tonnes of details, a simple sketch will do. Design questions are covered in the first part of this guidebook on designing your future. Picture of the Manhattan bridge under construction in 1909 adapted from a public domain image via Wikimedia commons w.wiki/32Rg

- Chapter 11: *Surviving your future* looks at the next steps. Once you've landed a job, how will you survive and thrive outside (and after) University
- Chapter 12: *Achieving your future* looks at evidence you can collect of your learning and development using various kinds of certifiable evidence
- Chapter 13: *Researching your future* discusses if a Masters degree or a PhD right for you?

0.5.3 Supporting your future

The third part of this book, contains supporting material that will help the design, build and test phases described above. You'll need good support to help with the stresses and strains of building your future shown in 7

- Chapter 14: *Ruling your future* provides *Ten Simple Rules for Coding your Future*, this book in a nutshell
- Chapter 15: *Hacking your future* invites you to put yourself in the employers shoes by hacking other people's CVs
- Chapter 16: *Moving your future* looks at opportunities outside of capital cities like London
- Chapter 17: *Hearing your future* invites you to listen to students stories of their transition from education to employment
- Chapter 18: *Actioning your future* gets you to think about your actions by emphasising verbs on your job applications
- Chapter 19: *Scheduling your future* is the live synchronous sessions for this course, if you're not participating in these, schedule a time every day or week for personal development

Figure 7: Huge supporting chains on the Clifton Suspension Bridge in Bristol allow heavy loads pass over the Avon valley bridge. You'll need good support to cope with the stresses and strains of building your future. Clifton suspension bridge picture adapted from original by Nic Trott via Wikimedia commons w.wiki/32tu

- Chapter 20: *Reading your future* lists everything cited in this guidebook.

0.6 Your future themes

This guidebook aims to help you build a bridge from where you are now to where you'd like to be in the future. Each chapter of the book contains the following recurring themes:

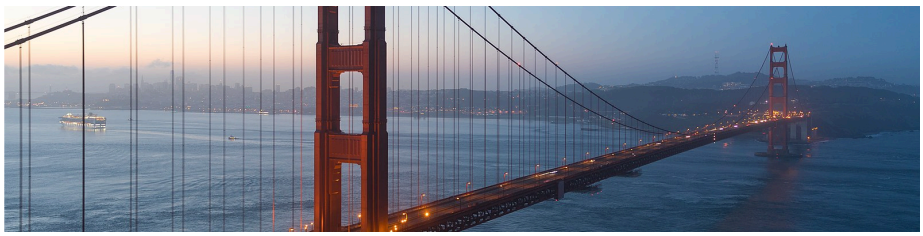


Figure 8: This guidebook will help you build a bridge to your future. Picture of the iconic Golden Gate Bridge in California during the blue hour adapted from an original by Frank Schulenburg (CC BY-SA) on Wikimedia Commons w.wiki/37kY

1. **Learning** your future: What you will learn from any given chapter
2. **Watching** your future: videos and animations for you to watch
3. **Listening** to your future: audio and podcasts for you to listen to
4. **Speaking** your future: articulating from a script or by improvisation, particularly when preparing for interviews
5. **Discussing** your future: breakpoints invite you to pause your code and think about the variables and parameters you are using. Can they be

improved? Reflect and discuss.

6. **Reading** your future: reading stuff because its good for your mind, body and soul. Read The Friendly Manual. `RTFM`. Read THIS Friendly Manual.
7. **Writing** your future: written exercises using natural language
8. **Quizzing** your future: quick quizzes to be done in real-time live scheduled sessions described in chapter 19 (synchronously) and in your own time (asynchronously)
9. **Assessing** your future: activities to be assessed by yourself, your peers, an employer or an academic (depending on who and where you are)
10. **Challenging** your future: coding challenges are designed to take you out of your comfort zone by encouraging you to experiment with your thoughts, discussions and actions
11. **Signposting** your future: the most useful resources that I recommend you read, listen to or watch

0.7 Contributing to your future

If you'd like to contribute this guidebook, I welcome constructive feedback from critical friends, see figure 9.

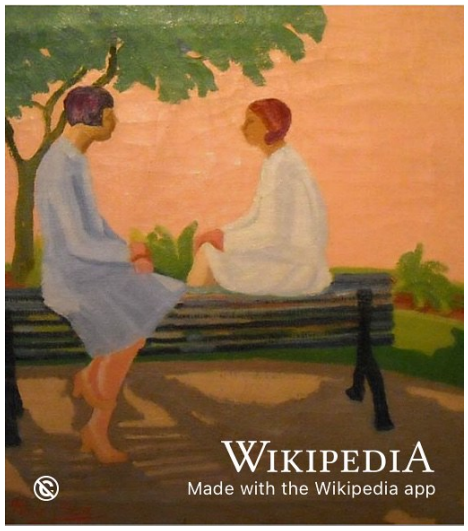


Figure 9: Can you be a supportive but critical friend of this guidebook? Public domain image of a painting *Friendship* by Petrona Viera via Wikimedia Commons w.wiki/3WjY adapted using the Wikipedia App

I'm looking for feedback and contributions on everything in this guidebook from the small things like typos, grammatical errors and spelling mistakes through to bigger issues for each chapter such as:

- Does the chapter make sense, is it clear?
- Does it strike the right tone, is it pitched at the right level? Not patronising? Too many platitudes?
- Are there too many motivational (or demotivational) quotations?
- Where is it too long and waffly (see figure 7.12) or too short?
- Are there too many (or too few) pictures? What needs more illustration?
- Is it well scoped? Too broad or too narrow?
- Are the stated learning objectives met by the chapter?
- Are the activities clear? Can students understand why the activities are recommended? What other activities could be added?
- Will it make sense to global readers e.g. will students from China and India understand the quirks and idioms of English language and culture
- Are there too many metaphors? Mixed metaphors? Awkward analogies? Idiotic idioms? Annoying alliterations?
- Too many citations? Not enough citations? Missed any key citations?
- What's missing?
- Where are the unstated assumptions? Where is the unconscious bias?
- What are the issues with equality, diversity and inclusion?
- Are there too many musical references or annoying emoji? Please bear in mind I'm trying to strike an irreverent, light-hearted and playful tone to improve readability
- What else should be ruthlessly edited out?

All suggestions welcome! Don't be shy. There are several ways you can contribute, depending on how comfortable you are with Git:

0.7.1 For git contributors

If you're familiar with git and markdown you can github.com/join and:

- Raise new issues at github.com/dullhunk/cdyf/issues/new
- Click on the `Edit this page` link, which appears on the bottom right hand side of every page published at cdyf.me when viewed with a reasonably large screen (not a phone)
- Contribute at github.com/dullhunk/cdyf/contribute and help with existing issues at github.com/dullhunk/cdyf/issues
- Fork the repository, make changes and submit a pull request github.com/dullhunk/cdyf/pulls. If you need to brush-up on your pulling skills see makeapullrequest.com
- From the command line, clone the repository and submit pull requests from your own setup:

```
git clone https://github.com/dullhunk/cdyf.git
```

Most of the guidebook is generated from RMarkdown, that's all the `*.Rmd` files. So markdown files are the only ones you should need to edit because everything else is generated from them including the `*.html`, `*.tex`, `*.pdf`,`*.epub` and `*.docx` files.

0.7.2 For everyone else

If you don't want to (or can't) use Git and github.com then you can:

- Add comments by annotating cdyf.pdf using your personal weapon of choice (tablet, reMarkable or whatever) and emailing your updated version to me
- Add comments by annotating cdyf.epub and emailing your updated version to me
- Suggest changes by editing the Microsoft Word version at cdyf.docx. The text is all there, but the images are all over the place. This is because the pagination, typesetting and graphic placement algorithms in Word aren't anything like as good as the LaTeX ones used to create cdyf.pdf from cdyf.tex.[1] Make sure you've turned on track changes in Word. Track changes is Microsoft Word's killer feature that allows your corrections to be easily identified from the original text.
- Just email me suggestions for improvements

Any corrections or suggestions will be gratefully received and noted in the acknowledgements section 0.8, unless you tell me otherwise. I welcome all improvements, big and small.

0.8 Acknowledgements

The content of this book is based on hundreds of conversations I have had with students of Computer Science, Mathematics, Physics and Engineering, since 2012. It is also based on conversations I've had with many of their employers too.

0.8.1 Thank you students

First and foremost, I'd like to thank all the students who have helped with this book, both directly and indirectly see figure 10.

So, if you have studied some flavour of Computer Science at the University of Manchester since 2012, there's a high probability you have contributed to this

[1] Don't say I didn't warn you!

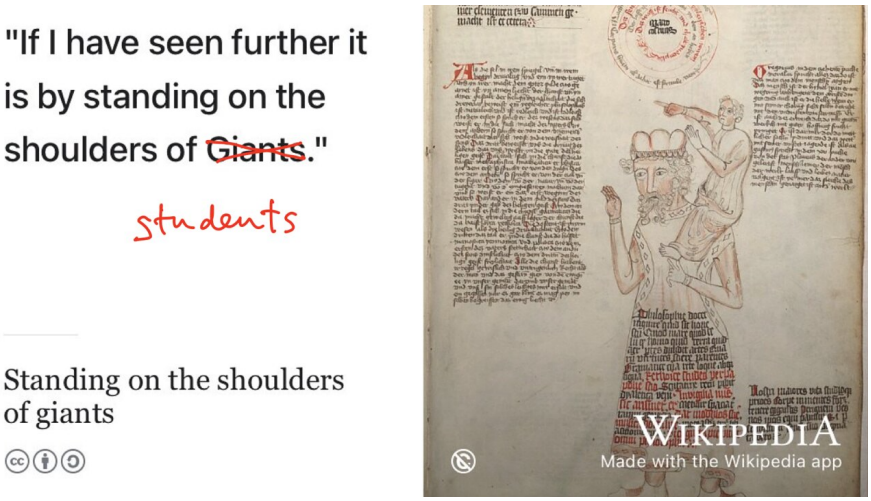


Figure 10: If I have seen further it is by standing on the shoulders of ~~giants~~ students. (Newton, 1675) Public domain image of Orion carrying his servant Cedalion on his shoulders via Wikimedia Commons w.wiki/_zZ2E

book. Thank you for having the courage to tell me your stories. Thank you for being ambitious, hard working, talented, fearless, creative, inspirational and listening to me (sometimes). It has been my pleasure and privilege to work with you all.

I'd especially like to thank industrial experience (IE) students who have completed a year in industry as part of their degree as well as those who have done summer internships, either as part of the Master of Engineering (MEng) program or otherwise, particularly Sami Alabed, Luke Beamish and Petia Davidova. In addition, the PASS leaders and facilitators, UniCSmcr.com, HackSoc, CSSoc and Manchester Ultimate Programming members have all been influential on the content of this book. I've learned heaps by manually trawling through thousands of your CVs too, so if you've shown me a copy of your CV, thanks! If you sent me a CV and I didn't reply, I apologise. There are limits to what is humanly possible. Chapter 7 on *Debugging your future* (self assessment) and chapter 15 on *Hacking your future* (peer assessment) are based on the most common bugs (or are they features?) I've seen in CVs.

So, thank you students for being studious.

0.8.2 Thank you employers

Thanks to all the organisations who have employed students from the Department of Computer Science as either summer interns, year long placements or

Figure 11: Posing on the BBC Breakfast red sofa with the winning student team at the BBC / Barclays University Technology Challenge (UTC) in MediaCityUK, Salford, Greater Manchester

graduates. A big chunk of this guidebook documents their experience of employers and their graduate recruitment programs.

Thanks to Niall Beard and Sharif Salah at Google for introducing me to Google's Technical Writing course. (Googler, 2019)

So, thanks employers for employing our students.

0.8.3 Thank you colleagues

I've also had significant support from colleagues in the Department of Computer Science (@csmcr) and support staff at the University of Manchester. (@ManUniCareers, @alumniUoM, @OfficialUoM)

I would especially like to thank Jim Miles for encouraging me to write a book shortly after he offered me a job. I thought he was joking (about the book) but it actually turned out to be another one of Jim's great ideas. Thanks Jim.

I'd also like to thank the only three people in the whole world who've had the misfortune of reading all of my PhD thesis; Robert Stevens, Anil Wipat and Steve Pettifer. I suspect it was as painful for you to read as it was for me to write it. Thanks Robert for your relentless patience and giving me a well timed, well aimed kick up the (proverbial) arse to write this book in the Midland Hotel, Manchester at the May ball.

Thanks Steve Furber for playing bass guitar in our "boy band" Tuning Complete. Thanks to Carole Goble for patiently re-teaching me how to write by covering early drafts of my MSc thesis in red ink and less patiently (on subsequent revisions) swear words.

0.8.3.1 Thanks to academic staff

Thanks to past and present academic colleagues, PhD students and teachers at the University of Manchester (and elsewhere) who have contributed to this guidebook and the environment it was written in. We are bound together by the power of weak ties (section 8.6) alongside stronger forces and friendships. They include (in alphabetical order):

Pinar Alper, Sophia Ananiadou, Mikel Egaña Aranguren, Constantinos Astreos, Terri Attwood, Sam Bail, Robin Baker, Richard Banach, Riza Batista-Navarro, Michael Bada, Niall Beard, Sean Bechhofer, Lynne Bianchi, Helena Björn van Praagh, Stewart Blakeway, Petrut Bogdan, Caroline Bowsher, Linda Brackenbury, Judy Brewer, Nick Brown, Mihai Bujanca, Oscar Corcho, Christian Brenninkmeijer, Andy Bridge, Andy Brass, Andy Brown, Gavin Brown, Terry V. Callaghan, Grant Campbell, Angelo Cangelosi, Peter Capon, Andy Carpenter, Nicola Carrier, Barry Cheetham, Ke Chen, Sarah Clinch, Ian Cottam, Brian Cox, Simone Di Cola, Paul Dobson, Clare Dixon, Danny Dresner, Nick Drummond, Warwick Dunn, Doug Edwards, Iliada Eleftheriou, Anas Elhaig, Suzanne Embury, Michael Emes, Alvaro Fernandes, Jonathan Ferns, Michele Filannino, Nick Filer, Paul Fisher, Steve Furber, Andre Freitas, Aphrodite Galata, Matthew Gamble, Jim Garside, Kristian Garza, Chris Gilbert, Danielle George, Richard Giordano, Birte Glimm, Carole Goble, Rafael Gonçalves, Antoon Goderis, Roy Goodacre, Graham Gough, Bernardo Cuenca Grau, Peter R. Green, Keith Gull, John Gurd, Luke Hakes, Robert Haines, Guy Hanke, Simon Harper, Phil Harris, Jonathan Heathcote, Lloyd Henning, Gareth Henshall, Andrew Horn, Farid Kahn, Matthew Horridge, Ian Horrocks, Toby Howard, Roger Hubbold, Luigi Iannone, Jane Ilsley, Jules Irenge, Daniel Jameson, Caroline Jay, Mirantha Jayathilaka, Huw Jones, Simon Jupp, Yevgeny Kazakov, John Keane, Douglas Kell, Catriona Kennedy, Rachel Kenyon, Chris Knight, Joshua Knowles, Dirk Koch, Nikolaos Konstantinou, Christos Kotselidis, Ioannis Kotsiopoulos, Oliver Kutz, Alice Larkin, Peter Lammich, John Latham, Kung-Kiu Lau, Margi Lennartsson Turner, Dave Lester, Peter Li, Zewen Liu, Phil Lord, Mikel Luján, Darren Lunn, Matthew Makin, Nicolas Matentzoglu, Paul Mativenga, Erica McAlister, Mary McGee Wood, April McMahon, Merc and members of the Manchester University Mountaineering Club (MUMC), Simon Merrywest, Eleni Mikroyannidi, Colin Morris, Norman Morrison, Georgina Moulton, Boris Motik, Christoforos Moutafis, Tingting Mu, Ettore Murabito, Mustafa Mustafa, Javier Navaridas, Kostas Nikolou, Aleksandra Nenadic, Goran Nenadic, Steve McDermott, Jock McNaught, Mary McGee-Wood, Pedro Mendes, Sarah Mohammad-Qureshi, Tim Morris, Jennifer O'Brien, Tim O'Brien, Steve Oliver, Pierre Olivier, Mario Ramirez Orihuela, Stuart Owen, Ali Owrak, Pavlos Petoumenos, Luis Plana, Jackie Potter, Malcolm Press, Colin Puleston, Paul Nutter, Ignazio Palmisano, Dario Panada, Michael Parkin, Bijan Parsia, Jon Parkinson, Norman Paton, Jeff Pepper, Steve Pettifer, Rishi Ramgolam, Allan Ramsay, Alasdair Rawsthorne, Farshid Rayhan, Alan Rector, Giles Reger, Graham Riley, David Robertson, Jeremy Rodgers, Clare Roe-

buck, Jeremy Rodgers, Mauricio Jacobo Romero, Nancy Rothwell, William Rowe, Oliver Rhodes, David Rydeheard, Graham Riley, Daniella Ryding, Ulrike Sattler, Ahmed Saeed, Pejman Saeghe, Rizos Sakellariou, Pedro Sampaio, Sandra Sampaio, John Sargeant, Andrea Schalk, Viktor Schlegel, Renate Schmidt, Jonathan Shapiro, Liz Sheffield, Bushra Sikander, Lemn Sissay, Vangelis Simeonidis, Kieran Smallbone, Alastair Smith, Stian Soiland-Reyes, Irena Spasic, David Spendlove, Robert Stevens, Alan Stokes, Shoaib Sufi, James Sumner, Peter Sutton, Neil Swainston, John H. Tallis, Paul Taplin, Federico Tavella, Chris Taylor, Tom Thomson, Dave Thorne, David Toluhi, Tony Trinci, Dimitri Tsarkov, Daniele Turi, Jake Vasilakes, Laura Vasques, Delia Vazquez, Giles Velarde, Chiara Del Vescovo, Markel Vigo, Andrei Voronkov, Niels Walet, Alex Walker, Louise Walker, Dieter Wiechart, Igor Wodiany, Katy Wolstencroft, Natalie Wood, Chris Wroe, Crystal Wu, Lisheng Wu, Yifan Xu, Viktor Yarmolenko, Yeliz Yesilada, Serafeim Zanikolas, Xiao-Jun Zeng, Jun Zhao, Liping Zhao, Ning Zhang and Evgeny Zolin.

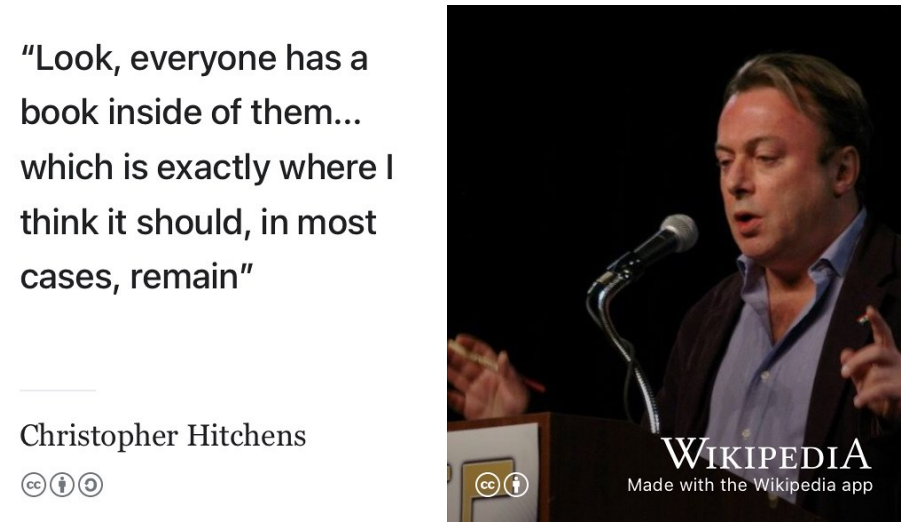


Figure 12: Optimists will tell you that “everyone has a book in them…”, but pessimists like Christopher Hitchens will add that “…in most cases that’s exactly where it should remain”. (Hitchens, 1997) Despite Hitchens amusing trademark scepticism, I am optimistic about the power of natural languages, especially in books. CC BY portrait of Christopher Hitchens by ensceptico via Wikimedia Commons w.wiki/3YK7 adapted using the Wikipedia app

So thanks academics for being even more sceptical that Christopher Hitchens, see figure 12. Thanks academics for being academic.

0.8.3.2 Thank you professional services staff

Thanks also to the superb support staff (past and present) from professional services, especially the Academic Support Office (ACSO), Student Support Office (SSO) and external affairs office in the Kilburn building. Professional services staff continue to make all the magic of teaching and learning possible: Alyx Adams, Cassie Barlow, Jennie Ball-Foster, Emma Bentley, Christine Bowers, Karen Butterworth, Chris Connolly, Ellie Crompton, Jean Davison, Gavin Donald, Miriam Cadney, Chris Calland, Ben Carter, Hannah Cousins, Holly Dewsnip, Tammy Goldfeld, Penney Gordon-Lanes, Amelia Graham, Iain Hart, Kath Hopkins, Lynn Howarth, Yvonne Hung, Susie Hymas, Radina Ivanova, Dan Jagger, Alex Jones, Jessicca Kateryniuk-Smith, Mike Keeley, Stephanie Lee, Dominic Laing, Gill Lester, Jez Lloyd, Ruth Maddocks, Cameron Macdonald, Tony McDonald, Karon Mee, Anne Milligan, Rachel Mutters, Matthew Oakley, Alyson Owens, Chris Page, Melanie Price, Chris Rhodes, Graham Richardson, Martin Ross, Julian Skyrme, Elaine Sheehan, Angela Standish, Martine Storey, Bernard Strutt, Jannine Thomas, Joseph Tirone, Daisy Towers, Anna Warburton-Ball, Richard Ward, Sarah White, Elizabeth Wilkinson, Andrew Whitmore, Lisa Wright and Mabel Yau.

And Wendy. We all miss you and love you Wendy. #JusticeForWendy Fight the Power! (Ridenhour et al., 1989)

So, thank you colleagues for being collegiate. You make the University of Manchester an enjoyable place to work.

0.8.4 Thank you scientists

Beyond Manchester there is a wider academic community of scientists, engineers and scholars that have influenced this guidebook:

- Thanks to Sally Fincher and Janet Finlay whose report Computing Graduate Employability: Sharing Practice (Fincher and Finlay, 2016) has had a big influence on this guidebook.
- Thanks to Steven Bradley, Quintin Cutts and Jane Waite for helping me setup and run SIGCSE journal club. Thanks to all the journal clubbers too, both regular and irregular including Brett Becker, Ceredig Cattanach-Chell, James Davenport, Rodrigo Ferreira, Colin Johnson, Nicola Looker, Jim Paterson, Sue Sentance, David Sutton and Moshe Vardi. Many of our journal club conversations have fed directly into the content of this guidebook
- Thanks to David Malan (@malan) for CS50 which continues to be an inspiration to me and many others. (Malan, 2010, 2020, 2021) Thanks to Cristian Bodnar for inviting David to run CS50 in Manchester in 2017 which was a great introduction to David's work (Malan, 2017)

- Thanks to Laurie Santos (@lauriesantos), for *The Science of Well-being* (TSOWB) (Santos, 2021) which was been a significant influence on this book had a gradual but dramatic effect on my personal and professional life. I've tried to distill some of the ideas into chapter 3 on *Nurturing your future*
- Thanks to Hadley Wickham (@hadley) and Garrett Grolemund (@garrettgman) for *R for Data Science* (Wickham and Grolemund, 2017) which helped me get started with R and bookdown. If you're reading this page in some kind of web browser, the stylesheet used here is re-used from r4ds.had.co.nz
- Thanks to Athene Donald and Stephen Curry at Occams Typewriter for writing stuff that has entertained and inspired me
- Thanks to Jonathan Black (@JonathanPBlack) for his book *Where am I Going, Can I Have a Map?*, his *Financial Times* columns and videos. (Black, 2017, 2021)
- Thanks to David Alan Walker for his book *Energy, Plants & Man* which inspired the conversations and pictures idea behind this book. (Walker and Walker, 1992)

So thanks scientists (and engineers) for being scientific.

0.8.5 Thank you Bath

As a graduate of the Postgraduate Certificate in Education (PGCE) in Science at the University of Bath (graduated 2011), I have been heavily influenced by the fantastic work of PGCE science course leaders Caroline Padley (Physics), Steve Cooper (Chemistry), Malcolm Ingram (Biology) and fellow students on the course.

Thanks Bath for the initial teacher training (ITT), the medicinal Aquae Sulis and the beautiful Cotswolds Area of Outstanding Natural Beauty (AONB).

0.8.6 Thank you Shaftesbury

Thanks to Stuart Ferguson, David Booth, Chris Almond, David Ball, Caroline Dallimore, Mr Travers, Caroline Moss and all the other staff and students at Shaftesbury School who hosted my first PGCE teaching placement. Thanks also to my fellow Shaftesbury/Bath trainees Katharine Platt, Harriet Edwards, Vicky Dury and Joan Shaw for sharing their hard won knowledge through peer learning. Thanks Joan for keeping me awake on the long and winding west country roads to and from deepest darkest Dorset. Thanks for sharing the heavy burden of driving too.

So thanks Shaftesbury for lessons on top of Gold Hill and the Hovis Advert, one of Britain's best-loved adverts. (Scott, 1974)

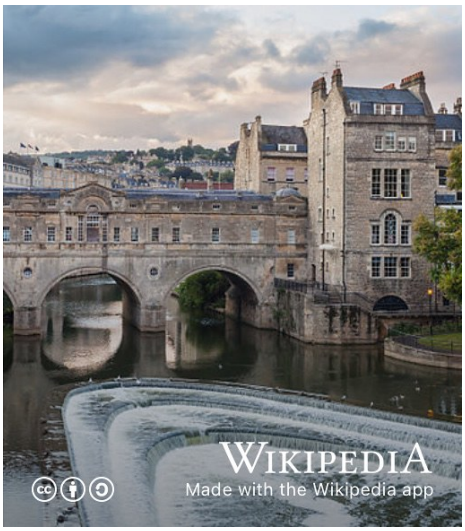


Figure 13: Named after its Roman Baths, the City of Bath is home to the University of Bath which was named *Sunday Times* University of the Year in 2011. Picture of Pulteney Bridge by Diego Delso, delso.photo, License CC-BY-SA via Wikimedia Commons at w.wiki/3VWY adapted using the Wikipedia app

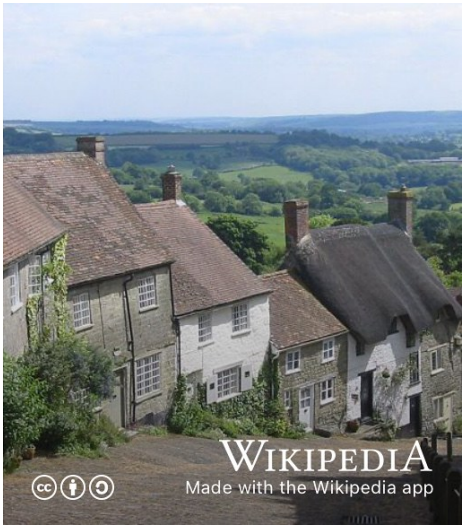


Figure 14: Shaftesbury in Dorset is the home of Gold Hill and Shaftesbury School. Image of Gold Hill by Sean Davis via Wikimedia Commons w.wiki/3LhD adapted using the Wikipedia app.

0.8.7 Thank you Swindon

Thanks to headteacher Clive Zimmerman, his team of staff, Mr M. Carter , Mr K. Thomas and the students of Greendown Community School (now Lydiard Park Academy) in Swindon, Wiltshire for hosting my second PGCE teaching placement. It was fun teaching you about waves using @Alom Shaha's jelly babies and kebab sticks shown in figure 15.

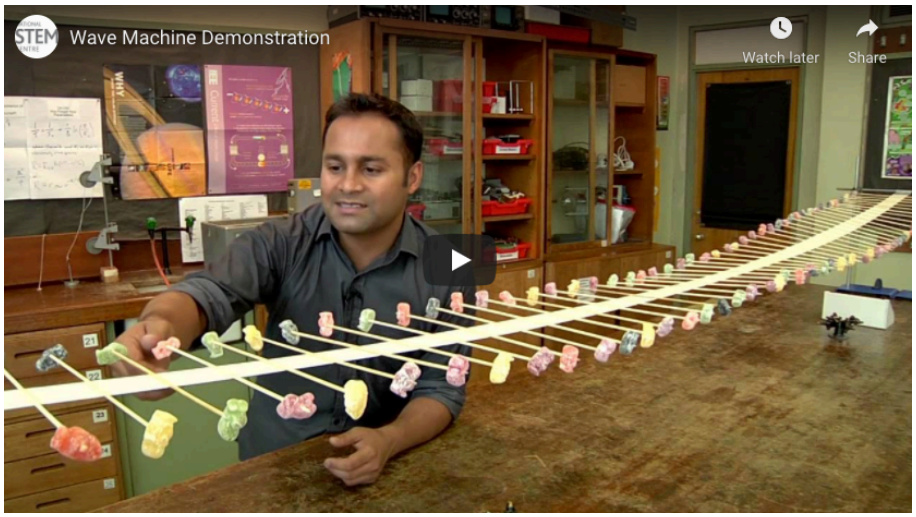


Figure 15: Alom Shaha demonstrates his awesome wave machine. Physics and jelly babies, what's not to like? (Shaha, 2014)

So thanks Swindon for being great and western and Swindon Town Football Club, the best football team in the whole of the West Country.

0.8.8 Thank you Stockport

Thanks to headteacher Joanne Meredith, her team of staff and the students at St. Annes R.C. High School, Stockport for hosting my Newly Qualified Teacher (NQT) year. Thanks to Keith Doran and other members of the *alternative staff room* for your emotional, moral and practical support throughout the year. According to the *Manchester Evening News*, St. Anne's is "the forgotten school" (Johnson, 2020; Gill and Statham, 2021), see figure 16, but I'll never forget you or the lessons you taught me.

So thanks Stockport for being Stockport, the magnificent Stockport Viaduct and for The Hatters! It's all that matters, Stockport Hatters.

Figure 16: Good governance is crucial to good schools. Many forgotten schools like St. Anne's R.C. High School, and the thousands of children in the UK they educate every year, need help from skilled people like you on their governing boards. Why not serve your local community as a "critical friend" on the governing board of a school? All ages are welcome, but especially younger governors, see where are all the young school governors? (Tickle, 2015) Take a look at governorsforschools.org.uk. Fair use image via Wikimedia Commons w.wiki/3Swt adapted using the Wikipedia app

0.8.9 Thank you schools

Thanks to all the schools who interviewed (but rejected me) for my Newly Qualified Teacher (NQT) year. Doing interview lessons, meeting your students and your senior leadership teams was a gruelling but fascinating magical mystery tour of the UK education system, both public and private. Although unsuccessful, these interviews were very productive failures:

- Wrightlington School, Radstock, Somerset, see their amazing Orchid project wsbeorchids.org run by Simon Pugh-Jones
- The Cooper School, Bicester, Oxfordshire, see their teacher in my pocket project
- St John's Marlborough, Wiltshire - not to be confused its posher and more famous next door neighbour Marlborough College
- Oasis Academy, Brislington, Bristol
- Redland Green School, Redland, Bristol
- The John of Gaunt School, Trowbridge, Wiltshire
- Didcot Girls' School, Didcot, Oxfordshire
- Cheltenham Ladies' College, Cheltenham, Gloucestershire[2]
- Blackburn College, Lancashire "I read the news today, oh boy! Four thousand holes in Blackburn, Lancashire" (Lennon and McCartney, 1967)

So thanks schools, for schooling.

0.8.10 Thank you Manchester

Thanks to Greater Mancunians Paul Bason, Martin Bryant, Darren Dancey, Shaun Fensom, Tony Foggett, Katie Gallagher, Emma Grant, David Haikney, Ross Keeping, Tony McGrath, Chris Marsh, Geraint North, Martyn Spink, Julian Tait, Rachel Thompson, Wesley Verne, Paul Vlissidis and Travis Walton for friendly Northern support and advice.

So thanks Manchester for being Mancunian.[3]

0.8.11 Thank you Moravians

Thanks to Thsespal Kundan, Principal of the Moravian Institute in Rajpur, Dehradun, Uttar Pradesh, India for hosting me and my friend Doug fresh out of high school on a gap year. We learned loads as visiting supply teachers of English and Mathematics, thanks to an introduction from a mutual contact Angus

[2] As a newly trained Jedi knight, freshly armed with a PGCE, I was anxious for my first teaching job and momentarily considered using my pedagogical powers on the "dark side" of the force: private education. (Green and Kynaston, 2019) Forgive me for I have sinned!

[3] Manc-you very much. Sorry, can't resist. I'll get my coat…

Barker. Thanks also to the Moravians in Manchester at Fairfield High School for Girls for hosting undergraduate students as part of coding their future.

Figure 17: The Moravian Institute lies in the foothills of the Himalayas between Dehradun in the Doon Valley and the hill station of Mussoorie. Situated between the Yamuna and Ganges, the institute was founded in 1963 by the late Reverend Eliyah Thsetsan Phuntsog in Ladakh, Jammu & Kashmir state to provide education for Tibetan refugees fleeing from their homeland across the Himalayas.

So thanks Moravians (and Angus) for life changing and formative experiences.

0.8.12 Thank you influencers

Some of the most important influences on this guidebook are people I've only met very briefly, virtually or not at all (yet).

- Thanks to Gayle Laakman McDowell (@gayle), for her cracking series of books (McDowell, 2014, 2015; McDowell and Bavaro, 2013; Bavaro and McDowell, 2021) which have been very useful resources both for students I've worked with and me personally
- Thanks to Yihui Xie (@yihui) and contributors to bookdown.org, the software used to produce this book alongwith the comprehensive and well-written documentation on using it. [Xie (2017); Xie (2015); Xie et al. (2020);]
- Thanks to Bronnie Ware for her *The Top Five Regrets of the Dying* (Ware, 2011) which helped me to re-align my priorities when they were all out of kilter
- Thanks to blogging blokes on the interwebs whose words I've enjoyed reading. Your writing provides an existence proof that engineers and scientists should also be good communicators:
  - Tim Bray at ongoing
  - Paul Downey at whatfettle.com
  - Paul Graham at paulgraham.com
  - Peter Norvig at norvig.com

  - Neil Saunders at What you're doing is rather desperate
  - Greg Wilson at third-bit.com
- Thanks to Sophie Milliken for *From Learner to Earner: A recruitment insider's guide for students wanting to achieve graduate job success* (Milliken, 2019) which draws useful distinctions between graduate jobs and graduate schemes

So, thanks influencers for being influential.

0.8.13 Thank you githubbers

Thanks to everyone who has contributed via github. I will credit *any* github contributors here, small or large. Even the typos, it all counts. You can easily add yourself to this roll call (see section 0.7) by correcting my delibreate mitsakes.

Keith Mitchell (@apiadventures), Jan Machacek (@janm399), Zee Somji (@ezeethg), Tsvetankov (@Tsvetankov), teobalmos (@teobalmos), Richard Gourley (@richardgourley), captaman0 (@captaman0), iliketohelp (@iliketohelp)

So, thanks githubbers for cloning, forking, pulling, adding, committing and pushing.

0.8.14 Thank you Wikipedians

Thanks to all the thousands of editors and engineers that make Wikipedia one of the greatest communities on the internet, see figure 18.

Special thanks to English speaking Wikipedians Abd Alsattar Ardati, Caroline Ball, Alex Bateman, Dan Brickley, John Byrne, Lucy Crompton-Reid, Daria Cybulska, Andrew Davidson, Paul Gardner, Madeleine Goodall, Aaron Halfaker, Melissa Highton, Eoin Houston, Darren Logan, Magnus Manske, Andy Mabbett, Charles Matthews, Ewan McAndrew, Joshua Minor, Peter Murray-Rust, Richard Nevell, Frank Norman, Rod Page, Bhavesh Patel, Mike Peel, Martin Poulter, Joseph Reagle, Dario Taraborelli, Sara Thomas, Denny Vrandečić, Ian Watt, Alice White, Jessica Wade, Taha Yasseri for insights, inspiration, support, software, data, pictures and guidance. Thanks also for educating me on issues of equality, diversity and inclusion, especially gender and race.

So, thanks Wikipedians for being Wikipedia.

0.8.15 Thank you Bryan

Many of the illustrations for this book have been drawn by the very talented Bryan Mathers @BryanMMathers shown in figure 19.

Figure 18: A small fraction of the Wikipedia community that works to give free access to the sum of all human knowledge to every single person on the planet. CC BY-SA picture of some Wikipedians gathered at the annual Wikimania conference in 2012, adapted from an original by Helpameout on Wikimedia Commons w.wiki/3YLJ using the Wikipedia app

Bryan is an artist, visual thinker and entrepreneur, who also happens to have a Bachelors degree in Computer Science from the University of Glasgow. His combined skills in art, science and engineering made him the perfect fit for illustrating this guidebook. You can find out more about Bryan at bryanmathers.com and visualthinkery.com. I'm *sooo* glad we randomly bumped into each other at a conference shown in figure 8.9.

So, thanks Bryan for your witty illustrations, this book wouldn't be the same without your visual thinkery.

0.8.16 Thank you friends

Thanks to my friends, especially those who I've enjoyed singing, dancing and live music with. I hope we can sing and dance together to live music again before too long.

So, thank you friends for your friendship.

0.8.17 Thank you family

To my family: mum, dad, wife, son, brother, sister, , and extended family: I'm lucky to have been taught by you and that you've always been

Figure 19: Bryan Mathers Self portrait by Visual Thinkery is licensed under CC-BY-ND

there when I needed you.

So, thanks to all my family for your unconditional love. Σ .

0.9 About me

Hello, my name is Duncan Hull and I wrote this guidebook for undergraduate and postgraduate students as part of my job at the University of Manchester where I'm a lecturer (Assistant Professor) in the Department of Computer Science.

So what's *my* story? Like many people, my path has been what Helen Tupper and Sarah Ellis call a "squiggly career" rather than classic linear one. (Tupper and Ellis, 2020, 2021) I've been gainfully employed as a paperboy, supermarket cashier, shelf stacker, sausage factory worker, pork pie filler, chef, dogsbody, field assistant, database administrator, deli counter server, consultant, match-day steward, envelope stuffer, high school teacher, postdoc, research scientist, software engineer, lecturer, external examiner, tutor and scholar.

I've done a range of voluntary work too, serving as a competition judge, fundraiser, code club & coderdojo leader, rabble rouser, digital council member, school governor, curator, librarian, beer drinker, wikipedia trainer, journal clubber and editor. But as Ronnie Lane and Ronnie Wood (figure 20) once said:

"I wish that I knew what I know now, when I was younger."

Ooh La La (Faces song)

Figure 20: Hindsight is a great teacher. I wish that I knew what I know now, when I was younger, see Ooh La La (Lane and Wood, 1973) I've written some of what I know now in this guidebook, I hope you find it useful.

This guidebook documents some of what I know now, that I wish I'd known, when I was younger. If you're starting your career, I hope you find these insights useful. I've sat on both sides of the interview table, as interviewer and interviewee. I have had some spectacular failures, alongside some modest successes, and have included personal stories where they are relevant.

Most of what I have learned about employment comes from listening to, and watching students interact with employers as they take the first tentative steps in their careers. I've documented some of what they taught me, so reading this book may help you learn from some of their successes and failures.

0.10 Legal stuff

I am not a lawyer (IANAL) but any opinions expressed in this guidebook are my own and not representative of my current employer, the University of Manchester. This guidebook does not therefore, represent University policy.

0.10.1 Licensing

The *text* of this guidebook is published under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 License (CC-BY-NC-ND) license. This means you can copy and redistribute the written material provided that:

- You provide full attribution by linking directly to the original source
- You do not use the material for commercial purposes
- You do not make any derivative works

See the full license (CC-BY-NC-ND) for details.

The *images* used in this guidebook are published under different licenses, depending on their source. For example, Bryan Mathers illustrations are licensed CC-BY-ND, see figure 21. Other images have different licences, for example, images from Wikimedia Commons (commons.wikimedia.org) are published under CC-BY or CC-BY-SA, fair use or public domain. Each figure caption gives details of the pictures licence.

0.10.2 Your privacy

This site is hosted on netlify.com, see the netlify privacy policy. This site also uses netlify analytics (server side) and Google Analytics (client side) to understand our audience better. Both comply with the General Data Protection Regulation (GDPR). If you want to, you can opt out using the Google Analytics Opt-out Browser Add-on.

Figure 21: Images in this guidebook are published under different licences, see each figures caption for details. Bryan Mathers illustrations are licensed CC-BY-ND, which means you should link to the original artwork, the creator profile and the licence terms. CC attribution artwork by Visual Thinkery is licenced under CC-BY-ND

Some of these services use cookies. These can be disabled in your browser, see allaboutcookies.org/manage-cookies

So now that we've dispensed with the formalities, let's look at why should you bother reading this guidebook in the first place.

Part I

DESIGNING

Chapter 1

Rebooting your future

The first part of this book is about designing your future. So before we get started, we need to reboot and tackle a fundamental design issue. Why the hell would you want to bother with your future? Why should read this guidebook when you have so many other things to do right now? So:

- You are a busy person, YES!
- Your time is a precious and finite resource, YES!
- You could be spending that precious time right now in lots of other ways, YES!
- There are mountains of self-help guides and courses already, YES!
- Do you really need *yet another* guidebook? YES!

You need this guidebook because it is different to all the other guidebooks! It will help you design, test, build, code and debug your future in computing.

Before we do that, we need to reboot. Come with me down the rabbit hole in figure 1.1 and let me explain…

1.1 What you will learn

After reading this chapter you will be able to:

- Reboot your future by
  - Setting your expectations for using this guidebook, and open some doors to your future
  - Travelling down the rabbit hole into the underworld of employment
  - Discuss some of the gaps and differences that exist between education and employment and how you can bridge them

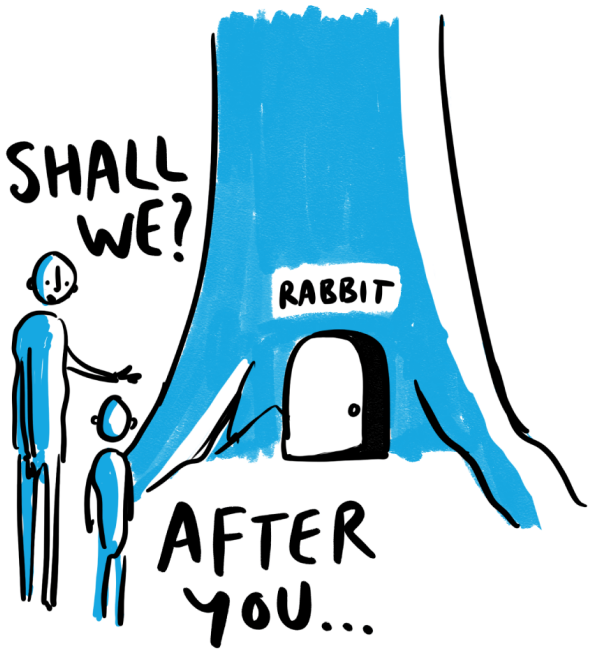


Figure 1.1: Shall we go down the rabbit hole? Rabbit Hole learning by Visual Thinkery is licensed under CC-BY-ND

1.2 Let's go down the rabbit hole

In the novel *Alice's Adventures in Wonderland* (Caroll, 1865), the protagonist Alice follows a white rabbit down a hole. What she discovers is a strange underground world populated by weird and wonderful characters. The world of work can sometimes be a mysterious underworld where you adventure in wonderland accompanied by colourful characters.

You will spend lots of time in this wonderland, potentially as much as 80,000 hours of your life. (Investor, 2019, 2020) So join me down the rabbit hole, it's fun (honest), and sooner or later you'll have to come down here anyway. So open up the door to the new possibilities in your future.

1.3 Opening your future

Studying at University opens new doors to your future, some of which will take you down rabbit holes. As the poet Lemn Sissay puts it (figure 1.2):

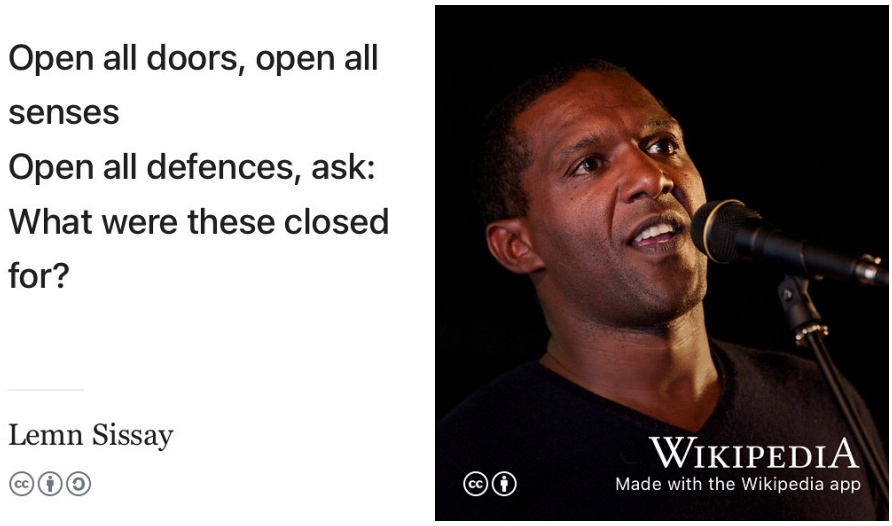


Figure 1.2: Open all doors, open all senses, open all defences, ask: What were these closed for? From *Inspire and be Inspired* by Lemn Sissay whose poetry is even better when you hear it, rather than just read it. (Sissay, 2015) Portrait of Sissay speaking in 2010 by Philosophy Football via Wikimedia Commons w.wiki/3VYT adapted using the Wikipedia app

1.4 Maximising your future

As well as opening your future, studying at University about *investing* in your future. You're spending lots of your time and money at University. Hopefully, you've picked a subject that stimulates and challenges you intellectually while allowing you to find and develop your unique talents. But there's another reason that you probably chose to study at University and that was to improve your job prospects. This guidebook will:

1. Help you maximise the return on the substantial investment of time and money (ROI) you've put into your study
2. Give you an overview of important professional issues that are sometimes neglected or sidelined in University curricula
3. Highlight and review essential resources beyond this guidebook that will help with the above

All of the resources that can help you are scattered around in lots of different places. There are books, there are videos, there are podcasts, there are websites and jobs boards. There are online courses, blogs, social media, newspaper columns, journal articles, marketing material and many other good resources. It is overwhelming.

1.5 Your future is your responsibility

When Andy Stanford-Clark started working at IBM, fresh out of University, his boss gave him the career advice shown in figure 1.3:

Andy is now Chief Technology Officer (CTO) and IBM Master inventor in the UK so it was probably good advice. Another, slightly more positive way of putting the advice is, the person who cares *most* about your career is you. So while there are people who can help design and build your future, ultimately it is **YOU** who has to take responsibility for the implementation (if you like, the `code`). The sooner you get coding the better.

At University, there are lots of people can help design and build your future: peers, friends, academic staff, your careers service, employers and your wider social and professional networks but ultimately it is *your* responsibility to sort out whatever comes next. That might sound obvious but don't wait for somebody else to do it for you, because it probably won't happen.

1.6 Your degree is not enough

You've worked incredibly hard to get the grades you needed to get into University. You've spent (or are spending) a significant amount of time and money

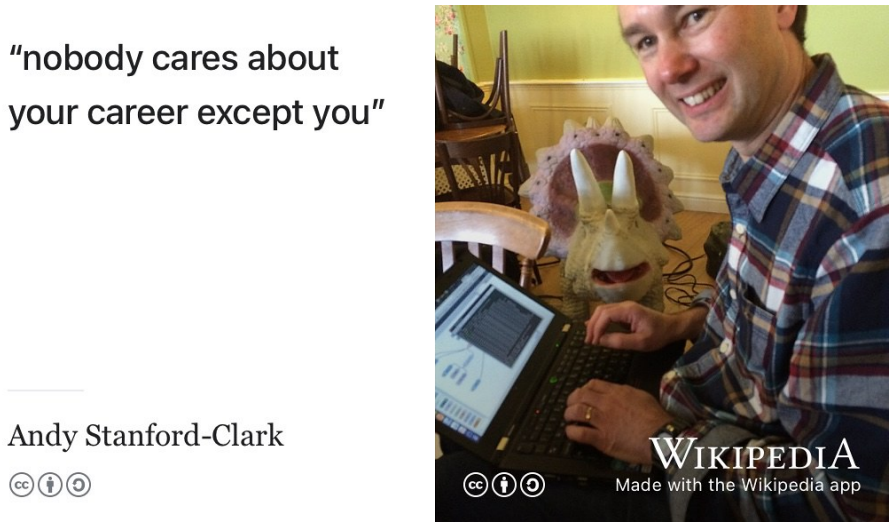


Figure 1.3: Nobody cares about your career except you. Quote via Andy Stanford-Clark (Stanford-Clark, 2019) to an unattributed IBM boss. Image of Andy by Gizmo~enwiki via Wikimedia Commons w.wiki/3TSn adapted using the Wikipedia app

studying whatever it is you are studying your chosen discipline.

Under these circumstances, you might be tempted to believe that the world owes you something in return for all your hard work. Unfortunately that's not the case.

At some point during or after your study, you might find yourself applying for a graduate job or graduate scheme. EVERYONE applying for these opportunities will have a degree or be rapidly on their way to getting one. So having a degree isn't going to set you apart much from your competition. Even having a first class degree (Coughlan, 2019a; Borrett, 2019) may not distinguish you that much your competitors. Some employers would rather not know (or don't care) what University you went to, so your education might not make you stand out as much as you might like anyway. (Agnew, 2016; Garner, 2015)

What **will** distinguish you from your competitors will be your experience (see chapter 5), your projects (see section 7.5.4), your communication skills (see chapter 4) and any awards or honours you might have picked up along the way. If you think that your degree will be enough to get you the job you want, bear in mind that:

1. There are more and more graduates, the UK for example recently passed the milestone of 50% of young people going into higher education. This compares to just 15% of over 18s who stayed in higher education in 1980 (Coughlan, 2019b)

2. The increase in the number of graduate schemes and graduate jobs has not kept pace with this growth in graduates which means that each graduate job or graduate scheme has more and more graduates applying for it
3. There are lots of graduates in your discipline. In the UK, for example, around 9,000 students graduate every year in Computer Science. If you're studying in the UK, what makes you different from the other 8,999 computer scientists graduating in your year?

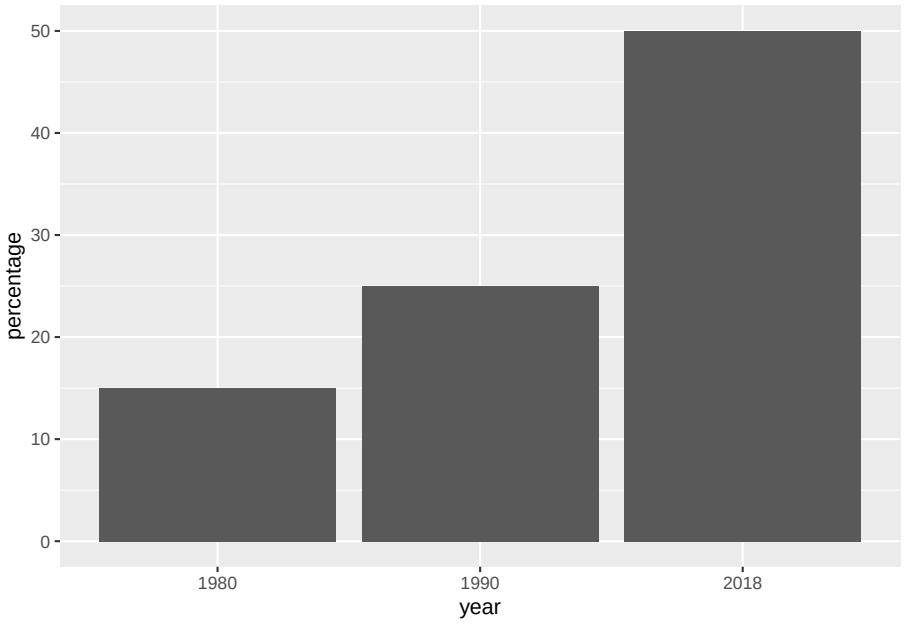


Figure 1.4: Percentage of young people in the UK going into higher education between 1980 and 2018. Over the last forty years, the proportion of young people going into higher education has more than doubled from 15% in 1980 to over 50% in 2018. Data taken from BBC news article on the symbolic target of 50% at university reached (Coughlan, 2019b)

> Computing is one of the largest subject areas in UK higher education, and is taught in almost every institution, graduating around 9,000 students every year –Sally Fincher (Fincher and Finlay, 2016)

Now, don't be disillusioned by the statistics because a degree can open doors to many careers in computing. What the data in Figure 1.4 show is that you'll need to look beyond your formal education to distinguish yourself from your competition. Your degree can certainly help you start a career, but it is typically not enough by itself.

1.7 It's too late when you graduate

You might be tempted to postpone making difficult career decisions. I'll do it tomorrow. I'll do it next week. I'll do it next year. I'll finish this assignment. I'll finish this exam. I'll finish this semester. I'll finish my degree first, see figure 1.5. Procrastination is a part of the human condition. Software engineer Paul Graham calls this good and bad procrastination (Graham, 2005).

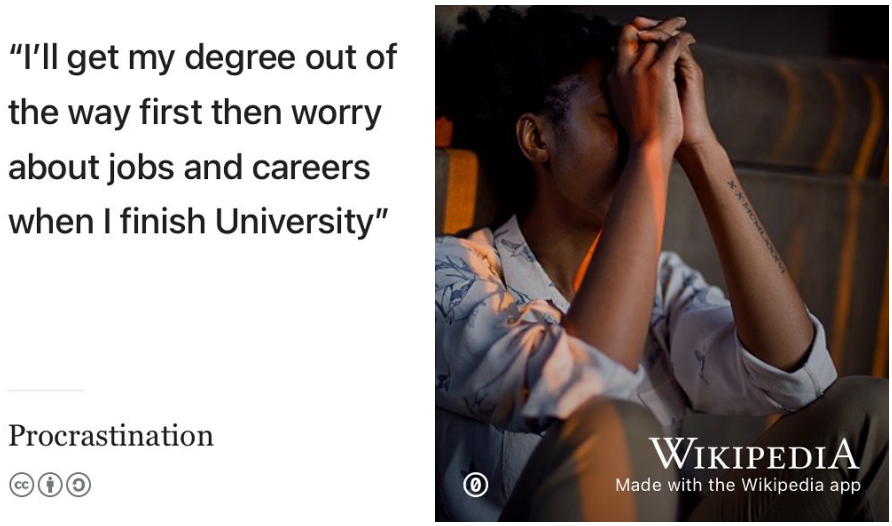


Figure 1.5: Procrastination: the attitude of "I'll get my degree out of the way first then worry about jobs and careers when I finish University" is bad procrastination. It's too late when you graduate to start thinking about what might come next (Graham, 2005) Stresssed procrastinator picture by MismibaTinashe-Madando on Wikimedia Commons w.wiki/3TXo

Postponing decisions about your career is usually bad procrastination. It probably doesn't help that many of issues described and discussed in this book are typically not closely integrated into the curriculum in Higher Education. You'll often find them on the edges, or completely outside of, standard University curricula. Broadly speaking, the professional issues described in this book are usually covered by pastoral support systems, counselling services, careers services, trade organisations, professional bodies, student unions and their societies.

Despite being sidelined, these issues matter and it is in your own self interests to start thinking about them right now. According to recent estimates by *Investors in People*, the average person spends **80,000 hours** working during their lifetime. (Investor, 2020) So, *whatever* you end up doing after University, you'll be spending a lot of time doing it. Difficult decisions often get sidelined but it is never too early to start thinking about them and doing something.

If you want to work for a big name like those in section 5.3.1 or 8.2.1, many of the larger graduate employers expect you to have *some* experience (see chapter 5) *before* you graduate. A large chunk of vacancies on graduate schemes are filled people who have already been employed as interns or placement students within that (or another) organisation. So the sooner you start investigating employers by getting some experience the better decisions you'll be able to make about what comes next. It's (usually) too late when you graduate.

That doesn't mean you have to know EXACTLY what you want to do when you finish. Lots of students don't and I certainly didn't when I graduated. I'd done a gap year teaching in India, two summer internships (in Sweden and the United States) and a year-in-industry in the UK and I *still* graduated with **no clue** as to what I wanted to do next! The important thing is that you make a start, and sometimes knowing what you **don't** want to do is just as valuable as knowing what you want to do.

Computer scientists call this problem "search space reduction", (Ferrari et al., 2008) because you have a feasible region of future possibilities and you need to narrow down the candidates. You could think of coding your future as an optimisation problem. Start optimising now because it's too late when you graduate.

1.8 Yes, this WILL be on the exam

Students love to ask their teachers "*will this be on the exam*"? The short answer is **YES** (and **NO**)! Yes, this will be on the exam, but NO the exam won't be set by your University. Unlike other courses you've done, the examinations for this course aren't set by your University but by employers. Roughly speaking, there are three kinds of examinations that you'll need to get good at, shown in Table 1.1

Table 1.1: Examining your future: The "exams" used by employers, what gets assessed and the grades you can get. For written "exams" see chapters 4 and 7, for speaking "exams" see chapter 10 and for your employee "exams" see chapter 11.

Examination	What examiners are assessing	Grade
CV, application form covering letter	• Should we invite you to interview ? • Can you communicate well in writing? • What experience do you have?	pass/fail

Examination	What examiners are assessing	Grade
Interview	• Should we offer you a job? • Can you communicate well verbally? • Can you communicate well nonverbally?	pass/fail
Employee performance	• Should we promote you? • Should we give you a pay rise? • Should we extend your contract?	pass/fail

So, *yes*, this will be on the exam, but *no*, the exams are obviously not set, administered, invigilated and marked by academics at your University. The exams are set by employers and the results are **brutally binary**:

- **PASS**: you've got the interview, job or promotion or…
- **FAIL**: none of the above. Next!

One of the challenging things about employers exams are, they typically do not have the bandwidth to give applicants useful feedback, other than a simple pass or fail. When it comes to job applications software engineer Gayle Laakmann McDowell calls this the "black hole". The gravitational force of employers black holes is so strong that no CV or Résumé can escape, we'll say more about this in chapter 7 on debugging your future.

It's a similar story with interviews, if you fluffed and interview question or came across badly, it can be really difficult to find out from the employer what you did wrong.

1.9 Practicing your future

There are practical exercises, for you to get your hands dirty with. Each chapter incorporates activities including individual exercises, group exercises, quizzes and points for wider discussion. You'll get a lot more out of this guidebook by doing the activities, rather than just reading it.

1.10 Navigating your future

There are **lots** of resources out there that offer self-help, career advice and techniques for self-improvement. It can be hard to know where to start, or even how to find your way around the mountains of advice.

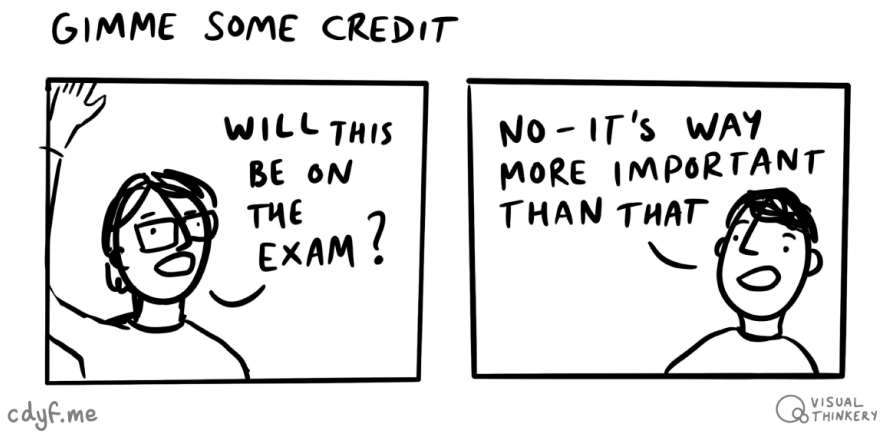


Figure 1.6: So *no* this will not be on the exam set by University, but *yes* it will be on the exams set by employers. Some of the most important exams you sit at (and after) University are set by employers. This guidebook will help you prepare for those exams and increase your chances of passing them. Gimme some credit figure by Visual Thinkery is licensed under CC-BY-ND

Figure 1.7: There are tonnes of resources out there offering advice on a huge range of professional issues. You can't read them all, but this guide aims to help you navigate the resources that will be most use to you

Lots of professional advice is readily available, but how will you navigate it? This book signposts you to what I think are the most important resources, each chapter has a signposts section, and they are all gathered together in the signpost at the end alongside everything (yes, EVERYTHING!) that this guidebook cites in the references, chapter 20.

1.11 Crediting your future

Get credit for your contributions. As well as being openly accessible on the web, this book is open source too. What this means is, you can contribute in several ways described in section 0.7. All the written content for this guidebook is licensed under CC-BY-NC-ND, see the license in section 0.10.1.

1.12 Your future is different

I wrote this guidebook because I needed a resource for students to help them design, build, test, hack and debug their futures. I needed a book that could help students compete for jobs while at University, or shortly after graduating. I could not find anything suitable that met all the requirements of the students I was teaching. So I wrote this one which contains some new material and recommends the best resources if you want to know more. These are found in the signposts sections of each chapter.

This book aims to combine these perspectives and to be different from existing resources in the following ways:

1.12.1 Your future is signposted

Some career resources claim (or imply) that they are the *all you will need* to solve a particular problem or worse: solve *all of your problems*! Just buy this book, do this course, watch this video, listen to this podcast and all your problems will go away! Rather than continue this trend, this book **signposts** some of the most useful resources, see figure 1.8.

Scientists call signposting **citation**, so I've signposted and cited sources in this guidebook so that you can :

1. Follow them if the destinations are interesting or useful
2. Check and verify any facts and claims I make in this book for yourself

While this guidebook cites lots of resources, some of them are more important than others. Each chapter summarises these in a signposts section. You'll find

Signposts allow users to navigate on long journeys, sometimes over difficult terrrain

Traffic sign

Figure 1.8: Wondering which way to go at the traffic sign? I've signposted the resources that will help you navigate the start of your professional journey. Which route will you take? Picture of a signpost in the Åland Islands, Finland by Sal via Wikimedia Commons w.wiki/3Xop adapted using the Wikipedia app

everything else in the references, chapter 20. University and public libraries may also have physical and electronic copies of some of the books listed here.

We're not suggesting that you read *all* these books right now, but that if a particular chapter has piqued your interest, these signposts are good places to keep going, if you haven't already read them. I hope you'll find these signposts handy for navigating the mountains of advice. Not all who wander are lost.

1.12.2 Your future is guided

This guidebook to your future accompanies a course that has been co-designed by students for students, with input from academics and employers. It unites several disparate themes into one coherent story, from fundamental questions about identity and wellbeing through to more applied and practical advice on job hunting, career progression and life after University. Resources that do this are typically scattered around in many different places. There is usually no narrative to tie them all together to help students navigate the mountains of advice as embark on the first stages of their careers.

Although this is a course guidebook used in the second year undergraduate teaching, you don't need to be enrolled on the course to benefit from reading it, watching the videos and doing the exercises and coding challenges.

1.12.3 Your future is constantly updated

You are reading the alpha version, the Minimum Viable Product (MVP) of this guidebook, last updated on 06 July, 2021. That's software engineer talk for saying it isn't finished yet. Subsequent versions, will be continuously and iteratively released on a daily and weekly basis. They will include:

- More quizzes for better interactivity
- More videos on the Coding your Future YouTube channel
- Audio interviews with Students in the *Coding your Future podcast* in chapter 17
- More illustrations throughout the book
- Improved content, finish incomplete chapters
- Fix bugs and typos
- Your suggestions for improvements and corrections, via github etc see section 0.7

I'm taking a release early, release often (Raymond, 1999) approach to publishing this guidebook, you could call it agile book development, see figure 1.9 (Schuh, 2004)

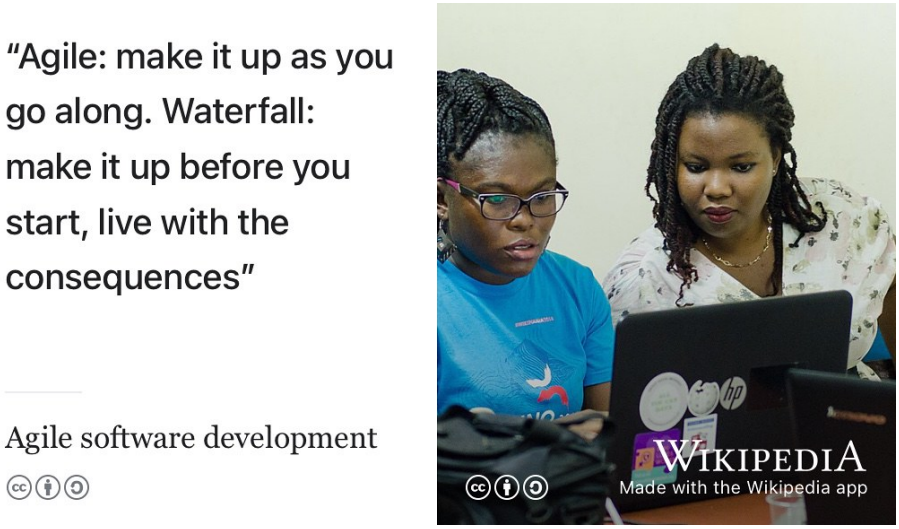


Figure 1.9: Agile software developers make it up as they go along, whereas waterfalling software developers make it all up at the beginning and then live with the consequences. It's the same with natural language engineering (books). I'm making it up as I go along, using agile book development methods. Women who code image by Justice Okai Allotey via Wikimedia Commons w.wiki/3Xnk adapted using the Wikipedia app

1.12.4 Your future is personal

A lot of scientific and technical writing is written in the third person or passive voice, which is fine for academic writing, but can alienate readers. I have opted to use first person narrative where possible as it is shorter, and hopefully more engaging for you to read. (Poundstone, 2013) Where relevant, I've told personal stories to illustrate key points.

1.12.5 Your future has no paywall

You don't need to pay anything to read this book online because its not hiding behind a paywall, see the license terms (CC-BY-NC-ND) in section 0.10.1. Publishing this guidebook online makes it findable and accessible, something that isn't true of lots of knowledge locked up inside other books.

Because this guidebook is online, it is searchable, browsable and linkable. You can link to whatever level you like, top level, chapter level and to every section and subsection level. Everything important has a Uniform Resource Locator (URL).

1.12.6 Your future has audio & video

This book is not just words and pictures, but includes audio and video as well, especially:

1. videos produced by third parties that are worth watching
2. audio produced by third parties that are worth listening to, either individual episodes or whole series
3. short videos produced by me, which augment the written content of this book, see the Coding your Future YouTube channel
4. the coding your future podcast which interviews undergraduate students

1.13 Engaging with your future

I've tried to make the content of this book as engaging as possible by including pictures and conversations. *Your future* is deliberately playful and light-hearted. If you think this guidebook can be improved, let me know via the mechanisms described in section 0.7. I always welcome constructive feedback, especially when it comes via a pull request. Engage, see figure 1.10.

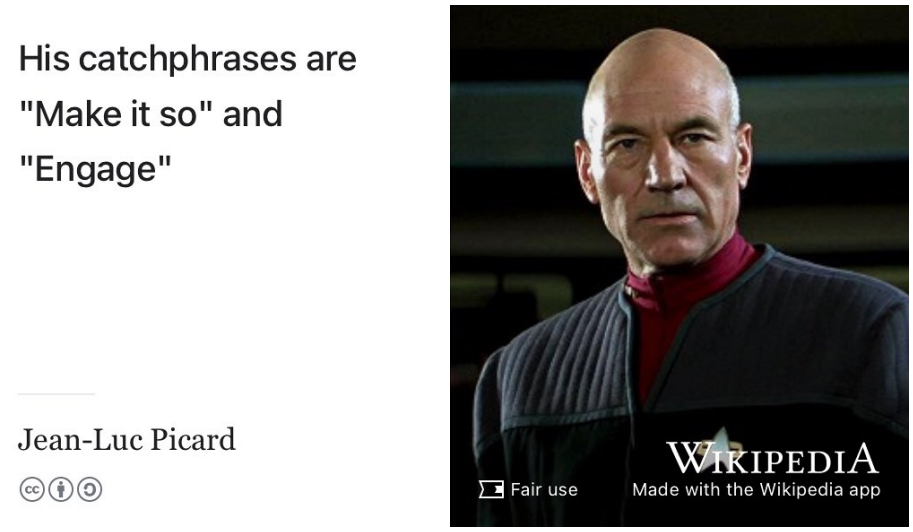


Figure 1.10: This is Captain Jean-Luc Picard of the Starship *Enterprise*. Engage! Fair use image of actor Patrick Stewart performing in *Star Trek* adapted using the Wikipedia app. Make it so.

1.14 Signposting your future

Each chapter in this book has a signposts section, highlighting key reading, watching or listening you could do next. This chapter has addressed the question of **why should you bother coding your future**? The answer is that your future is your responsibility and no-one elses. There are lots of people can help shape your future, but none more than yourself. Software engineer Robert C. Martin argues this point in his book *The Clean Coder: A Code of Conduct for Professional Programmers*. (Martin, 2011)

What's good about *The Clean Coder* is that it is short (only 200 pages), well written and to the point. The main part of the book covers professional issues in software engineering, some of which I discuss further in chapter 11, *surviving your future*.

1.15 Summarising your future

If all that was too long, didn't read (TL;DR) for you, then you'll be relieved to hear that each chapter has a TL;DR (executive) summary.

The TL;DR for this chapter is, you should read this guidebook because it is different to all the other guidebooks. It will help you take responsibility for maximising your future. No-one else is going to do this for you. Your degree

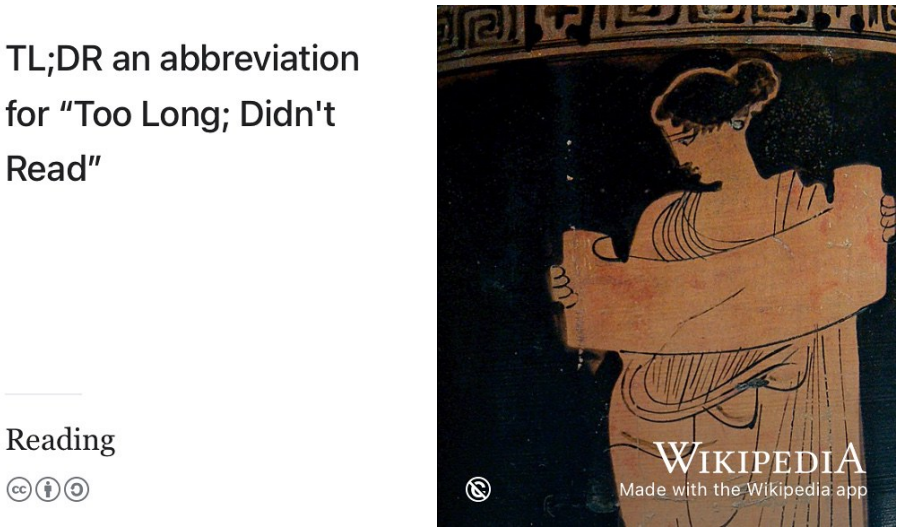


Figure 1.11: There's always more stuff you should be reading. If this guidebook is a bit **Too Long, Didn't Read** (TL;DR) then each chapter has a brief summary at the end. Public domain picture of ancient greek muse reading a scroll that's probably too long via Wikimedia Commons w.wiki/3Xoh adapted using the Wikipedia app

will help open up future options, but it is not enough by itself so you'll need to maximise the return on your investment. Don't procrastinate because it's too late when you graduate and *YES* this will be on the exam (set by future employers). This guidebook has signposts to help you navigate, design, build, test, debug and code your future in computing.

It looks like the reboot has finished, so we're ready to go. In the next chapter, you will reflect on who you are. What's your story, coding glory?

Chapter 2

Knowing your future

Hello, who are you? What's your story? What are you good at, what do you like doing and what do you value? What are your hopes and dreams for the future? Tell me about your education and who you are. What unique talents are you finding and developing during your education? How are you striving to become the best possible version of you? Having good self knowledge will help you answer these big questions, which are important for your future. Knowing your future depends on knowing who you are now.

2.1 What you will learn

Reading this chapter and doing the activities will help you to

- Improve your self-awareness, knowing yourself better will help you to know your future more clearly
- Describe your story so far in terms of head, heart and hands:
  - What you know: what is in your **head**
  - Your values: what is in your **heart**
  - Things you have you done **hands**
- Identify your protected characteristics
- Check and be grateful for any privileges that you may have

2.2 What's your story, coding glory?

We're hardwired to love storytelling because it help us understand our world, see figure 2.2. We use stories to organise and communicate, so knowing your story is a crucial part of knowing who you are. What's your story?

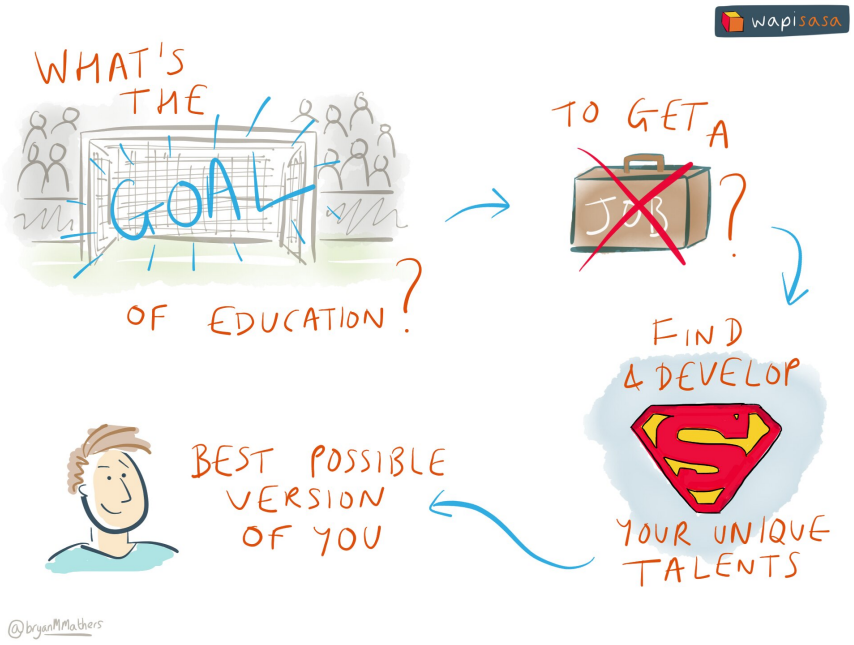


Figure 2.1: Your education is a crucial part of your story and who you are. The purpose of your education is not just to get you a job but to find and develop your unique talents. What are your unique talents? How are you developing them? Goal of Education sketch by Visual Thinkery is licensed under CC-BY-ND

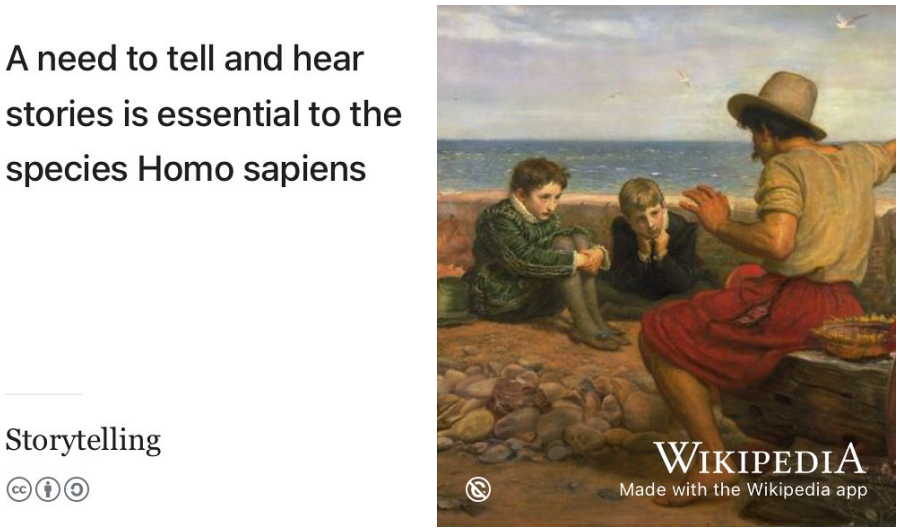


Figure 2.2: Storytelling is an ancient art and who doesn't love a good story? As a species *Homo sapiens*, we need to tell and hear stories to understand the world around us. What's your story, coding glory? Public domain image of a painting by John Everett Millais, with a seafarer telling the story of what happened out at sea, via Wikimedia Commons w.wiki/3VHM

Self-awareness, understanding who you are, is important for leading a healthy and happy life, and likely to be an important factor in your future success. One way to develop self-awareness is to think about the finer details of your story. (Box and Mocine-McQueen, 2019) How did you get here, where are you going, what has inspired you? Who is the authentic you? (Ware, 2011) What are your hopes and dreams? By starting to answer these questions you will gain a better understanding of who you are. This includes strengths, weaknesses, motivation and values. (Bolles, 2019)

Your story is probably complex but you need to know it so you can distill the details of it into *much* shorter stories on your job applications described in section 7.5.

Universities offer many opportunities for self improvement, self discovery and developing your unique skills. One way to build your self-awareness is to reflect on your knowledge, values and skills. In Waldorf education this is characterised as "head, heart and hands". (Easton, 1997)

1. **Head**: What do you *know*?
2. **Heart**: What do you *value*, what motivates you?
3. **Hands**: What can you *do*? What have you *done* so far? What will you do in the future? Your actions define your impact, see chapter 18

Answering these questions will help you understand your story.

2.3 Ikigai: What is the meaning of life?

Many of the learning outcomes described above are non-trivial. You may have good self-awareness and be able to describe aspects of who you are in a matter of minutes. Other personality traits make take longer to realise. You can develop better self-awareness by describing four attributes shown in Figure 2.3, together these are known as your ikigai ( ) or "reason for being".

- what do you love doing?
- what are you good at?
- what does the world need?
- what can you be paid for?

You'll be lucky if you can find activities at the intersection of all four sets shown Figure 2.3. In practice, you may realistically only be able to achieve one, two or three. That said, it's still a valuable exercise to think about what is in each category for you.

2.4 Self assess your ikigai

Take a sheet of paper, draw the four overlapping rings shown in Figure 2.3, and spend five to ten minutes adding things in each ring.

- What are your values?
- What motivates you?
- Are there things you like doing that you aren't particularly good at?
- Why does that make them enjoyable?

Thinking about your ikigai will clarify your knowledge of yourself. Some parts of your identity are so important that they are protected by legislation, in the UK and in other countries. The next section looks at those.

2.5 Your protected characteristics

Some of your characteristics are protected. The Equality Act of 2010[1] protects you from discrimination at work or in education, based on what are known as "protected characteristics". (UK, 2020a). This means that:

[1] http://www.legislation.gov.uk/ukpga/2010/15/contents

Figure 2.3: Reasons for being, a concept in Japanese known as *ikigai*. Image by Emmy van Deurzen on Wikimedia Commons. According to ikigai, a meaningful life combines doing four things. 1. What you are good at 2. What you love 3. What the world needs and 4. What you can get paid for. Illustration by Nimbosa derived from works in the public domain by Dennis Bodor and Emmy van Deurzen, CC BY-SA 4.0 on Wikimedia Commons w.wiki/qWT

Figure 2.4: How well do you know yourself. Know who you are sketch by Visual Thinkery is licensed under CC-BY-ND

- Your **age** should not determine how you are treated
- Your **disabilities** should not determine how you are treated
- Your **gender** should not determine how you are treated (Saini, 2018; Damore, 2017; Lewis, 2017; Bates, 2016)
- Your **gender re-assignment** should not determine how you are treated
- Your **marriage** or civil partnership should not determine how you are treated
- Your **pregnancy** and maternity should not determine how you are treated
- Your **race** (including colour, nationality, ethnic or national origin) should not determine how you are treated (Eddo-Lodge, 2017; Saini, 2019)
- Your **religion** or beliefs should not determine how you are treated
- Your **sex** should not determine how you are treated (Price, 2019)
- Your **sexual orientation** should not determine how you are treated (Britton, 2019)

2.6 Breakpoints

Let's pause here. Insert a breakpoint in your `code` and slowly step through it so we can examine the current values of your variables and parameters.

```
* PAUSE
```

This chapter has looked at some big issues around identity, by inviting you to think about some fundamental questions. Another way to think about these

questions is as coding challenges. They are non-trivial questions to answer, it might take you weeks, months or even years to answer some of them. But they are worth spending time thinking about

- What are your values?
- What makes you happy?
- What do you want to get from your time at University?
- What do you want after University?
- Where do you see yourself in `x` years time?
- What are your privileges (if any), see section 2.7.4

The signposts in the next section may help tackle some of these coding challenges.

```
* RESUME
```

2.7 Signposts from here on identity

This chapter challenges you to reflect on who you are and what you're good at. We've only scratched the surface, so if you want to dig deeper you'll find the following resources useful:

- *The Top Five Regrets of the Dying*
- *What Colour is Your Parachute?*
- *How Your Story Sets You Free*
- A range of books about privilege

2.7.1 Your dying regrets?

One of *The Top Five Regrets of the Dying* (Ware, 2011) is that people wish they'd had the courage to live a life true to themselves, and not a life that others expected of them. Figuring out exactly who your authentic self is can be challenging. Bronnie Ware's book might help, it has some very moving, personal and insightful true stories of peoples regrets that will illuminate your own values and might just change your life. The top five regrets, outlined in the book are:

1. I wish I'd had the courage to live a life true to myself, not the life others expected of me
2. I wish I hadn't worked so hard
3. I wish I'd had the courage to express my feelings
4. I wish I had stayed in touch with my friends
5. I wish that I had let myself be happier

You need to be courageous to live a regret-free life but the alternative is to die full of regret, see Bronnie's video in figure 2.5.

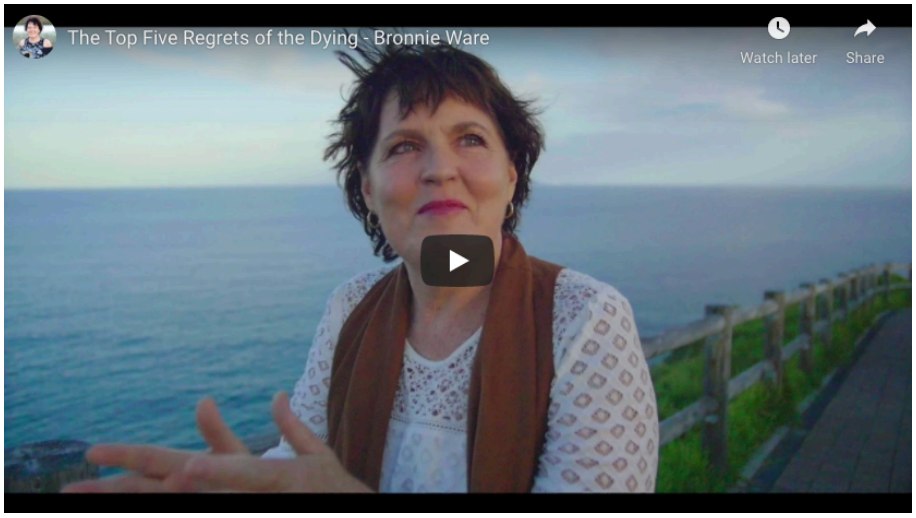


Figure 2.5: Palliative care nurse Bronnie Ware explains the top five regrets of the dying. [@youtube-bronnie] Bronnie learned a lot from looking after people on their deathbeds, then wrote it all down in a fantastic book [@regrets]. The image in the figure is a screenshot, you can [watch the two minute video here](https://www.youtube.com/watch?v=nayz3xJxRTA)

2.7.2 Colouring your parachute

Since first being published in 1972, over ten million copies of *What Colour is Your Parachute?* have been sold. It has been translated into 20 languages and is used in 26 countries. What is good about *Parachute* is that it has some useful *self-inventory* exercises that go beyond the introductory ones in this guidebook, particularly in the context of your future career. While the style and examples can be U.S. centric, it's a classic self-help book that looks at a broad variety of issues around job hunting. The author, Richard Nelson Bolles was a Harvard educated chemical engineer and he explains how you can't possibly decide what to do in five years time in the video in figure 2.6. Where do you see yourself in five years time? is a question some interviewers like to ask.

2.7.3 What's your story?

A useful technique for developing self-awareness is to think about what your story is. Heather Box and Julian Mocine-McQueen's book *How Your Story Sets You Free* (Box and Mocine-McQueen, 2019) takes a storytelling approach to

Figure 2.6: Where will you be five years from now? Best-selling author of *Colouring Your Parachute* Dick Bolles discusses the gaps between education and employment. [@youtube-bolles] The image in the figure is a screenshot, you can [watch the full 32 minute talk here](https://www.youtube.com/watch?v=oeP6Pm3Xf-8)

help you gain a better picture of who you are and what you value. What's good about this book is its short, less than 100 pages and contains practical exercises which extend those in this chapter.

2.7.4 Check your privileges

Reflecting on your identity should lead you to check any privileges you might have. Being grateful for any privileges you may have is also beneficial for your mental health which we talk about in chapter 3 so:

- **If you're white** a good place to start understanding your white privileges is *Why I'm No Longer Talking to White People About Race* by Reni Eddo Lodge (Eddo-Lodge, 2017) and *Superior: The Return of Race Science* by Angela Saini
- **If you're male** a good place to start understanding the privileges you have as a result of being a man is *Inferior* by Angela Saini (Saini, 2018)
- **If you're socially privileged** a good place to start understanding the privileges you have as a result of your class is *The Class Ceiling: Why it Pays to be Privileged* by Sam Friedman and Daniel Laurison (Friedman and Laurison, 2020). If you were privately (or selectively) educated in

Britain (or elsewhere) you should read *Engines of Privilege: Britain's Private School Problem* (Green and Kynaston, 2019)

- **If you're heterosexual** a good place to start understanding the privileges you have as a result of your sexual orientation is Ben Britton's presentation on *No sexuality please, we're scientists* (Britton, 2019) which covers bisexuality and homosexuality, including lesbian and gay homosexuality
- **If you're gender binary** a good place to start understanding the privileges you have as a result of being gender binary is Ben Britton's presentation (Britton, 2019) which also covers transgender, genderqueer, non-binary and plus identities

There is a lot more to your identity than your race, class, gender and sexual orientation, see your protected characteristics in section 2.5.

2.8 Summarising self awareness

Too long, didn't read (TL;DR)? Here's a summary:

This chapter has looked at who you are. Being self aware, understanding your strengths and weaknesses is key to getting what you want from your career. Questions about your identity are non-trivial, hopefully this chapter has started you thinking about who you are, what motivates you and what you want out of life. You need to keep thinking about your identity because some aspects of your identity may be constantly changing.

These are fundamental design questions you'll need to address when you starting building your future. We touched on understanding any privileges you may have as being important for understanding who you are but also in being beneficial for your mental health.

In the next chapter, we'll look at mental health in more detail.

Chapter 3

Nurturing your future

It doesn't matter if you are a student, an employee or even both at the same time. To be successful at studying or working, you need to take your well-being seriously. By well-being, I mean your health and happiness. Your health isn't just about your physical health but also your mental health and the two are very closely linked. It's all too easy when you are busy or stressed to neglect your well-being and then **bad-stuff™** happens. This chapter looks at your well-being, and how you can nurture it. Because looking after yourself now will also nurture your future.

3.1 What you will learn

By the end of this chapter you will be able to:

- Identify some of the symptoms of mental ill health in yourself and your peers, particularly anxiety and depression
- Describe five self-help techniques for improving mental health
- Describe services and other people you can approach if you (or someone you know) is being affected by mental ill health and self-help isn't enough
- Schedule activities for improving mental and physical health into your daily or weekly routine

DISCLAIMER : I am neither a medical doctor or a psychologist: If you're affected by mental ill health, you should speak to a trained professional. This chapter just gives you a quick overview of mental health and points you to where you can find out more.

Figure 3.1: You probably already knew that Alan Turing was an outstanding Computer Scientist, but did you know he was also a respectable athlete too? Turing ran, cycled and rowed to relieve stress, (Kottke, 2018) and came close to competing in the Olympics as a runner. This should come as no surprise, the connections between well-being and academic performance are widely documented. Image via Jonathan Swinton's biography *Alan Turing's Manchester*. (Swinton, 2019) The copyright holder for this image has been unidentifiable or unresponsive at their self-advertised contact details.

3.2 Mental ill health

Stress can lead to many kinds of ill health. Turing was put under lots of stress by his government bosses, people like Alastair Denniston and Stewart Menzies. (Tyldum, 2014) When asked why he punished himself so much in training, Alan Turing's reply is shown in figure 3.2.

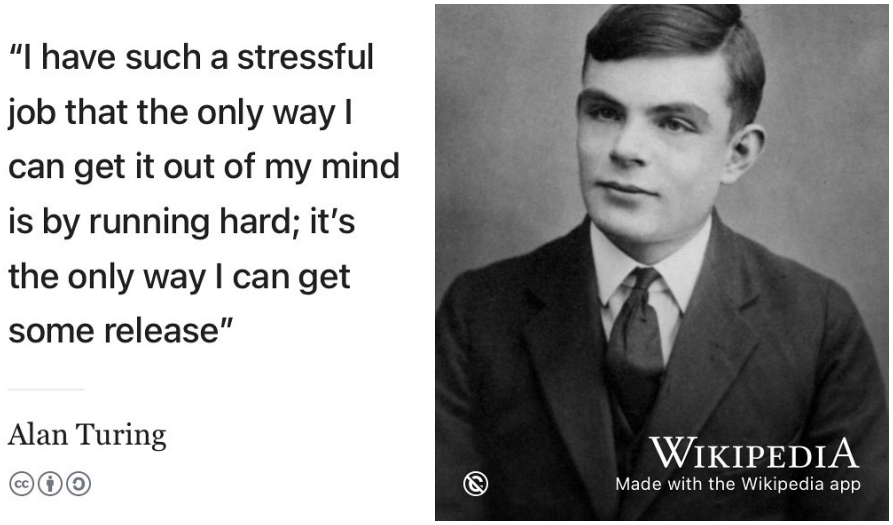


Figure 3.2: When asked why he trained so hard, Alan Turing replied: "I have such a stressful job that the only way I can get it out of my mind is by running hard; it's the only way I can get some release". Like many, Turing found running a relief from the mental pressures he was under in his job. (Kottke, 2018) Studying can be stressful too and put you under pressure. Your academic performance at University can be significantly improved by taking regular exercise and it will improve your mental health too. Public domain portrait of Alan Turing via Wikimedia Commons w.wiki/oZx adapted using the Wikipedia app

University is a positive experience for many people, however like Alan, you may also experience periods of stress. This may also be accompanied by anxiety, loneliness and depression. Financial, social and academic pressures alongside concerns about employability and an ongoing pandemic of COVID-19 can all have an impact on your wellbeing. Statistically, one in four of us will be affected by mental ill health during our lifetime. Two of the most common forms of mental ill health are:

- **Anxiety**: *persistent* feelings of unease, such as worry or fear
- **Depression**: a low mood that *lasts for a long time* and affects your everyday life

The *persistent* and *lasting a long time* are important here because while its part of the human condition to worry and feel low, that doesn't *necessarily* mean you are affected by poor mental health.

3.2.1 Anxiety

Anxiety is one of most common mental health disorders and can lead to depression, increased risk of suicide. Generalised Anxiety Disorder (GAD), a common form of anxiety is explained in the video in Figure 3.3 and at nhs.uk/conditions/generalised-anxiety-disorder. People who are affected by anxiety may struggle to function normally, and find routine everyday task difficult or impossible.

Figure 3.3: Generalised anxiety disorder is a condition characterised by excessive, persistent and unreasonable amounts of anxiety and worry about everyday things. (Desai, 2016) Note that the video takes an American perspective using American terminology such as DSM–5.

3.2.2 Depression

Millions of people around the world live with depression. If you are affected by depression it can be really hard to talk about it as there are many social stigmas around mental health. Thankfully depression is largely preventable and treatable. Recognising depression and seeking help is the first and most critical step towards recovery. To mark World Mental Health Day writer and illustrator Matthew Johnstone tells the story of how he overcame the "black dog

of depression" in the video in Figure 3.4 made in collaboration with the World Health Organization (WHO).

Figure 3.4: Matthew Johnstone explains how he overcame the affects of depression, using the metaphor of the black dog (Johnstone, 2012)

3.2.3 Drugs

Prescription medication can help some people with their mental health. For example, when I was affected by depression, Selective Serotonin Reuptake Inhibitors (SSRIs) worked for me, shown in Figure 3.5, but they don't for everybody. Sometimes the drugs don't work, they just make you worse. (Ashcroft, 1997)

Some doctors prescribe benzodiazepines for anxiety, which may be effective where SSRI's are not, but these can be addictive and have big side effects.

It is often worth considering cognitive behavioural therapy (CBT) before taking any medication. *The Science of Wellbeing* (TSOWB) at coursera.org/learn/the-science-of-well-being is an easy way to access some CBT free online. See the signposts section at the end of this chapter (Santos, 2021)

3.3 Look after yourself

Looking after yourself can serve to both prevent and treat mental health issues that can affect you in life. You might be your own worst critic, or perhaps when you're under pressure you neglect things that are proven to be beneficial for

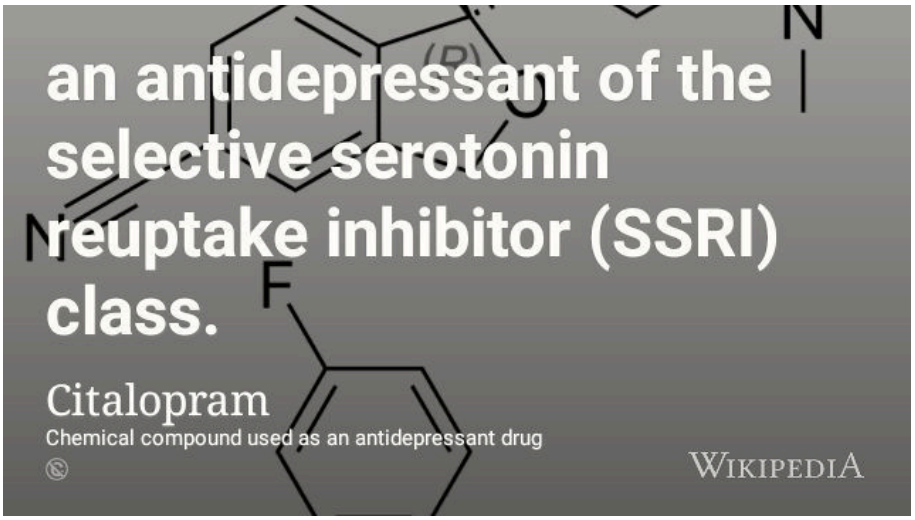


Figure 3.5: Citalopram is a type of antidepressant known as a Selective Serotonin Reuptake Inhibitor (SSRI). SSRI's can help some people who have been affected by depression. They work for some people (including me) but they don't for everybody. Skeletal formulae of Citalopram by Vaccinationist via Wikimedia Commons w.wiki/3Ddn adapted using the Wikipedia app.

your mental health, like sleep, exercise, mindfulness and friendship. Looking after yourself means at least three things:

- being mindful of your feelings and learning to manage your inner critic
- being kind to yourself in various ways
- deliberately scheduling protected time to do the non-work things that matter.

Harvard Psychologist Laurie Santos describes five evidence-based strategies for coping when times are really challenging and tough in the video in figure 3.7. Those strategies are:

1. **Exercise**: getting regular exercise improves both physical AND mental health.
2. **Gratitude**: research shows that being grateful can significantly improve your mental health. One way to do this is by keeping a gratitude journal, a log you fill in everyday of things you are grateful for (either small or big)
3. **Sleep**: actively developing healthier sleep patterns. Poor sleep hygiene can be both cause and effect of poor mental health. See the discussion of *Why we sleep* (Walker, 2018) in section 3.6
4. **Socialising**: prioritise time with friends and family, rather than turning inward or diving deeper into work

Figure 3.6: It's important not to neglect your body, mind, soul and social life when you're working hard. Look after yourself by Visual Thinkery is licensed under CC-BY-ND

5. **Mindfulness**: be mindful of emotions using the R.A.I.N. technique:
    - **R**ecognise: negative emotions
    - **A**ccept: accept emotions rather than fighting them
    - **I**nvestigate: notice how the emotion feels inside your body
    - **N**urture: be kind to yourself, step away from your emotions by distancing yourself from them.

It can help to think of negative emotions as coming from another person, an inner critic, rather than yourself. You are not your emotions and thoughts. Laurie explains the R.A.I.N. technique in figure 3.7.

So there are things you can do to help yourself, but you may also need to seek help from others.

Sometimes a desire to be productive by working hard has the opposite effect, because the sacrifices you make can be counter-productive.

> pic.twitter.com/D2SP4iJspT
> — lizandmollie (@lizandmollie) February 23, 2020

3.4 Help is available if you need it

If you are affected by mental ill health, particularly anxiety or depression, it can be hard:

Figure 3.7: Laurie Santos describes five coping techniques for improving well-being: Exercise, gratitude, sleep, getting social and meditation (Santos, 2020).

- to recognise that you need help in the first place
- to help yourself using self-help resources
- to ask others to help you

Even if you don't need help, it is important to recognise and understand the symptoms of mental ill health. It's quite likely that someone you know will suffer from mental health issues and as their friend or peer, it might be you that can help by encouraging them to get the help they wouldn't otherwise ask for.

You are not alone, help is available if you (or your friends) need it from a wide variety of sources:

3.4.1 Your University

There are lots of people who can help you:

- your personal tutor or other academic members of staff
- non-academic staff in the University, for example in Manchester contact the Student Support Office (SSO) studentsupport.manchester.ac.uk
- counselling services, for example contact counsellingservice.manchester.ac.uk. The counselling service offers help on dealing with anxiety, depression, exam stress, confidence and other issues.
- peers, flat-mates, family, friends etc. People close to you can help, although some people affected by mental health find it easier to discuss

mental health with a trained professional or volunteer because of the social stigmas. There are lots of services outlined below that provide this kind of service.

3.4.2 The National Health Service

As a student studying in the UK you are entitled to access free healthcare provided by the National Health Service (NHS) of the United Kingdom. To do so you'll need to be registered with your general practitioner (GP), see nhs.uk: Getting medical care as a student

Your doctor can advise you on medical treatment if required, see for example nhs.uk/conditions/antidepressants

3.4.3 Nightline

Nightline nightline.ac.uk is a confidential listening and information service run by students for students. Nightline is open 8pm till 8am every night during term time. It offers anonymous, non-judgmental and non-advisory support for students as described in figure 3.8.

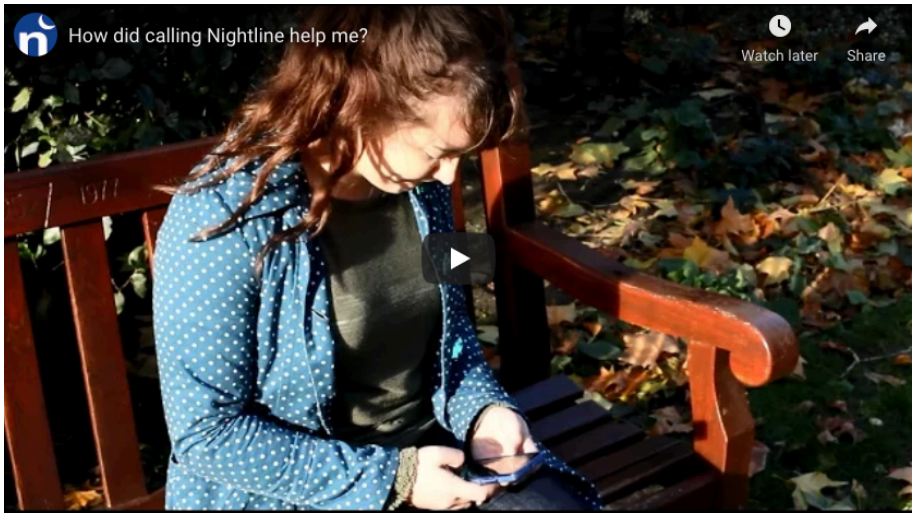


Figure 3.8: Students explain in their own words how calling Nightline helped them whilst at university. [@youtube-nightline]

Manchester students can contact nightline at nightmail@nightline.manchester.ac.uk and expect a reply within 48 hours. See manchester.nightline.ac.uk for details.

3.4.4 The Samaritans

The Samaritans are a charity who provide emotional support to anyone in the the United Kingdom and Ireland that:

- is suffering from emotional distress
- is struggling to cope
- is at risk of suicide

The name of the charity comes from the Parable of the Good Samaritan although the organisation itself is not religious. The Samaritans are available 24 hours a day, seven days a week, to talk confidentially about any problem, however big or small. See samaritans.org or telephone 116 123.

3.4.5 Students Against Depression

Students Against Depression (SAD) acknowledge the devastating impact that depression can have on those experiencing it, as well as on their friends, family and supporters. For further help in understanding and coping with suicidal thoughts, and emergency contacts in a crisis, visit studentsagainstdepression.org

Actor Ruby Wax has written about mental health and how the "internal critics" in our minds can send us mad in her book *Sane New World*. (Wax, 2014) She is interviewed by Students Against Depression in the video in figure 3.9 about using mindfulness to "dodge the bullets" of depression.

3.4.6 Papryus

Suicide is the biggest killer of under 35's in the UK. Papyrus believe that many young suicides can be prevented, they are a national charity that you can find out more about at papyrus-uk.org or telephone the free number 0800 068 4141.

3.4.7 Self-help services

Self-Help services are a mental health charity which helps people to help themselves, see selfhelpservices.org.uk or phone 0161 226 3871.

3.4.8 MIND

MIND provide advice and support to empower anyone experiencing a mental health problem. They campaign to improve services, raise awareness and promote understanding of mental health issues. Find out more at mind.org.uk and in the video in figure 3.10

Figure 3.9: Ruby Wax describes being affected by depression in her childhood and how mindfulness and cognitive behavioural therapy (CBT) provided an alternative to medical treatment enabling her to dodge the bullets of mental health. [@youtube-wax]

Figure 3.10: Stephen Fry, President of Mind, describes how MIND tackles misconceptions around mental health and social stigmas. [@youtube-we-are-mind]

3.4.9 Student minds

Student Minds empowers students to look after their own mental health, support others and create change, find out more at studentminds.org.uk and in the video in Figure 3.11 which describes why it is important to talk about student mental health.

Figure 3.11: Talking about mental health is a crucial part of helping those who are suffering from it [@youtube-student-minds]

3.4.10 Togetherall

Togetherall is an online community for people who are stressed, anxious or feeling low. The service has an active forum with round-the-clock support from trained professionals. You can talk anonymously to other members and take part in group or 1-to-1 therapy with therapists. Togetherall is for anyone aged 16 or over who wants to improve their mental health. The service is free for many universities. Find out more at togetherall.com and in the video in figure 3.12 which describes why its important to talk about student mental health.

3.5 Developing a growth mindset

Learning at University can be hard because you might have gone from being at (or near) the top of the class in high school to no longer being top of the class at University.

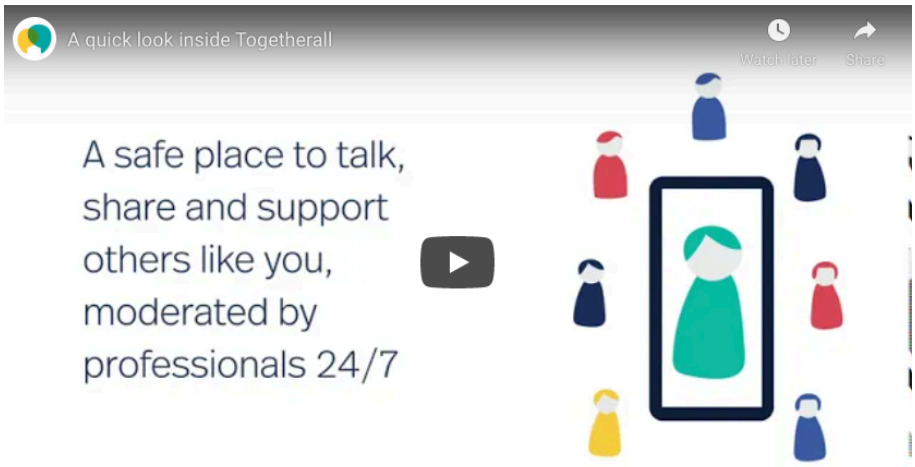


Figure 3.12: A quick look inside togetherall, an online community for people who are stressed, anxious or feeling low. [@youtube-togetherall]

Likewise the job hunting described in chapter 8 can take a heavy toll on your mental health because repeated rejection is an ordinary part of the process. It can be time consuming, stressful and demoralising. You may find your applications disappear into a black hole. They will be ghosted (ignored) by employers. Interviewers will blank you and refuse to give you meaningful feedback because they're too busy. This could happen multiple times. This is all *par for the course*, normal and expected, and is not necessarily a reflection on your abilities or potential. See the examples in the coding interview section 10.2.3. (Davidova, 2021)

Adopting a growth mindset can be a successful strategy for maintaining your wellbeing, see figure 3.13. If your grades aren't has good as you hoped or your search for employment is being met with repeated rejection, a growth mindset can help. Let's take rejection from potential employers as an example, there are two ways you can react to it:

1. ***Fixed mindset***: responding with a fixed mindset will mean you are likely to take rejection personally. You might even assume that this confirms what you've always suspected. You're not good enough or that you made some fatal mistake in your applications or interviews. Ouch.
2. ***Growth mindset***: by responding to rejection with a growth mindset, you focus on what happens next. Rejection is not failure but a "not yet" described in figure 3.14. Maybe you're not yet ready for that employer, but you'll definitely have a good idea of what you learned from the process and how can you do better next time.

Figure 3.13: A fixed mindset is monolithic like the Easter island statues. If you're not already, you should be wary of a fixed mindset. Fixed mindsets by Visual Thinkery is licensed under CC-BY-ND

According to Stanford psychologist Carol Dweck we can all be placed on a spectrum describing where we think our abilities come from. At one end, the fixed mindset assumes all kinds of abilities are fixed traits while at the other end, a growth mindset assumes these abilities can be developed over time. (Dweck, 2017) There is good evidence that adopting a growth mindset will make you a better learner who can cope with the inevitable failures and rejections in life better. This approach can be used in a range of different disciplines such as learning programming languages (Cutts et al., 2010), music (Davis, 2016) and job hunting.

3.6 Wellbeing signposts

This chapter has looked at your wellbeing and especially the role that both your mental health and physical health play in your future. Matt Haig's first-hand accounts of poor mental health will be comforting to anyone who is affected by mental ill health. Even if you're not affected, there is a 25% chance you will be at some point in your life. There's also a high probability someone close to you will suffer from mental health issues. It might be a colleague, friend, family member, fellow student or partner, so it is worth educating yourself on the issues by reading his two short books:

1. *Notes on a Nervous Planet* is a personal account of anxiety (Haig, 2019)
2. *Reasons to Stay Alive* is a personal account of depression (Haig, 2016)

Figure 3.14: Psychologist Carol Dweck explains the power of *not yet* and the growth mindset (Dweck, 2014)

What's good about Matt Haig's books is they are quick and easy to read, but give plenty of first-hand insight into what mental ill-health can do to people (including you). Matt describes his top five tips for good mental health in figure 3.15

There's plenty of evidence that social media can have a detrimental effect on health. Jaron Lanier's skeptical polemic *Ten Arguments for Deleting Your Social Media Accounts Right Now* (Lanier, 2018) is a thought-provoking romp through some of the pitfalls of social media that may have you reaching for the delete or un-install button fairly quickly. You don't have to be on social media, see figure 3.16.

If all these books are making you sleepy, neuroscientist Matthew Walker's *Why We Sleep: The New Science of Sleep and Dreams* may change your view on the importance of a good nights sleep. (Walker, 2018)

Finally, it's well worth taking a look at Laurie Santos course on *The Science of Wellbeing* (TSOWB) at coursera.org/learn/the-science-of-well-being. (Santos, 2021) TSOWB course provides an alternative to medication as it follows the principles of cognitive behavioural therapy (CBT).

TSOWB is the most popular course at Yale University and looks at some simple techniques you can use to improve your happiness. (Shimer, 2018) The course will help you increase your happiness and build more productive habits. Using the latest research, Santos describes the misconceptions about happiness and "annoying features" of your mind that can impair your well-being. The course takes about 19 hours to complete but you can spread this over a whole semester

Figure 3.15: Two of Matt Haig's top five tips for good mental health (Haig, 2018) include 1. Being more careful (and mindful) of social media and 2. Reading more books because books are good for your soul. Not just his book. Any book. Books are good for you. Trust me on this. (Forever, 2019)

Figure 3.16: Social media like LinkedIn clearly has its uses (see section 7.6.5) but you you don't *have* to be on it at all. Poor mental health is just one reason to be wary of social media, for nine other reasons read Jaron Lanier's polemic. (Lanier, 2018) Notes to Strangers by andy-leek.com photographed by Duncan Cumming

(or longer) if you choose. The short clip in figure 3.7 gives you a brief taster of Laurie's style and work.

3.7 Breakpoints

Let's pause here. Insert a breakpoint in your `code` and slowly step through it so we can examine the current values of your variables and parameters.

```
* PAUSE
```

- How would you describe your own state of mental health?
- Do you have friends or peers who are affected by mental ill health?
  - What are the signs they might be suffering?
  - How could you support or help them better?
- If you describe your own mental health as poor
  - Where can you go for self-help?
  - You are not alone but who can you talk to?

```
* RESUME
```

3.8 Summarising well-being

Too long, didn't read (TL;DR)? Here's a summary:

Anxiety and depression are serious conditions that are very likely to affect you or somebody close to you while you are at University. There's a one in four chance that you will be affected by mental health issues at some point in your life.

We've only talked about two particular mental health issues, anxiety and depression, but there are many other conditions such as phobias, obsessive-compulsive disorder (OCD), eating disorders, self-harm and more that are beyond the scope of this chapter. They do have one thing in common, and that is that talking about them is an important part of starting to develop better mental health.

If you are affected by mental ill health, talking about it is the first place to start, but often the hardest. In this chapter, I've outlined some ways you can help yourself alongside some of the services and people you can talk to if you need to.

Despite how you might feel, you are not alone.

Take my thoughts with you and when you look behind, you will surely see, a face that you recognise, you're not alone. (Kellett et al., 1997)

Chapter 4

Writing your future

Your soft skills will take a **life time** to develop and are **really hard** use. Why? Because soft skills are about *communicating* with and *understanding* other people so that you can work *together* as a team toward a shared goal. Your soft skills are hard. There are very few jobs where you work on your own in complete isoltion. For example, most software and hardware is designed, built, tested and used by teams of people. Many of these teams are large and have very diverse membership. This means that sooner or later you're going to have to master the dark arts of *working with other people* by developing and deploying your softer skills.

Figure 4.1: Unless you want to be a lighthouse keeper on a remote island, there are very few jobs where you don't have to work as part of a team with other people. Sorry to break the bad news! This means you need to constantly improve your softer skills and provide evidence of them to potential employers. Other people sketch by Visual Thinkery is licensed under CC-BY-ND

Communicating with other people and working in teams is inherently difficult because we're all human. There is good news and bad news…

- **THE GOOD NEWS** is, people can be diligent, humble, competent, honest, caring and reliable. They can be co-operative, generous, supportive, kind, thoughtful, intelligent, sensitive, understanding, punctual and professional too!
- **THE BAD NEWS** is, unfortunately people can also be lazy, stupid, ignorant, vain, incompetent, dishonest, unreliable, greedy, egomaniacal, unpredictable and moody. They can be proud, selfish, competitive, lustful, angry, envious, mean, busy, insensitive and thoughtless too. Some will disagree with you, boss you around, betray, exploit, misunderstand and mislead you, deliberately or otherwise. (Goble, 2007)

So communicating with and understanding other people can be hard work, but don't worry, **everyone** finds this challenging, it's not just you! It doesn't matter if you're an extrovert or an introvert, we *all* find communication difficult, and everyone can get better at it. This chapter takes a look at the softer skills and techniques you can use to improve your communication with other people, whatever mood they are in and whosever team they are on.

4.1 What you will learn

- Recognise the importance of written communication, both as a reader and a writer
- Identify examples of where written communication is crucial in science and engineering
- Improve your written communication skills using some simple writing and reading exercises
- Identify the importance of teamwork

4.2 Computing is your superpower!

Studying computer science gives you an awesome superpower. We will look at some of the reasons why in chapter 6 on *Computing your Future*. But for now, let us just acknowledge that hard technical skills like computing are highly sought after and valuable, both commercially and otherwise.

Your computational superpower is less powerful if it isn't complemented by a broad range of softer skills. Typically, these skills are not emphasised (by repeated assessment) in most computer science degrees. This not because soft skills aren't important but because they are hard to measure accurately.

For example, if I want to know how good you are at understanding the syntax and semantics of a programming language like Python, there are tried and tested techniques for doing this. However, if I want to know how good you are at using

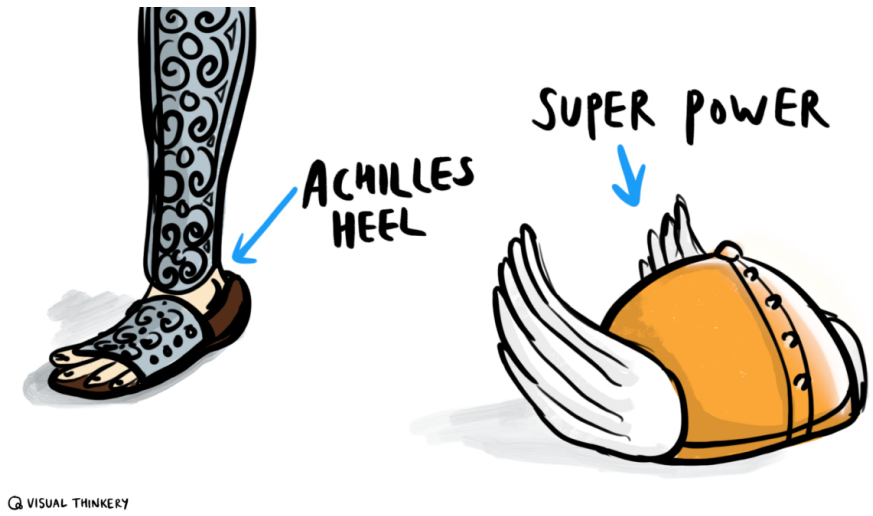


Figure 4.2: Computing is a superpower that gods like Hermes and mortal heroes like Achilles would probably have envied. (Fry, 2018) As a technical or *hard skill*, computing is a crucial weapon in your armoury but what are your weaker skills? What is your Achilles' heel? For some scientists and engineers, their weakness is their soft skills, such as communication and team work. This chapter looks at what you can do to improve them and convince employers that you are rounded individual with a healthy balance of soft and hard skills. Achilles heel to superpower by Visual Thinkery is licensed under CC-BY-ND

your communication skills to work in a team, negotiate, lead, resolve conflicts, persuade others, show empathy etc that's **much** harder to measure accurately.

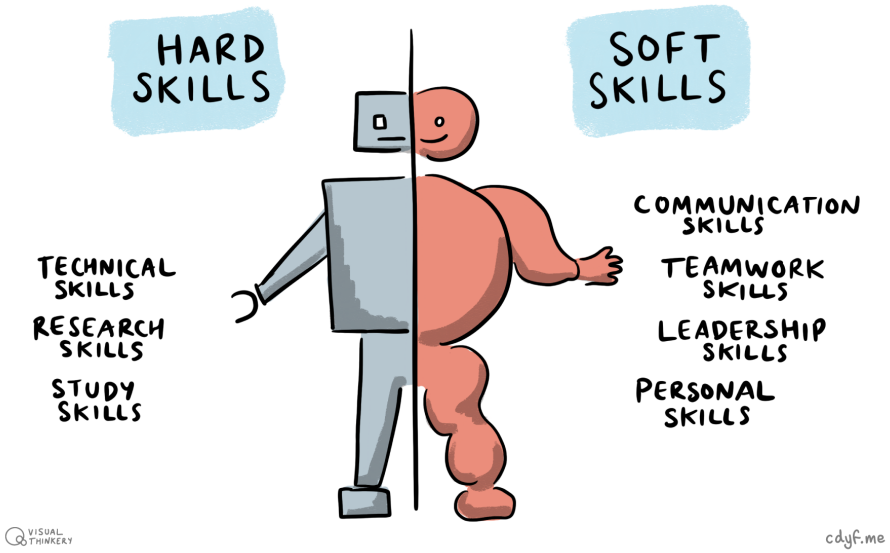


Figure 4.3: Hard skills and soft skills aren't much use without each other. You will need both to survive and thrive but most science and engineering education focuses on your hard skills, not your soft skills. Why? Because hard skills are often much easier to measure. Hard and soft skills sketch by Visual Thinkery is licensed under CC-BY-ND

Let's look at some of low-level communication skills (I/O) that they are built on.

4.3 Communication I/0

In terms of input and output, your fundamental communication skills are listening, speaking, reading and writing words in natural languages shown in table 4.1. These are the "assembly languages" of human communication. This might sound blindingly obvious, but these skills are often under-estimated or undervalued by engineers and scientists, especially undergraduates. Alongside verbal and written communication, there's also nonverbal language, or body language such as eye contact, gestures and facial expressions.

Engineers and scientists sometimes lack communication skills outlined in table 4.1. Think of your stereotypical scientist, clad in a white coat, unable to explain the complexities of their subject to people inside their lab, let alone outside of it. Then there is the nerdy software engineer stereotype who prefers the company of

Table 4.1: The inputs and outputs of the fundamental assembly languages of human communication

	Input	Output
Written natural language	Reading	Writing
Spoken natural language	Listening	Speaking
Nonverbal language	Observing other people	Being observed by others

computers to people. Yes, these are lazy and sometimes unhelpful stereotypes, but they express public perception of scientists and engineers as poor communicators. Don't perpetuate the stereotype by being a bad communicator.

4.3.1 The pen is mightier than the sword

The art of communication is a huge subject which extends far beyond the scope of this guidebook. So for the rest of this chapter, we'll focus on your superpower of written communication. The pen (and keyboard) is mightier than the sword, see figure 4.4.

"The pen is mightier than the sword" is a metonymic adage, created by English author Edward Bulwer-Lytton in 1839, indicating that the written word is a more effective tool for communication than violence

The pen is mightier than the sword

Figure 4.4: Need to arm yourself with another superpower for communication besides computing? It's in your own selfish interests to continuously develop your written communication skills because the pen is mightier than the sword. Public domain image of a drawing of Cardinal Richelieu by Henry Alexander Ogden via Wikimedia Commons w.wiki/3WHg adapted using the Wikipedia app

Written communication skills are important because:

1. **Good writing and reading are crucial in applications** for employment and further study. From writing CV's, covering letters, completing application forms and reading job specifications, and employer (or course) information, your ability to read and write natural languages is crucial to coding your future.
2. **Writing often gets neglected**: Written communication skills (both reading and writing) are sometimes sidelined in science and engineering degrees. This is particularly true in the "hard sciences". For example, communicating and solving problems using code or mathematics are usually the dominant forms of assessment in computer science courses. That's understandable given the subject, but tends to push natural languages like english to the sidelines.
3. **Good engineers are also good writers** Many engineers (and scientists) could significantly improve their written communication skills. Software engineers are notoriously bad at writing, see for example Why Computer Science Students Need Language, (Beaubouef, 2003) *Scientists Must Write* (Barrass, 2002) and The Real Reason Silicon Valley Coders Write Bad Software, (Meisler, 2012) just three examples amongst many others making exactly the same point. Employers like Google provide training (and a whole career path) for technical writers, see developers.google.com/tech-writing. I'm glad they exist, but these careers and courses wouldn't be needed if software engineers were better at documenting, explaining and communicating with other human beings in the first place!
4. **Writing good english is like writing good code**. Some of the skills you already have in coding can be transferred to written communication. Just like a good function or method in code should be well-defined with a clear purpose, your writing should also be clear and coherent. Well structured writing is a lot like well architected software too, with a clear separation of concerns (SoC)
5. **It is relatively easy to improve** your written communication skills, simply by reading and writing more. Reading and writing deliberately every day, will significantly improve these skills.

4.3.2 Natural language engineering

As a species, we've been writing stuff down for millenia in order to communicate with each other, see figure 4.5. If you stop to think about it, engineers and scientists spend a *lot* time communicating in writing. As well as engineering `code`, they also spend a significant amount of time engineering messages in natural languages like english.

Consider the following:

- email and instant messaging, Slack, Microsoft Teams, Discord, Zoom etc

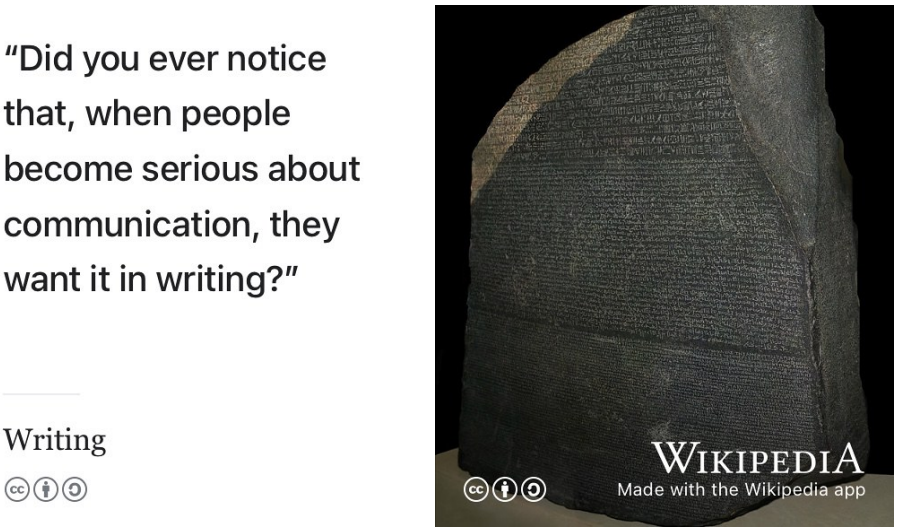


Figure 4.5: Have you ever noticed how when people become serious about communication, they want it in writing? CC BY-SA image of language in Ancient Egyptian using hieroglyphic, demotic and Ancient Greek written on the Rosetta Stone by Hans Hillewaert on Wikimedia commons w.wiki/3Ycn adapted using the Wikipedia app

- Posting on social media: LinkenIn, Facebook, WhatsApp, Twitter, blogs, Medium.com etc
- bug reports and messages in issue trackers like Jira, BugZilla, Github, Gitlab and Trello etc
- 'How to' and cookbook style articles and books
- API reference material
- in-code documentation `# comments in code`
- Self-documenting code that describes itself
- Executable specifications in test suites like cucumber.io
- Laboratory manuals and laboratory notebooks
- The one or two page summary for management
- reddit.com and hacker news news.ycombinator.com etc
- User documentation, release notes
- Case studies of software use
- Frequently Asked Questions (FAQ)
- Personal websites `YourPersonalDomain.com` if you have one
- Questions and answers on forums like stackoverflow.com
- Commit messages in version control systems like git and mercurial etc
- Architecture documentation and design specifications
- Literate programming natural language descriptions of computational logic (Knuth, 1984)

- Jupyter.org notebooks, code and natural language mixed together
- bookdown.org mixes code and natural language

What do they all have in common? They're all written in natural languages like the English language, but without them, the software or hardware they describe and discuss would be useless.

Making software isn't all about what you can do as an individual but rather how you communicate with and contribute to your team. It's easy to get carried away with your ego. Remember that most jobs require *lots* of softer people skills and collaboration, written communication is a huge part of that, see for example The Myth of the Genius Programmer. (Fitzpatrick and Collins-Sussman, 2009)

4.4 Writing your future

Hopefully I've convinced you that written communication skills (both as a writer and reader) are important soft skills that engineers often neglect. So how can you improve?

4.4.1 Dogfooding

Many employers test their products by trialling them on their own employees, this is known as eating your own dogfood shown in figure 4.6. Tasty, tasty dogfood.

Dogfooding is a great way to test your own writing. Let's say you've just written a covering letter. It's natural to read it over in your head to check for errors, before you send it. However, **reading it aloud** will pick up errors you may not have spotted by reading silently. There's something about articulating words out loud that flushes out errors you don't pick up when you read them silently. This is a tried and tested technique. It also means you're ready to vocalise those answers in an interview.

You might want to choose carefully where you do this as it might look a bit strange, but it works well. If you talk into a mobile phone while looking at a piece of paper, people won't notice you're talking to yourself. But you'll probably need some privacy as the stuff you're talking about is likely to be personal.

4.4.2 Try Google's Tech Writing course

Google have developed some excellent written communication courses for software engineers, and those looking to become technical writers:

Figure 4.6: Reading your own writing (aloud) is like eating your own dog food. It's a simple and proven technique for improving your written communication in job applications such as covering letters, CVs, personal statements and the like. Picture of a dog chowing on a tasty pig's trotter (omnomnom) by Denhulde on Wikimedia Commons w.wiki/3a2r adapted using the Wikipedia App

1. Technical Writing One: Technical Writing Fundamentals for Engineers developers.google.com/tech-writing/one
2. Technical Writing Two: Intermediate Technical Writing for Engineers developers.google.com/tech-writing/two

These courses run as part of the second year course COMP2CARS at the University of Manchester, see chapter 19 for details.

Google occasionally delivers these technical writing courses as free sessions open to the general public. For details, see developers.google.com/tech-writing/announcements for details.

4.4.3 Deliberate daily writing

Another technique for improving your written communication is to write something every day, that might be a personal diary that only you read, or it could be something more public like blog. Schedule a time every day, say 15 to 30 minutes when you will do this without fail. That writing could take several forms:

- public web log or blog

- gratitude journal see section 3.3
- private diary or personal laboratory notebook
- bullet journal. Some people swear by it, see bulletjournal.com

The technique of *30 minutes per day* can be a very effective way of getting things done, incrementally over time. In my experience it works for lots of things besides writing including getting exercise (discussed in chapter 3) to gardening. (Leendertz, 2006)

4.4.4 Deliberate daily reading

Reading other people's code will improve your software engineering skills. Likewise, reading other peoples writing will improve your natural language engineering skills. Read anything, it might be novels, magazines, newspapers, stuff online or any of the books cited in chapter 20. Find a time and place to do this every day and stick to it.

4.4.5 Reading the friendly manual

You don't get good at communicating with computers (coding) by just *writing* lots of code. You also need to *read* other people's code too and be able to understand and modify it. Likewise, you don't get good at communicating with people by just *writing* stuff in natural languages like english. You need to *read* stuff too. Books, manuals, software documentation, articles, use cases, novels, poetry, plays, magazines, newspapers etc. Reading this stuff will help you learn and you'll improve your written communication skills too. So Read The Friendly Manual. RTFM. Read THIS Friendly Manual and the stuff it cites, see figure 4.7

4.5 Breakpoints

Let's pause here. Insert a breakpoint in your `code` and slowly step through it so we can examine the current values of your variables and parameters.

```
* PAUSE
```

- Which of the communication skills in table 4.1 are your strongest?
- Which of the communication skills in table 4.1 are your weakest?
- What activities could you do to improve your weaker communication skills?

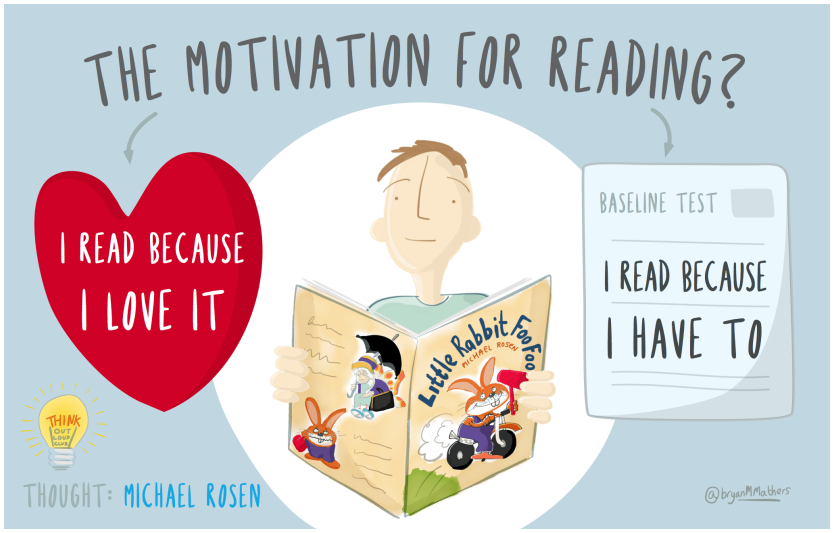


Figure 4.7: Read The Friendly Manual (RTFM), some of it you will love, some of it you won't. Either way, reading allows you to learn from other people's hard won experience whilst also improving your own written communication skills. Just like you improve your coding skills by reading and writing code, you will improve your written communication skills by reading and writing in natural languages. The motivation for reading by Visual Thinkery is licenced under CC-BY-ND with help from Michael Rosen

```
* RESUME
```

4.5.1 Coding challenges

- Write an article or blog post about something you care about, find a suitable venue for publication
- Take a course from outside computer science, where the main form of assessment is written essays or dissertations. Humanities departments are a good place to start. This will improve your written communication skills
- Not been reading many books lately? Pick a book to read just because its interesting, rather than because you have to.

4.6 Summarising your soft skills

Too long, didn't read (TL;DR)? Here's a summary:

You'll need both soft and hard skills to compete in the workplace. Don't underestimate the importance of softer skills, we've looked briefly at written communication skills in this chapter but that's only the tip of the soft skills iceberg.

Teamwork, negotiation, conflict resolution, public speaking, motivating others and leadership are also important soft skills too. How can you develop these skills while at University? How can you demonstrate to potential employers that you have these skills?

This chapter is under construction because I'm using agile book development methods, see figure 4.8.

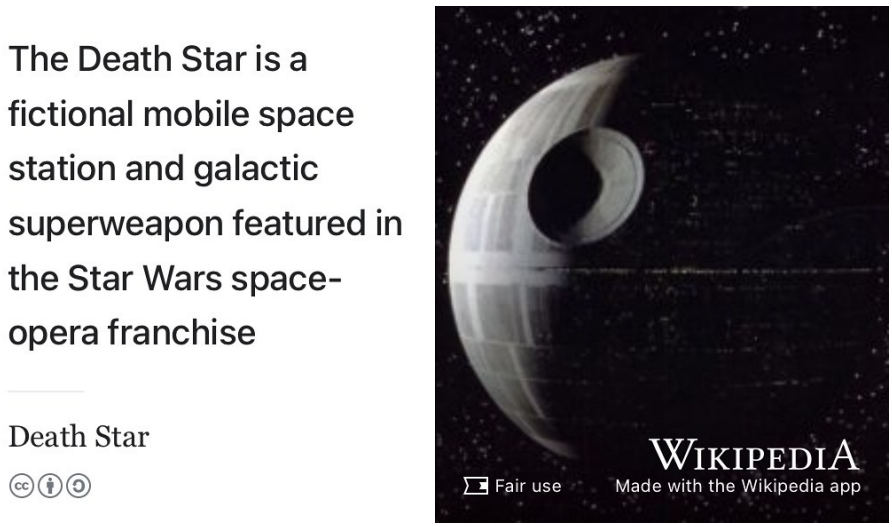


Figure 4.8: Just like the Death Star, this ~~galactic superweapon~~ chapter is under construction. Image of agile weapon engineering in *Star Wars* via Wikimedia Commons w.wiki/32PB adapted using the Wikipedia app

Chapter 5

Experiencing your future

So, tell me, are you experienced? Why is experience valuable and what kind of experience are employers looking for anyway? How can you get some more experience?

5.1 What you will learn

By the end of this chapter you will be able to

- Describe why having experience can improve your chances of getting interviews
- Identify what counts as experience and why it's valuable
- Recognise opportunities to get more experience before you graduate

5.2 Why is experience so valuable?

It's common for students to be focused on their grades, whether those grades are low, middling or or high. At the extremes, if you have got lower grades than you'd like, you might be anxious or unhappy about them. If you've got higher grades, you're probably focussed on keeping them high. Either way, you are *much more* than your grades, because your education is only a part of who you are. You are the sum total of your experiences, this is one of the reasons that experience is so valuable, see figure 5.3

Your experience tells a story about who you are and what you're capable of.

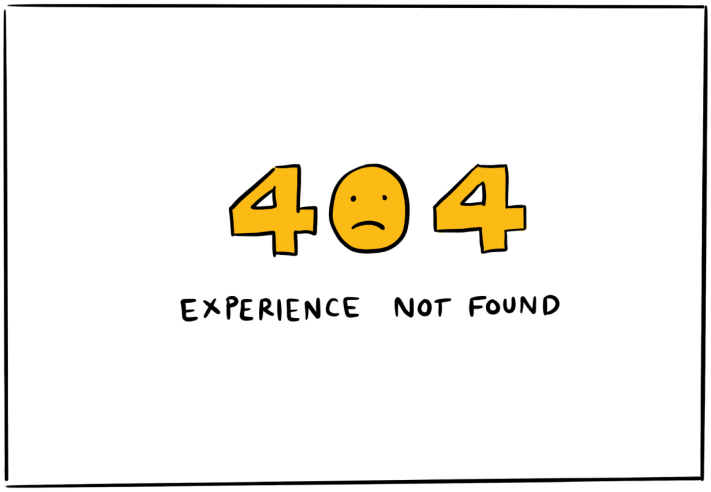


Figure 5.1: Do you respond with a sheepish `experience not found` error message when people ask about your experience? Is your experience like the classic page not found `HTTP 404`? The client sent you a valid request for your experience, but your server couldn't find it. Awkward. Embarrassing silence? Don't worry, there are some simple and easy ways to build your experience so that instead of negative 404's, you can respond with a cheerfully positive `200 (OK)`, as described in this list of HTTP status codes. We'll look at some of them in this chapter. Experience not found sketch by Visual Thinkery is licensed under CC-BY-ND

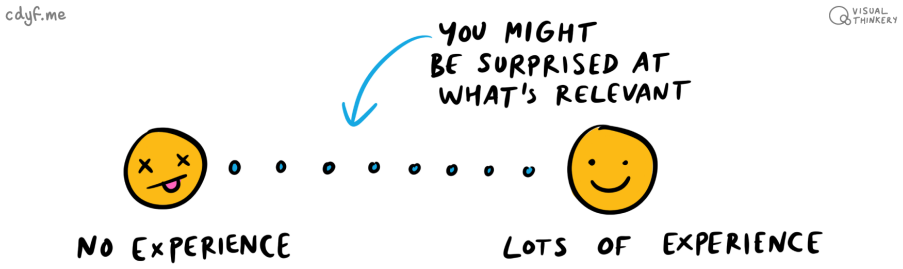


Figure 5.2: You might be surprised by which of your experiences are relevant, and what kinds of experience are relevant on your CV. What's relevant sketch by Visual Thinkery is licensed under CC-BY-ND

"experience" generally refers to know-how rather than descriptive knowledge (or in other words, on-the-job training rather than know-what or facts)

Experience

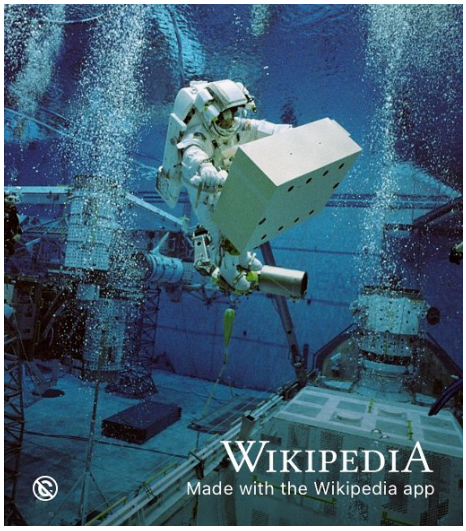


Figure 5.3: Experience is one of the best ways to develop *know-how*. While your formal education and academic study can help you develop *know-what*, you need to complement this knowledge with a range of experiences and on-the-job learning. This astronaut is training to work in microgravity by completing tasks underwater in a space suit. Like the astronaut, your education needs to combine academic study, with practical experience on-the-job. Public domain image of Christer Fuglesang training in the Neutral Buoyancy Laboratory (NBL) by NASA on Wikimedia Commons w.wiki/3WBf adapted using the Wikipedia App.

Table 5.1: Are you experienced? Terms used throughout this guidebook to describe experience, employment and their definitions

Experience	Description
Casual	Part-time or casual work, for example in hospitality or retail etc
Voluntary	Unpaid, both in technical and non-technical roles
Entrepreneurial	Self-employment, freelancing, contracting, "moonlighting" in a [side job](https://e
Insight	Usually no contract of employment. One to three weeks, sometimes known as wor
Internship	Fixed term contract of employment, typically 3 months full-time over summer, bu
Placement	Fixed term contract of employment, typically 12 months long and an assessed par
Graduate job	Full-time permanent contract typically working in one department of an organisat
Graduate scheme	Full-time permanent contract. Fast-track or high-flier managerial scheme, in your

5.3 Are you experienced?

So what counts as experience? Employers use terms to describe jobs and experience for undergraduates and graduates inconsistently. So I've defined and outlined terms for relevant kinds of experience shown in table 5.1 and we'll use these definitions throughout the book.

Some of the experience outlined in table 5.1 was probably what you were already thinking of as experience, however there are three important sources of experience that students often overlook:

1. **Voluntary work**: Any kind of work where you've given your time and labour to a community. That could be non-technical (working for a charity) or technical, such as contributing to open-source software, see section 5.3.3 and figure 5.4
2. **Casual work**: Working in hospitality or retail (etc) is often overlooked by students as an important source of relevant experience. It doesn't have to be technical to be relevant to employers, see section 5.3.5
3. **Student societies** Your students' union will have hundreds of official societies you can get involved in, and they'll be plenty of unofficial fringe communities too. As well as helping you develop new or existing interests, these societies give you an opportunity to serve a particular community of interest. Many societies seek members to take on positions of responsibility, above and beyond merely participating in their events

Before we discuss these experiences, lets look at some of the more conventional places for getting experience:

Figure 5.4: Volunteering is a great source of experience that employers value. That could mean volunteering for charitable causes, taking responsibility in a student society or getting involved in open source projects. Picture of volunteers cleaning up after Hurricane Sandy in 2012 by Jim Henderson via Wikimedia Commons w.wiki/3Z96 adapted using the Wikipedia App.

5.3.1 Big name experience

It's probably easier than you might think to get a big tech or big employer name on your CV. For example, many large employers run insight days, vacation schemes and spring weeks. These are often aimed at first year undergraduates, and are sometimes less competitive to get into than a longer term commitment such as a summer internship, year-long placement or even graduate job. A big name on your CV early in your degree can help it stand out later, as fluff bucket the grinning cheshire cat demonstrates on their CV shown in figure 5.5.

Other ways to get a big name on your CV include joining a big name competition or event, for example:

- Amazon hosts the Alexa challenge, see developer.amazon.com/alexaprize and the AWS Educate Challenge aws.amazon.com/education/awseducate/university-challenge
- Apple hosts the Swift Student Challenge developer.apple.com/wwdc21/swift-student-challenge
- Facebook has hackathons, see facebook.com/hackathon and developers.facebook.com
- Google hosts several events including:
  - Code Jam, HashCode and Kick Start codingcompetitions.withgoogle.com

Figure 5.5: It's easier than you might think to get a big name on your CV, sometimes these can help your application stand out from the competition. Big name sketch by Visual Thinkery is licensed under CC-BY-ND

  - Summer of Code summerofcode.withgoogle.com (Googler, 2021)
  - Developer Student Club Leads developers.google.com/community/dsc/leads
  - Inside Look buildyourfuture.withgoogle.com/programs/inside-look

- IBM hosts the annual Call for Code developer.ibm.com/callforcode unlike other competitions, these have a corporate social responsibility (CSR) themes for the benefit of society at large
- Microsoft hosts the Imagine Cup imaginecup.microsoft.com
- There are many other big employers that sponsor competitions, you can find them listed at devpost.com, Major League Hacking mlh.io and Hacker Earth hackerearth.com etc

Big names can look good on your CV, but they are not the only way to make your CV stand out.

5.3.2 Small name experience

Any experience will help your CV stand out. Smaller employers have the advantage that they tend to be less picky than big names so it is often easier to get a foot in the door. It might not be what you see yourself doing for long, but the experience gained in a small company can be invaluable.

5.3.3 Open source experience

Open source software projects are a great way to get some solid experience of software engineering, see for example Why Computing Students Should Contribute to Open Source Software Projects. (Spinellis, 2021) There's two ways to get started:

1. Raise a new issue via the project's issue tracker, such as github issues (Octocat, 2020)
2. Fix a bug by picking existing issues. (Robertson, 2020) It might sound trivial, but fixing a bug demonstrates that you can collaborate with others, understand the architecture and toolchain being used (which might be complex) and solve problems. See firstcontributions.github.io and the `<good first issue>` tag which helps new contributors identify starting points, see goodfirstissue.dev for some aggregated examples.

There are lots of different motivations for getting involved in open source, shown in figure 5.6. Whatever your motivation, contributing to open source software is fun, you'll learn heaps and it will look *great* on your CV. Open source software is widely used by, so contributing is a great way to get some real world experience of software development. Many open source projects are funded by employers both large and small, and you can get paid to develop open source software through projects like Google's Summer of Code. (Googler, 2021)

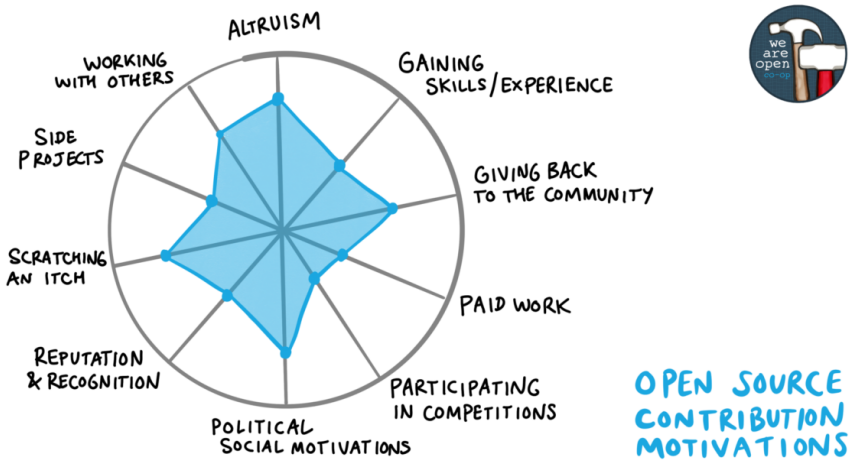


Figure 5.6: There are lots of good reasons for getting involved in open source software, gaining skills and experience of real software engineering in the wild is just one of them. Open Source Motivations by Visual Thinkery is licensed under CC-BY-ND

If you're looking for a project to get stuck into, here are Diomidis Spinellis tips for getting started (Spinellis, 2021):

1. Choose a project with several active contributors, so that there is a community to help you
2. Choose a relatively popular project (with some GitHub stars) so that you can avoid abandonware but…
3. … Avoid "blockbuster" projects like tensorflow or vscode , so that your contributions will not get lost in the politics and bureaucracy of a large project
4. Verify that you can build and run the project from your own setup
5. Ensure the project regularly accepts pull requests from outsiders, so that yours contributions will have a chance of being accepted
6. Contribute a trivial fix to start with to test your ability use the project's workflows

5.3.4 Voluntary experience

Experience as a section of your CV usually means *paid* work. However, experience in the context of this chapter means anything where you can show you've been part of a bigger team, taken responsibility for something or tried to make the world a better place somehow. These include:

- Volunteering: Doing voluntary work is a good way to pick up new skills
- Being involved in societies: e.g. taking responsibility for things in a society
- Getting involved in a community, either physical or online

5.3.5 Casual experience

You may already have experience of paid employment as a casual or part-time worker. This could include jobs such as waiting tables, serving in a bar or working in other areas of hospitality or retail, for example as a Saturday job.

It is important to recognise that these jobs have value. Many students make the mistake of overlooking their casual work experience because they disregard it as non-technical or consider it "low-skilled". In the section 7.5 we saw that one of the stories you want to tell in your job applications is that you:

1. take responsibility
2. achieve things
3. are nice to have around

Doing casual work can demonstrate all of these things. For example, from the ages of 12 to 18 I was a paperboy, except unlike the one selling newspapers in

A paperboy is someone – often an older child or adolescent – who distributes printed newspapers

Paperboy

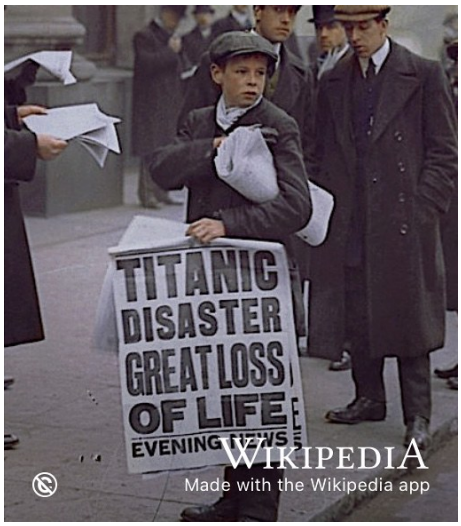


Figure 5.7: Casual and part-time work tell an important story about you on your CV. For example, from the age of 12, I was a paperboy, delivering newspapers directly to the doors of paying customers. This demonstrates reliability and work ethic, because I did this in all weathers (sun, wind, rain, snow, hangovers etc) for six years. If you have casual experience like this, don't forget to include it in your CV. Public domain image of the Titanic paperboy, Ned Parfett selling newspapers in London via Wikimedia Commons at w.wiki/35HA

the street in figure 5.7, I delivered newspapers directly to the doors of paying customers every morning. This was not a particularly highly skilled job, but it *does* demonstrate:

1. work ethic: getting up early *every* morning (including Saturdays). Sometimes work is about just turning up everyday!
2. taking responsibility and being reliable
3. understanding the value of money by earning a wage

I've often spoken to students who neglect to tell me about their paid work in retail or hospitality. "But it's not technical" they say, "it's low skilled and irrelevant". However, serving customers demonstrates your ability to provide good customer service and work as part of a team, often under pressure, see figure 5.8. This is good evidence of the "nice to have around" bit that Jonathan Black refers to (Black, 2019) and is something your formal education will not typically provide any evidence of. So don't fall into the trap of discounting the value of casual or part-time labour.

Figure 5.8: Early in your career, casual work in hospitality or retail, such as a supermarket like Budgens where I used to work as a teenager, is worth mentioning on your CV. If you have any experience of this kind, make sure you mention it and describe the skills you developed. Think carefully about the verbs can you use to describe casual experience, see chapter 18.

5.4 Breakpoints

Let's pause here. Insert a breakpoint in your `code` and slowly step through it so we can examine the current values of your variables and parameters.

```
* PAUSE
```

- What experience do you have to date?
- What activities could you do to get some more experience?

```
* RESUME
```

5.5 Summarising your experience

Too long, didn't read (TL;DR)? Here's a summary:

This chapter is under construction because I'm using agile book development methods, see figure 5.9.

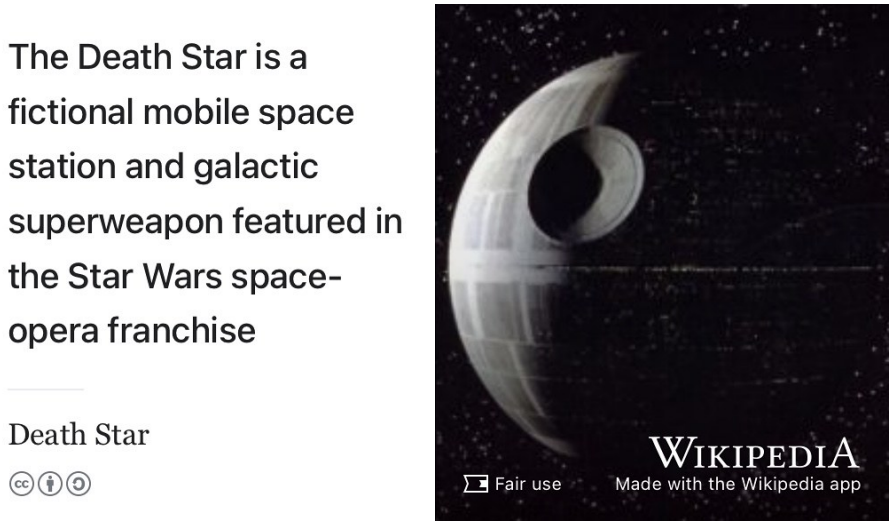


Figure 5.9: Just like the Death Star, this ~~galactic superweapon~~ chapter is under construction. Image of agile weapon engineering in *Star Wars* via Wikimedia Commons w.wiki/32PB adapted using the Wikipedia app

Chapter 6

Computing your future

It's difficult to think of any aspect of our lives that hasn't been changed by the invention of the digital computer, just 70 short years ago. Consequently, computing is a crucial skill in a wide range of careers across every sector of business and society. You don't have to have studied Computer Science at University to take advantage of all the exciting opportunities provided by computing. This chapter looks at why computing is a subject for everyone. If you're studying computing, this chapter isn't aimed at you, unless you are struggling to stay motivated with your subject!

6.1 What you will learn

Reading this chapter and doing the activities will help you to:

- Identify where you can get started with computing, if you're not studying computer science as a major part of your degree
- Describe why NOT studying computer science doesn't necessarily "lock you out" of computing as a career

But why should everyone be studying computing? There are social and economic arguments:

6.2 Computing is for everybody

At school, everyone learns to read, write and do maths. These are sometimes known as the three Rs but:

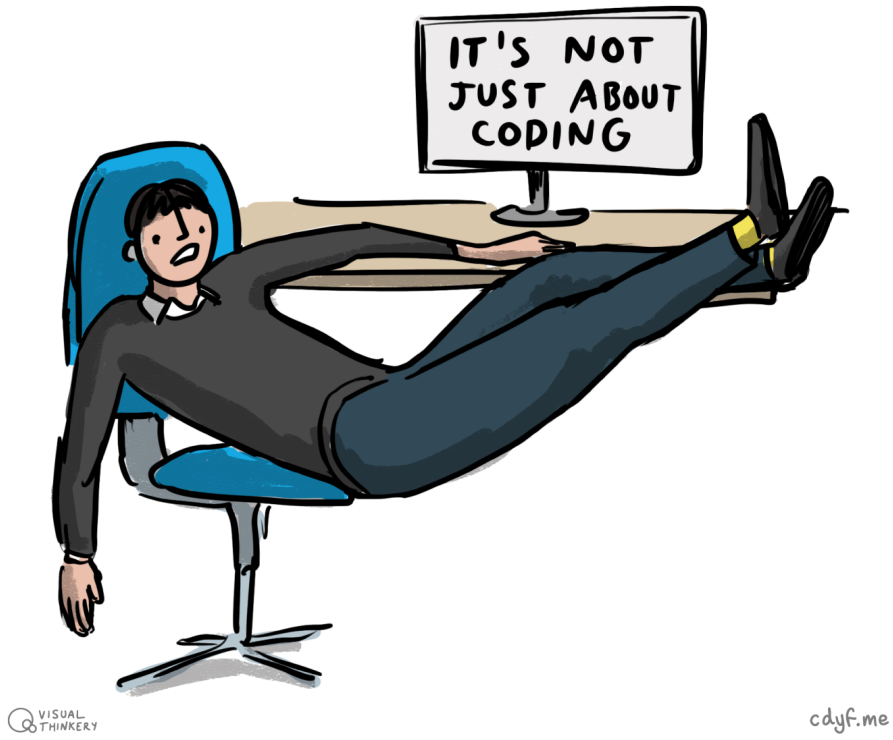


Figure 6.1: Computing is much more than coding, this chapter looks at what computing can do for your future. CV work sketch by Visual Thinkery is licensed under CC-BY-ND

- Why did you learn to read and write? Was it so that you could become a professional writer?
- Why did you study mathematics? Was it so that you could become a professional mathematician?

Of course not, that would be ludicrous! You learned to read and write because they are fundamental tools for expressing yourself and communicating with other people. You studied maths so that you could develop numeracy, reason about the world around you, analyse data and solve problems.

So why should everyone learn about computing? Is it so that everyone can become software engineers? Again, this is patently ludicrous.

Everyone should study computing for the same reasons everyone studies maths and english at school. Like writing, computing is one of the most creative tools for expression and communication that we have today. Just like mathematics, studying computing will also help you to solve important problems too. Sam Aaron, creator of Sonic Pi, puts exactly this case for creative computing in his TEDx talk (Aaron, 2016) shown in figure 6.2.

Figure 6.2: Sam Aaron puts the creative case for computing by discussing programming as performance in his TEDx talk. (Aaron, 2016) The image in this figure is a screenshot, watch the 18 minute video on programming as performance here.

Computing is also an intellectually stimulating and challenging subject to study in its own right. If you don't believe me, I'm not going to make the case here. Have a look at Silvio Peroni's free computational thinking and programming book at comp-think.github.io which is written for people with a background in the humanities. (Peroni, 2021)

6.3 Software is eating your future

Whatever future world you enter into after you graduate, there's a good chance it has already been eaten by software. In 2011, the software engineer and billionaire investor Marc Andreesen outlined why (in his opinion, figure 6.3) software is eating the world, in *The Wall Street Journal* (Andreessen, 2011).

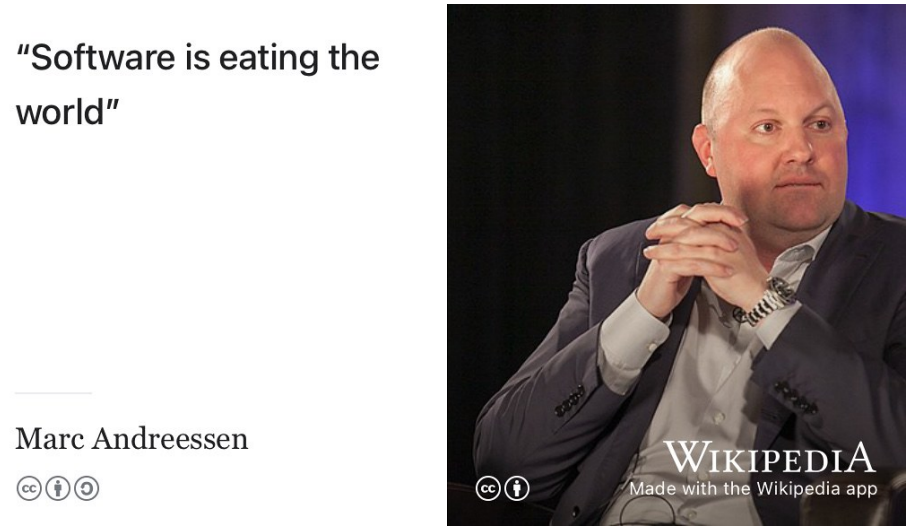


Figure 6.3: Whatever world you enter after you graduate, software has either eaten it, currently eating it or working out how to do so. Andreessen explains why software is eating the world and your future with it. Portrait of Marc Andreesen by JD Lasica on Wikimedia Commons w.wiki/3V48 adapted using the Wikipedia app

Unfortunately, many people lack the digital skills required to take advantage of all the opportunities provided by software. Robert Sedgwick at Princeton University has, like many others, argued that Computer Science should be required of all undergraduate students. (Sedgwick, 2019) We're not there yet because computing is a subject that has historically been siloed in Computer Science Departments, but this is changing as we'll see in this chapter. It's not that everyone should jump ship to Computer Science, but that:

- Computing is too important to be left to Computer Scientists
- Computing is too important to be left to men (Spärck-Jones and Runciman, 2007)

Whatever subject you are currently studying, adding some computing to your education will empower you with the computational thinking skills you need to be an active creator, not just a passive consumer in modern society. Computing can open up new opportunities for you and improve your social mobility.

6.4 Computing is eating the world

Besides the social arguments, there are also strong economic reasons for studying computing. It's not just software that's eating the world, but its combination with hardware that dominates the list of the world's largest corporations by market capitalisation, shown in figure 6.4. What use is software without hardware?

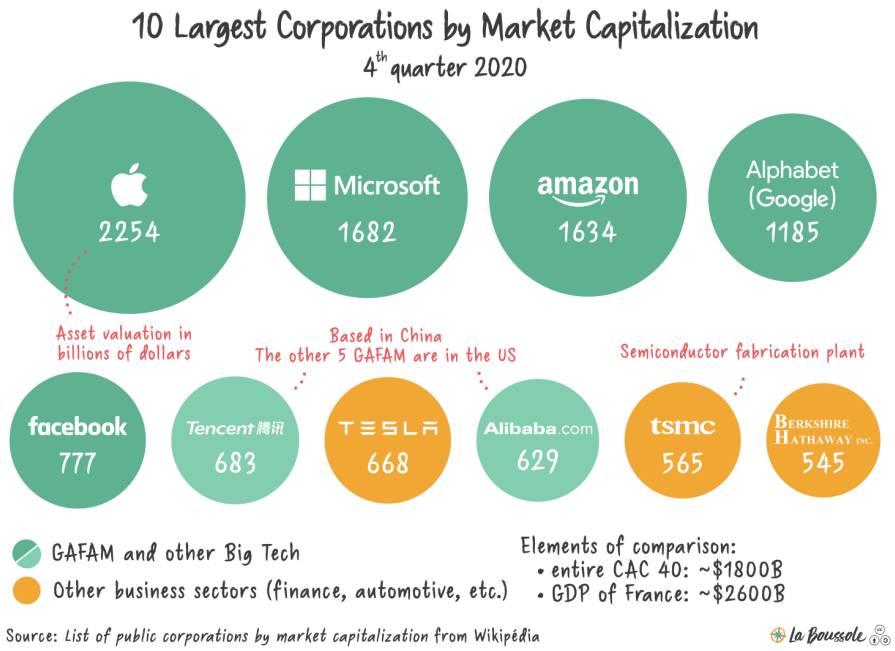


Figure 6.4: If stock markets are anything to go by, computing is eating the world. It would be impossible for Apple, Microsoft, Amazon, Google and Facebook to exist without computing. The economic weight of Big Tech graphic by YBSLE on Wikimedia Commons w.wiki/3KEU, note that Saudi Aramco has mistakenly been left of this map. The Visual Capitalist gives a more complete picture. (Ross, 2021)

Even if you don't want to work for any of these global monopolies, their success is good news for *all* students of computing because it shows how important computation is to society, both commercially and otherwise. For example, demand for software developers is high, comparable to teaching and nursing in terms of raw numbers. In the UK, the most common jobs for graduates from 2017-18 are shown in Figure 6.5, based on data taken from an article on the graduate labour market in 2021 (Ball, 2020)

So there's lots of choice and lots on offer, wherever you are in the world.

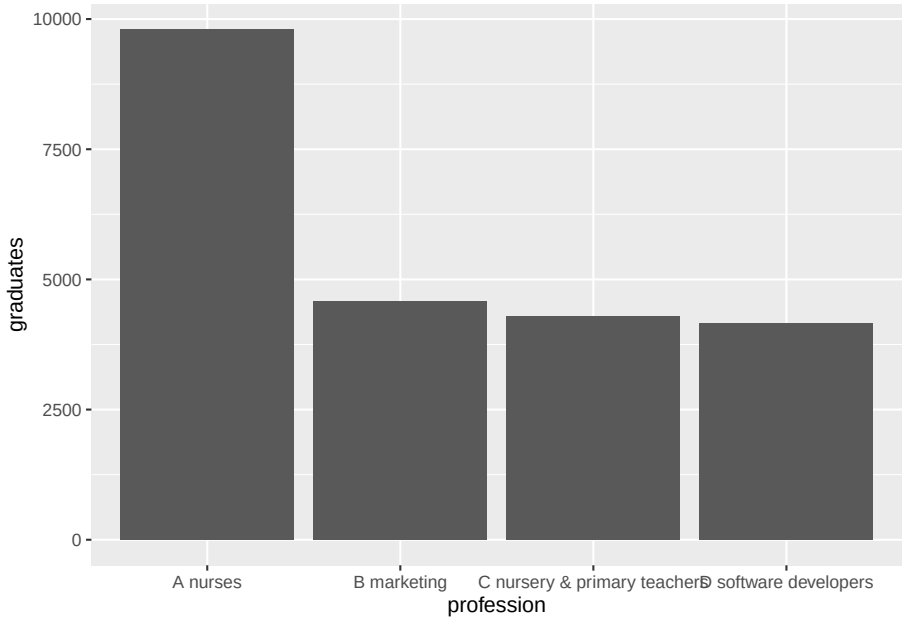


Figure 6.5: The most common jobs for graduates in the UK in 2017-18, demand for software developers is high according to data published by Charlie Ball (Ball, 2020)

6.5 Play your joker: Computational joker

Because of its social and economic importance, computing also gives you flexible career options. If academic disciplines are playing card suits, then Computer Science is the joker in the pack shown in figure 6.6. A versatile card, the computational joker can be played with (and without) any of the traditional four suits: diamonds, clubs, hearts and spades. That's because computing is a science *and* an art. It allows us to study human society and culture, so it's part of the humanities too (see digital humanities and computational social science for example). Last but not least, computing is also an engineering discipline and a branch of mathematics too. What all this means is that the computational joker is a wild card that can be played *whenever and wherever* you like, making it an incredibly powerful but dangerous card, depending on the game you are playing (see chapter 9).

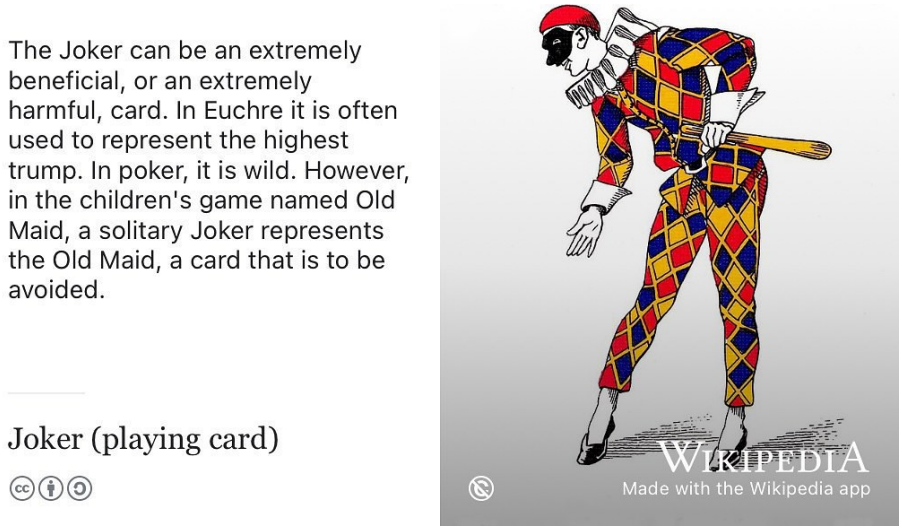


Figure 6.6: Computer Science: The joker in the pack. Public domain image of the Jolly Joker, a vintage Masenghini Italian playing card via Wikimedia Commons w.wiki/35EW adapted from the joker playing card using the Wikipedia app.

The flexibility of computing as a career means you have a broad range of options on where you can apply your computational skills. You don't *have* to be studying Computer Science to take advantage of these opportunities, but it helps.

6.6 Summarising computing your future

Too long, didn't read (TL;DR)? Here's a summary:

This chapter is under construction because I'm using agile book development methods, see figure 6.7.

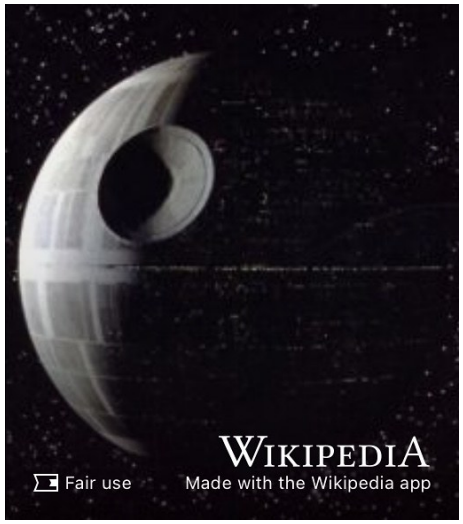


Figure 6.7: Just like the Death Star, this ~~galactic superweapon~~ chapter is under construction. Image of agile weapon engineering in *Star Wars* via Wikimedia Commons w.wiki/32PB adapted using the Wikipedia app

Part II

BUILDING

Chapter 7

Debugging your future

It's all very well *designing* your future but now you need to actually engineer it by *building* and *testing*. An obvious place to start is with your CV, because that's where most people get going. How can you create a bug-free CV, résumé or completed application form? How can you support applications with a strong personal statement or covering letter? These documents are crucial part of your future so how can you debug them?

7.1 What you will learn

By the end of this chapter you will be able to

- Structure and style the content of your CV and résumé appropriately
- Describe your story[1] of your relevant experience, projects and education etc
- Identify and fix bugs in CV's by:
  - Constructively criticising other people's CVs
  - Asking for, listening to, and acting on constructive criticism of your own CV
- Quantify and provide evidence for any claims you make you on your CV

7.2 Beware of the black hole

Before we get started, let's consider some advice from software engineer Gayle Laakmann McDowell. Gayle is an experienced software engineer who has worked

[1] actually a collection of short stories

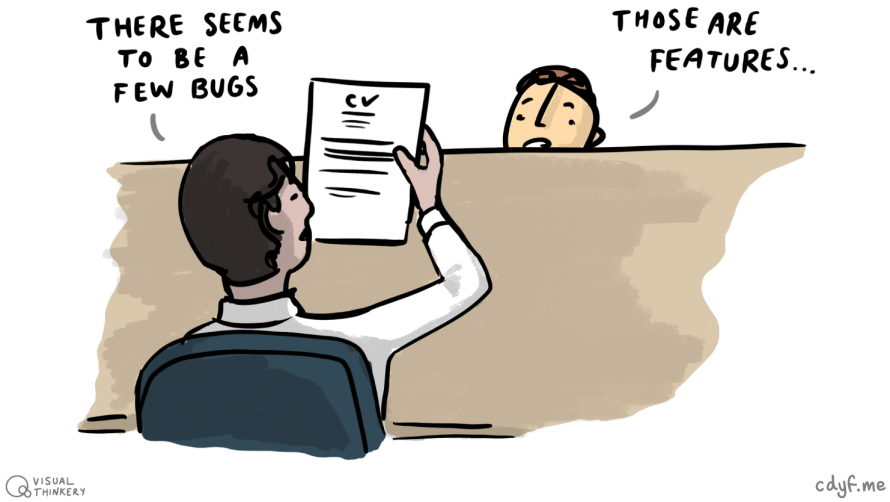


Figure 7.1: Is that a bug or a feature in your CV? To stand a chance of being invited to interview, you'll need to identify and fix any bugs in your written applications. If you don't, your application risks being sucked into a black hole and will never be seen again. Features not bugs picture by Visual Thinkery is licensed under CC-BY-ND

at the biggest technology employers in the world, Apple, Microsoft and Google. She's also authored a cracking series of books on technology careers, particularly *Cracking the Coding Interview* (McDowell, 2015) which we'll discusses in chapter 10. Gayle refers to the employer "black hole" described in figure 7.2.

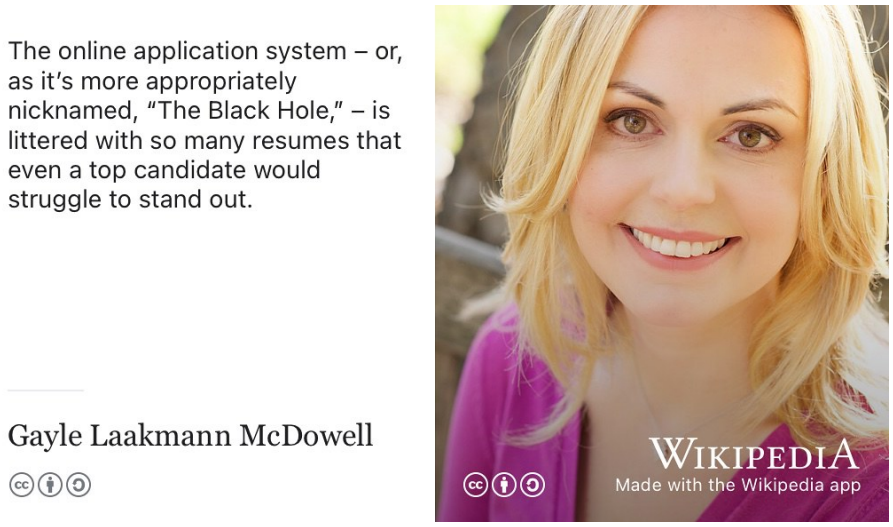


Figure 7.2: Beware of what software engineer Gayle Laakmaan McDowell calls the employer "Black Hole", especially if you're applying to large employers. "Getting through the doors, unfortunately, seems insurmountable. Hoards of candidates submit résumés each year, with only a small fraction getting an interview. The online application system – or, as it's more appropriately nicknamed, *The Black Hole*, – is littered with so many résumés that even a top candidate would struggle to stand out." (McDowell, 2011, 2014) Laakmann portrait by Gayle Laakmaan is licensed CC BY 4.0 via Wikimedia Commons w.wiki/wiu

If you're applying to big employers, you'll need to create a CV that is good enough to stand out before it disappears over the event horizon and into the employment black hole. It needs to be good enough to persuade an employer to invite you to an interview. You can start with an employer-agnostic CV but you may need to come back and revisit the issues in this chapter once you have identified some target employers, so that you can customise and tailor your CV and written applications.

7.3 It's not bug, its a feature

It's an age old trope in Computer Science that engineers use to cover their mistakes, passing off their accidental bugs as deliberate features of their work, see 7.3.

Figure 7.3: Do you have software bugs or undocumented features on your CV or résumé? Although tolerated in software, bugs in your CV, résumé and written applications can be fatal. Picture of gold-dust weevil *Hypomeces squamosus* by Basile Morin is licensed CC BY SA via Wikimedia Commons w.wiki/3E62

Nobody likes buggy software, but we unfortunately routinely tolerate badly-designed, low quality, bug-ridden software in our everyday lives. (Mann, 2002; Charette, 2005)

In contrast, buggy CVs are rarely tolerated, they will usually end up in the bin. Even a tiny defect, like an innocent typo, can be ~~fetal~~ fatal. Most employers (particularly large and well known ones) have to triage hundreds or even thousands of CV's for any given vacancy. This means they are looking for reasons to REJECT your CV, rather than ACCEPT it, because that's a sensible strategy for shortlisting from a huge pool of candidates for interview. A buggy CV, application and covering letter could ruin your chances of being invited to an interview, see chapter 10.

Like writing software, the challenging part of writing a CV isn't the *creation* but in the *debugging*. Can you identify and fix the bugs before they are fatal?

DISCLAIMER : If you ask three people what they think of your CV, you will get three different and probably contradictory opinions. CV's are very subjective and personal. The advice below is based on common sense, experience and ongoing conversations with employers. What makes a good CV will depend on the personal preferences and prejudices of your reader. There are some common sense debugging rules, which are described in this chapter.

While referring to this guide, remember that:

- The main purpose of your CV is to get an interview, not a job. Your CV should catch attention and provide talking points for an interview
- Your CV will be assessed in seconds, rather than minutes so brevity really is key
- Bullet points with verbs first (see section 7.6.4) will:
  - allow your reader to quickly scan your CV (employers don't read CVs, they scan them) (Richards, 2019a)
  - highlight your key activities
  - avoid long sections of prose (which the reader will very probably skip anyway) you can use your beautiful prose on your covering letter, see section 7.9

You're not trying to tell your *whole* life story from section 2.2 but to distill the essentials into several short stories which can be summarised into a handful of bullets or sentences. It's a bit like the blurb or synopsis on the back of novel, can you entice the reader into wanting to find out more?

7.4 Is it a bug or a feature?

Wherever criticism of your CV comes from, don't take it personally - it is probably one of the first you have written. Think of your current CV as an alpha or beta version that you continuously test, release and redeploy. There are many chances to debug and improve your CV during your study but before potential employers read it. The aim of this chapter is to help you improve your CV, whatever stage you are at. Employers often grumble that Computer Science graduates lack written communication skills. Written applications and CV's are a common example of this.

1. **EDUCATION**: Is your year of graduation, degree program, University and expected (or achieved) degree classification clear? Have you mentioned things you are studying now, not just courses you have finished? See section 7.5.2
2. **STYLE**: Does it look good, decent layout, appropriate use of LaTeX or Word or whatever? Are there any spelling mistakes, typos and grammar? Don't just rely on a spellchecker, some typos can only be ~~spitted~~ spotted by a human reader
3. **LENGTH**: Does it fit comfortably on (ideally) one page (for a Résumé) or two pages (for a CV)? See section 7.6.3
4. **STRUCTURE**: Is the structure sensible? Is it in reverse chronological order? Most important (usually recent) things first? Not too many sections or anything missing? See section 7.5
5. **VERBS FIRST**: Have you talked about what you have actually done using prominent `verbs`, rather than just what you think you know? Avoid long sections of prose, see section 7.6.4

6. **RESULTS**: Have you also demonstrated and *quantified* the outcomes of your actions where possible, see Context, Action, Result & Evidence (CARE) in section 7.6.2

7.5 Structure your CV

How you structure your CV will depend on who you are and what your story is. Recruiters at Google suggest four or five sections, that follow a header section. Before we look at those, lets look at some general points about CVs, watch the videos shown in Figure 7.4.

Figure 7.4: Recruiters at Google, Jeremy Ong and Lizi Lopez outline some tips and advice for creating your résumé. (Ong and Lopez, 2019) The image in this figure is a screenshot, you can watch the eight minute résumé tips video here.

As Jonathan Black, director of the careers service at the University of Oxford has pointed out, (Black, 2019) a key part of your story that you want to communicate in your CV is that you :

1. take responsibility
2. achieve things
3. are nice to have around

How can you demonstrate this? Watch the short video in Figure 7.5.

Quantify and provide evidence of any claims you make. This can turn meaningless assertions described in figure 7.5 into meaningful evidence. So for example:

Figure 7.5: Jonathan Black, head of the careers service at the University of Oxford, explains how to create a top notch CV by replacing meaningless assertions with meaningful evidence. (Black, 2019) The image in this figure is a screenshot, you can watch the 11 minute video on creating a top notch CV here.

```
* Achieved excellent results
```

...is a bit vague, what were the results exactly, can you measure them somehow? Or at least describe them?

```
* Worked in a team
```

So you worked in a team? `Worked` is vague. What was your role and contribution exactly? How long did the project last? How many people were on your team? What was the result of your action? What's the story?

7.5.1 Your header

The first thing in your CV is the header, a simple section giving your name, email, phone number and any links shown in the CV in Figure 7.6 for Alan Turing. That's it!

Your header doesn't need to include any more information than your name, email, phone and any links. This means your birth date, marital status, photo[2] and home address aren't relevant and you don't need to give multiple phone

[2] what you look like should *not* be a factor in an employers decision to interview you

Figure 7.6: Keep the header of your CV simple. Just your name, email, phone number and any relevant links are all you really need. Any additional information risks wasting valuable space and distracting your reader.

numbers or emails either, just one of each will do. If an employer wants to invite you to an interview, they'll get in touch by email, phone (or possibly LinkedIn) so other contact details are irrelevant at this point. After your header I suggest you have about five sections that cover some or all of the following:

1. Education: the formal stuff
2. Experience: paid work
3. Projects: personal, social, side or University projects
4. Leadership and awards
5. Optional section

Let's look at each of these sections in turn:

7.5.2 Your education

Unless you have significant amount of experience, the education section of your CV is likely to be the first real section, after the header. Your education section needs to strike a balance between:

- Describing in enough detail what you've studied and any projects you've completed at University as part of your formal education
- Keeping it short and sweet and avoiding getting bogged down in the details.

You've invested a significant amount of time and money in getting your degree. At this stage, your degree justifies more description than a terse one line `BSc Computer Science`. You'll need to say more than Pen Tester and Rick Urshion (for example) but not as much as Mike Rokernel, who has given *way* too much information on his degree. You don't need to name every single module and give a mark for each. Neither do you need to give your result to FOUR significant figures (it happens). Two significant figures will do just fine. You might like to pick out relevant modules, or the ones you got most out of. Employers like Google encourage applicants to emphasise courses on data structures and

algorithms, but you'll need to tailor your description to the role and be brief. On a one page CV, you might only have two or three lines to describe your higher education.

7.5.3 Your experience

Experience is where you can talk about any paid or voluntary work experience you have. Don't discount casual labour, such as working in retail or hospitality, these demonstrate your work ethic and ability to deal with customers, often under pressure. You are more than just a techie, so anywhere you've worked in a team is experience worth mentioning, even if that team was just two people. Two people is still a team.

Figure 7.7: Are you experienced? What have you done outside of your formal academic education? Experience sketch by Visual Thinkery is licensed under CC-BY-ND

If you don't have much experience, don't worry, there are plenty of opportunities to get some. For details, see chapter 5 *experiencing your future*.

7.5.4 Your projects

The `Projects` section of your CV is a where you can describe all other things you get up to. These might include:

- personal side projects
- social responsibility projects

- open source projects
- entrepreneurial projects
- University projects

They will most likely be unpaid because paid work fits better under the heading experience, see section 7.5.3. Perhaps you've completed some courses outside of your education such as a massive open online course (MOOC) or similar. Hackathons and competitions, fit well here too. (Fogarty, 2015) You don't *need* to have won any prizes or awards, although be sure to mention them if you have. Participating in hackathons and competitions clearly shows the reader that you enjoy learning new things. Demonstrating an appetite for new knowledge and skills will make your application stand out. If you're looking for some inspiration for side projects, Dani Stefanovic's build-your-own-x repository is a good starting point. Building and creating new things is a great way to understand them, just ask Richard Feynman shown in 7.8. One way of doing this is with open source projects which we describe in section 5.3.3.

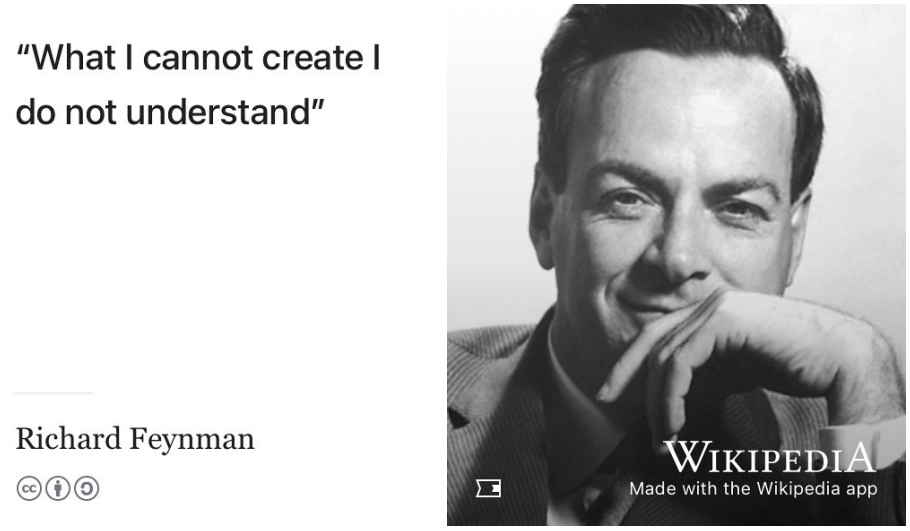


Figure 7.8: Physicist Richard Feynman once chalked "What I cannot create, I do not understand" on his blackboard at the California Institute of Technology while teaching the The Feynman Lectures on Physics. (Way, 2017) Creating software and hardware in personal side projects is a great way to build new understanding *and* help your CV stand out see github.com/danistefanovic/build-your-own-x. Public domain image of Richard Feyman by The Nobel Foundation on Wikimedia Commons w.wiki/3Xoy adapted using the Wikipedia app

Any longer projects you've done at University are worth mentioning. Your projects are important because they differentiate you from everyone else in your year group. Try to be *more* descriptive than this:

```
* First year team project
```

or perhaps

```
* Second year team project
```

or even just

```
* Final year project
```

By themselves, those project names are pretty opaque. They are OK for giving the context of your story but don't give the reader much else to go on. What was the story (the context, action, result and evidence (`CARE`) we described in section 7.6.2) of those projects? How many people were in your team? How long did you collaborate for? What did you build? What was it called? What did it do? What roles and responsibilities did you have in the team? Was their conflict in the team? How did you resolve it? How did you motivate the free-riders in the team to contribute?

This is all excellent CV fodder!

It's often better to describe what YOU did before you describe what the software, hardware or project did. Your reader is likely to be more interested in the former. Let's imagine you've developed a piece of software called `WidgetWasher`. You might describe it like this:

```
* WidgetWasher is a web service that washes widgets
* Makes use of an HTTP API and secret keys
* Tested WidgetWasher on a range of different operating systems
* Collaborated with one other contributor over two days
* Designed and implemented an API
```

Instead, you could reverse the order to change the emphasis like this:

```
* Designed and implemented an API
* Collaborated with one other contributor over two days
* Tested WidgetWasher on a range of different operating systems
* Makes use of an HTTP API and secret keys
* WidgetWasher is a web service that washes widgets
```

The latter has all the same information, but by reversing the order, you've emphasised what *you* did, rather than what the software did.

7.5.5 Your leadership & awards

If you can demonstrate leadership, you may want to dedicate a whole separate section for it. This is also a good place to add any prizes you've won. If you've been granted any interesting awards or honours be sure to mention them. You'll typically need a bit more than:

```
* Awarded scholarship / prize
```

Congratulations, but how many people were awarded that prize? How many applicants or entrants were there and what percentage were successful? Was it a regional, national or global award? How frequently is the award given? It is unlikely that your reader will have heard of the award unless it is widely known. So if you're going to mention awards, give the context where you can.

7.5.6 Your optional extras

If you have anything else you want to highlight besides your `education`, `experience`, `projects` and `awards` you *may* still have room for one more optional extras section. Try to come up with a better name than `Miscellaneous` (which sounds like a dumping ground) or you might decide to have a dedicated `skills` section but see 7.5.7.

7.5.7 Your skills?

You may be tempted to dedicate a whole section on your CV to skills, particularly the technical ones. Maybe it makes you feel good listing them all in one place like a stamp collection. If you're going to have a `skills` section, keep it short. Why? Let's imagine, that like Rick Urshion, you include Python in a long list skills, with its own dedicated section. There are at least four problems with Rick's not so skilful approach:

1. **No `Context`** to give the reader an idea of where he's developed or used his Python skills. Was it during his education, as a part of his work experience or his personal projects? We don't know because he doesn't say.

2. **No `Actions`** or *evidence* to back up his claims of having skill with Python. So Rick claims he knows Python. So what? What did he *do* with those python skills? We don't know because he hasn't told us.
3. **No `Results`** given for what the outcome of using the skills was. Did he save his employers some money? Did he make something more efficient? Did he learn some methodology? We will never know.

4. **No `Evidence`** to support his claims. Perhaps he DOES have Python skills, perhaps he DOESN'T. Is he telling lies and peddling fake news (see section 8.3.4)? It's difficult to tell.
5. **No `C.A.R.E.`** There's no story told for that skill, see figure 7.9. This makes for a very dull and boring read. Yawn. NEXT!

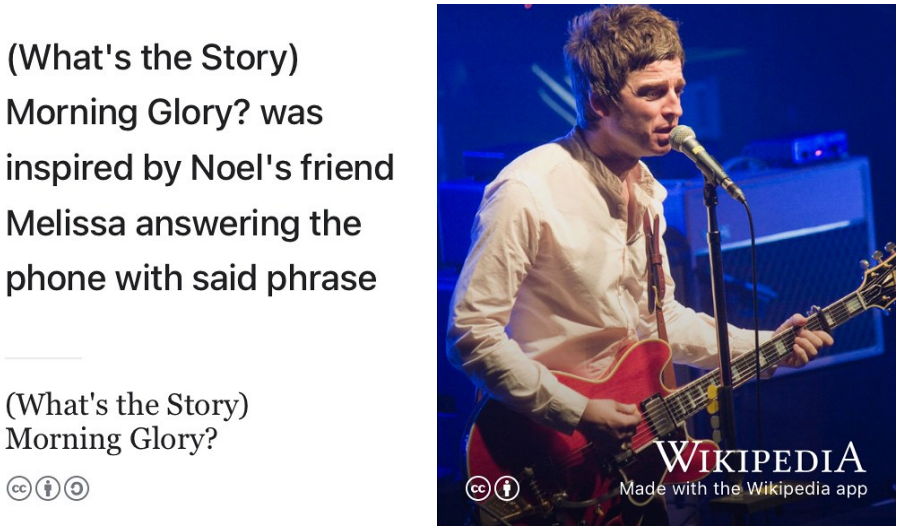


Figure 7.9: (What's your Story) ~~Morning~~ Coding Glory? What is the Context, the Actions, the Results and the Evidence for the stories that you are trying to tell? Show your `C.A.R.E.` in storytelling. CC BY portrait of Noel Gallagher by alterna2.com on Wikimedia Commons w.wiki/3bimy adapted using the Wikipedia app

So, this doesn't mean Rick shouldn't mention his Python skills. Where he can, he needs to give us the context, action, result and evidence (`C.A.R.E.`) of his story described in section 7.6.2. This will make his Python story much more convincing and interesting to read. Showing a bit of C.A.R.E. will improve his chances of being invited to interview.

This applies to soft skills too, not just hard technical skills. Best to mention the context in which you've used any skills you mention on your CV. So, if you're going to have a skill section:

- keep it short (one or two lines maybe) but personally I'd avoid dedicating a whole section to it
- stick to your strongest and most relevant skills that you are comfortable to answer questions on in your interview, rather than an exhaustive encyclopædic inventory

- avoid listing office products of Microsoft (e.g. Word etc), Google or Apple as a skill, they are *not* generally very interesting skill because everyone has them. Don't waste space talking about it unless you've done something interesting with them, like some advanced integration with other software. Cloud services are a slightly different matter, see section 11.4.1.

If you're a computer scientist, you also have demonstrable "meta" skills like the ability to learn things quickly. You can also think logically, reason, problem solve, analyse, generalise, criticise, decompose and abstract - often to tight deadlines. These computational thinking skills are future-proof and will last longer than whatever technology happens to be fashionable right now. Employers are often more interested in these "meta" capabilities and your potential than in any specific technical skills you may or may not have.

7.6 Birds eye view

Having looked at the sections you're likely to have, we'll take a birds eye view of your whole CV. The issues in this section apply to the whole of your CV, rather than individual sections.

7.6.1 Your style

Making your CV look good can take ages, but a well presented CV will stand out. While its worth making an effort to style carefully and consistently, you need to be be wary of the huge time sink of typography.

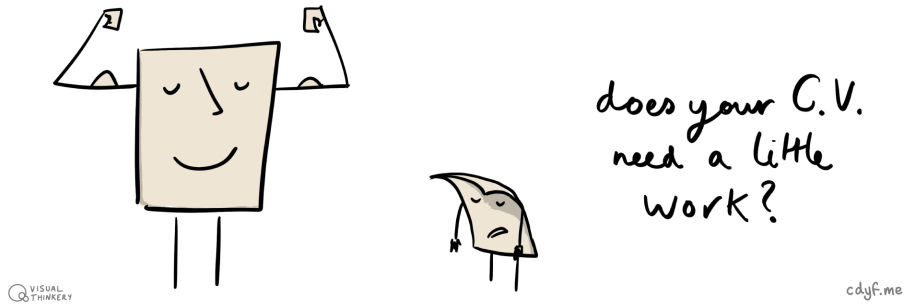


Figure 7.10: Does your CV need a little work? The truth is your CV is never finished, you will be continuously developing, debugging and releasing it throughout your life. It's such a crucial document because it will determine if you are interviewed, so its important to spend time getting it right. CV work sketch by Visual Thinkery is licensed under CC-BY-ND

Whatever your typographical style is, portable document format (PDF) is the safest way to deliver it. It's called portable for a reason. While Microsoft Word is fine for editing, it is difficult to ensure that a Word document doesn't get mangled by transmission via the web or email. PDF is much safer, you can be more confident that it will work well on a range of different operating systems and devices. Try opening a Word document on any smartphone or tablet and you'll see what I mean. It helps if you can give the file a descriptive name so `ada_lovelace.pdf` is a better filename than `my_cv.pdf`.

It's fine to author your CV in Microsoft Word, but you'll want to save as PDF to make it more platform independent. LaTeX and overleaf can be used to create professional PDFs and have many templates, see Getting started with LaTeX (LaTeX4year1) if you've not used LaTeX before, or you need to refresh your memory. (Hull, 2020)

7.6.2 What's your story, coding glory?

One way to structure descriptions of items within each section of your CV is to use **C**ontext, **A**ction, **R**esult and **E**vidence (C.A.R.E.) to tell your stories. This method can also be useful for structuring answers to interview questions, especially if you get nervous. So for example, rather than just listing `Python` as a skill, you should tell the reader more about the context in which you've used python, what you actually did with it and what the result was. You really need to spell it out.

- **CONTEXT**: So you've used Python, but in what context? As part of your education? For a personal project? As a volunteer? In a competition?
- **ACTION**: What did you *do* with Python? Did you use some particular library? Did you integrate or model something?
- **RESULT**: What was the result and how can you measure it? You picked up some new skills? What was the impact? Perhaps you made something that was inefficient and awkward into something better, cheaper or faster? Some things are hard to measure but you should quantify results where you can.
- **EVIDENCE**: Where evidence exists, you should point to it. It might be a certificate, a badge (see chapter 12) or the actual software itself

You don't have to stick rigidly to the order C.A.R.E. as long as they appear somewhere. For example, recruiters at Google (see figure 7.4) advise candidates to describe their experience and projects using this simple pattern:

```
* Accomplished [X], as measured by [Y], by doing [Z]
```

Where *accomplished* is `Result`, *measured by* is `Evidence` and *doing* is the `Action`. So instead of just saying:

```
* Generated reports for end users
```

You could say:

```
* Generated daily reconciliation report for team by automating workflow of 8 different
```

The latter is better because it is more specific, captures the result (`accomplishment`), by giving evidence (`8 different tasks`) and talks about the actions (the `doing` part). Choose the verbs you use carefully, see chapter 18 for examples.

7.6.3 Your length

How long should your CV be? Many people start with a two page CV, which is a sensible starting point shown in figure 7.11. It is also advisable to create a one page Résumé. (David, 2017)

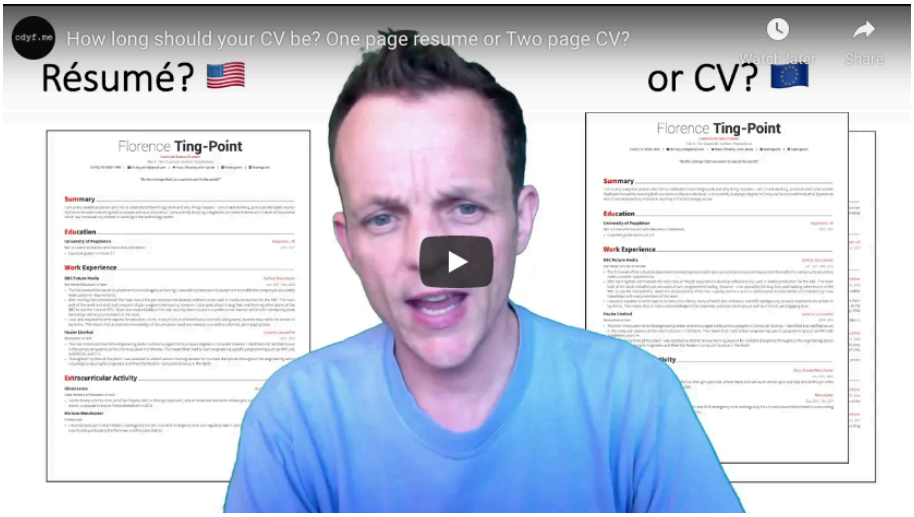


Figure 7.11: How long should your CV be? Should you write a two page European style CV or an American style résumé (one pager)? (Hull, 2021) The image in the figure is a screenshot, you can watch the five minute video on how long your CV should be here.

At this stage in your career you *should* be able to fit everything on to one page. However, it can be challenging and time consuming squeezing it all on, see figure 7.12.

"I would have written a shorter letter, but I did not have the time."

Lettres provinciales

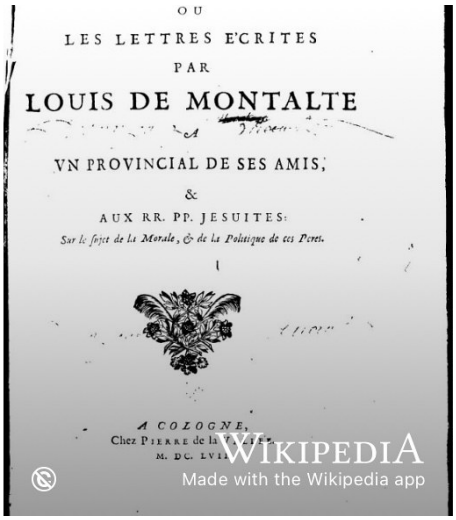


Figure 7.12: I would have written a shorter ~~letter~~, ~~CV~~, Résumé but I did not have the time. This quote (or meme) is frequently attributed to Blaise Pascale's *Lettres provinciales* (O'Toole, 2012). Public domain image by Gallica on Wikimedia Commons w.wiki/3Uzn

It takes more time to write less. Writing a one page résumé is a valuable exercise, because it forces you to distill and edit out any filler or fluff, which you sometimes find on two page undergraduate or graduate CVs. It is much better to have a strong one-page résumé than a weaker two-page CV that is padded out with filler to make up the space, as described in the video in figure 7.11. Adding more features (pages and content) to your CV doesn't necessarily make it better. Sometimes adding more features to your CV will make it worse, as shown in figure 7.13.

If you're struggling to fit all the information onto a one page résumé, revisit each section and item carefully. Is there anything you can drop? Can you save a wasteful word here, or a lazy line there? Check for any spurious line breaks because every pixel counts. Don't throw your two page CV away, it is still a good store of stuff you might want to add to customised one-page résumés.

7.6.4 Verbs first: lead with your actions

A simple but effective technique for emphasising what you have done, rather than just what you know, is to start the description of it with a verb. Employers don't just want to know what you know, but what you have actually done. So, for example, instead of saying e.g.

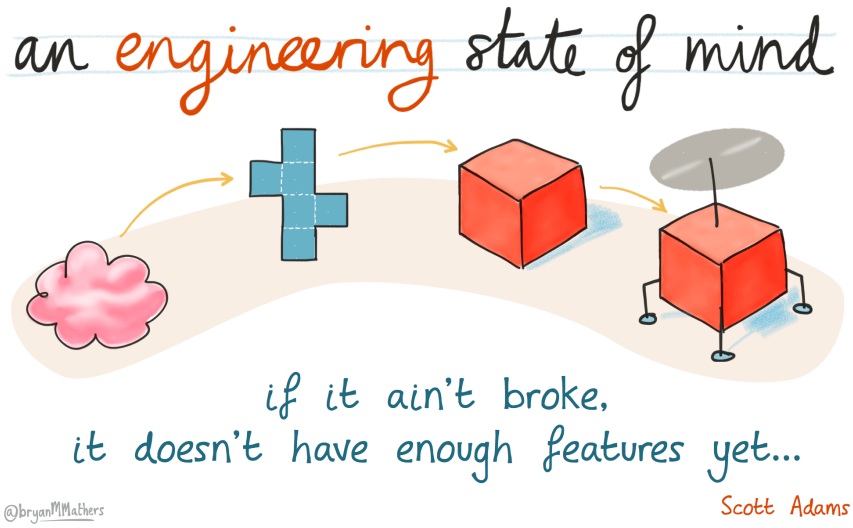


Figure 7.13: If it ain't broke it doesn't have enough features yet. Adding more features to software doesn't necessarily make it better. Likewise, adding more pages and content to your CV or résumé won't always improve it. It's often better to be precise and concise, rather than bloated and potentially more buggy. An engineering state of mind by Visual Thinkery is licensed under CC-BY-ND, with help from Dilbert cartoonist Scott Adams

```
* In my second year CS29328 software engineering module I used Java, Eclipse and JUnit to test ar
```

you could say:

```
* Built and tested a large open-source codebase using Eclipse, Ant, JUnit and Jenkins"
```

followed by:

```
* Added and deployed new features to a Massively Multiplayer Online Role-Playing Game (MMORPG) ir
```

The latter examples get to the point much quicker and avoid the problem of using `I, me, my...` too much which can sound self-centred and egotistical. Although your CV is all about you so it is natural to have a few personal pronouns in there, but too many can look clumsy and give the wrong impression. Choose the verbs you use carefully, see chapter 18 for examples.

7.6.5 Your links

Augmenting your CV with web links (hyperlinks) can add important context to your story, without adding too many words or taking up valuable space. An example using LinkedIn is shown in figure 7.14

Figure 7.14: Adding links is a good way to augment your CV. If you're adding LinkedIn, make sure you customise your public profile URL, to remove the default random alphanumeric string at the end, like the `038b37` example here. (Hoffman, 2020) You can also remove any ugly `http::`, forward slashes `//` and `www` in URLs which are distracting noise. Just make sure links are clickable and actually work. Neither do you need to waste valuable space telling people what the link is, like in the first example, the domain name already tells you its a LinkedIn profile.

Links are also a great way to add evidence and substantiate any claims that you make. They allow your reader to *read between the lines* and make inferences from the information you've provided them with. For example, you might says things like:

```
* Built a thing called example.com
```

Reading between the lines: "I like building things. Look at this thing I built just for fun, its really cool". Or you might say:

```
* Elected as a representative for hacksoc.com
```

Reading between the lines: "I was part a bigger thing you might not have heard about but you can find out about here". You might also say:

```
* Competed at hack-to-the-future.com part III
```

Reading between the lines: "I really enjoy learning from other people by going to hackathons and competitions"

… and so on. So links are crucial features of your CV and an interested reader *may* even follow them. Treat links with respect and they will support your goals and help your readers. Invest some time thinking about how you word the link text, and how they would be understood out of context. Make sure that:

- your hyperlinks are readable and descriptive (Richards, 2019b)
- your hyperlinks are clickable in the PDF. Don't expect your reader to cut and paste (or even type) URLs, they are too busy. If they are clickable, people are much more likely to follow them
- your hyperlinks are paper-proof. Some people still print CVs so the phrase `click here` won't work well on printed paper. See to print or not to print — a CV, that is (Garone, 2014)

Besides LinkedIn you could include public profiles from github.com, devpost.com, hackerrank.com and stackoverflow (Hamedy, 2019). You can also link to personal projects or your blog if you have one. Obviously, you need to be careful about what you link to and what employers can find out about you online. They *will* Google you. So keep it professional and, as we discussed in a section 3.6, be wary of social media.

7.6.6 Robot proofing your CV

It's a good idea get feedback from as many different sources as you can on your CV. By *sources* I don't just mean humans, but also robots. Larger employers will use automated Application Tracking Systems (ATS) to log and trace your application. These "résumé robots" (if you like) are unlikely to have arms and legs like the one in Figure 7.15, but they *will* be looking for keywords and

standard headings in your CV. You can get automated feedback from a range of different automated systems, though it is a good idea to remove any personal information like phone numbers and emails before using these free services. You might also want to check what the services privacy policy says about what they do with you personal data. Résumé robots include:

- careerset.io, a free service provided by a UK based company, careerset Ltd.
- resume.io, a free service provided by a Dutch company, Imkey BV
- jobscan.co, a free service provided by an American company

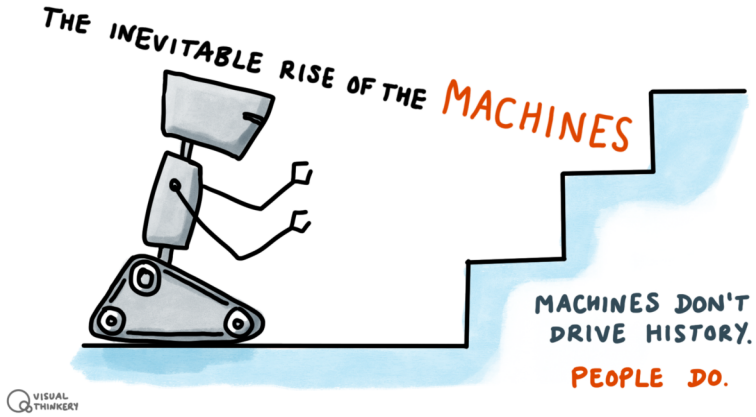


Figure 7.15: Although they often struggle to get up the stairs, résumé robots are likely to play an important role in any decisions to invite you to a job interview, especially if you're applying to bigger companies. Make sure your CV is résumé robot friendly by feeding it through a robot . Machines by Visual Thinkery is licensed under CC-BY-ND

Besides providing feedback on the content of your CV, using these systems can help address issues such as the use of tables or layout which may cause problems for some systems. For example, some systems ignore the second column of a two-column CV because they can't identify it. Some things to check with automated CV screens:

- Have you used standard headings for the sections? Non-standard sections maybe ignored or misunderstood
- Have you used appropriate verbs to describe your actions?
- Is your layout and design robot friendly? Sometimes tables and two column layouts can get horribly mangled, see what happens to tables and columns in an applicant tracking system (Shields, 2019)

7.6.7 Your references

You might be tempted to put your referees details on your résumé. Don't bother because;

- references waste valuable space. You can say much more interesting things about yourself than who you referees are
- references aren't needed in the early stages of a job application. They are typically taken up much later, when you've been offered or are about to be offered the job
- references give personal information out. Do you really want to be giving personal details out to anyone that reads your CV? It could easily be misused.

It's not even worth saying `references available on request` - that just wastes space as well and is implied information on every CV anyway.

7.7 Breakpoints

Let's pause here. Insert a breakpoint in your `code` and slowly step through it so we can examine the current values of your variables and parameters.

```
* PAUSE
```

- How long is your CV? How long should it be?
- Should you have a personal statement on your CV, like Mike Rokernel has for example?
- One column or two column layout?
- Should you put education or experience first? Which is most important?
- How many of my hobbies and personal interests should I list? (Cheary, 2021)
- How many employers actually read cover letters?

```
* RESUME
```

7.8 Checklist: Big Bad Bugs

Here is a quick check-list for debugging your CV before you send it off to an employer:

1. Has it been reviewed by several people? Because according to Linus's law "given enough eyeballs all bugs are shallow" (Raymond, 1999)

2. Have you reviewed other people's CV's? This will help you write a better CV, see chapter 15
3. Have you got a second opinion from a "résumé robot"? Is it robot proof? See section 7.6.6
4. Does the style look good? Is it easy on the eye? Is there adequate whitespace, not too much or too little? See section 7.6.1
5. Have you eaten your own dogood, see section 4.4.1? Is *everything* relevant? e.g. no swimming certificates from ten years ago?
6. Is your year of graduation, degree program, University and expected (or achieved) overall degree classification clear? See section 7.5.2
7. Have you spell-checked both manually and with software?
8. Have you added context using relevant hyperlinks that an interested reader can click on? See section 7.6.5
9. Is it in reverse chronological order? Most recent things first. Can your timeline be easily scanned, e.g. all dates left or right aligned for easy reading?
10. Have you avoided using too many personal pronouns? `I, me, my ...` everywhere?
11. Have you made it clear what you have actually done using prominent `verbs`? See chapter 18
12. Have you given sufficient information on your education without going into too much detail? Have you mentioned courses you are studying now (or next semester)? See section 7.5.2
13. Have you quantified and provided evidence of claims you make where you can? See section 7.5
14. Is it balanced, including both technical and non-technical (softer) skills? See section 5.3.5
15. Does it have a good, clear structure? Not too many headings, around five sections for a one-pager see section 7.5.1?
16. Have you clearly distinguished between paid, unpaid and voluntary experience? Have you included all of your experience including casual work, see section 5.3.5?
17. Does it fit comfortably on exactly one page (résumé) or a two pages (CV)? Not one-and-a-half pages or more than two? See section 7.6.3

7.9 Covering letters & personal statements

Applications and CV's are often accompanied by covering letters or include some kind of personal statement. Whereas a lot of your CV is essentially a bulleted list of facts and statements, a covering letter or personal statement gives you a chance to *really* demonstrate your fluent written communication skills in clear prose. If you're going to have a `personal statement` or `profile` on your CV keep it short, unlike Mike Rokernel who waffles on for ages without providing any evidence. Usually this kind of information is better in your covering letter.

Let's say you're applying for a widget engineering position at `Widget.com`. There are three things you need to cover in approximately this order:

1. **Why them?** Why are you applying to `Widget.com`
2. **Why that role?** `Widget.com` employees have many roles and responsibilities, so what is it about widget engineering that attracts you?
3. **Why you?** Why should they employ you? What makes you stand out from all the other widget engineering candidates? What is your Unique Selling Point (USP) or points?

7.9.1 Does anyone actually READ covering letters?

Some employers will read your covering letter very carefully, others less so. It's not clear which employers will bother and which won't.

Even if nobody reads your covering letter, it is still worth writing one because it forces you to rehearse standard interview questions shown above. See e.g. www.careers.manchester.ac.uk/applicationsinterviews/cl for further information. Think of it as practicing the lines of your elevator pitch.

7.10 Debugging summary

Too long, didn't read (TL;DR)? Here's a summary:

This chapter we've looked at how to debug your CV. It's important to try and squash any of the bugs we've described here, before an employer see's your CV. The checklist above in section 7.8 is a good place to start.

Chapter 8

Finding your future

So you've successfully debugged your future, see chapter 7. How can find an interesting job? How can use your CV, covering letter and any other communication to persuade employers to invite you to an interview? What techniques exist and how can you use your networks to help you? Where can you look?

8.1 What you will learn

At the end of this chapter you will be able to:

- Formulate job search strategies, by role, by sector, time, size and location
- Identify opportunities for finding work, online and face-to-face
- Identify people in your existing networks who can help you
- Grow your networks and use them to your advantage
- Apply your search strategies to advertised (and unadvertised) opportunities
- Evaluate what employers have on offer
- Describe some of the problems with recruitment, both for employers and yourself

8.2 Where can you for look for jobs?

The marketplace for job searching and job hunting advice is incredibly crowded. Employers spend huge amounts of money on recruitment and this is reflected in the enormous range of job websites, which are often accompanied by advice on job hunting. I've broken resources down here into three categories, student specific, more general resources and recruiters.

Figure 8.1: Coding your future is all very well, but how do you actually get a job? This chapter looks at job searching and networking. Yes but… sketch by Visual Thinkery is licensed under CC-BY-ND

8.2.1 Student and graduate specific resources

The following job finding resources are specifically aimed at undergraduate students and graduates:

- gradcracker.com for engineering and technology students, you can filter e.g. by Computing/Technology jobs, from the publishers of the popular gradcracker toolkit
- ratemyplacement.co.uk is a leading UK job resource for undergraduates seeking placements and internships.
- targetjobs.co.uk graduate jobs, schemes and internships from the people behind The Guardian 300 top graduate employers
- milkround.com, placements and graduate positions from the people behind The Times Top 100 Graduate employers
- graduateland.com, placements and graduate positions around Europe
- prospects.ac.uk, a jobs board accompanied by job searching advice
- InsideCareers.co.uk is good if you're looking for jobs in the financial sector
- varsitycareershub.co.uk, targeting students from Loxbridge but many of the employers recruit much more widely
- Year in Industry if you're looking for a year in industry
- Your University careers service. University jobs boards are good places to look for opportunities that are specifically targeted at students of the University where you are studying. So if you're studying at the University of Manchester it's careerconnect.manchester.ac.uk (UoM login required)

8.2.2 More general resources

The following job finding tools are aimed at a wider audience (not just students and graduates) but will be useful to you nonetheless.

Google job search shown in figure 8.2 is a good starting point. It doesn't index *every* job listings site, see Google's job hunting service comes to UK (Kelion, 2018), but its a pretty good place to start.

- Google job search indexes jobs advertised by many of the resources mentioned in this chapter. You can use google job search use to find internships, placements and graduate jobs anywhere in the world, as well as saving vacancies and setting up job alert notifications by email. If you haven't used it already try the searches below. Unlike other sites, Google job search works by indexing embedded microdata structured with schema.org/JobPosting. Keywords like `job` and `intern` in an ordinary (vanilla) google search will trigger the job search product, some examples:

  -google.com/search?q=software+engineering+intern+manchester

  -google.com/search?q=business+analyst+intern

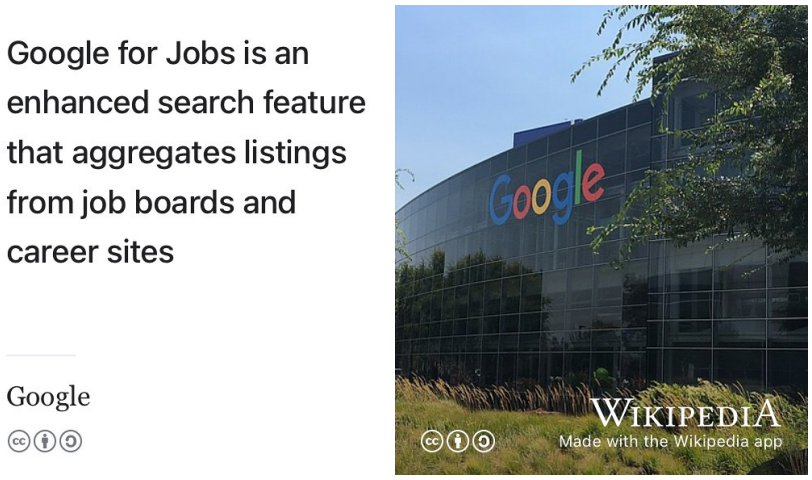


Figure 8.2: Keywords like `job` and `intern` in an ordinary google search will trigger Google's job search product, an enhanced search feature that aggregates listings from many different jobs boards. See the text below for examples but note that European terms like `placement` are not keywords. CC BY-SA picture of the Googleplex in California by The Pancake of Heaven via Wikimedia Commons w.wiki/3X4t adapted using the Wikipedia app

-google.com/search?q=graduate+hardware+engineer

-google.com/search?q=graduate+software+job+london

-google.com/search?q=data+scientist+intern

-Google job search is an impressive product, see grow.google/job-seekers but it doesn't index *everything*. If you're looking for a job AT google, they have moved from jobs.google.com to careers.google.com, see also section 5.3.1

- LinkedIn advertises job vacancies, is frequently visited by recruiters and you can often apply for jobs directly on LinkedIn (although making fast applications is not always a good thing). See university.linkedin.com/linkedin-for-students, linkedin.com/learning/learning-linkedin-for-students and figure 8.3.

- glassdoor.co.uk is like tripadvisor for jobs. Find out what it's *really* like to work for given employers from current and former employees. A student oriented version can be found at glassdoor.com/Students, this means you can use it without writing a review of a previous employer (which is what non-student users have to do to access the content)
- HiPEAC jobs (High Performance and Embedded Architecture and Compilation) is good for jobs in hardware, supercomputing and related fields

Figure 8.3: Linkedin is a social media service which allows employers to advertise job vacancies online and candidates like you to apply for them. Social media caveats aside (see section 3.6), LinkedIn can be useful tool for networking with other professionals and finding a job. Image via Wikimedia Commons adapted using the Wikipedia app

- Indeed.co.uk, adzuna.co.uk, cwjobs.co.uk, fish4.co.uk, reed.co.uk, totaljobs.com, monster.co.uk, jobs.smartrecruiters.com, workinstartups.com, cv-library.co.uk, jobs.ac.uk are general jobs boards that also advertise jobs for students and graduates, alongside many other vacancies.
- stackoverflow.com/jobs jobs via StackOverflow. You've cut and pasted the code… they advertise technical vacancies too
- Otta.com for people with 0-10 years experience. From engineering to sales, discover jobs & internships at London's most innovative companies.

8.2.3 Recruiters

Recruiters can help you find work and they operate in every industry sector. They are sometimes called "head-hunters", and there are two basic kinds that can help you:

1. Recruiters employed directly by an employer, for example in the human resources (HR) department of a given organisation.
2. Recruiters who are self-employed or work for a recruitment agency. They typically earn money from the number of interview candidates and successful hires they provide for their clients.

Recruiters are usually not technical people, so don't expect them to have lots of knowledge about software engineering (for example) - that isn't usually their

skill set. Although recruiters can help you, it is worth being wary of recruiters as shown in figure 8.4, especially if they work for an agency rather than being directly employed by the organisation you are interested in.

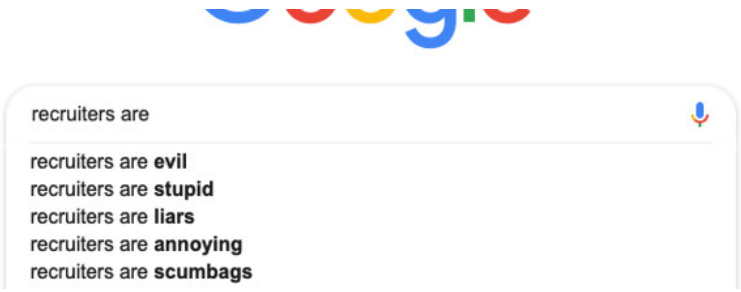


Figure 8.4: The autocomplete algorithm of a well known search engine gives you an idea of what *some* people think about *some* recruiters. This doesn't mean you should avoid recruiters completely, just be careful how you use them and remember who they work for.

Some recruiters are very good and can help you. For example, there are some recruitment agencies that specialise in helping employers recruit graduates, these may be useful to you. However some recruiters are not very good, and don't provide a valued service for employers or potential employees like you. This is why you sometimes see `no recruiters` or `no agencies` on job adverts. So be wary of recruiters, and remember that some recruiters work primarily for their clients (employers) not you.

In most cases you shouldn't have to pay recruiters up front but job scammers will sometimes pose as recruiters so beware. Talking of job scammers, there's some things you need to be wary of when you are job hunting:

8.3 Buyer beware

When you're looking for job you're acting as both a buyer *and* a seller.

1. **SELLING**: You're selling your services in a marketplace, for the best price you can get
2. **BUYING**: You're buying into the culture and values of an employer (see section 9.4), who are trying to sell themselves to you.

As a buyer and seller, you should be wary of the following:

- Job scammers: section 8.3.1
- Over-specified jobs: section 8.3.2

- Unpaid internships: section 8.3.3
- Overselling: section 8.3.4
- Underselling: section 8.3.5
- Rejection: section 8.3.7
- The rollercoaster: section 8.3.8

8.3.1 Beware of the job scammers

Most job adverts are legitimate but you are vulnerable when you are job hunting. You may become more vulnerable over time if you are getting repeated rejections (remember: repeated rejection is quite normal). Unfortunately there are some shady characters out there looking to exploit your vulnerability through various kinds of employment fraud. (Hinds, 2017) You should be wary of anyone asking you for:

- Money up front - be very suspicious
- Excessive personal data such birth dates, passport numbers and bank details. These could be used for identity theft, fraud or other criminal activities
- See for example google.com/search?q=job+scams

Reputable employers (and jobs boards) will not try to scam you, but you should beware of job scammers if you find yourself looking for employment off the beaten track.

8.3.2 Beware of over-specified jobs

Employers and recruiters routinely over-specify job descriptions. A good example of this is, when the Swift programming language was released in 2014 at the Apple Worldwide Developers Conference (WWDC) in California, job adverts instantly appeared asking for programmers with `5 years experience in Swift`! How can *anyone* have five years experience in a programming language that's only just been made public?! Aside from the people who developed the language, like Chris Lattner, the recruiters and employers must have had a *long* search trying to find their candidate. It must have taken them at least five years!

> Job requirements: pic.twitter.com/4QB1oPyxU0
>
> — Mubeen (@Mubeeen_) July 28, 2020

The moral of this Apple story above is, if you don't meet *all* the criteria in a job specification, that shouldn't stop you applying. Most job specifications

are over-specified as wishful employers dream up their ideal candidate.[1] Many employers will over state their requirements in the hope they get their dream candidate. You might look at the job description and think, I've only got 70% of what they're asking for, so I won't bother applying. The reality is, if you've got 60% of what they are asking for, you should definitely apply. It's unlikely that *anyone* will meet 100% of the job requirements.

If you see things on job adverts you don't understand or are not sure about, go and find out about them. There's a good chance it will be similar to something you already know about, or can self-educate yourself to fill any gaps.

8.3.3 Beware of unpaid internships

In the United Kingdom, it is illegal to employ people without paying them a salary. However, there are exceptions which can allow employers to take on unpaid interns depending on how they classify their employment status. See for example:

- Employment rights and pay for interns gov.uk/employment-rights-for-interns
- Targetjobs position on the law on unpaid internships: know your rights
- This article on Why I Regret Doing an Unpaid Internship (Louise, 2019)
- For more horror stories see google.com/search?q=unpaid+internships

In science, technology and engineering, unpaid internships are much less common than in other sectors as demand for skilled engineers and scientists is generally high. Some employers, particularly startups, may offer company equity (such as shares) as an alternative to a salary - again you should be wary of this. Unless you're very lucky, the chances are those shares will probably be worthless. Like many people, I don't endorse unpaid internships, and I recommend you avoid them completely. They are a form of modern slavery.

8.3.4 Beware of overselling

When people try to sell you something, you will naturally be wary of overselling and fake news, see figure 8.5.

There's two kinds of fake news that are common in the jobs marketplace:

1. Employers overselling themselves. Recruitment can be a bit of a beauty contest, with everyone trying to show you their best side. Some employers

[1] Using a relationship analogy, describe your ideal romantic partner. PAUSE. How many people are actually likely to have ALL of those attributes?

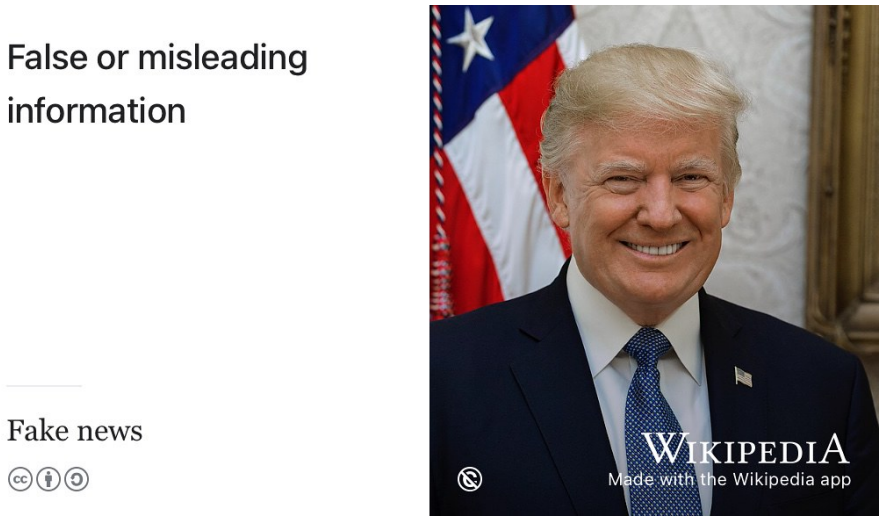


Figure 8.5: Beware of false or misleading information in the jobs marketplace. Are employers *really* as good as they say they are? Are *you* as good as you say you are? Or is it fake news? Public domain official portrait of Donald Trump taken by Shealah Craighead via Wikimedia Commons w.wiki/3XoY adapted using the Wikipedia app

may make promises they can't deliver but a quick look on sites like glassdoor.com will help you evaluate employers. Even better, talk directly with actual employees of the organisation, both current and former. Is their employer really as good as they say they are?

2. You exaggerate your achievements: It can be tempting to oversell yourself in the marketplace. An experienced reader or interviewer will be able to spot your fake news and find you out, see chapter 7 *debugging your future*

So beware of fake news and overselling.

8.3.5 Beware of underselling

Likewise, you should make sure you don't undersell yourself. Know your value (financial), know your values (see chapter are 2) and try to understand how that fits with a given employer. What are the employers stated values? On the financial side, it is easy to find out about salaries, for example see:

- technical intern salaries in the UK (Wodiany, 2018)
- graduate salaries in the UK (Grove, 2018)
- The Highest Paid Internships and Placements in the UK (Louise, 2021)

Salaries in the UK for interns (and graduates) range from minimum wage to £50k and over, with everything in between. So beware of under-selling yourself, know your value. Some employers see students as a form of cheap labour that can be exploited because you're not "qualified" until you graduate. I'd think twice before working for such an employer, computing skills are in demand and there are plenty of other employers who will treat with more respect.

8.3.6 Beware of the time sink

Finding employers that you are interested in and submitting high quality job applications takes time. Many students under estimate the time needed to job hunt. It can be a very time consuming process for everyone, both employers and candidates alike, see figure 8.6.

Figure 8.6: Playing the game of job hunting can be a big drain on your time and some time wasting is unfortunately inevitable. Employers will waste some of your valuable time and you'll probably waste some of theirs too. Beware of the recruitment time sink. Public domain image of a DualShock PlayStation controller by Evan Amos on Wikimedia Commons w.wiki/3VFp adapted using the Wikipedia app

Even after you have managed to:

1. Identify and articulate your skills and knowledge, see chapter 2
2. Understand what you're interested in, see chapter 2
3. Update your CV, see chapter 7
4. Consider all your options, see chapter 9
5. Target employers or sectors of interest

.. the actual business of applying described in this chapter can be very bureaucratic. Any interviews you have will take time to prepare for (see chapter 10) and you've got loads of other calls on your time like studying and having a social life.

One way to tackle this problem is to schedule some time every week when you work on applications, see chapter 19. However, there's no getting away from the fact that finding a job will be time consuming. Beware of the time sink.

8.3.7 Beware of rejection

For most people, rejection is a normal part of applying for jobs. Rejection can take a heavy toll on your mental health described in chapter 3 and it is usually a struggle. Some employers won't even bother to reply to reject you, welcome to the employer black hole described in section 7.2 and shown in figure 8.7.

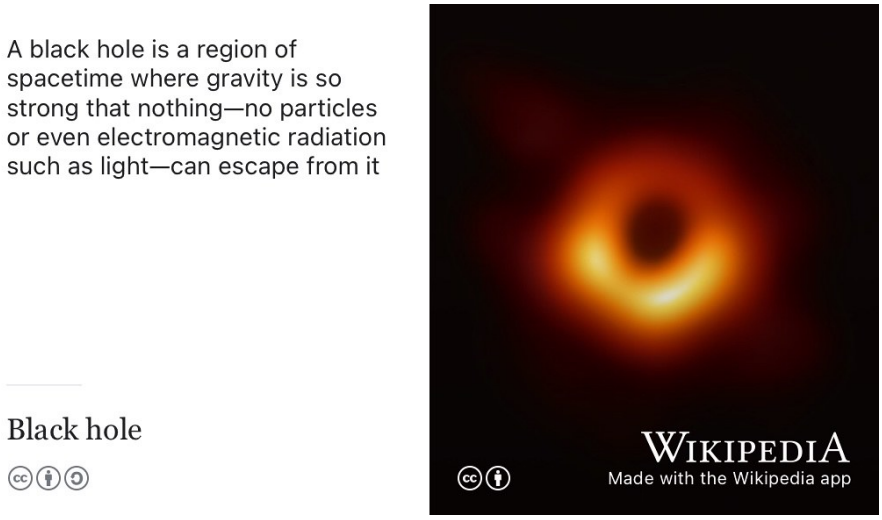


Figure 8.7: Rejection is a normal part of applying for jobs, some of your applications may disappear without trace into employers "black holes", especially if they are large and have a strong gravitational force on job applicants like you. This *doesn't* mean you shouldn't bother applying, but that you need to think about how to make your application stand out. CC-BY image of the supermassive black hole in Messier 87 created using the CHIRP algorithm by the Event Horizon Telescope team via Wikimedia Commons w.wiki/3RCa

If you're getting too many rejections or all your applications are sucked into black holes, it might be because you need to :

- debug your CV and job applications some more (see chapter 7)

- develop a better job search strategy (see this chapter, chapter 8)
- broaden your job search (see chapter 9)
- pay attention to your mental health, rejections can make you anxious and depressed (see chapter 3)

So beware of rejection, it's a normal part of job hunting.

8.3.8 Beware of the rollercoaster

There are highs and lows in job hunting, you will ride the job search rollercoaster shown in figure 8.8. There will be highs, you'll be invited to interviews, but there will be also be lows too, such as the inevitable rejections we discussed in the previous section 8.3.7. It will be a rollercoaster, which ends on the high of a job offer you accept. Fasten your seatbelt, enjoy the ride and good luck with your applications and interviews!

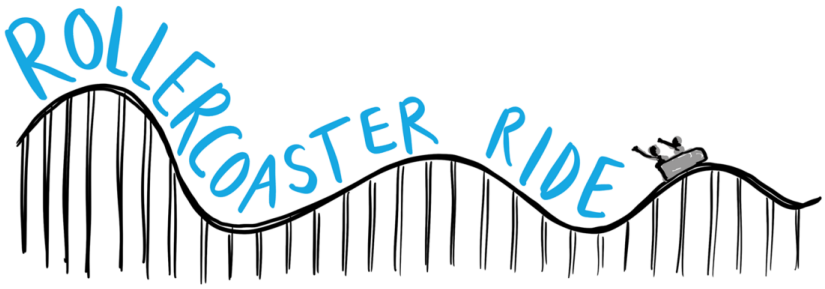


Figure 8.8: Are you ready to ride the emotional rollercoaster of job hunting? A tech project is like… sketch by Visual Thinkery is licensed under CC-BY-ND

So beware of the rollercoaster, it has ups and down. Before we move on to talk about broadening your options, lets looks at some basic job search strategies you can use to get started.

8.4 Job search strategies

Now that you're aware of some pitfalls you need to think about developing a range of different strategies for job hunting, and change the strategy during the

Table 8.1: Where, when and how to apply for vacancies, internships, placements, graduate jobs and schemes in large multinational corporations and small to medium sized enterprises (SMEs)

	Large corporations	SMEs
Where	Advertise broadly	Less likely to advertise on big jobs boards
When	Vacancies earlier in the academic year	Vacancies tend to be later in the academic year
Type	Unlikely to consider speculative applications	May consider speculative or informal applications
How	Multistage applications, several rounds of interviews	Shorter application processes

year. At the beginning of the academic year in September you might target large multinational organisations. If you're not successful, you could switch to smaller employers later in the academic year. Table 8.1 shows summarises some of the when and where some employers advertise.

8.5 Breakpoints

Let's pause here. Insert a breakpoint in your `code` and slowly step through it so we can examine the current values of your variables and parameters.

```
* PAUSE
```

- What are your current job search strategies?
- How could they be improved or tuned?
- How many jobs should you apply for?

- Why is it important to build your network?
- How can recruiters help you?
- Why do recruiters have a bad reputation?
- How long does it take to apply for a job?
- Should I optimise for *quality* or *quantity* of job applications?
- How can you deal with the inevitable rejections that come during job hunting?

```
* RESUME
```

8.6 The power of weak ties

Your close network probably won't change that much, the friends and family you trust and rely on. Its important to recognise the importance of more

casual acquaintances, or what sociologist Mark Granovetter calls "weak ties". (Granovetter, 1973)

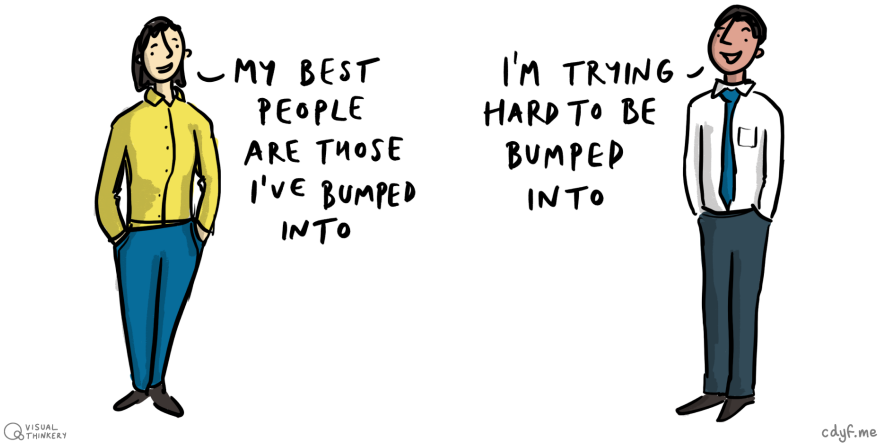


Figure 8.9: It's not what you know, its who you know. Networking and personal contacts can be more useful than just knowledge and skills alone, when seeking employment. Networking is an essential part of any job search, your networks can help you now and in the future. One of the things looked at in this chapter is how to build and use your networks to help find the job you're after. The simplest networking technique is bumping into people, but you need create opportunities for that to happen. Bumped into sketch by Visual Thinkery is licensed under CC-BY-ND

Weak ties are people you don't know as well, but are important for a range of reasons. Research has shown that building networks of weak ties is good for your mental health and can give you an edge in job hunting. (Leslie, 2020) Granovetter showed that many job opportunities came through weak ties, rather than strong ones. This is true not just of jobs early on in your career (like now) but also later too. So it is in your interests to continually foster weak connections and be open to serendipitous meetings where you bump into people, as in Figure 8.9. "Bumping into" here, could mean either physical or virtual.

8.7 Summarising search

Too long, didn't read (TL;DR)? Here's a summary:

We've looked at search techniques that will help you find opportunities you care about. Figuring out what you want to do is tricky at times but it usually works out well in the end.

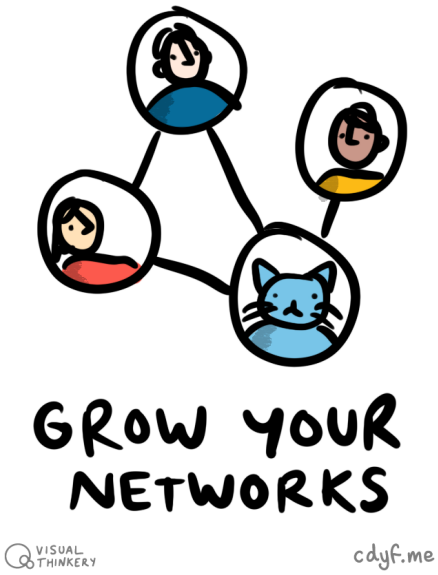


Figure 8.10: Who is in your network? Grow and use your network, both the strong ties and the weak ties. Weak ties are often the most important when it comes to job hunting. Networks sketch by Visual Thinkery is licensed under CC-BY-ND

This chapter is under construction because I'm using agile book development methods, see figure 8.11.

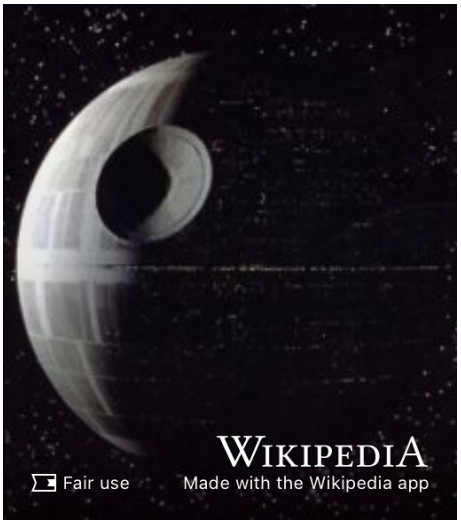


Figure 8.11: Just like the Death Star, this ~~galactic superweapon~~ chapter is under construction. Image of agile weapon engineering in *Star Wars* via Wikimedia Commons w.wiki/32PB adapted using the Wikipedia app

Chapter 9

Broadening your future

Do you feel like the *weird edge case* pictured in Figure 9.1? Do typical graduate destinations such as large multi-national corporations, not really make you want to *Shake Your Thang*? (Isley et al., 1988) Perhaps you want to:

- use your technical skills responsibly and ethically to make the world a better place?
- start your own business and make money for yourself, rather than other people?
- work in computing in roles beyond software engineering?

Broadening your initial job search described in chapter 8 will open up more opportunities on your horizon. This chapter will broaden those horizons and get you to think about some of the less obvious options, because I *love* weird edge cases and you should too.

Many technology jobs exist outside of technology companies, (Assay, 2020) because a lot of software is written to be used rather than sold. Consequently, many employers create bespoke software to fit the needs of their business. The people who build it are often employees, rather than people employed by a technology company. In the United States for example, ninety percent of IT jobs are outside the traditional tech industry. Technical jobs outside the technology sector often have the advantage of being more accessible than those within a very competitive technology sector. (Markow et al., 2019)

9.1 What you will learn

- Describe the less obvious careers that computer science can lead to, besides software engineering, including:

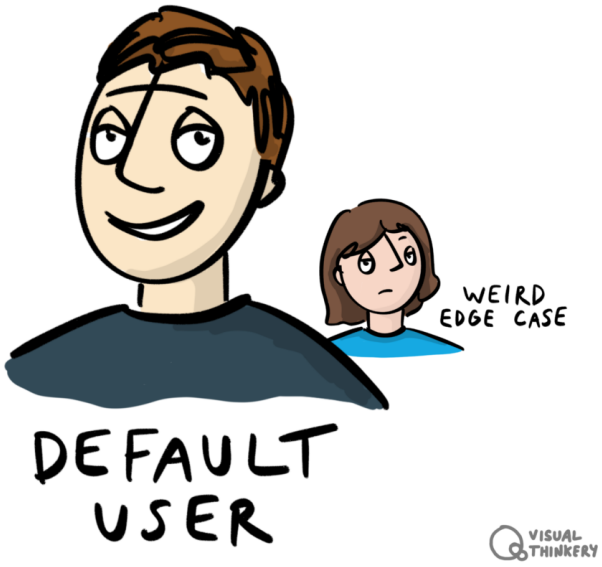


Figure 9.1: Are you a weird edge case? By default, many graduates choose a graduate scheme with big brand, often a blue-chip multinational employer. While working for these kind of employers has many benefits, they are not the whole story. This chapter looks at some of the alternatives. Default user by Visual Thinkery is licensed under CC-BY-ND

  - Starting a business or joining a startup
  - Working outside of the technology sector
  - Working outside of the private sector (governments, non-profits etc)
  - Roles allied to software engineering that require you to be a conversational programmer

- Recognise the social responsibility accompanying the power held by computer scientists
- Match and critically evaluate the values of an employer with your own values and ethics

9.2 Beyond software engineering

The phrase software engineering has been around since Margaret Hamilton (figure 9.2) wrote code for NASA's Apollo program. However, the practice of software engineering has been around even longer right back to Ada Lovelace.

Figure 9.2: The role of software engineer has been around for a long time but there are plenty of other roles for computer scientists beyond software engineering. Margaret Hamilton in 1969 standing next to the (printed) navigation software that she and her MIT team produced for the Apollo program. Public domain image via Wikimedia Commons w.wiki/3YJW adapted using the Wikipedia app

Software engineers (or software developers) are one of the most popular roles for graduates (see e.g. figure 6.5) but there are plenty of affiliated roles that computer scientists go into besides software engineering.

- Data scientist
- Product manager or owner

- Project manager
- Software architect
- Business analyst
- Technical writer, see section 4.4.2
- Technical sales and marketing
- Test engineer (QA)
- Usability engineer (any Human–computer interaction HCI)
- Security engineer, penetration testing etc
- DevOps / sysadmin
- Consultant What do these roles entail?

9.3 With great code comes great responsibility

Computer scientists wield tremendous power in the twenty first century. We know that:

- With great power comes great responsibility (Parker, 1962)
- With great code comes great responsibility (Goldman and Schlesinger, 2018)

Given the growing power of computing in the twenty-first century, computer scientists have a duty to society to use that power responsibly and justly. How can they do so? Do computer scientists need to sell their soul to the highest bidder?

9.4 Do you need to sell your soul?

You'll sometimes hear people saying you need to sell your soul to get a job, shown in figure 9.4. See for example:

- Soul sold for less than £12 (Malham, 2002)
- Am I Selling My Soul to Work for My Company? (Bell, 2021)
- google.com/search?q=selling+your+soul+to+your+employer

So when you're searching for jobs and researching potential employers, one of the first things you need to find out is what the values and ethical principles of an employer are see section 8.3. It's a quick way to evaluate what makes an organisation who they are. Most employers publish their values and ethics openly, here's a small selection to give you a flavour:

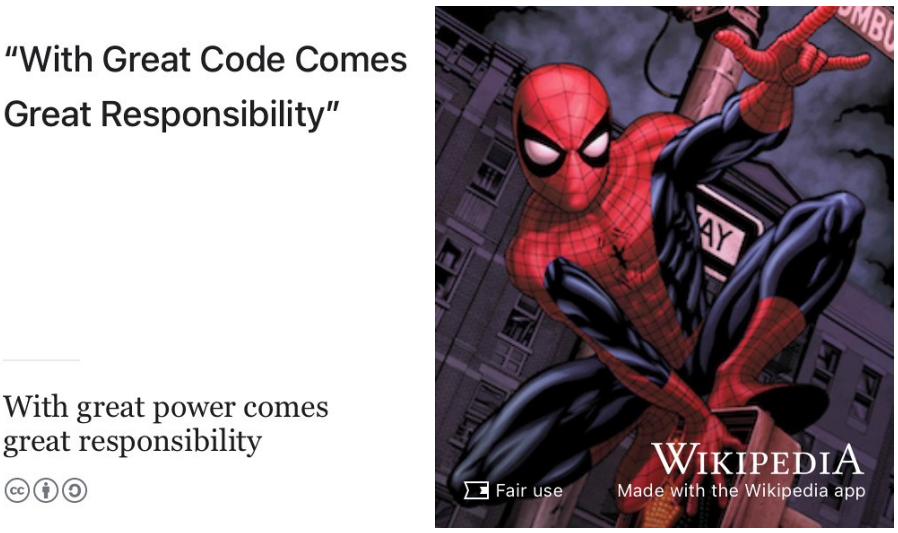


Figure 9.3: The greater your code, the greater your superpower. The greater your superpower, the greater your responsibility. What powers does computing give you and how can you use that power responsibly? (Parker, 1962; Goldman and Schlesinger, 2018; Shapiro et al., 2021)

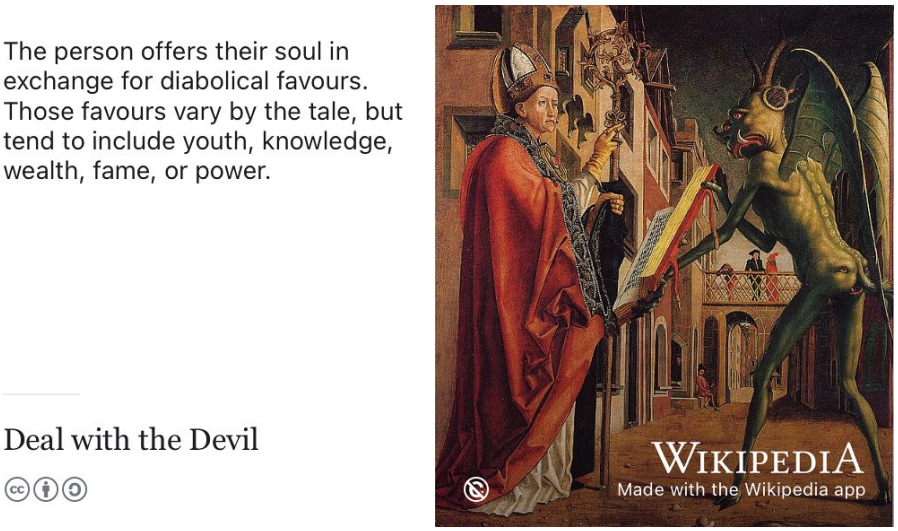


Figure 9.4: In European folklore, doing a deal with the devil is a motif that recurs in culture. Wealth and power are just two of the items exchanged for a persons soul as part of diabolical deal. Public domain image of a painting of diabolical dealing by Michael Pacher on Wikimedia Commons at w.wiki/3VvX adapted using the Wikipedia app

- Amazon amazon.jobs/en/principles
- Microsoft microsoft.com/en-us/about/corporate-values
- Apple apple.com/compliance
- Google ai.google/principles
- Morgan Stanley morganstanley.com/about-us/morgan-stanley-core-values

With offices in more than 42 countries and more than 60,000 employees, the firm's clients include corporations, governments, institutions, and individuals

Morgan Stanley

Figure 9.5: Morgan Stanley is an American multinational investment bank and financial services company headquartered in New York City. The firms clients include corporations, governments, institutions and individuals. CC-BY picture of Morgan Stanley HQ in Times Square by Ajay Suresh on Wikimedia Commons w.wiki/3Vnt adapted using the Wikipedia app

Let's look at Morgan Stanley (figure 9.5) as an example, I've chosen these values because they are brief and self-explanatory:

1. Do the right thing: act with integrity
2. Put clients first: listen to what the client is saying and needs
3. Lead with exceptional ideas: win by breaking new ground
4. Commit to Diversity and Inclusion: value individual and cultural differences
5. Give back: serve communities generously with expertise, time and money

Look at these values carefully, or choose the values of another employer you're interested in. What do they mean to you?

9.5 Breakpoints

Let's pause here. Insert a breakpoint in your `code` and slowly step through it so we can examine the current values of your variables and parameters.

```
* PAUSE
```

- How closely do a given employers values align with your own? You may need to revisit section 2.3.
  - You might not get a 100% match but you're unlikely to enjoy working for an employer where your values don't match very well at all
- Are the stated values of an employer the whole story?
- Is there anything missing?
- Are there any unwritten rules? What an employers *says* and *does* may be contradictory. Actions speak louder than words.

```
* RESUME
```

Once you've thought about these questions, you stand a much better chance of working out if a given employer is a good match for you. So do you have to sell your soul as shown in figure 9.6? It depends on what you value and if an employer shares those values with you.

Figure 9.6: Here's a dilemma: Do you need to sell your soul to your employer? If so, how much can you get for it? What percentage stake of your soul will they ask for and how much are you willing to give? How do your values align with those of your employer? Soul selling dialog box sketch by Visual Thinkery is licensed under CC-BY-ND

9.6 Summarising your alternatives

Too long, didn't read (TL;DR)? Here's a summary:

This chapter is under construction because I'm using agile book development methods, see figure 9.7.

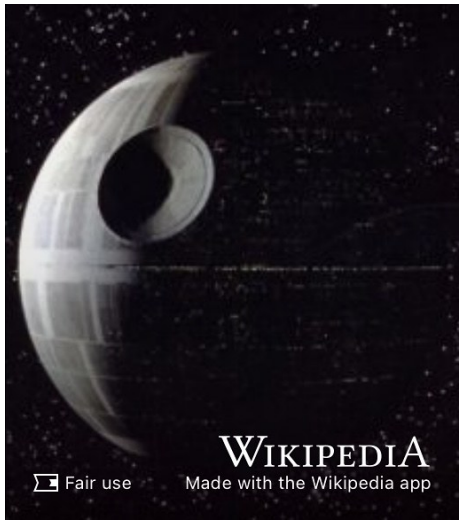


Figure 9.7: Just like the Death Star, this ~~galactic superweapon~~ chapter is under construction. Image of agile weapon engineering in *Star Wars* via Wikimedia Commons w.wiki/32PB adapted using the Wikipedia app

Chapter 10

Speaking your future

Congratulations, you've been invited to an interview. It might be a telephone, video or face-to-face, or it might even be as part of an assessment centre where you'll be asked to complete several other tasks and tests. Being invited to an interview means that your CV, along with any accompanying covering letters, application forms or digital portfolios, have hit the target. All that reading and writing has paid off. BULLSEYE!

Having passed the first stage, you move onto speaking and listening which you'll do in a live interview. One of your goals is to convince the interviewers that you can articulate yourself clearly, and communicate well using spoken natural language, while listening carefully to questions that they ask. This is fundamental skill discussed in section 4.3 on communication I/O.

If you've got an interview, you can feel good about having a bug-free CV shown in Figure 10.1. But now you have a new set of problems. How can you prepare for the interview? What kinds of interviews exist and what questions might you be asked? If they offer you a job, how will you negotiate the terms, conditions and salary? Do you *really* want the job anyway and are they the kind of people you actually want to work with everyday? You'll be giving this employer:

- most of the hours of your day
- most of the days of your week
- most of the weeks of your year
- something like the next two years of your life (Black, 2017) as the first part of up to 80,000 hours

So you want to ensure employers are a good match and not going to waste your time.

Figure 10.1: If you have got an interview, then you have proved that your CV is bug free. That doesn't mean your CV is perfect, it just means that it is good enough to get you an interview with that particular employer. Congratulations! What comes next? Bug free sketch by Visual Thinkery is licensed under CC-BY-ND

10.1 What you will learn

By the end of this chapter you will be able to:

- Identify kinds of interviews you might be invited to
- Anticipate common interview questions, technical and non-technical
- Prepare questions for your interviewer by researching the employer
- Formulate strategies for negotiating job offers
- Calm your interview nerves

10.2 Interviews

Broadly speaking there are two basic kinds of interviews

- technical or coding interview.
- non-technical interview, sometimes called competency based interview or HR interview (human resources)

These can be conducted in various modes:

10.2.1 Modes of interview

Interviews can be conducted in various modes:

- telephone (no visual contact)
- pre-recorded pieces to camera (this is you talking to your webcam)
- teleconference (zoom / teams etc) with cameras and microphones turned on
- real-time face to face

10.2.2 Competency interviews

Competency interviews are there to test your soft skills, find out what you're like, how you work in a team, if you can communicate well. Here are some common competency interview questions. Imagine you are going on a stage, prepare lines that answer these questions, rehearse them out loud in front of a mirror (or a critical friend).

1. What roles can you play in a team?
2. Tell me about a time when you showed integrity and professionalism

Figure 10.2: Got a video interview? Toggle your mute button. Bob… by Visual Thinkery is licensed under CC-BY-ND

3. Can you give an example of a situation where you solved a problem in a creative way?
4. Tell me about a big decision you've made recently. How did you go about it?
5. Give an example of a time you handled or resolved conflict
6. How do you maintain healthy working relationships with your colleagues?
7. Describe a project where you had to use different leadership styles to reach your goal
8. Give me an example of a challenge you faced and tell me how you overcame it
9. How do you influence people in a situation with conflicting agendas?
10. Tell me about a time that you made a decision and then changed your mind.
11. Tell me about a time when you achieved success even when the odds were stacked against you.

For more examples of non-technical questions see:

- google.com/search?q=competency+based+interview+questions
- careers.manchester.ac.uk/applicationsinterviews/interviews

Since there's already tonnes of information on competency based interviews, the rest of this chapter will focus on technical interviews, also known as coding interviews.

10.2.3 Coding interviews

Many employers using technical or coding interviews to assess your harder skills. Not every employer does, but they are widely used by employers. Preparing for coding interviews is a good way to become a better engineer, so even if you don't have to face a series of really tough coding interviews, it is worth knowing about them.

There are a lots of resources to help you prepare for and practice coding interview questions, the best place to start is *Cracking the Coding Interview* by Gayle Laakmaan McDowell. (McDowell, 2015) As well as reading Gayle's book, there are lots of online resources to help you prepare for coding interviews. Before we look at those, University of Manchester Computer Science graduate Petia Davidova explains in figure 10.3 what she learned from failing several coding interviews at big technology companies.

Figure 10.3: Petia describes her worst software engineering interview failures. (Davidova, 2021) Petia demonstrates her growth mindset (section 3.5) and productive failure(s). Although she failed her interviews, she learned lots from the process and went on to get a job she wanted. The image above is a screenshot, you can watch the full 16 minute video on software engineering fails here.

Coding interviews can be tough, but preparing for them, and doing them will make you a better engineer. So if you spectacularly wipeout in your coding interview, reflect and think how can you improve next time? Perhaps you need to

- Read up on some more data structures
- Familiarise yourself with more algorithms

- Practice thinking out loud (verbally) by doing a mock technical interview?

All of these things will help both your general professional development and your chances of success in future technical interviews. Thankfully there are plenty of resources out there for helping you get better at coding interviews so let's have a look at some.

10.2.4 Project Euler

Project Euler provides a wide range of challenges in computer science and mathematics. The challenges typically involve solving a mathematical formula or equations, see projecteuler.net and figure 10.4

Figure 10.4: Since its creation in 2001 by Colin Hughes projecteuler.net has become internationally popular, with new problems added approximately every two weeks. (Somers, 2011) Public domain image of a painting of Leonhard Euler by Jakob Emanuel Handmann on Wikimedia Commmons w.wiki/3WAV adapted using the Wikipedia app.

10.2.5 Topcoder

TopCoder is a platform for competitive programming online. You can complete on your own directly online using their code editor. Single round matches are offered a few times per month at a specific time where you compete against others to solve challenges against the clock, see topcoder.com

10.2.6 Codewars

Codewars allows you to challenge yourself on kata ( ), created by the community to strengthen different skills. Master your current language of choice, or expand your understanding of a new one. Find out more at codewars.com and see figure 10.5

Figure 10.5: Just like martial arts, you can practice the computational arts with kata ( ) on codewars.com and elsewhere. CC BY-SA Picture of world champion Emmanuelle Fumonde by Thierry Caro on Wikimedia Commmons w.wiki/3XeE adapted using the Wikipedia app.

10.2.7 Leetcode

LeetCode is a platform to help you enhance your skills, expand your knowledge and prepare for technical interview see leetcode.com.

10.2.8 HackerRank

HackerRank allows developers to practice their coding skills, prepare for interviews and get hired. HackerRank allows users to submit applications and apply to jobs by solving company-sponsored coding challenges. Some employers use hackerrank as part of the interviewing process, so these are not just academic exercises.

Hacker Rank provide a discussion and leaderboard for every challenge, and most challenges come with an editorial that explains more about the challenge, see

hackerrank.com and figure 10.6

Figure 10.6: HackerRank's programming challenges can be solved in a variety of programming languages (including Java, C++, PHP, Python, SQL, JavaScript) and span multiple computer science domains. HackerRank logo from Wikimedia Commons at w.wiki/3WAP adapted using the Wikipedia App.

10.2.9 Pramp

Pramp offers free mock technical interviewing platform for engineers. Pramp, **Pra**ctice **m**akes **p**erfect, was founded in 2015 by Rafi Zikavashvili and David Glauber. As engineers, they were frustrated by the lack of resources to help them prepare for coding interviews. Find out more at pramp.com

10.2.10 ICPC

More than 50,000 students worldwide from more than 3,000 universities in 111 countries participate in over 400 on-site competitions as part of the International Collegiate Programming Contest (ICPC) see icpc.global and figure 10.7.

ICPC is organised by the Association for Computing Machinery (ACM), a global community which advances computing as a science and a profession.

There are subregional contests for ICPC, so in the UK there is the United Kingdom and Ireland Programming Competition (UKIEPC) which is part of the Northwestern Europe European Regional Contest (NWERC).

an algorithmic programming contest for college students. Teams of three, representing their university, work to solve the most real-world problems, fostering collaboration, creativity, innovation, and the ability to perform under pressure.

International Collegiate Programming Contest

Figure 10.7: In their own words, ICPC is "an algorithmic programming contest for college students. Teams of three, representing their university, work to solve the most real-world problems, fostering collaboration, creativity, innovation, and the ability to perform under pressure. Through training and competition, teams challenge each other to raise the bar on the possible. Quite simply, it is the oldest, largest, and most prestigious programming contest in the world." (Hacker, 2021)

UKIEPC has been held annually since 2013 to help universities pick teams to travel to NWERC. Ask your University if they are involved, see ukiepc.info. If they aren't involved yet, you could encourage them to join. It's not just about winning, it's also about taking part.

10.3 Breakpoints

Let's pause here. Insert a breakpoint in your `code` and slowly step through it so we can examine the current values of your variables and parameters.

```
* PAUSE
```

Let's imagine you're applying to work at an employer called `widget.com`. During you're application you need to find out:

- What the main products and services that the organisation provides?
- Who are their clients or customers?
- Who are their biggest competitors?
- What are their values, principles and ethical policies? See section 9.5
- What sector do they principally operate in?
- Who are the market leaders in that sector?
- How is the sector changing, for example how is technology having an impact on their business?

```
* RESUME
```

10.4 Summarising interviews

Too long, didn't read (TL;DR)? Here's a summary:

We've looked at a range of platforms and competitions that can help you prepare for coding interviews. These won't just make you better at coding interviews, they'll make you a better engineer too, whatever stage you're at.

This chapter is under construction because I'm using agile book development methods, see figure 10.8.

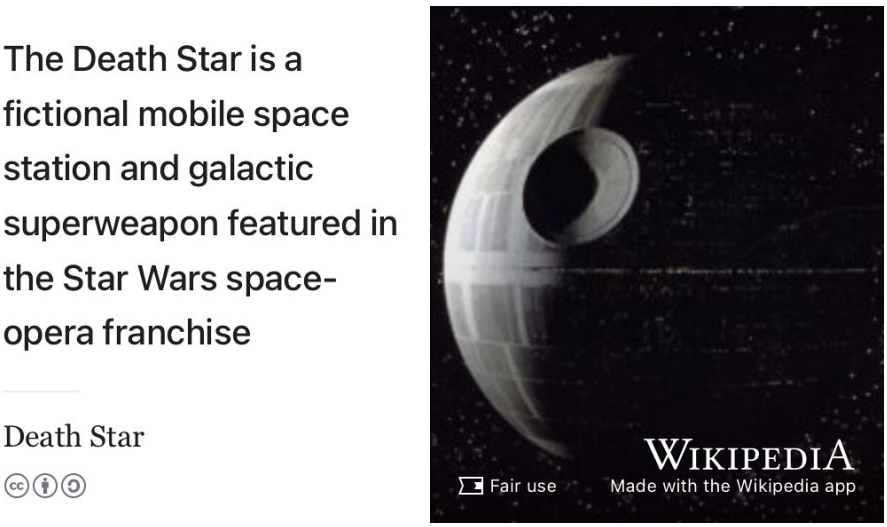


Figure 10.8: Just like the Death Star, this ~~galactic superweapon~~ chapter is under construction. Image of agile weapon engineering in *Star Wars* via Wikimedia Commons w.wiki/32PB adapted using the Wikipedia app

Chapter 11

Surviving your future

Congratulations, you've just accepted an offer of employment. You nailed that interview (or interviews) and you're just about to embark on the exciting journey from the world of study to the jungle of employment. This *might* be your first SERIOUS job, so what do you need to survive and become a professional? What survival skills will you need, see figure 11.1. Even better, how can you thrive in your new role and take on the challenges that are coming your way? How will you optimise your trajectory to reach new heights?

11.1 What you will learn

At the end of this chapter you will be able to

- Manage your manager so that you can:
  - Survive the workplace
  - Thrive in the workplace
  - Avoid diving in a workplace environment
- Collect evidence of new workplace skills and knowledge that you develop
- Reflect on new workplace skills and knowledge that you need to develop

11.2 Survive, thrive or dive?

Starting a new job is a bit like starting a new relationship, except that it is professional rather than romantic. You've searched for and found a partner. You've been through the courtship of recruitment, this might have been quick or may have had many rounds of first and second "dates" (interviews). Once

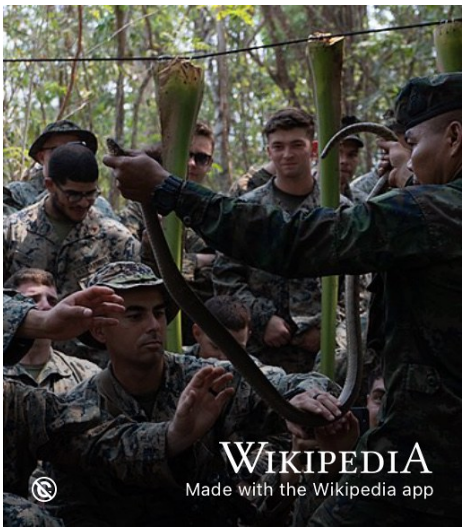


Figure 11.1: The world of employment can be a bit of a jungle where you struggle for existence. What survival skills will you need and how can you go beyond merely surviving to positively thriving as a professional? Public domain image of jungle survival lessons by United States Army on Wikimedia Commons w.wiki/3WBj adapted using the Wikipedia app.

you start employment, you are both committed to each other in a serious relationship. Simply put, there are three scenarios for you as a new employee. You'll survive, thrive or dive.

11.2.1 Survive

Your new job will go OK, you'll meet the expectations of your employer and become and valued employee. If your employer has a probationary period, you'll pass your probationary review without any problems. Most employees probably fit in to this category.

11.2.2 Thrive

Your new job will go brilliantly, you'll exceed the expectations of your employer. If you're on a fixed term contract, such as a summer internship or year long placement, they'll make you a job offer during or soon after your contract of employment expires. If you're on a more permanent contract, such as a graduate job or graduate scheme you'll be promoted and given more responsibility.

You're doing really well if you can impress your manager. Some lucky people make it into this category.

11.2.3 Dive

Your new job will go badly, you will struggle to fit in and won't meet the expectations of your employer. Once you were like star-crossed lovers, (see figure 11.2) but the relationship has turned sour could take a dive into tragedy. (Shakespeare, 1597; Goble and Wroe, 2004)

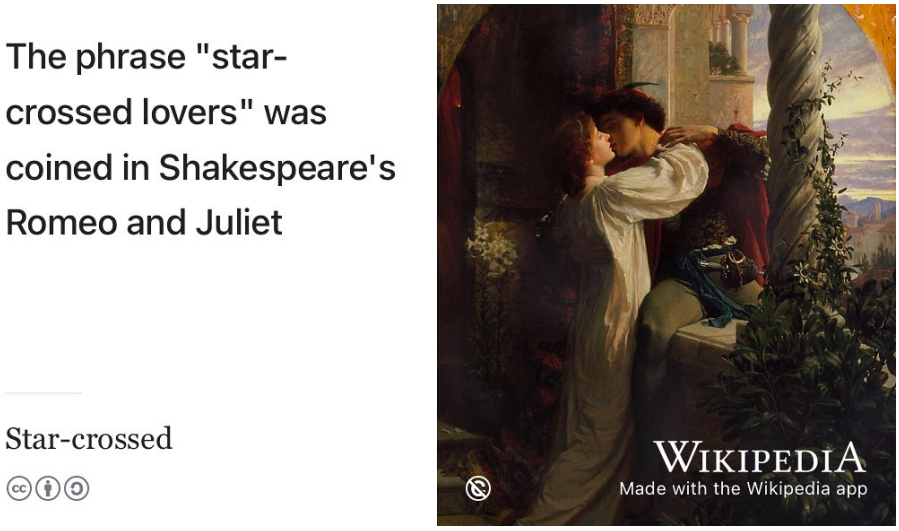


Figure 11.2: Your relationship with your employer is obviously a professional one rather than a romantic one, but that doesn't mean it can't end in tragedy like Shakespeare's star-crossed lovers in *Romeo and Juliet*. (Shakespeare, 1597) What can you do to keep your relationship with your employer healthy and happy? Public domain image of a painting of Romeo and Juliet by Frank Dicksee via Wikimedia Commons w.wiki/3DfJ adapted using the Wikipedia app

There are several relationship problems that could lead to you breaking up with (or being dumped by) your newly estranged lover.

- **Relationship problems**: Your relationship with your manager(s) is not going well. You've tried solving problems informally by talking to your manager but you're not satisfied with the response and want to raise a formal grievance complaint in writing. (UK, 2021c)
- **It's not you, it's me**: You might ultimately decide to you want to hand in your notice to terminate your contract of employment and leave. (UK, 2020b)
- **It's not me, it's you**: If things get really bad, your employer may take disciplinary action against you (UK, 2021a) and in the worst case scenario, you'll be fired (dismissed). (UK, 2021b)

Dismissal is rare, but it **does** happen, even to interns and placement students. In this scenario in the UK, the employer has a duty to do everything they reasonably can to prevent this from happening. It's not in your employers interests to fire you because they've invested a lot of time and money in you by this point. If they have sensible recruitment procedures, those procedures will root out unsuitable candidates long before they make it to the workplace where they can cause real and lasting damage to the organisation once in post.

All employers have procedures for making sure that you can agree on work that suits both of your needs. Better employers will have better procedures to ensure this happens. Employers don't want their employees to "dive" and will try prevent this from happening wherever possible.

11.3 Managing your manager

Building a good relationship with your manager(s) will be key to determining which of the *dive, survive or thrive* scenarios above plays out. At University, you didn't have a manager. Yes you had deadlines, but you didn't have a boss. That changes when you're an employee so it's in your interests to understand what your boss expects of you.

Software engineer Julia Evans has authored a series of programming zines, there's one called HELP! I have manager! (Evans, 2018) you might find useful. It will help you understand your managers job better so that you can work together more effectively. It will help you survive and thrive, not dive because it covers:

- understanding your manager's job
- setting clear expectations
- talking about problems early
- reviewing performance and getting promoted
- asking for specific feedback

> ask for specific feedback pic.twitter.com/wu4uWS0Ftc
>
> — Julia Evans (@b0rk) September 21, 2018

The zine has the benefit of being aimed aimed at engineers just like you. Thoroughly recommended! You might also enjoy Julia's other more technical zines such as HTTP: learn your browser's language (Evans, 2021) and Oh Shit, Git! (Sylor-Miller and Evans, 2021)

11.4 Stay in school

As you develop new skills and knowledge at work, it is a good idea to collect evidence of what you've done. Whatever your career path, you'll need to keep your CV updated. One way to think of the evidence is as *badges*, digital or otherwise. Your employer may already have training schemes that recognise and reward your accomplishments. These badges may be generic or specific to the particular sector you are working in. See chapter 12 on *Achieving your future* for more details.

11.4.1 Technical badges

Some examples of technical badges include:

- Microsoft Certifications docs.microsoft.com/en-us/learn/certifications
- Amazon Web Services Certification aws.amazon.com/certification
- Google Cloud Certification cloud.google.com/certification

Just three examples, there are many others covering both technical and non-technical skills. In many cases, your employer will encourage and possibly pay for you to get these certifications.

11.4.2 Non-technical badges

You are more than just a techie, so make sure you develop your non-technical skills as well. We introduced softer skills in chapter 4, but there's plenty of other skills to think about:

- Building resilience
- Negotiating and managing conflict
- Leadership, influence and change
- Having difficult conversations
- Emotional intelligence
- Public speaking
- Active listening

11.4.3 Online learning

There are many online platforms for building your skills and knowledge, some examples include:

1. coursera.org

2. edx.org
3. egghead.io
4. futurelearn.com
5. khanacademy.org
6. linkedin.com/learning
7. open.edu/openlearn
8. pluralsight.com
9. skillshare.com
10. udemy.com
11. youtube.com

The choice of online learning can be bewildering. Some platforms provide free resources, others do not, but your employer may already pay for some services making them free to you while you are an employee. If that's the case, make good use of the services while you can. There's a useful comparison of four different online learning platforms here. (Chen, 2020)

11.4.4 Other training courses

Your employer may provide other courses you can go on. Again, you should make the most of these if and when they are available.

Whatever job you're doing, stay in school. Take advantage of any training on offer or go and find courses that help you develop professionally and personally. Remember that **you** are the person who cares most about your career, see section 1.5.

11.5 Breakpoints

Let's pause here. Insert a breakpoint in your `code` and slowly step through it so we can examine the current values of your variables and parameters.

```
* PAUSE
```

Besides collecting evidence and managing your manager, you need to manage yourself too. A proven way to do this is to periodically reflect on your work. Your employer may have procedures to help you do this, such as performance reviews or one-to-ones with your manager on a regular basis. Whatever the setup, you will benefit from taking time to reflect on:

1. What have you been doing?
2. WWW: What Went Well?
3. EBI: Even Better If?

11.5.1 What have you been doing?

Briefly describe your roles and responsibilities. What projects have you worked on? What were the main technologies that you used? As well as describing this to colleagues, you should aim to communicate this with non-specialists, people outside your field. How would you describe your job to your friends and family and terms they would understand?

11.5.2 WWW: What went well?

Are there any projects you are particularly proud of? What new knowledge or skills have you learned or improved? Remember to include both non-technical as well as technical aspects of your job. Non-technical skills include organisation, time-management, confidence, communication etc.

11.5.3 EBI: Even Better If?

What areas have you identified for improvement in the future? Again, this includes non-technical as well as technical skills.

11.5.4 Your managers view

If you asked your manager the same questions, would they come up with the same answers? Are there any differences between your view and your managers view of your work? If so, why do they differ?

```
* RESUME
```

11.6 Summarising survival

Too long, didn't read (TL;DR)? Here's a summary:

This chapter is under construction because I'm using agile book development methods, see figure 11.3.

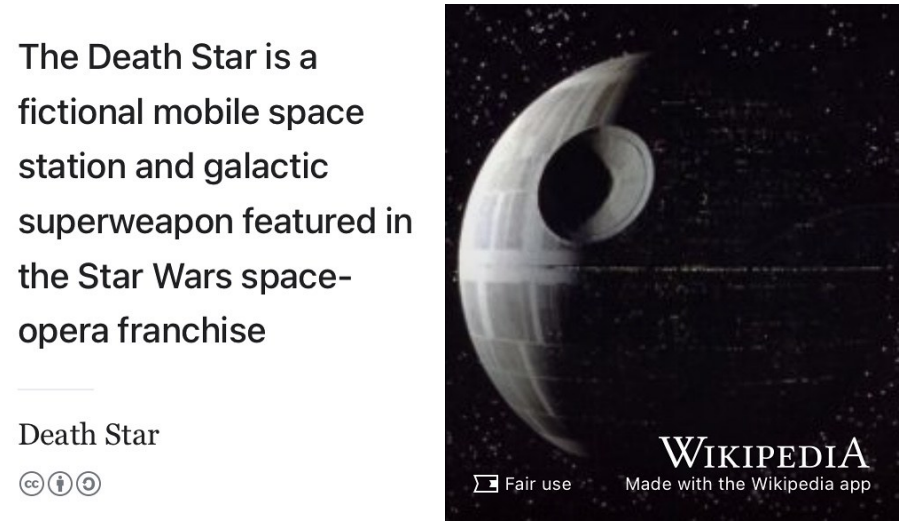


Figure 11.3: Just like the Death Star, this ~~galactic superweapon~~ chapter is under construction. Image of agile weapon engineering in *Star Wars* via Wikimedia Commons w.wiki/32PB adapted using the Wikipedia app

Chapter 12

Achieving your future

Learning is a lifelong process, a `while` loop in which you continuously develop new skills and knowledge. As you loop the loop, you will collect evidence of your personal and professional achievements. Some of these achievements can be certified or "badged" in various ways. This evidence can be collected as part of your professional identity and reputation, the jar called `ME` in figure 12.1.

Evidence is a key part of your "Context, Action, Result and Evidence" (CARE) story described in section 7.6.2. So what evidence can you collect and how you can certify or badge it?

12.1 What you will learn

After reading this chapter you will be able to:

- Describe some the evidence you can collect and badge to show your achievements:
  - during University
  - after University and throughout your professional career
- Identify any gaps in your personal and professional achievements
- Plan activities and set goals for future achievements that will help you to continue grow professionally and personally

12.2 Academic badges

One kind of badge you get when you finish University is your degree certificate like the one shown in figure 12.2. A degree certificate is an important

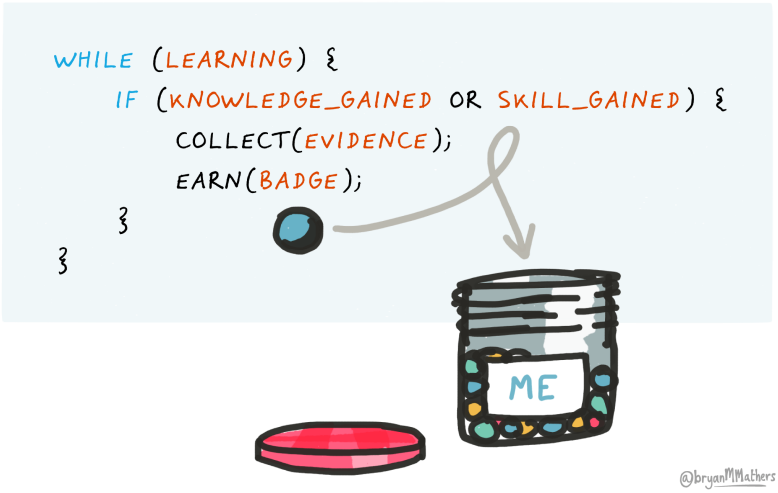


Figure 12.1: Learning is a lifelong loop where you constantly acquire knowledge and skills. You can collect evidence of your development, some of which can be certified or badged during University and throughout your professional career. Computing Badges by Visual Thinkery is licenced under CC-BY-ND

offline physical (paper) badge that marks a milestone in your career. If you like gaming, its a huge `achievement unlocked` that will take you to the next level. Your certificate also tells people that you were a member of a particular University community and that you mastered your chosen discipline to some level, Bachelors, Masters or PhD.

Figure 12.2: Level up, achievement unlocked! A degree certificate is a milestone that provides evidence of your academic knowledge and skills gained while at University. A certificate is just an offline badge by Visual Thinkery is licensed under CC-BY-ND via Doug Belshaw

Degree certificates are an important badge, but they don't give very many details of your professional and personal story while at University. You could give more details by providing:

- your overall degree classification: first, second, third etc
- your individual module grades, for example in an academic transcript or by listing them on your CV, see section 7.5.2
- your projects, see section 7.5.4
- your portfolio of work, if you have one

This data give a *bit* more detail that a degree certificate does not but it is limited to purely academic achievements. You are *much* more than your grades, because there's a lot about your character that is difficult or impossible to measure, see figure 12.3.

Figure 12.3: Your grades give more detail than a degree certificate but they still don't say much about you. I am more than just my grades sketch by Visual Thinkery is licensed under CC-BY-ND

Employers will often want to see more detailed evidence of your character and your softer skills than those provided by degree certificates and grades. While academic achievements paint some broad brushstrokes of your professional identity shown in figure 12.4, they don't help employers see the finer details or much of the evidence.

12.3 Digital badges

Digital badges provide a solution to this problem, just like your degree certificate is a verified badge of your achievements, a digital badge does the same but in a digital way. Rather than being physical, a digital badge is virtual and transferable. It's just a `*.png` graphic file which has been digitally signed and contains metadata. This means it can be displayed on your CV, on social media such as LinkedIn or "stacked" into a digital portfolio collection as shown in figure 12.5.

The Mozilla Foundation have defined an open standard for digital badges (Foundation, 2019) so that badges can be issued, hosted and certified by different organisations. (Belshaw, 2019)

A digital badge has certified data locked inside (shown in figure 12.6) which

Figure 12.4: What kind of picture would you like to paint of your professional identity? Open Badges paint a better picture... by Visual Thinkery is licenced under CC-BY-ND

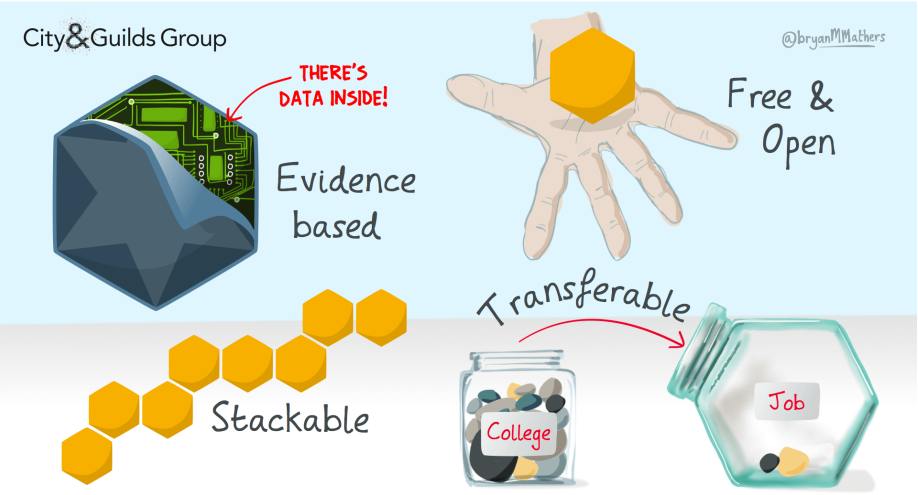


Figure 12.5: Digital badges have certified evidence inside, can be transferred between education and employment and can be stacked into collections or portfolios. Properties of Open Badges by Visual Thinkery is licenced under CC-BY-ND for the City and Guilds of London Institute

details the achievement it has been awarded for including:

- `recipient` that's you!
- `issuer` the organisation awarding the badge, e.g. Poppleton University
- `badge name` e.g. PASS leader badge (see figure 12.7)
- `badge image` e.g. a digital logo
- `evidence URL` a link to evidence

All this information is coded so that only the recipient and issuer can manipulate it, for example by associating an email address with it.

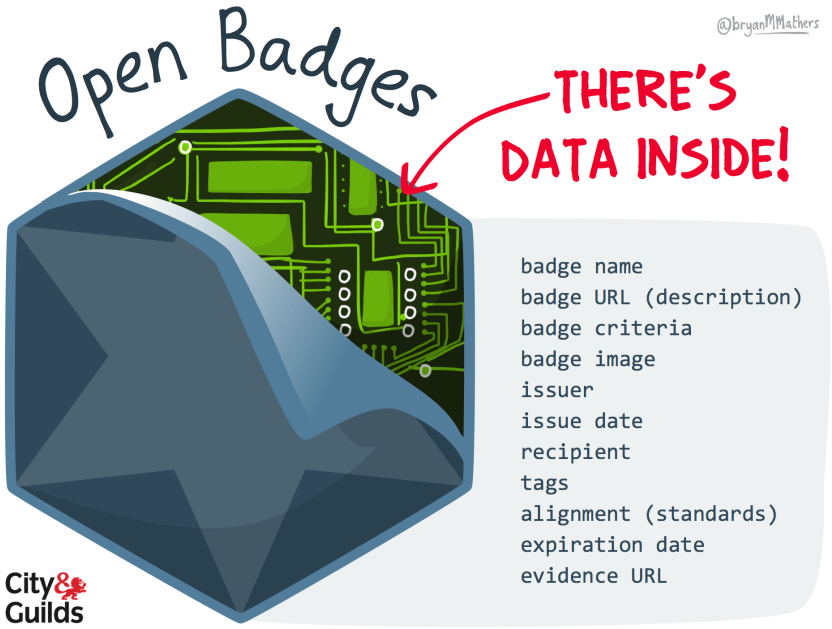


Figure 12.6: Open digital badges have data, like the `issuer` and `recipient`, locked inside them so they can be verified. There's data inside open badges by Visual Thinkery is licenced under CC-BY-ND for the City and Guilds of London Institute

Anyone such as your University or employer can issue badges, so for example, the University of Manchester issues badges for leaders of its Peer Assisted Study Scheme (PASS) www.peersupport.manchester.ac.uk. An example of a PASS leader badge is shown in figure 12.7.

Badges can be used for a wide range of different kinds of achievements shown in figure 12.8.

Figure 12.7: An example of a digital badge awarded to Peer Assisted Study Scheme (PASS) leaders at the University of Manchester who have mentored and helped others students during their academic study. If you're a University of Manchester student, you can see other badges available at wiki.cs.manchester.ac.uk/index.php/Badges (UoM login required)

1. **Membership** being a member of an organisation or group
2. **Participation** participating in communities
3. **Capability** demonstrating capability with some skill
4. **Mastery** mastering a set of skills

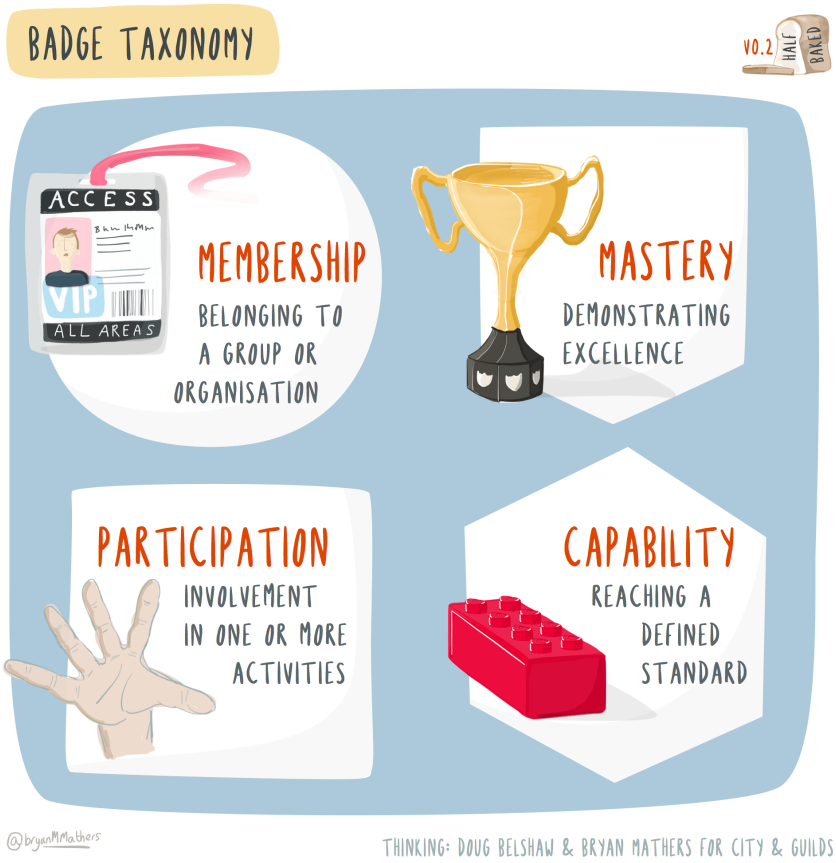


Figure 12.8: There are many different achievements which badges can be awarded for including membership, participation, capability and mastery. Badge taxonomy by Visual Thinkery is licensed under CC-BY-ND

12.4 Other digital badges

There are other digital badges for evidencing your achievements besides the open ones described in this chapter. Like open badges, they also provide certifiable evidence of professional and personal achievements, see figure 12.9:

- Amazon Web Services badges credly.com/organizations/amazon-web-services/badges
- Certificates from Microsoft and Google, see section 11.4.1
- Certificates from edx.org cs50.harvard.edu/x/2020/certificate
- Certificates from coursera.org coursera.org/professional-certificates
- Certificates from redhat redhat.com/en/services/certifications etc

Figure 12.9: An example of an Amazon Web Services badge awarded by credly.com for an AWS certified developer.

We have focussed on technical achievements here, but there are non-technical achievements too.

12.5 Breakpoints

Let's pause here. Insert a breakpoint in your `code` and slowly step through it so we can examine the current values of your variables and parameters.

```
* PAUSE
```

- Besides the badges and certifications already discussed, what others exist?
- For the skills and knowledge you already have, what evidence do you have for it?

- Where are the gaps in own skills or knowledge?
- What evidence can you collect that you are developing these skills and knowledge?
- What parts of this evidence are you able to badge and certify?

```
* RESUME
```

12.6 Summarising your achievements

Too long, didn't read (TL;DR)? Here's a summary:

Open digital badges let you take your achievements from the many places you learn, and combine them into a portable portfolio that tells a digital story about your learning. The data inside a badge is shown in figure 12.10 and gives an employer a more detailed and evidenced view of your professional development than traditional physical badges like degree certificates.

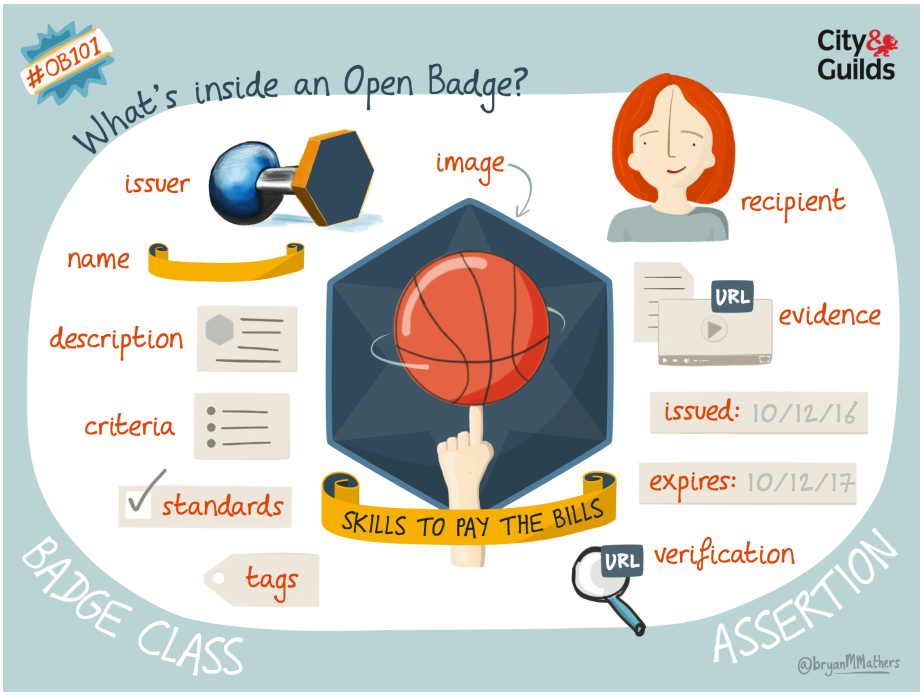


Figure 12.10: Open digital badges have certifiable metadata locked inside. What's inside an open badge? by Visual Thinkery is licenced under CC-BY-ND for the City and Guilds of London Institute

Chapter 13

Researching your future

So you want some more, eh? Your undergraduate degree has whetted your appetite. What are the options for postgraduate study and research? Where can they take you and are they worth investing your time and money in? You are a curious character. You like the idea of pushing the boundaries of human knowledge a little further, maybe you even fancy yourself as the next Ada Lovelace or Alan Turing?

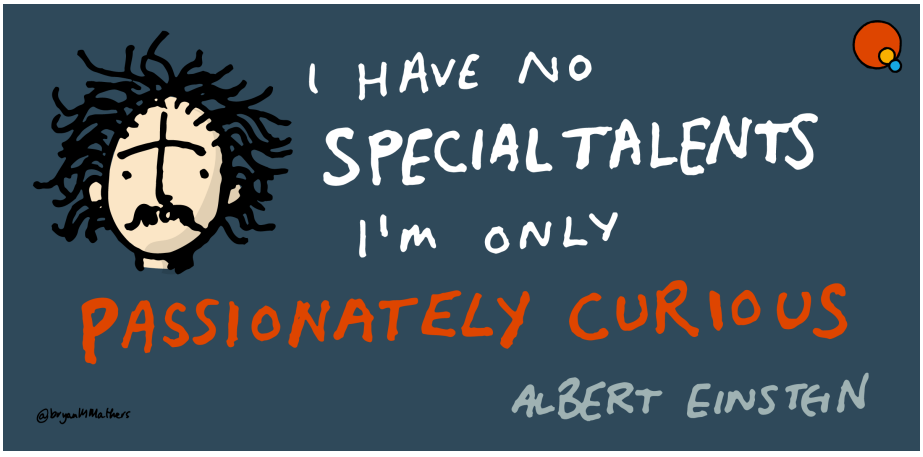


Figure 13.1: Are you passionately curious? Is further study or research the right path for you? Are you the next Einstein or Einsteiness? This chapter looks at the options. Curiosity by Visual Thinkery is licensed under CC-BY-ND

13.1 What you will learn

At the end of this chapter you will be able to:

- Describe the costs of postgraduate study and research
- Describe the benefits of postgraduate study and research

13.2 Where to start

A good place to start if you're looking for a masters or PhD are:

- Apply directly to Universities for postgraduate study, if there is a specific group or course you are interested in. See also:
- findamasters.com for postgraduate study, a directory of Masters degrees and postgraduate qualifications at universities around the world
- findaphd.com for postgraduate research, a large database of PhD opportunities
- jobs.ac.uk also lists PhD opportunities, not just in the UK

13.3 Breakpoints

Let's pause here. Insert a breakpoint in your `code` and slowly step through it so we can examine the current values of your variables and parameters.

```
* PAUSE
```

1. When is the best time to do a masters, straight after your undergraduate degree or after working for a while?
2. How much does a Masters degree improve career prospects?
3. How much does a PhD improve improve career prospects?
4. Is postgraduate study and research really worth all the pain and suffering?
5. What careers can a PhD lead to?

```
* RESUME
```

13.4 Signposts from here on research

A good place to start if you're thinking about doing a PhD (or trying to get through one) is *How to get your PhD: A Handbook for the Journey* by Gavin

Brown. (Brown, 2021) I wish I'd had this book when I was a PhD student! I'm not just saying that because Gavin is a colleague of mine but this is a genuinely useful book which quickly tackles a wide range of issues you'll encounter during a PhD from the technical to the psychological. The second half also contains a range of short viewpoints on doing a PhD from people including Nancy Rothwell, Victoria Burns, Steve Furber, Lucy Kissick, Hiranya Peiris, Melanie Leng, Jeremy Wyatt, David Hand, Carolyn Virca, Shakir Mohamed, Jonny Brooks-Bartlett and Jennifer Polk. If you're serious about doing a PhD, you should read Gavin's guidebook.

> "How to get Your PhD: A Handbook for the Journey" (@OUPAcademic https://t.co/Q9dTmWQ9iF) is out today. Preview on Google Books https://t.co/US8O2EeaQf / #howtogetyourphd (14/14)
>
> — Gavin (@profgavinbrown) March 1, 2021

13.5 Summarising further study and research

Too long, didn't read (TL;DR)? Here's a summary:

This chapter is under construction because I'm using agile book development methods, see figure 13.2.

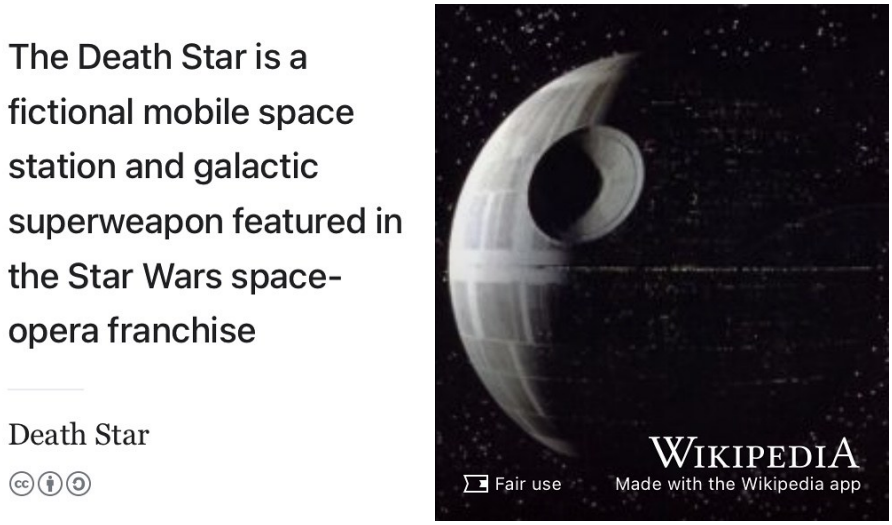


Figure 13.2: Just like the Death Star, this ~~galactic superweapon~~ chapter is under construction. Image of agile weapon engineering in *Star Wars* via Wikimedia Commons w.wiki/32PB adapted using the Wikipedia app

Part III

SUPPORTING

Chapter 14

Ruling your future

In 2005, the scientist and engineer Phil Bourne starting publishing a series of articles which distilled people's hard won knowledge into *Ten Simple Rules*. (Bourne, 2005) Over a decade more than 1000 rules were published in over 100 articles in the scientific journal *PLOS Computational Biology*. (Bourne et al., 2018) These articles offer a huge range of advice from making the most of a summer internship (Aicher et al., 2017) to teaching programming (Brown and Wilson, 2018) and even winning a Nobel Prize. (Roberts, 2015) Articles as lists, or "listicles" as they are sometimes known, are a convenient way to summarise key points. So here are *Ten Simple Rules for Coding Your Future*: the too long, didn't read (TL;DR) summary of this guidebook.

14.1 Know who you are

There is a lot more to you than your degree. Yes, you've spent (or will be spending) three or four years getting your degree. Use this time to identify your weaknesses and work out how to improve them. Knowing your future depends on knowing who you are now, see chapter 2 and figure 14.2.

14.2 Look after yourself

Look after yourself mentally and physically. Studying at University can be enjoyable but it can also make you stressed, anxious and depressed. Choose your reference points carefully, try not to compare yourself to the person at the top of the class. Ask yourself, am I doing better than last time? Be kind to yourself because nurturing yourself now will nurture your future, see chapter 3.

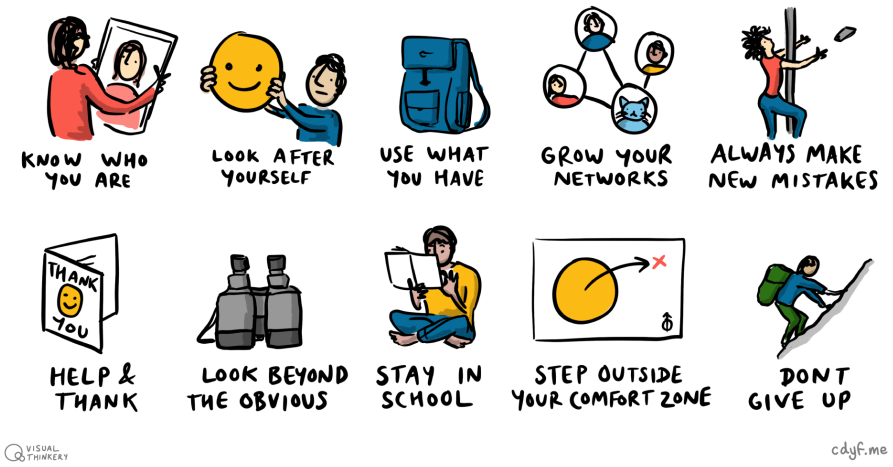


Figure 14.1: Ten Simple rules for coding your future. Know who you are, look after yourself, use what you have, grow your networks, always make new mistakes, help and thank, look beyond the obvious, stay in school, step outside your comfort zone and (*most importantly*) don't give up! Figure by Visual Thinkery is licensed under CC-BY-ND

Figure 14.2: How well do *really* you know yourself? Know who you are sketch by Visual Thinkery is licensed under CC-BY-ND

14.3 Use what you have

This rule is borrowed from software engineer Greg Wilson @gvwilson in figure 14.3, who probably adapted it from a quote frequently misattributed to Theodore Roosevelt (Brewton, 2014).

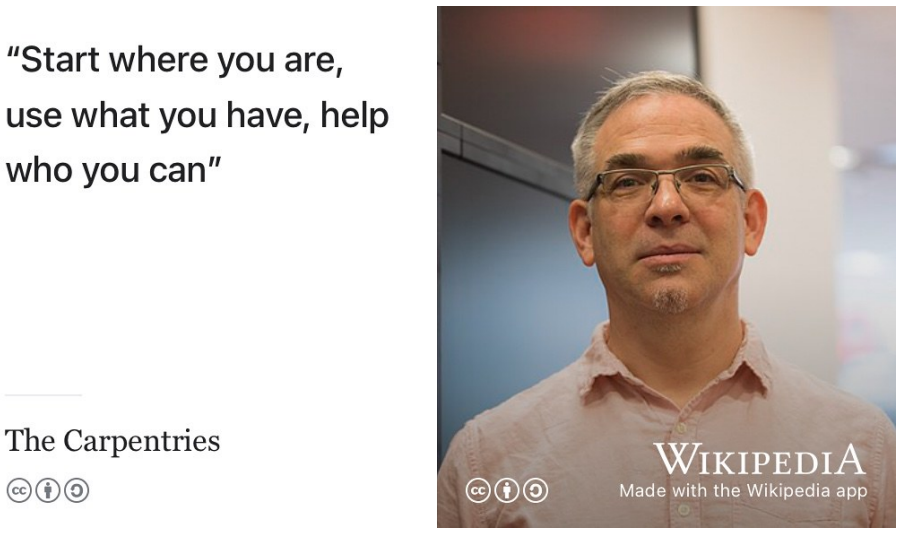


Figure 14.3: "Start where you are, use what you have, help who you can." — Greg Wilson at third-bit.com and software-carpentry.org. CC BY Portrait of Greg Wilson at The Carpentries via Wikimedia Commons w.wiki/3a6V adapted using the Wikipedia App.

When you're job hunting, it can be competitive and cut-throat. You can sometimes fall into poor habits of mind:

- "*If I had a better degree from a different university, I'd be more successful…*" see section 1.6
- "*If I had more experience, more voluntary work, more internships, I'd stand a better chance…*" see section 5.3
- "*If I'd done more projects and extra-curricular activities…*" etc see section 7.5.4
- "*If I'd got better grades at school and Uni…* " see section 12.2
- "*If only I'd worked harder…* " see 2.7.1
- "*If I was more confident at speaking and interviews …* " see chapter 10 on *Speaking your future*
- "*If I'd been to a different school, I could be more successful…*" see the 93percent.club (Nye, 2021; Verkaik, 2021)

Coulda. Woulda. Shoulda. This is all nonsense from your inner critic that you

should acknowledge, see section 3.3, then distance yourself from. Start from where you are, use whatever you have and help who you can.

14.4 Build your networks

Build your networks, use all the people you can, because people can help you. Remember that the weaker ties in your network (see section 8.6) may be more important than your stronger ties, especially when it comes to finding jobs.

Figure 14.4: Your network can help you before, during and after your employment, so be sure to grow your network when you can. This includes both the stronger ties of trusted friends alongside the weaker ties too, described in section 8.6. Both are important. It's all about people by Visual Thinkery is licensed under CC-BY-ND

It's not what you know, it's who you know.

14.5 Always make new mistakes

You can classify your mistakes and failures into two categories:

1. Productive mistakes: those you learnt from
2. Unproductive mistakes: those you didn't learn anything from (and risk repeating)

Mistakes and failure are inevitable in life, but productive mistakes are going to help you much more that unproductive ones (Petroski, 1992). That doesn't just mean you should "*fail fast, fail often*" (Babineaux, 2013) or "*move fast and break things*", but to conciously learn from any mistakes you make so that you don't repeat them. One way to turn unproductive mistakes into productive ones is deliberately and conciously reflect on why you made them. This is part of the growth mindset we discussed in chapter 3.

In a growth mindset, mistakes can be good, but the fear of making them is not. You are more likely to take more chances when you're unafraid to fail, and this will improve your chances of success.

Many education systems around the world don't teach people how to fail, because they put too much emphasis on success (as measured by grades) rather than progress and learning. (Lahey, 2016) So as the angel investor Esther Dyson once said, "Always make new mistakes", see figure 14.5

Figure 14.5: Mistakes are inevitable in life, so there's no shame in making them especially if they are new. Making *new* mistakes can be a form of productive failure that you learn from rather than a source of unproductive failure that you repeat (old mistakes). Portrait of Esther Dyson by Christopher Michel (CC BY-SA) via Wikimedia commons w.wiki/3TEY adapted using the Wikipedia App.

So:

- If you've got some harsh feedback on your CV, how can you make less buggy in the future?
- If you've applied to lots of companies and not even had a reply yet, how you improve your job search strategy?

- If you've neglected to develop interests and projects outside of work, how can you rebalance?
- If you crashed and burned in an interview, how can you use the experience to do better next time?
- If you failed to get the promotion you thought you deserved, what will you do differently in the future

14.6 Help and thank who you can

There are good reasons to be grateful, showing gratitude doesn't just help other people, it helps you too.

Join a team by helping someone, be a team player, help others, thank others for their help.

14.7 Look beyond the obvious

Be flexible in approach. Not just big employers, there are startups, SME's you've never heard of. It's not just London (see chapter 16), and other big cities. Look beyond graduate schemes, look beyond graduate jobs. Broaden your horizons and your job search, see chapter 9.

You are not just a techie, Either.

14.8 Stay in school

Part of what you learn during your education is *how* to learn. But your learning shouldn't finish when you leave University. Computer science is a young and rapidly changing discipline which means you can not afford to be left behind. Never stop learning, see chapter 12.

14.9 Step outside your comfort zone

Am I being insensitive asking people to step outside their comfort zone when we've all be stretched beyond breaking point during COVID-19?

We're going to need to continue to step outside of our respective comfort zones in order to meet the challenges we face around the world.

This takes courage, but that's often when you learn most. So step outside your comfort zone if you're feeling brave enough to learn.

14.10 Don't give up

Learn to live with rejection, don't take it personally

14.11 Ten simple summaries

This chapter is under construction because I'm using agile book development methods, see figure 14.6.

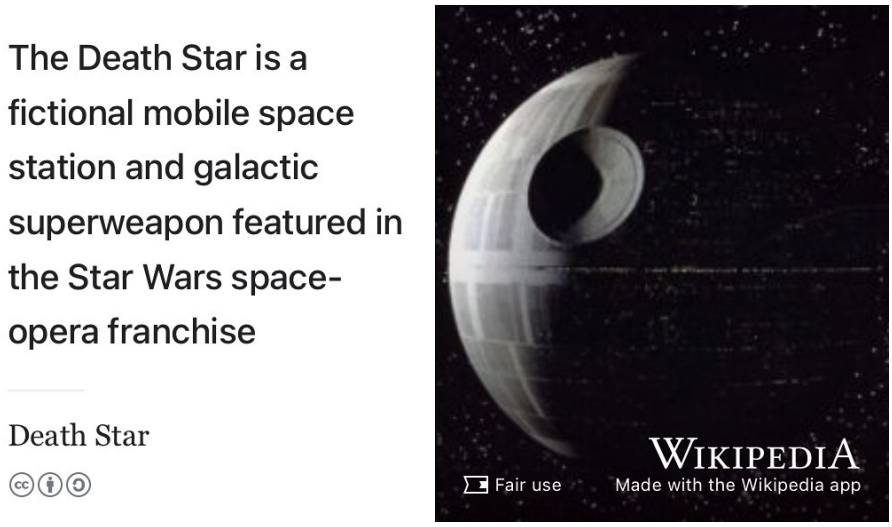


Figure 14.6: Just like the Death Star, this ~~galactic superweapon~~ chapter is under construction. Image of agile weapon engineering in *Star Wars* via Wikimedia Commons w.wiki/32PB adapted using the Wikipedia app

Chapter 15

Hacking your future

It's very easy to overlook mistakes in your own writing. That's true of any written communication such as a covering letter, personal statement, email or any message that you write. Mistakes are particularly common in CVs (or résumés) because you can spend *hours* carefully polishing the words and the formatting but not see a fatal error at the top of page one. Hacking[1] other people's CV's will help you improve your own. You may need to use ingenious hacks like the one on the bridge in figure 15.1. Returning to the bridge analogy from section 0.5.1, fixing other peoples bridges will help you improve your own bridge. You'll build better bridges to more interesting and ambitious destinations.

Figure 15.1: Your CV is a bridge which enables you to cross from where you are now to where you'd like to be in the future. Like the bridge in this picture, the CV's in this chapter are all faulty in some way, can you fix them? You may need to use ingenious hacks and kludges like the one shown here on the Million Dollar Bridge in Alaska . Public domain image of kludgy repairs adapted from an original by Jet Lowe on Wikimedia Commons w.wiki/3Uvn

[1]The term hacking is a horribly overloaded word with many different meanings, but I'm using it here to mean an "ingenious temporary solution to a problem"

15.1 Hack their CVs

A useful hacking technique is to take somebody elses stuff and fix or improve it. That works for written communication as well as actual `code`. The dogfooding technique described in section 4.4.1 is a useful hack which you can use by:

1. Eating your *own* dogfood by reading your own written work ALOUD
2. Eating *somebody elses* dogfood by:
    - Hacking a friends or peers CV by swapping with them and giving them constructive feedback
    - Hacking the fictitious CVs below by ranking them against a job advert. Who would you want to interview and why?
3. Eating your own dogfood again, re-reading and re-editing repeatedly
4. Persuade someone else to eat your dogfood - get feedback from as many people (and bots see section 7.6.6) as you can

So, here are some fictitious CVs for you to hack, from students of Computer Science. They are based on CVs I've seen, warts and all, with personal information removed and anonymised. Can you spot their triumphs and tragedies? Can fix their CVs and work out which candidate is best for the sample job description at CoolTech in section 15.4? Can you hack their future?

Special thanks to Toby Howard and Sean Bechhofer for coming up with some of these silly names for fictional students in computing. Please direct any complaints about the terrible geeky puns to Toby and Sean! Thanks also Ben Carter and Penny Gordon Lanes in the careers service at the University of Manchester, some of these CVs are based on examples they have collected and anonymised.

15.2 Breakpoints

Let's pause here. Insert a breakpoint in your `code` and slowly step through it so we can examine the current values of your variables and parameters.

```
* PAUSE
```

When you read these CVs make a note of:

1. **What Went Well?** (`WWW`) What do you like about any given CV, what have they done well?
2. **Even Better If?** (`EBI`) What could be fixed or improved, can you hack it?

3. **Their Rank order** (`1,2,3...`) Who is top of your list to interview? Who is going in the bin and why?

```
* RESUME
```

Imagine the person is real, what would you tell them about their CV if they'd given it to you for advice without hurting their feelings? How could you be a critical friend by giving them actionable feedback?

15.3 Sample CVs

Hacking other people's CVs will help you improve your own because you're putting yourself in the shoes of your reader. Here are some samples:

15.3.1 Penelope Tester

Penny Tester, or Pen as her friends call her, loves cybersecurity and reverse engineering. She has a real passion for finding vulnerabilities in software and hardware. Just don't call her a hacker she hates that word, see 15.2.

Figure 15.2: Penny Tester's full CV can be viewed at cdyf.me/Penelope_Tester.pdf

15.3.2 Rick Urshion

Rick is a big fan of functional programming and loves expressing himself using languages like Lisp, Haskell, Clojure, Erlang and Scala. He really hates side-effects but tries to avoid getting into a state about it. His critics say he can be inefficient but Rick insists he's just lazy, see figure 15.3.

15.3.3 Marge Conflict

Marge loves version control and her superpower is resolving people's differences.

(Marge is currently updating her CV)

Figure 15.3: Rick Urshion's full CV can be viewed at cdyf.me/Rick_Urshion.pdf

15.3.4 Michael Rokernel

Mike loves operating systems, but not if they get too bloated, see figure 15.4.

Figure 15.4: Mike Rokernel's full CV can be viewed at cdyf.me/Mike_Rokernel.pdf

15.3.5 Florence Ting-Point

Flo loves maths and is a particularly big fan of floating-point arithmetic, see figure 15.5.

Figure 15.5: Flo Ting-Point's full CV can be viewed at cdyf.me/Flo_Ting-Point.pdf

15.3.6 Peter Byte

Peter and his twin sister Peta, both love big data, machine learning, statistics, data science and Artificial Intelligence (AI). They come from a big family with nine siblings, Deca, Hector, Kilo, Megan, Giga, Terry, Exa, Zita and Yotta. They are wildly ambitious, but critics say the Byte family have been terribly over-hyped, see figure 15.6.

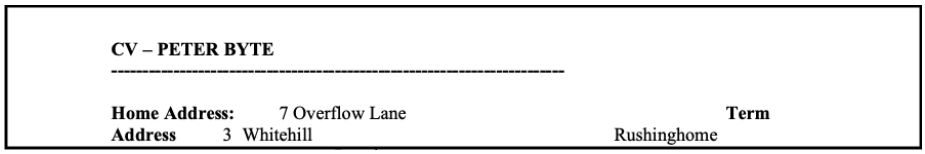


Figure 15.6: Peter Byte's full CV can be viewed at cdyf.me/Peter_Byte.pdf

15.3.7 Polina Morphism

Polly loves object-oriented programming. She has lots of siblings, and a cousin called, Isa.

(Polina is currently working on her CV)

15.3.8 Neil Pointer

Neil is a mature student who loves the C programming language, see figure 15.7. The Pointer family are sometimes misunderstood, but Neil compensates for this with his excellent memory management skills and efficiency. He has a younger half-brother, Neil Pointer-Exception, from his fathers second marriage. Neil Pointer-Exception prefers Java .

Figure 15.7: Neil Pointer's full CV can be viewed at cdyf.me/Neil_Pointer.pdf

15.3.9 Bryn Hanby-Roberts

The last CV is a real one. Bryn kindly gave his permission to share it with you, see figure 15.8. Bryn graduated in 2016, his CV is longer as he has five years of experience under his belt but it provides a useful counterpoint to the examples above. Thanks Bryn.

15.4 Sample CoolTech Job advert

We're looking for bright and geeky graduates to join our software engineering team. No experience is required, and many of our successful applicants have

Figure 15.8: Bryn Hanby-Roberts full CV can be viewed at cdyf.me/bryn.pdf a snapshot taken in 2021 from bryn.co.uk

never programmed before. If you think logically and enjoy problem solving, then you have the potential to become a great developer.

A career at CoolTech will challenge you every day. In your first few weeks you will be solving real-world problems as you help to develop software used by professionals across the world.

You'll be part of an agile development team, working on one of the largest real-time databases in the world. You'll work on a wide variety of projects, ranging from Artificial Intelligence assisting clinicians with early diagnosis of cancer to an iOS app helping patients manage their diabetes.

Developers at CoolTech are involved in the full software cycle, and work closely with all teams across the company to scope out new projects as they design, develop and deploy our products.

Chapter 16

Moving your future

Finding a job often involves moving to another city or even another country. So a simple way to improve your job prospects is to broaden your job search (chapter 8) and consider all the alternatives, geographically and otherwise. Some of the alternative options are outlined in chapter 9. This chapter looks at some UK alternatives to London.

Figure 16.1: Like many capital cities, London dominates the economy of its country. There are clearly some fantastic employers offering great opportunities in London but they do not represent everything that is on offer in the UK. There are plenty of good career options outside of capital cities like London if you don't want to live and work in a huge metropolis. Not just London Image by Sharon Dale (Dale, 2018)

For example, if you're seeking work or further study in the United Kingdom, there are lots of possibilities in Loxbridge (**Lo**ndon, **Ox**ford and Cam**bridge**). While Loxbridge undoubtedly offers many fantastic opportunities for professional growth and development, it does not represent *all* of the best opportunities that exist in the UK.

So, it's wrong to assume that capital cities like London are where *all* the best

opportunities are. That's true in the UK and also in many other countries too. You might need to consider moving your future.

This is a partial list of employers in the North West of England (aka the Northern Powerhouse) that recruit Computer Science students. This is not a comprehensive list of all tech companies in the North West, but will give you a quick flavour of employers in the Manchester, Leeds, Liverpool, Sheffield and Northern England.

16.1 Hit the North, Not Just London

You don't have to move to London to find top employers, there are plenty located here in the North West if you'd like to stick around after you graduate and Hit the North. (Smith et al., 1987) Northerners have often argued that Northern England offers a better quality of life than London we couldn't possibly comment other than to say its horses for courses. The North West Tech Community calendar, provides a window on (and networking opportunities with) many of employers based in the North West technw.uk if you want to find out more. To quote sharon dale, its NotJustLondon (Dale, 2018)

The list of employers in this page is currently being migrated from git.io/manc, visit that page for a list of employers from the North West.

16.2 techUK and technation.io

In addition to these techuk.org and technation.io provide more information on technology businesses outside of London.

16.3 Guest lectures from employers

The Department of Computer Science welcomes external speakers and hosts a number of guest lectures from a wide range of collaborators in industry and academia. There are several ways the department can host guest lectures

- Scheduled lectures for COMP101, a first year course with 400+ enrolled students. During 2021, these talks take place on Mondays from midday to 12.50pm on zoom
- School research seminars , see examples of past seminars
- Technical talks arranged as part of a scheduled course, speak to the course leader for the relevant course. These tend to be more advanced courses aimed at students later in their degrees or as part of postgraduate study.

- One-off talks arranged *ad hoc* that are not part of scheduled series of lectures
- Seminars and talks from speakers arranged and booked by students, for example unicsmcr.com, The University of Manchester Computer Science society

16.3.1 Examples of COMP101 lectures

Some example guest lectures for COMP101 are shown below to give a flavour of the kind of talks that are appropriate

- Hacking the Hacks, delivered by NCC Group
- Debunking the myths associated with User Experience, delivered by American Express
- The Business of Intellectual Property, delivered by Imagination Technology)
- How to Break a Hacker's Mind, Web Application Vulnerabilities Exposed, delivered by Morgan Stanley
- 100 billion ARM chips, delivered by ARM
- Software at Airbus
- Computing in the Community, delivered by CodeClub & Manchester Girl Geeks
- How to be a brilliant software engineer, delivered by Apadmi

16.3.2 Interested in speaking?

For COMP101, we are always looking for good speakers who can engage large groups of students on interesting topics that they care about and relate to Computer Science. For COMP101, there are a limited number of guest lecture slots (around 20) which run through term-time starting in November and finishing in early June. Its important that speakers

- Give much more than a sales pitch for an organisation, by providing insight into a technical subject
- Talk about content that relates to the (very broad) syllabus of COMP101
- Engage, interact, educate and entertain. Students vote with their feet (by not turning up) if they think a lecture won't be interesting
- If you would like to propose a guest lecture for COMP101, please contact Duncan Hull. For all other external seminars and events, see the links above.

Since lockdown, lectures have been online, which has allowed for more interaction, typically via the chat dialogue. We can monitor and moderate the chat, feeding questions to the speaker when it's appropriate.

Chapter 17

Hearing your future

This podcast features interviews with students as they come to end of their undergraduate degree or shortly after they graduate. We interview students to find out more about their journey from student to professional. During the interview, we ask students to tell us:

- What's their story? Tell us about themselves
- Where does their interest in computing come from
- Which organisation they were employed by, why and how did they chose them
- Tell us about their role in the organisation
- How did they find the job and what other jobs did they look for?
- What was the most enjoyable or rewarding part of working for their employer
- What advice would they give to students looking for placements
- What are their plans for the future
- What's the most interesting thing happening in computer science / technology right now?
- What are their favourite work related radio shows or podcasts?

17.1 Episode 1: Raluca Cruceru

Interview with Raluca Cruceru careers.cern/Raluca, a software engineer at CERN, will be published here shortly. Raluca graduated with BSc in Human Computer Interaction with Industrial Experience in 2020.

17.2 Episode 2: George Dunning

Upcoming interview with George Dunning, software engineer at steama.co in Manchester will be published here. George graduated with a BSc in Computer Science with Industrial Experience in 2020.

17.3 Episode 3: It could be you!

If you'd like to be interviewed, get in touch.

Chapter 18

Actioning your future

Employers are often more interested in what you have *done*, rather than what you just *know*. Your actions are a key part of your story we discussed in section 7.6.2. A simple technique for emphasising the `action` in your stories is to lead descriptions of your projects, education and experience with carefully chosen `verbs`, see section 7.6.4 for examples.

18.1 Your actions define your impact

Your actions define your impact, see figure 18.1. What stories you can tell of your actions to date? What verbs best describe how you achieved a result or had an impact? What was the context, action, result and evidence (CARE) we discussed in section 7.6.2 of each (short) story?

By leading with verbs you will highlight what you have *actually done* and how you did it, rather than what you know. See the verbs first section 7.6.4 of chapter 7 *debugging your future*.

18.2 What you will learn

By the end of this chapter you will be able to:

- Emphasise your *actions* when describing your education, projects and experience
- Reflect on
  - what skills you already have
  - what skills you need to develop
- Demonstrate those skills explicitly and quickly in job applications

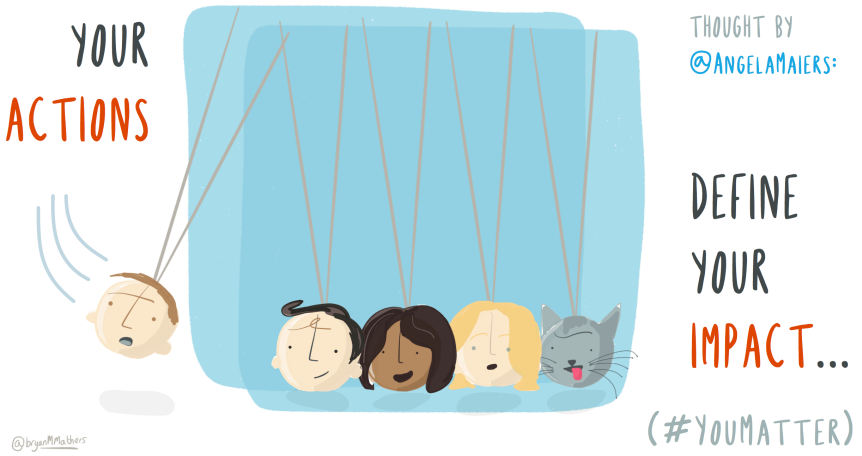


Figure 18.1: What action have you taken and what stories can you tell about the results and your impact? What are the best verbs for highlighting your actions? Your actions define your impact by Visual Thinkery is licensed under CC-BY-ND via Angela Maiers

18.3 Breakpoints

Let's pause here. Insert a breakpoint in your `code` and slowly step through it so we can examine the current values of your variables and parameters.

```
* PAUSE
```

Quickly scan your CV, covering letter or application form for VERBS:

- Where are the verbs?
  - Are verbs buried deep in long sections of prose or do they prominently lead descriptions of your activities?
- Have you over-used certain verbs (like `worked` or `assisted` for example) or been repetitive (like `developed`)
- How can you increase the variety of verbs you have used (without exaggerating or lying)?
- Which verbs are stronger than others and why?
- Are there any categories of verbs you can't provide evidence for, such as leadership (see section 18.6) or influencing (see section 18.11)?
- What activities or projects could you do that would help you develop these missing skills?

```
* RESUME
```

18.4 Team verbs

Some verbs to demonstrate how you have worked and communicated with others in a team.

- `administered`
- `advised`
- `assisted`
- `coached`
- `collaborated`
- `encouraged`
- `explained`
- `instructed` if you helped others
- `interviewed`
- `organised`
- `participated`
- `attended`
- `presented`
- `recommended`
- `recruited`
- `suggested`
- `volunteered`

18.5 Engineering verbs

Verbs to demonstrate your engineering and technical skills.

- `adapted` (e.g. new features)
- `added` (e.g. new features)
- `analysed` (e.g. the requirements)
- `architected`
- `assigned` (e.g. bugs to team members)
- `automated` (e.g. builds and tests etc)
- `built`
- `branched` (e.g. git)
- `configured`
- `designed` (e.g. greenfield software development)
- `cloned` (e.g. git)
- `debugged` (e.g. brownfield development)
- `developed`

- `deployed`
- `documented`
- `experimented`
- `gathered` (e.g. requirements)
- `implemented` (e.g. an algorithm)
- `installed`
- `integrated` (e.g. different systems)
- `made`
- `merged` (e.g. git)
- `migrated`
- `modified`
- `solved`
- `specified`
- `upgraded`
- `tested`

18.6 Leadership verbs

Some verbs to demonstrate how you have used your initiative and taken the lead:

- `established`
- `created`
- `decided` (you've had the power to make decisions)
- `devised`
- `directed`
- `facilitated`
- `introduced`
- `launched`
- `led`
- `managed`
- `mentored` if you've helped less experienced people
- `motivated`
- `supervised`

18.7 Improving verbs

Verbs that demonstrate how you have improved a situation by taking resposibility for something:

- `delivered`
- `completed` if you finished something

- edited
- enhanced
- generated
- increased make sure you quantify it, see section 7.6.2
- refined
- resolved a conflict
- saved money, time, resources etc
- transformed

18.8 Scientific verbs

Verbs that demonstrate your analytical and scientific skills

- assessed
- calculated
- discovered
- estimated
- evaluated
- identified
- interpreted
- investigated
- measured
- modelled (in a computational or mathematical sense)
- proved
- quantified for example in benchmarking
- researched

- reviewed
- tested

18.9 Winning verbs

Verbs for demonstrating your achievements and honours

- achieved
- attained
- awarded
- nominated
- recommended
- selected (you were chosen for something)
- mastered
- won

18.10 Organising verbs

Verbs to demonstrate your planning and organisational skills:

- `arranged`
- `prepared`
- `scheduled`
- `organised`
- `planned`
- `produced` making things (but not just software)
- `revised`

18.11 Influential verbs

Verbs that demonstrate how you have influenced and persuaded others:

- `bought` if you've had purchasing power
- `guided`
- `demonstrated`
- `illustrated` if you have graphical skills for example
- `influenced` could even include social media influencing
- `liaised`
- `negotiated`
- `marketed`
- `mediated`
- `promoted`
- `presented`
- `publicised`
- `sold` an idea, product or service
- `written`

18.12 Summarising your actions

Too long, didn't read (TL;DR)? Here's a summary:

Actions speak louder than words, or as suffragette and political activist Emmeline Pankhurst frequently said "Deeds not Words", see figure 18.2. Your CV needs to emphasise your deeds and actions using words, those words are verbs.

Leading with verbs is a simple but powerful technique that enables you to provide evidence (rather than assertion) for the skills and knowledge you have. Choose your verbs carefully. Which verbs are missing from your CV? These verbs can help you identify gaps in your professional and personal development.

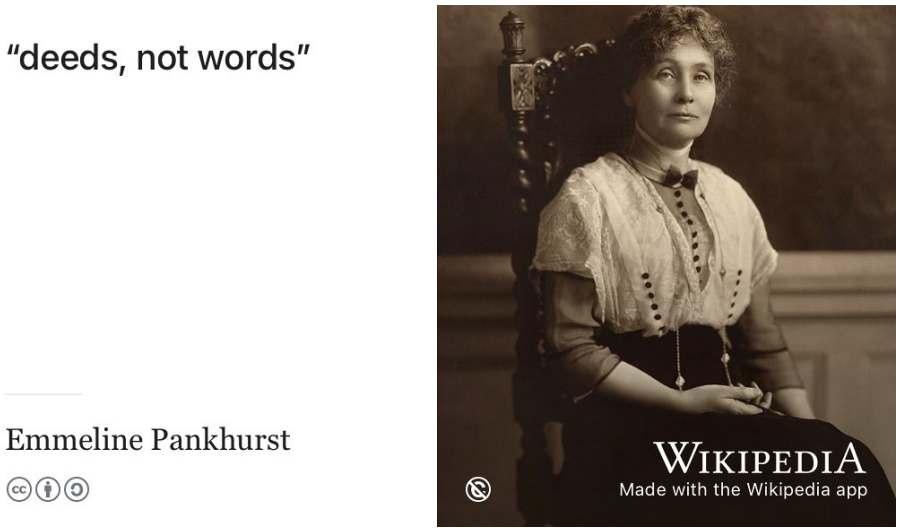


Figure 18.2: "Deeds not words" was the rallying cry of suffragette Emmeline Pankhurst. Emphasise the deeds on your CV by using carefully chosen verbs. Public domain image of Emmeline Pankhurst by Richard Gordon Matzene restored by Adam Cuerden on Wikimedia Commons w.wiki/3bPa and adapted using the Wikipedia app

Chapter 19

Scheduling your future

You might find it a bit scary thinking about your future. You might be tempted to procrastinate making important decisions about your future, see figure 19.1. There is a risk of getting stuck in a `do-nothing` or busy waiting loop. This guidebook is here to help you break out of that loop. One way to breakout of an unproductive loop is to schedule some personal development time every week where you work on job applications. Doing good applications takes time and you'll probably find you can't do as many applications as you might like.

If you're a University of Manchester student, the live *Coding your Future* (COMP2CARS) sessions are also here to help. They'll be back in September 2021, see manchester.ac.uk/discover/key-dates. In the meantime see the *Wednesday Waggle* at waggle.cs.manchester.ac.uk

COMP2CARS complements the second year tutorials (COMP2TUT) at the University of Manchester and takes place in the same slot as COMP2TUT when you meet your personal tutor. See the timetable studentnet.cs.manchester.ac.uk/ugt/timetable.

For small group sessions and one-to-one meetings with your tutor, please turn your camera on, see section 19.1.

19.1 Cameras on or off?

I would normally expect participants in small meetings (not large ones like lectures) to turn their cameras on but I understand that there are good reasons why people may not be willing/able to and won't explicitly ask you to.

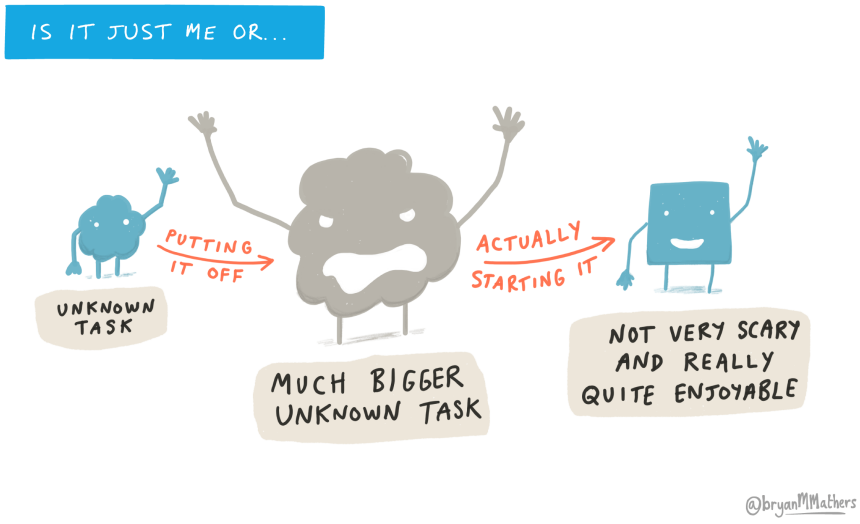


Figure 19.1: The biggest waste of time is the time spent not getting started on a project. Your future might seem big and unknown but it's really not as scary as you might think and getting started can be surprisingly enjoyable. New Project? Every time… by Visual Thinkery is licenced under CC-BY-ND

19.1.1 Why turn cameras on?

There has always been a question around whether to turn cameras on during online meetings but it is even more obvious with online meetings becoming the norm rather than the exception. I see a direct benefit in using cameras in small, personal meetings where many of us make use of visual cues to aid the flow of conversation – at the very least it's easier to identify who is talking. Additionally, it can help people get along – people might feel more 'listened to' if they can see somebody listening and personally I find it easier to remember names etc if I have a face to match the names to. I will generally have my camera on and will continue to keep it on even if I am the only one.

19.1.2 Why not?

There are lots of reasons why not. Most obviously, if you don't have access to a camera. But you may also be in an environment which you prefer others not to see, you may have anxiety around the issue, or your connection might be too slow. There are many other perfectly reasonable reasons for you not to put your camera on and you should not feel pressured to do this. If you simply say "Sorry, I can't turn my camera on today" then I won't ask any further and I will never explicitly ask you to turn it on.

19.1.3 Being appropriate

You should already be treating online meetings like physical ones e.g. turning up on time, being prepared, listening, engaging etc. Similarly, if people can see you then you should ensure you are wearing appropriate clothes (wearing clothes is the absolute minimum here!) and in an appropriate place (the bathroom is probably not appropriate) as you would for a physical meeting.

19.1.4 Respecting others

If other people have decided to turn their cameras on then I ask that you show them respect by not recording anything without explicit permission. I won't touch on the legality of this as I believe that basic respect for each other should be enough to prevent any issues. You will take part in larger meetings where recording may be standard and in such cases this should be made explicit.

Thanks to Giles Reger and Sarah Clinch for the text above

Chapter 20

Reading your future

If you want to have your future read, you can read all about it right here. These books, journals, websites, magazines will help you to read your future. There is no need to gaze into any crystal balls! Get reading your future now, because reading is good for your mind, body and soul. Libraries give us power, see figure 20.1.

Figure 20.1: Libraries give you power, the power to read your future. These references are your digital library to search and browse. Panorama of the British Museum Reading Room by David Iliff on Wikimedia Commons w.wiki/3BEs

This chapter lists everything (and I mean *everything*) cited in this book which you might find overwhelming. For a more easily digestible versions see the signposts section of any chapter in this book.

Since you are reading the pdf version, all the references can be found in the bibliography section following this page.

Bibliography

Aaron, S. (2016). Programming as performance.

Agnew, H. (2016). Big four look beyond academics: Firms want to make offers based on potential, rather than personal circumstance. *Financial Times*.

Aicher, T. P., Barabási, D. L., Harris, B. D., Nadig, A., and Williams, K. L. (2017). Ten simple rules for getting the most out of a summer laboratory internship. *PLOS Computational Biology*, 13(8):e1005606.

Andreessen, M. (2011). Why software is eating the world. *Wall Street Journal*.

Ashcroft, R. (1997). The drugs don't work. In Potter, C., editor, *Urban Hymns*. The Verve, Hut Records. https://www.youtube.com/watch?v=ToQ0n3itoII.

Assay, M. (2020). Want an IT job? look outside the tech industry. *TechRepublic*.

Babineaux, R. (2013). *Fail Fast, Fail Often: How Losing Can Help You Win*. Tarcher.

Ball, C. (2020). What might the graduate labour market look like in 2021? *WONKHE*.

Barrass, R. (2002). *Scientists Must Write: A Guide to Better Writing for Scientists, Engineers and Students*. Routledge.

Bates, L. (2016). *Everyday Sexism: The Project that Inspired a Worldwide Movement*. A Thomas Dunne Book for St. Martin's Griffin.

Bavaro, J. and McDowell, G. L. (2021). *Cracking the PM Career: The Skills, Frameworks, and Practices to Become a Great Product Manager*. CareerCup.

Beaubouef, T. (2003). Why computer science students need language. *ACM SIGCSE Bulletin*, 35(4):51–54.

Bell, C. (2021). Am i selling my soul to work for my company? *thevectorimpact.com*.

Belshaw, D. (2019). 10 platforms for issuing open badges. *weareopen.coop*.

Black, J. (2017). *Where am I Going and Can I Have a Map? How to take control of your career plan - and make it happen*. Robinson.

Black, J. (2019). How to write a top-notch CV. *Financial Times*.

Black, J. (2021). Dear jonathan: Careers column. *The Financial Times*.

Bolles, R. N. (2019). *What Color Is Your Parachute? A Practical Manual for job-hunters and career-changingers*. Random House USA Inc.

Borrett, A. (2019). Is a first class degree really that important? many graduate recruiters are weighing up other skills alongside academic achievement. *Financial Times*.

Bourne, P. E. (2005). Ten simple rules for getting published. *PLOS Computational Biology*, 1(5):e57.

Bourne, P. E., Lewitter, F., Markel, S., and Papin, J. A. (2018). One thousand simple rules. *PLOS Computational Biology*, 14(12):e1006670.

Box, H. and Mocine-McQueen, J. (2019). *How Your Story Sets You Free*. Chronicle Books.

Brewton, S. (2014). Squire bill widener vs. theodore roosevelt.

Britton, B. (2019). No sexuality please, we're scientists.

Brown, G. (2021). *How to get your PhD: A Handbook for the Journey*. Oxford University Press.

Brown, N. and Wilson, G. (2018). Ten quick tips for teaching programming. *PLOS Computational Biology*, 14(4):e1006023.

Caroll, L. (1865). *Alice's Adventures in Wonderland*. Macmillan.

Charette, R. N. (2005). Why software fails. *IEEE Spectrum*, 42(9):42–49.

Cheary, M. (2021). Hobbies and interests: Should I include them in my CV? *reed.co.uk*.

Chen, C. (2020). Online learning is growing faster than ever — we compared 4 of the top platforms to help you decide which one makes sense for you. *businessinsider.com*.

Coughlan, S. (2019a). 'grade inflation' means 80% more top degree grades. *BBC News*.

Coughlan, S. (2019b). The symbolic target of 50% at university reached. *BBC News*.

Cutts, Q., Cutts, E., Draper, S., O'Donnell, P., and Saffrey, P. (2010). Manipulating mindset to positively influence introductory programming performance. In *Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE '10*. Association for Computing Machinery.

Dale, S. (2018). #NotJustLondon — what's that then? *21st Century Mindset*.

Damore, J. (2017). Google's ideological echo chamber.

David, J. (2017). How long should your CV be? *cv-library.co.uk*.

Davidova, P. (2021). My worst software engineering interview fails: Failing my facebook and google interviews.

Davis, V. W. (2016). Error reflection: Embracing growth mindset in the general music classroom. *General Music Today*, 30(2):11–17.

Desai, R. (2016). Generalized anxiety disorder (gad) - causes, symptoms and treatment. *osmosis.org*.

Dweck, C. (2014). The power of not yet. *TEDxNorrköping*.

Dweck, C. (2017). *Mindset: Changing The Way You think To Fulfil Your Potential*. Robinson.

Easton, F. (1997). Educating the whole child, "head, heart, and hands": Learning from the waldorf experience. *Theory Into Practice*, 36(2):87–94.

Eddo-Lodge, R. (2017). *Why I'm No Longer Talking to White People About Race*. Bloomsbury Publishing.

Evans, J. (2018). Help! i have a manger! how to understand your manager's job and do awesome work together. *Wizard Zines*.

Evans, J. (2021). Http: Learn your browser's language. *Wizard Zines*.

Ferrari, V., Marin-Jimenez, M., and Zisserman, A. (2008). Progressive search space reduction for human pose estimation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Fincher, S. and Finlay, J. (2016). *Computing Graduate Employability: Sharing Practice*. Council of Professors and Heads of Computing CPHC.

Fitzpatrick, B. and Collins-Sussman, B. (2009). Google i/o: The myth of the genius programmer. *Google Developers*.

Fogarty, T. (2015). Hackathons are for beginners. *medium.com*.

Forever, B. (2019). Ten reasons why books are our best friends. *The Times of India*.

Foundation, M. (2019). Open badges v2.0: Ims final release. *IMS global*.

Friedman, S. and Laurison, D. (2020). *The Class Ceiling: Why it Pays to be Privileged*. Policy Press.

Fry, S. (2018). *Heroes: The myths of the Ancient Greek heroes retold*. Penguin Random House UK.

Garner, R. (2015). Ey: Firm says it will not longer consider degrees or a-level results when assessing employees. *independent.co.uk*.

Garone, E. (2014). To print or not to print — a cv, that is. *bbc.com worklife*.

Gill, E. and Statham, N. (2021). Head who promised stability for 'forgotten school' with nine leaders in 10 years is leaving. *Manchester Evening News*.

Goble, C. (2007). The seven deadly sins of bioinformatics. In *Bioinformatics Open Source Conference (BOSC) Special Interest Group (SIG) at the 15th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB)*, Vienna, Austria. Slideshare.

Goble, C. and Wroe, C. (2004). The montagues and the capulets. *Comparative and Functional Genomics*, 5(8):623–632.

Goldman, P. and Schlesinger, Y. (2018). With great code comes great responsibility: Announcing the responsible computer science challenge. *medium.com*.

Googler, A. (2019). Google technical writing courses: Every engineer is also a writer.

Googler, A. (2021). Google summer of code student guide. *Github*.

Graham, P. (2005). Good and bad procrastination. *paulgraham.com*.

Granovetter, M. S. (1973). The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380.

Green, F. and Kynaston, D. (2019). *Engines of Privilege: Britain's Private School Problem*. Bloomsbury Publishing.

Grove, M. (2018). Graduate salaries in the UK. *Luminate*.

Hacker, A. (2021). International Collegiate Programming Contest (icpc). *icpc.global*.

Haig, M. (2016). *Reasons to Stay Alive*. Canongate Books Ltd.

Haig, M. (2018). Top 5 tips for good mental health in a social media age. *Waterstones*.

Haig, M. (2019). *Notes On A Nervous Planet*. Canongate Books Ltd.

Hamedy, R. (2019). How to build a reputable StackOverflow profile. *medium.com*.

Hinds, D. (2017). Jobseekers being targeted by scammers. *BBC News*.

Hitchens, C. (1997). Everyone has a book inside them. *Cable-Satellite Public Affairs Network (C-SPAN)*.

Hoffman, R. (2020). Customize your public profile URL. *linkedin.com*.

Hull, D. (2020). Getting started with latex: A laboratory manual. *latex4year1.netlify.app*.

Hull, D. (2021). How long should your cv be: One page resume or two page CV? *Coding Your Future*.

Investor, A. (2019). Let's make work better. *investorsinpeople.com*.

Investor, A. (2020). How to bring yourself to work and allow others to do the same. *investorsinpeople.com*.

Isley, O., Isley, R., and Isley, R. (1988). Shake your thang. In *A Salt with a Deadly Pepa*. Salt-N-Pepa, London Records.

Johnson, H. (2020). The 'forgotten school' with nine head teachers in 10 years - and the man with a plan to turn it around. *Manchester Evening News*.

Johnstone, M. (2012). I had a black dog, his name was depression. *World Health Organization*.

Kelion, L. (2018). Google's job hunting service comes to UK. *BBC News*.

Kellett, T., Taylor-Firth, R., and Boyle, R.-A. (1997). You're not alone. In Kellett, T. and Taylor-Firth, R., editors, *Extra virgin*. Olive, RCA Records.

Knuth, D. E. (1984). Literate programming. *The Computer Journal*, 27(2):97–111.

Kottke, J. (2018). Alan turing was an excellent runner. *kottke.org*.

Lahey, J. (2016). *The Gift of Failure: How the Best Parents Learn to Let Go So Their Children Can Succeed*. Harper.

Lane, R. and Wood, R. (1973). Ooh la la. In *Ooh La La*. The Faces, Warner Bros.

Lanier, J. (2018). *Ten Arguments For Deleting Your Social Media Accounts Right Now*. Bodley Head.

Leendertz, L. (2006). *The half-hour allotment: Royal Horticultural Society*. Frances Lincoln, London.

Lennon, J. and McCartney, P. (1967). A day in the life. In Martin, G., editor, *Sgt. Pepper's Lonely Hearts Club Band*. The Beatles, Parlophone. https://www.youtube.com/watch?v=usNsCeOV4GM.

Leslie, I. (2020). Why your 'weak-tie' friendships may mean more than you think. *bbc.com worklife*.

Lewis, P. (2017). 'i see things differently': James damore on his autism and the google memo. *The Guardian*.

Louise (2019). Why i regret doing an unpaid internship. *Rate My Placement*.

Louise (2021). The highest paid internships and placements in the uk. *Rate My Placement*.

Malan, D. J. (2010). Reinventing CS50. In *Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE '10*. Assoication for Computing Machinery.

Malan, D. J. (2017). CS50 at scale. *YouTube.com*.

Malan, D. J. (2020). Teaching from home via zoom. *medium.com*.

Malan, D. J. (2021). Toward an ungraded CS50. In *Proceedings of the 52nd ACM technical symposium on Computer science education - SIGCSE '21*. Association for Computing Machinery.

Malham, G. (2002). Ey: Firm says it will not longer consider degrees or a-level results when assessing employees. *bbc.co.uk*.

Mann, C. C. (2002). Why software is so bad: For years we've tolerated buggy, bloated, badly organized computer programs. but soon, we'll innovate, litigate and regulate them into reliability. *MIT Technology Review*.

Markow, W., Coutinho, J., and Bundy, A. (2019). Beyond tech: The rising demand for IT skills in non-tech industries. *Burning Glass*.

Martin, R. C. (2011). *The Clean Coder: A Code of Conduct for Professional Programmers*. Prentice Hall.

McDowell, G. L. (2011). Ex-googler reveals strategies to land interview with google. *cnbc.com*.

McDowell, G. L. (2014). *Cracking the Tech Career*. John Wiley & Sons Inc.

McDowell, G. L. (2015). *Cracking the Coding Interview: 189 Programming Questions and Solutions*. CareerCup, 6 edition.

McDowell, G. L. and Bavaro, J. (2013). *Cracking the PM Interview: How to Land a Product Manager Job in Technology*. CareerCup.

Meisler, B. (2012). The real reason silicon valley coders write bad software. *The Atlantic*.

Milliken, S. (2019). *From Learner to Earner: A recruitment insider's guide for students wanting to achieve graduate job success*. Rethink Press.

Newton, I. (1675). Letter to robert hooke. *Historical Society of Pennsylvania*.

Nye, C. (2021). Being proud of going to state school. *BBC News*.

Octocat (2020). Mastering issues. *github.com*.

Ong, J. and Lopez, L. (2019). Create your resume for google: Tips and advice. *Google*.

O'Toole, G. (2012). If I Had More Time, I Would Have Written a Shorter Letter. *quoteinvestigator.com*.

Parker, P. (1962). With great power comes great responsibility. *wikipedia.org*.

Peroni, S. (2021). *Computational Thinking and Programming*. github, Bologna.

Petroski, H. (1992). *To Engineer Is Human: The Role of Failure in Successful Design*. Vintage.

Poundstone, W. (2013). *Are You Smart Enough to Work at Google?* Oneworld Publications.

Price, H. (2019). *What is sexual harassment at work? How to tell when lines are crossed in the workplace*. BBC.

Raymond, E. S. (1999). *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly Media.

Richards, S. (2019a). *Content Design London: Readability Guidelines*, chapter Why people do not read online. readabilityguidelines.co.uk.

Richards, S. (2019b). *Content Design London: Readability Guidelines*, chapter Treat links with respect. readabilityguidelines.co.uk.

Ridenhour, C., Sadler, E., Boxley, H., and Boxley, K. (1989). Fight the power. In Lee, S., editor, *Do the Right Thing*. Public Enemy, The Bomb Squad.

Roberts, R. J. (2015). Ten simple rules to win a nobel prize. *PLOS Computational Biology*, 11(4):e1004084.

Robertson, R. (2020). Fixing a bug on my open source project: From start to finish. *dev.to*.

Ross, J. (2021). The biggest companies in the world in 2021. *Visual Capitalist*.

Ryder, S. (1988). Wrote for luck. In Hannett, M., editor, *W.F.L. Think About the Future Mix*. Happy Mondays, Factory Records.

Ryder, S. (2019). *Wrote for Luck: Selected Lyrics*. Faber and Faber.

Saini, A. (2018). *Inferior: How Science Got Women Wrong and the New Research That's Rewriting the Story*. Harper Collins.

Saini, A. (2019). *Superior: The Return of Race Science*. Harper Collins.

Santos, L. (2020). Five favorite coping tips for world mental health day. *The Happiness Lab*.

Santos, L. (2021). The science of well-being at yale university.

Schuh, P. (2004). *Integrating Agile Development In The Real World*. Charles River Media, Inc., Rockland, MA, USA, 1st edition.

Scott, R. (1974). Boy on a bike. *bfi.org.uk*.

Sedgwick, R. (2019). Should computer science be required? *Inside Higher Ed*.

Shaha, A. (2014). Wave machine demonstration.

Shakespeare, W. (1597). *Romeo and Juliet*.

Shapiro, B. R., Lovegall, E., Meng, A., Borenstein, J., and Zegura, E. (2021). Using role-play to scale the integration of ethics across the computer science curriculum. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. ACM.

Shields, J. (2019). What happens to resume tables and columns in an ats? see for yourself. *jobscan.co*.

Shimer, D. (2018). Yale's most popular class ever: Happiness. *The New York Times Magazine*.

Sissay, L. (2015). Inspire and be inspired. *YouTube.com*.

Smith, M. E., Smith, B., and Rogers, S. (1987). Hit the north. In *The Frenz Experiment*. The Fall, Beggars Banquet.

Somers, J. (2011). How i failed, failed, and finally succeeded at learning how to code. *The Atlantic*.

Spinellis, D. (2021). Why computing students should contribute to open source software projects. *Communications of the ACM*, 64(7):36–38.

Spärck-Jones, K. and Runciman, B. (2007). Computing is too important to be left to men. *ITNOW*, 49(4):18–20.

Stanford-Clark, A. (2019). Innovation begins at home. *Alliance Manchester Business School*.

Swinton, J. (2019). *Alan Turing's Manchester*. Infang Publishing.

Sylor-Miller, K. and Evans, J. (2021). Oh shit, git! recipes for gitting out of a git mess. *Wizard Zines*.

Tickle, L. (2015). Where are all the young school governors? under 30s aren't signing up to governance in the same way they want to be environmental campaigners or trek for charity. *The Guardian*.

Tupper, H. and Ellis, S. (2020). *The Squiggly Career: Ditch the Ladder, Embrace Opportunity and Carve Your Own Path Through the Squiggly World of Work*. Portfolio Penguin.

Tupper, H. and Ellis, S. (2021). Squiggly careers and the end of the traditional career path. *ted.com*.

Tyldum, M. (2014). The imitation game.

UK, G. (2020a). Discrimination: your rights and protected characteristics. *www.gov.uk*.

UK, G. (2020b). Handing in your notice. *www.gov.uk*.

UK, G. (2021a). Disciplinary procedures and action against you at work. *www.gov.uk*.

UK, G. (2021b). Dismissal: your rights. *www.gov.uk*.

UK, G. (2021c). Raise a grievance at work. *www.gov.uk*.

Verkaik, R. (2021). A bullingdon in reverse: how working-class student club is taking on elitism. *The Guardian*.

Walker, D. and Walker, R. (1992). *Energy, Plants and Man*. Oxygraphics Ltd;, 2 edition.

Walker, M. (2018). *Why We Sleep: The New Science of Sleep and Dreams*. Penguin Books.

Ware, B. (2011). *Top Five Regrets of the Dying*. Hay House UK Ltd.

Wax, R. (2014). *Sane New World*. Hodder.

Way, M. (2017). What I cannot create, I do not understand. *Journal of Cell Science*, 130(18):2941–2942.

Wickham, H. and Grolemund, G. (2017). *R for Data Science*. O'Reilly UK Ltd.

Wodiany, I. (2018). Interns in technology salaries 2018 report. *github.com*.

Woodruff, W. (2003). *Beyond Nab End*. Abacus.

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2 edition.

Xie, Y. (2017). *bookdown: Authoring Books and Technical Documents with R Markdown*. Chapman and Hall/CRC, Boca Raton, Florida.

Xie, Y., Dervieux, C., and Riederer, E. (2020). *R Markdown Cookbook*. Chapman and Hall/CRC, Boca Raton, Florida.