Don't Panic

A CODEBREAKER'S GUIDE TO GETTING UNSTUCK

Crackerjack

ABSTRACT

The Hitchhiker's Guide to the Galaxy is, among other things, a book designed to help its reader navigate a universe that is confusing, chaotic, and occasionally hostile. This guide has far more modest ambitions: to help you remain at least somewhat functional while staring at a block of ciphertext that appears to have been designed specifically to ruin your evening. If there is one thing worth remembering throughout the rest of this guide, it is the same advice printed in large, friendly letters on the cover of the Hitchhiker's Guide: DON'T PANIC.

What this guide is (and isn't)

This is not a guide on how to break codes.

There are plenty of places you can go if what you want is the mechanics: how to run frequency analysis, how to implement a hill-climb attack, or which button to press next in Python. If that's what you're looking for, this guide will probably feel a bit frustrating.

This is a guide on how to be a codebreaker. Or at least, a guide I wish I had when I first started.

It's about what to do before you know what cipher you're even looking at. How to think when nothing makes sense yet, how to recognise progress when there is no readable plaintext in sight, and how to stay functional and vaguely sane when you're stuck, because at some point, you absolutely will be.

Codebreaking, especially with something like the National Cipher Challenge, is very rarely a neat sequence of steps. More often than not, it's long periods of uncertainty sprinkled with small insights that only feel obvious in hindsight.

Most of the difficulty in codebreaking comes from uncertainty like not knowing where to start, or whether you're making progress, and not knowing whether the idea you're attached to is actually helping.

In short, if you've ever stared at a block of ciphertext and thought to yourself, *'I have no idea what I'm doing'*, then you're in the right place.

Contents

Being Stuck

> *Not all those who wander are lost*
>
> *J.R.R. TOLKIEN*

Transition from Familiar Problems

Most problems you have probably met before come with a kind of invisible scaffolding.

In mathematics, you are given an equation and expected to apply a known method. In physics, you identify the model, plug in the numbers, and hope the units work out in your favour. Even when a problem is difficult, there is usually a sense of what kind of solution it requires. You may not immediately know how to proceed, but you recognise the structure of the task. You know what tools are available, what progress might look like, and roughly how far you are from an answer.

Why Codebreaking Feels Different

Codebreaking does not usually begin with that same clarity.

When you first encounter a difficult cipher, nobody hands you a tidy label saying this is a substitution cipher or this one is Vigenère but with a twist[1]. There is no neat little signpost telling you whether the symbols represent letters, groups of letters, sounds, numbers, or something much weirder.

Instead of being guided towards a method, you are presented with something that has been deliberately designed to conceal its structure. You are no longer just trying to solve something difficult. You are trying to work out what the thing even is before you can begin solving it properly. Which is, unfortunately, much less glamorous than it sounds and often involves staring at repeated symbols until your brain starts assigning emotional significance to bigrams.

Without a clear framework, it becomes very easy to panic slightly and start

[1] I must say this isn't strictly true. Depending on how generous Harry and Jodie are feeling, they might let a small hint or two slip before the first deadline!

throwing methods at the problem in quick succession. You try one thing. It doesn't work. You try another. Then another. And after a while, you begin to get the horrible creeping sense that the cipher is not impossible, exactly, but that it has somehow become personal.

On Feeling `Stuck'

In most areas, being stuck has a fairly obvious meaning: you have reached a point where you do not know how to continue.

In codebreaking however, I think the meaning is more subtle. What feels like being stuck is often not a failure to make progress, but a failure to recognise the problem. The difficulty may lie not in applying a method, but in identifying which methods are even relevant.

If you expect yourself to recognise the correct approach immediately, then any delay starts feels like a mistake. Being stuck, in this sense, usually means that you do not yet know what kind of problem you are solving.

The Point of the Challenge

At first, it is very tempting to think of codebreaking as an exercise in applying techniques efficiently. But if the goal were simply to run frequency analysis, implement known algorithms, or write a program that could quickly decode a given cipher, then much of the difficulty would disappear. Once the method is identified, the rest becomes largely mechanical.

Competitions like the Cipher Challenge go beyond this. The difficulty (and the point) lies in working out what to do in the first place. It lies in recognising structure, forming hypotheses, and deciding which ideas are worth pursuing. The challenge is not just to execute a method, but to discover which method, if any, applies.

In this sense, codebreaking is less about computation and more about interpretation. It is an exercise in reasoning under uncertainty, and learning how to think when the problem has not yet revealed itself.

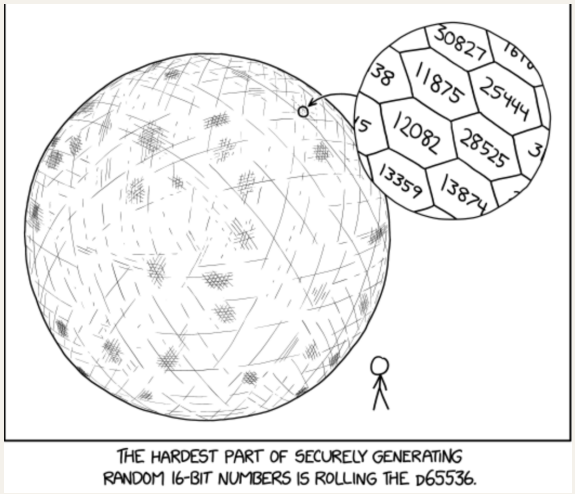

Figure 1.1: 2626: d65536

Redefining Progress

> *Success is the product of daily habits-not once-in-a-lifetime transformations*
>
> *JAMES CLEAR*

Marginal Gains

In elite sport, there is an idea referred to as marginal gains[2]. The principle is simple: instead of looking for one large breakthrough, you focus on making many small improvements. Each small change on its own might seem insignificant, but together, they compound.

One of the most well-known examples of this comes from British cycling. In the years leading up to their Olympic dominance, the team became almost absurdly committed to improving everything by tiny amounts. They monitored training, nutrition, recovery, sleep, equipment, even the things that seemed too small to be worth bothering with. Now what stood out was not the scale of any single improvement, but the accumulation of many small ones.

What Progress Usually Looks Like

In most settings, progress is often visible. Think about solving an equation step by step, with each step bringing you closer to a solution. When answering an exam question, you move through a sequence of known methods, gradually building towards a final answer. Even when the work is difficult, there is usually a clear sense of direction. You can see yourself getting closer.

This familiarity shapes how you expect progress to feel, and when those signals are absent, it is natural to assume that nothing is happening.

At the beginning of a cipher, there is often no clear indication that you are getting anywhere. You may spend several minutes observing, testing ideas, and ruling things out, without producing anything that resembles a solution. Compared

[2] I first read about this in James Clear's Atomic Habits, an excerpt from which can be found here

to more familiar problems, this can feel like a lack of progress.

Why It Doesn't Feel Like Progress

If this process is genuinely productive, why does it often feel like nothing is happening?

The difficulty is that small improvements do not feel like progress. There is no intermediate output to confirm that you are moving in the right direction. And without that feedback, it is easy to interpret the absence of a solution as the absence of progress.

If progress is defined only has having solved the cipher, then everything before that point starts to feel like failure. Which is a terrible definition, not least because it means you can spend hours doing genuinely intelligent work and still come away feeling like you have achieved nothing. This is the same mistake as ignoring marginal gains in sport because each individual improvement seems too small to matter.

And I say that not from a position of great wisdom, but as someone who has absolutely stared at a ciphertext for ages, made several useful observations, and then convinced myself I was getting nowhere because none of them had yet turned into plaintext.

A lot of early insights only become meaningful in hindsight. A pattern that looks irrelevant at first may later turn out to be the thing everything hinged on. But in the moment, it rarely feels like revelation.

Narrowing the Problem

At the start of a cipher, there are usually many possible interpretations. The text could represent letters, numbers, symbols, or groups. It could be the result of a simple substitution, a transposition, or one of those ciphers that makes you briefly question whether language itself was a mistake. Without additional information, these possibilities can feel equally plausible.

When you notice that certain patterns repeat in a consistent way, you begin to rule out some explanations. When the number of distinct symbols is too high or too low for a particular method, that method becomes less likely. When a hypothesis fails to produce consistent results, it can be set aside.

Over time, this process narrows the range of possibilities. You are not moving

directly towards a solution, but making the problem smaller and more defined. Each step removes options until what remains is manageable enough to analyse more directly.

Looking at it this way, codebreaking is less about finding the answer and more about reducing what the answer could be.

A More Useful Definition

Instead of asking: *Have I solved the problem?* Try asking: *Do I understand the problem better than I did before?*

If the answer is yes even in the slightest, then progress has been made. You may not yet have a solution, or may not even be close to one. But if you have ruled out possibilities, identified structure, or clarified your assumptions, then you have moved forward.

This shift in perspective is important because it changes how you approach the process. It allows you to value the marginal gains and the incremental steps that lead to understanding rather than focusing only on the final result.

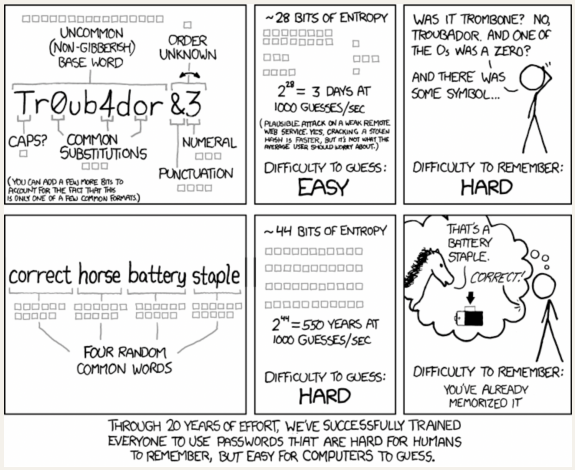

Figure 2.2: 936: Password Strength

The Starting Line

> *Slow is smooth, and smooth is fast*
>
> *MILITARY PROVERB*

Speed vs Accuracy

As much as this challenge rewards speed, it also rewards accuracy. In the later challenges especially, I often found that acting too quickly meant committing to an idea before the problem had been properly understood. When an initial idea fails, it is easy to move on to another, and then another again, without ever fully engaging with the structure of the cipher itself.

A more effective approach might be to first pause. After all, there is a reason races begin with *On Your Marks... Get Set... Go!* and not just someone yelling *GO!* at a group of already stressed people. The pause is not separate from the start, it's part of the start. It gives you a second to settle, to look ahead, to position yourself, and to begin with some sense of direction rather than pure adrenaline.

Codebreaking is no different. So before attempting to solve anything, it helps to take a step back and look carefully at what is actually in front of you. At this stage, your goal is not to crack the cipher, but to understand what you're dealing with.

What Is Actually There

When you look at the ciphertext as it is, ask questions about what you can see. Think of going through steps of a mental flowchart to problem solving

Some questions you might want to consider could be:

- What characters appear?
- How many distinct characters are there?
- Do certain patterns in the text repeat?

- Are there deliberate spaces in the text?

These questions do not require any specialised knowledge as they are less about applying techniques, and more about paying attention to the information that is already present.

Defining the Unit

One of the most important questions you can ask yourself early on is: What is one unit of this ciphertext?

It might be natural to assume that each character corresponds to a letter. But ciphers are under no obligation to be that cooperative. A single symbol might represent a letter, or a pair of letters, or a number, or a code. Maybe the groupings of such symbols or characters only make sense when they appear in a certain pattern.

For example, in Morse code, individual dots and dashes are not letters. Only when they are grouped in the correct way do they form meaningful units, so treating each symbol independently would not lead to a solution. And if you get the unit wrong, you can spend a very long time being impressively wrong in a highly committed way.

Recognising the correct unit is not always immediate. But simply knowing that this is a question worth asking can save you from making assumptions too early, which is one of the more common ways to accidentally make your own life harder.

Asking Better Questions

If there's anything I've learned, it's that quality of your thinking is determined by the quality of the questions you ask.

At the beginning of a cipher, it is easy to focus on questions that are too specific about the type of cipher. But those questions assume that you already understand the problem well enough to classify it.

More useful questions are broader and more structural such as:

- What patterns are present?
- What is and isn't consistent?
- What assumptions am I making?

- What would have to be true for this to work?

Notice how the questions above don't have quick answers, but they do guide your attention in a more productive direction in the sense that they help you build a clearer picture of the problem before committing to a particular approach.

The purpose of the starting line is not to immediately produce an answer, but to begin understanding the problem in a structured way. By observing carefully, questioning assumptions, and resisting the urge to act too quickly, you create the conditions for meaningful progress later on.

Figure 3.3: 1323: Protocol

Abandoning Ideas

> *It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts.*
>
> *SIR ARTHUR CONAN DOYLE*

The Difficulty of Letting Go

There is a well-known tendency in business decision making called the sunk cost fallacy. Once time, effort or resources have been invested into something, it becomes much harder to away even when continuing is no longer the best option.

A classic example is watching a film you are not enjoying. Half an hour in, you realise it is not very good. The plot is weak, the dialogue is unbearable, and at least one character appears to have been written by someone who has never met another human being. But instead of turning it off, you keep watching, because you have already spent thirty minutes on it and the whole thing is only ninety.

The same thing could happen in codebreaking. After spending a long time developing an idea, testing it and trying to make it work, it becomes difficult to let go. The time you have invested starts to feel like a reason to continue even when the evidence might no longer support it.

Every year that I have competed in the challenge, this has been one of the most difficult things to deal with. Not because the problem was particularly complex, but because I was then faced with a choice: continue with an idea that feels promising, or abandon it and return to uncertainty.

Why We Hold On

It's very natural to become attached to an idea once you have invested enough of yourself into it. And by 'yourself', I don't just mean your time. I also mean your hope, concentration, and sense that this has all been leading somewhere.

The more work you have done, the more difficult it becomes to let go. Each small success reinforces the belief that you're close to a solution, even when the overall picture does not quite fit. An idea that is 'almost perfect' can be quite dangerous because it creates the illusion that you are only one step away from solving the problem.

There is also a tendency to adjust the problem to fit the idea. If something does not align perfectly, it can be tempting to treat it as an exception, or to assume that it will make sense later. But over time, this can lead to increasingly complicated explanations that are difficult to justify, but equally difficult to abandon.

When an Idea just isn't Working

Letting go of an idea becomes easier when you know what to look for. This is genuinely difficult, but gets easier with time and practice and patience.

Some common signs that an idea may not be useful include:

- The explanation only works for part of the ciphertext
- You need to summon exceptions to make it fit
- Small inconsistencies are being ignored or explained away
- The reasoning becomes more complicated over time, rather than simpler

A useful principle here is that good explanations tend to simplify the problem. So if an idea requires increasing effort to maintain, it is often a sign that it is not the right one.

Returning to the Problem

After abandoning an idea, it is important to return to the problem itself, rather than immediately searching for a replacement.

Go back to the ciphertext. Look again at what is actually there. Revisit the observations that still hold. Ask yourself what remains true even now that your favourite idea has collapsed dramatically in front of you.

By returning to the structure of the problem, you allow new ideas to emerge from what you can observe, rather than from what you hope might work.

Again, abandoning an idea does not mean starting again from nothing. Even if a particular approach does not lead to a solution, the process of exploring it may still have produced useful information. You may have ruled out certain structures, identified patterns that remain consistent, or clarified which assumptions do not hold.

These point is that these insights remain valuable. The goal here is not to avoid being wrong. That would be both impossible and extremely boring. The goal is to be wrong in ways that sharpen your understanding, to fail usefully, to let the bad ideas die without dragging you down with them.

Figure 4.4: 1820: Security Advice

Using Context

> *Amateurs hack systems, professionals hack people*
>
> BRUCE SCHNEIER

Codebreaking with Cribs

During WW2, much of the Allied effort[3] in breaking encrypted messages relied on cribs, which is a just a fancy word for fragments of plaintext that were expected to appear in the decoded messages.

The important thing is that a crib is not just a random guess. It is an educated guess. Wartime messages were often predictable in ways that had nothing to do with the cipher itself. They followed routines, contained expected phrases, weather reports tended to mention weather, military communications tended to mention military things. Human beings, even under conditions of extreme secrecy, are still frustratingly prone to habit.

By aligning these expected fragments with intercepted ciphertext, codebreakers were able to recover information about the encryption process. This essentially provided a way into systems that would have otherwise been extremely difficult to analyse directly.

The point being, these breakthroughs did not come purely from analysing symbols in isolation.

Choosing Cribs

One of the most useful habits you can build in the Cipher Challenge is learning to choose cribs intelligently. That means not just guessing words you want to be there, but thinking carefully about what is reasonable to expect.

And where do those expectations come from? Context.

The ciphers are not dropped into a vacuum.

[3] This became extremely important when it came to breaking the Enigma

Fortunately for National Cipher Challenge, the ciphers are not dropped into a vacuum. Every single piece of information is intentional [4]. They are built around stories, themes, recurring characters, and intentional framing. The plaintext is very rarely arbitrary, which means that before you even begin solving the cipher, you often already know more than you think you do. The Case Files section will prove especially useful here, but other examples include:

- Titles may hint at key(s) or type of cipher used
- Recurring named figures across the challenges
- Specific dates or years hinting towards historic events
- Certain themes being reflected in the structure of the ciphertext. Eg: telegraph machine in 2024(10B) and playing cards in 2025(10B)

One of the most useful habits you can develop is learning to think like the setter. Code setters are still people, and people have tendencies. They reach for certain kinds of wordplay, references, and structures that feel satisfying to build. If you were designing this challenge, what kinds of keys, themes, or hidden signals might you want someone to eventually notice? Everything is intentional, and everything is given to you.

Using Context Cautiously

While context is incredibly powerful, it must be used with care. The same instinct that helps you identify a likely crib can also lead you to see what you want to see rather than what is actually there. An educated guess that fits the theme perfectly might still be a red herring, and the more elegant it feels, the more dangerous it becomes. So apply the same scepticism to context-driven ideas as you would to any other problem. Remember the sunk cost fallacy, and be willing to challenge your own assumptions.

Context also plays a useful role when you've acquired a solution. For instance:

- Does the length of your solution match the ciphertext (only if a one-to-one mapping is expected, but you get the idea)
- Does the solution make sense within the theme of the challenge
- Does the solution have any missing/incomplete sections?

[4] And even if something isn't, that's a feature, not a bug

Remember that spaces don't matter in your solution as all spaces and punctuation are removed from submissions when they are checked.

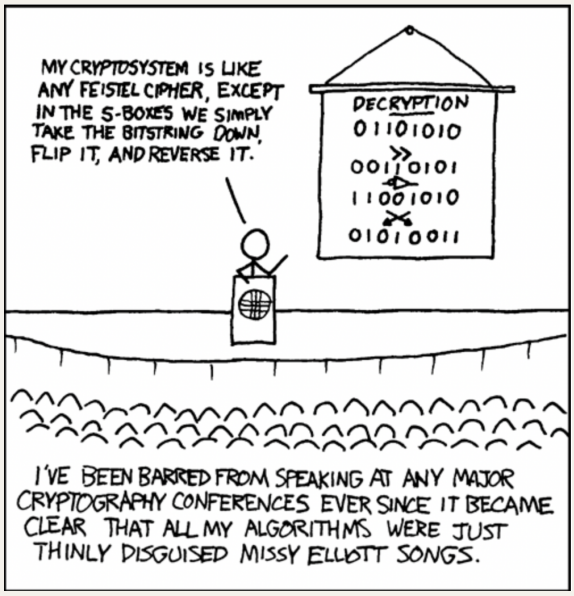

Figure 5.5: 153: Cryptography

Teamwork

> *If I have seen further it is by standing*
> *on the shoulders of giants*
>
> *SIR ISAAC NEWTON*

Methods of Competing

One of the fun things about the Cipher Challenge is that you can do it entirely on your own.

For a while, that can feel quite nice. Working alone means you can move at your own pace. You can follow your own ideas without needing to explain them. You can disappear down whatever rabbit hole you like without anyone asking you why you are suddenly convinced that the key length is 17 based on what is, at best, a fragile emotional argument. But sooner or later, you will get stuck. And when you are working alone, there is no one there to pull you out of it. No one to question your assumptions. No one to tell you, gently or otherwise, that your current idea might be… not your best.

That is where teams become interesting. A good team does not just divide the work. It changes the way you think. It gives you multiple perspectives, different instincts, different ways of approaching the same problem. It gives you people to bounce ideas off, which is often the difference between being stuck for hours and being unstuck in five minutes because someone else noticed the one thing you didn't.

That said, not every team works like that. Sometimes, when the challenges get harder, people drift away. Life gets busy. Motivation dips. The problem starts to feel too big. And suddenly, the group that began full of enthusiasm becomes smaller, and eventually, sometimes, just you.

I've experienced both sides of this. From being part of a really engaged team, and also finding myself as the last person still actively working on a challenge. And in a slightly unexpected way, that showed me the kind of teammate I wanted to be. Someone who stays. Someone who contributes. Someone who doesn't disappear

when things stop being easy. Having joined a new school for sixth form, I had the fun change of forming a new team from scratch. Working together on the cipher challenge later turned out to be one of the easiest ways to get to know people. It's genuinely an amazing experience to work together to solve a problem that no one fully understands.

Learning from Each Other

One of the biggest advantages of working in a team is the opportunity to learn from one another.

For instance, when I first started competing in the NCC, I could barely code. But working alongside friends who shared this enthusiasm for problem-solving created an environment in which I saw my confidence in coding improve. At the start of my sixth form year, I had just over a year of coding experience and could identify most commonly used ciphers in the challenge, but I didn't really see myself as the 'coding person'[5]. Yet, through the process of contributing to our team's shared GitHub repository, I found myself working on annealing and modified Playfair ciphers. Whether it was optimising the hill climb parameters for various programs, or spotting a silly inequality error in the quadgram evaluation block, the shared teamwork made even the most daunting ciphers feel achievable.

At the same time, programming isn't the only skill that matters. Some people are amazing at coding and automating processes, whereas others are better at spotting patterns, recognising when something doesn't quite it, and knowing when to move onto another idea.

A good team brings all of these strengths together.

Shared Thinking

An idea that feels completely convincing in your own head can start to unravel the moment you attempt to put it into words. And while that can be mildly embarrassing, it is also extremely useful. On the other hand, you explain an idea, someone else listens, and then they take it in a slightly different direction. They notice something you didn't. They reinterpret it. They connect it to something else. And suddenly, what started as a half-formed thought becomes something

[5] The only Python programs I recall writing were an IOC calculator and a simple Vigenère cipher solver for a known key

much more solid.

There's also something to be said for the experience itself. The shared frustration, the moments where nothing makes sense, and the occasional breakthrough when something suddenly clicks. That shared experience is a big part of what makes this competition memorable.

Teamwork is not always smooth. People work and think at different speeds and sometimes ideas clash and disagreements happen. When things get difficult, it's easy for communication to break down.

It helps to remember that everyone is working towards the same common goal. So taking a step back, listening to each other and being willing to let go of your own initial ideas to intake new information can make a big difference.

Working Practically

Listed below are some things that helped our team work more efficiently, and I hope you might find something useful here:

- Sharing Progress - it can be useful to have something like a team Discord server or any group chat so that you can communicate and keep each other updated with progress
- Divide and Explore - different people can test different ideas at the same time which can speed up the process of ruling things out
- Shared resources - my team kept a shared GitHub repo where we stored all our scripts and solutions to most common ciphers. Having a shared codebase can be very useful in the long term and make later challenges easier to approach
- Team captain login - since only the 'Team Captain' account can submit solutions, it might be worth to make sure more than one person has access to it in case the team captain is unavailable

There's a lot to be said for solving a problem on your own. But there is also something genuinely special about solving one with other people, about being confused then figuring things out together. And in the end, those are often the parts you remember most. Not just the solution, but the people you got there with.

Figure 6.6: 1513: Code Quality

Tools and Resources

> *A fool with a tool is still a fool*
>
> *GRADY BOOCH*

Getting Started

A common misconception about the competition is that you need to be good at programming to take part. Spoiler: you don't.

For context, Year 10 was the first time I participated in the challenge, and I only wish I had discovered it sooner. The exhilaration of cracking 10A the night before the final deadline, armed with nothing but a pen, paper, and an Excel spreadsheet, remains one of my most memorable moments.

For a lot of the challenge, you can get pretty far without much coding at all, it'll be slow and a bit tedious, but absolutely doable. That said, just because we can solve everything by hand doesn't mean we should.

My first two of years doing the Cipher Challenge I barely knew how to code, so I relied on very simple tools like:

- reading from a text file
- finding and replacing characters
- String manipulation
- frequency analysis

You can also use a spreadsheet or any text editor to do this. But the above are just a few lines of Python each, and you can learn them as you go.

When dealing with any programming problem, the hard part is usually knowing what to do rather than how to do it. The 'how' comes with time and practice, but once you know what type of cipher/problem you're dealing with, things become more manageable.

For example, if you're working with a transposition cipher, you can abstract the problem by thinking of it as just picking a key length, splitting the ciphertext into rows of that length, and ordering the columns and seeing if it looks more or less like English, and making small tweaks in your row/column layout as you go along. You can do all of that using a spreadsheet without writing a single line of code. But once you understand the process of solving that cipher, you can look up how to implement something like a hill-climbing attack to automate the tweaking and speed up the decryption process.

I learned Python during GCSEs, but just the basics to get me a good grade. Having never done any programming before that, most of my coding experience came from writing solutions for the cipher challenge problems, which in turn helped me in A-Level and just improving thinking skills in general. There is a difference between learning programming in a classroom and learning because you have a genuinely annoying ciphertext in front of you and would quite like it to stop being annoying. While studying Mathematics at university now, I realised that most of the things I've learned is thanks to the experienced gained by taking part in this competition.

PAST CHALLENGES

One of the most useful things to do is to go back and look at past challenges. Conveniently for the competitors, there exists the unofficial archive of National Cipher Challenge which has problems dating back all the way to 2002.

Past challenges are helpful for two reasons. Firstly, they give you practice. Which is obvious, but important. Secondly, they help you build pattern recognition. You start to notice the kinds of ciphers that appear more often in later rounds, the kinds of twists the setters seem to enjoy introducing when they are feeling particularly mischievous.

When it came to tackling 10B, our team ended up going back through previous later round challenges and making a list of the kinds of ciphers that had appeared before. From there, we started writing helper scripts for the ones that came up more often, just so we could automate the more repetitive parts and spend more time actually thinking.

LEARNING WITH PROJECTS

I genuinely think one of the best ways to learn anything is to build things that interest you. Every now and then I start some new maths or coding project,

often with absolutely no guarantee that I will finish it, but I almost always end up learning a lot in the process. So pick something you enjoy and do it for fun, not just because you think you 'should'.

As much as I loved writing code for solving challenges, there were definitely times when it became stressful, and if you find that happening, it's usually a sign you should switch things up for a bit. Sometimes that means working on a different problem, and sometimes it means doing something completely different[6].

There are lots of excellent books and resources out there, and people much more experienced than me will happily recommend them if you post in the Forum. But personally, I have mostly learned by searching things as I needed them. I often find big books overwhelming when I do not yet know what is relevant. It is much easier to care about a concept once you have already encountered the problem that makes it useful.

It makes a massive impact being around people who share a similar goal to you than doing everything by yourself. When I first started programming, I never imagined I'd be able to solve the kinds of problems I can today. It can be really daunting at first, especially when you're looking at other people's code and it all feels incomprehensible, but you have to keep things in perspective and trust the learning process, it genuinely does add up over time.

Also remember, you don't need to do everything at once. It may feel too much, but just try and focus on implementing one concept, one solver, or learning one skill at a time. Everyone starts somewhere.

The Official Rules

There are also plenty of online tools that can solve ciphers in one click. But the competition has specific rules about what counts as fair use, so it's worth knowing them so you don't accidentally invest time in something you can't use.

> You may use spreadsheets and text editors to help you tackle the challenges and any software we provide on the Official BOSS Deciphering Tools page. You can also use any software that you write yourself. You should not use deciphering tools you find elsewhere

[6] At one point I picked up crocheting, and I vividly remember crocheting while debugging my A-Level code because I was getting so frustrated with a program that refused to run. Weirdly, that kind of break actually helped more than staring at the screen ever did. This sounds ridiculous, but then again, so does a lot of programming.

> on the web and doing so can result in disqualification.

In short, you're not allowed to import you way to victory, i.e. don't use existing tools and libraries designed for cipher cracking. But you can write your own programs and use libraries like numpy.

The challenge site also has many good resources under the BOSS Training Division to help you get started with codebreaking and programming.

If during the competition, you have more questions whether something is within the rules or not, always ask in the Forum.

The Forum

The Cipher Challenge Forum has to be one of the most valuable resources in the entire competition. As a reminder, below are the Forum rules reiterated:

> Some House Rules for the Forum
> You can post comments and questions about the challenges, and share ideas with others here on the forum.
> The Forum is staffed by Harry and Jodie together with members of ELF, the Elf League of Forum moderators. Remember that those you are talking to on the Forum are people too, even the elves. They have feelings, just like you, so keep it friendly and polite.
> We want to keep you all safe, so please do not post, or attempt to post anything personal. Here you can be anyone you want to be as long as you don't tell anyone who you are!
> Above all, please, please, please don't give hints, tips, or above all solutions to a live challenge without asking us first! This is an important rule and we don't want it compromised.
> If you want to post anything at all about the challenges, post it here, so that we can moderate it and everyone can read it. You might be tempted to use other channels like Discord, but that is a breach of security and not the way we do things here.
> Remember the three Ws: Be witty, be wise and be warm! Let's make the forum a great place to hang out.
> Harry

I have loved posting and contributing to the forum both as a competitor and in the past year as an alumnus. Some of my favourite moments now come from posting on it and seeing someone else discover the joy of problem solving.

Sometimes it's just a small hint, or someone asking a question that nudges another person in the right direction, other times it's reading about someone finally crack a cipher they've been stuck on for days.

Learning is much easier when you're not doing it alone. There's a very high chance that whatever problem you're stuck on has been hit by someone before[7] and you get have this space where you can not only communicate with like-minded people, but be part of this shared experience. And if you're lucky, you might even get a reply from Harry himself.

Use of Generative AI

WHAT AI CAN'T REPLACE

At the 2025 Prizegiving at Bletchley, guest speaker Rob Eastaway gave a brilliant talk about (counter-)intuition and how often our instincts can lead us down the incorrect solution. But one moment that stuck with me was when he mentioned how AI almost never responds with 'I don't know'. Try it yourself, ask your favourite LLM, "Why don't you ever say 'I don't know'?" and see what happens.

We hear a lot about AI at the moment, and how 'intelligent' it's becoming, but there are still things that it can't replace. In many ways, I believe that problem solving begins with the willingness to admit you don't yet know the answer, and the curiosity to keep exploring until you do.

AI can certainly generate answers, but it can't replace the experience of solving something meaningful together with other people. It can't replace the moment when someone on the forum posts a small hint that suddenly unlocks a problem you've been stuck on for hours. It can't replace the pride you feel when another student finally cracks a cipher they have been wrestling with for days, and it definitely can't replace the strange, lovely experience of building friendships around shared confusion and a mutual desire to bully ciphertext into becoming plaintext.

No conversation with an AI can replace the feeling of another human choosing to spend their time helping you through something difficult. And equally, no algorithm can replace the sheer joy of finally cracking a ciphertext after hours of effort. And no matter how sophisticated the tool becomes, that part can't be automated.

[7] funnily enough this is true in both programming and life…

What I've realised over the years is something simple: one of the things humans can do that AI cannot is share in the joy of someone else finally solving something difficult. Looking back, the Cipher Challenge is one of the places where I've seen that most clearly.

Figure 7.7: 1838: Machine Learning

The Long Game

> *It is not the mountain we conquer but ourselves*
>
> *SIR EDMUND HILLARY*

Challenge 10B

There is a particular kind of madness reserved for Challenge 10B. If this is your first year, you'll probably discover that soon enough. And if you've done this before, you already know the deal.

One of the most difficult things I find with 10B every year, apart from the cipher itself, is knowing when to stop and take a break. Or more accurately, knowing when to stop *for now*.

When you're deep in the middle of some bizarre and increasingly unhinged line of thought about letter frequencies or key lengths or whether Harry has personally decided to ruin your evening, taking a break feels almost impossible. It's so easy to fall into that trap of 'I'm so close… just a bit longer…' and then 5 hours later your eyes hurt, brain's fried, and you're resisting the urge to throw your laptop out the window.

It feels completely counterintuitive, some of the best insights I've had came when I was doing something completely unrelated like rowing or cooking.

And yet, even knowing that, I still find it difficult to step away when I am in the zone, at least in what feels like the zone but is sometimes actually just the early stages of cognitive decline. Over time, though, I have found a few things that help. None of them solve the cipher for you, but they do make it much easier to return to it as a somewhat functioning human being.

WRITE DOWN EVERYTHING YOU KNOW

Every hour or just pick a timeframe, jot down what you know. Not what you hope is true, or what your current favourite theory requires to be true. Just what you actually know. Write down repeated n-grams, likely cribs, structural observations.

Even the obvious things, especially the obvious things.

This also has the very useful side effect of reminding you that you probably have made more progress than it feels like. When you are stuck in the middle of something, it is very easy for all the small observations to blur together into one giant sense of getting nowhere. But once you write them down, you often realise that the problem has become more defined than it was an hour ago.

HAVE A SNACK

Your brain is doing a ridiculous amount of work while you are problem-solving, and there is a reason athletes are constantly being reminded that food is fuel. You are not less of a codebreaker because you paused to eat a snack.

CHECK IN WITH YOUR TEAM IF YOU HAVE ONE

It's very easy to disappear into your own rabbit hole of weird little theory ecosystem. You start becoming emotionally attached to ideas that would sound completely deranged if you said them out loud, which is precisely why it helps to say them out loud. Even if only to have someone else gently inform you that your beautiful theory about the spacing pattern is, in fact, nonsense.

So take a moment to sync and share weird hunches with your team, and remind each other to take breaks as well.

SETTING EXPECTATIONS

During Cipher Challenge season, especially towards the end, I have found it helpful to make it very clear to my family and people around me that I will probably be a bit mentally elsewhere for a while so they know what to or not to expect from me. I find that this definitely does reduce some background stress.

SLEEP

I don't think I need to explain this further. I appreciate this might be boring advice, especially to students, but sleep is not a scam. It's objectively one of the most effective forms of problem-solving available to you, so please remember to sleep at a somewhat sensible time for a reasonable number of hours. There is no medal for staying up the longest.

STEPPING AWAY

Slightly unconventional, but I have a hoodie I only wear when working on the cipher challenge[8].

When I take a break, I physically change out of it. It sounds silly but that very act of changing the hoodie helps me switch mental gears and actually rest. As much as we all love the challenge, I'm sure we all have other things (including food and sleep) that need getting done and I just found that having a physical reminder helped me compartmentalise.

There's a reason the codebreakers at Bletchley had hobbies and didn't spend every waking hour staring at ciphertexts. Alan Turing for example used running as a way to relieve stress from his intense mental work.[9].

The people we now imagine as these intellectual giants were still, at the end of the day, human beings with brains and bodies and limits. And if even they needed hobbies and movement and time away from the work, then you are probably allowed to go outside for a bit.

Sticking With It

Having done the cipher challenge for the five years, my only real regret was not sticking with 10B till the end in my second year.

I gave up too early and convinced I was going nowhere after working on it for just over a day, only to see others break through with just a bit more persistence.

What I've learned since then is that hints do come, but you've got to have patience. They're mostly be subtle or buried under obscure forum comments, or phrased in a way that only makes sense after you've cracked it.

But the point is this: you can be much closer than you think while still feeling completely lost. And if you stop too early, you never get to find that out.

Celebrating the Little Wins

It's very easy to let the whole thing feel overwhelming. So try and celebrate the little victories. Remember that marginal gains eventually add up.

[8] Sadly it doesn't have any actual powers, but having a dedicated codebreaking hoodie does make me feel extra geeky

[9] Turing could've also possibly qualified for 1948 London Olympics had he not suffered from an injury. You can read more of such facts about him here

Winning is great, obviously. I am not about to pretend that nobody cares about doing well. Of course they do. I did too. It sounds cliché to say *value the process over the outcome* but it's actually true. Performance brings pressure with it, and the more you care about something, the more tempting it becomes to turn it into a referendum on your intelligence, your effort, and possibly your worth as a human being.

This is, to be clear, a terrible idea. One of the easiest ways to ruin something you love is to only let it matter if you are succeeding at it visibly. And the Cipher Challenge is far too interesting to reduce to that.

Do not let the desire to do well take away the reason you started in the first place. Do not let the need to perform strip all the curiosity and delight out of the process. Because underneath all the frustration and the confusion, there is something genuinely beautiful here: the experience of wrestling with a difficult idea and slowly, stubbornly, beginning to understand it.

It's quite funny but I think that if a cipher has annoyed you enough to make you want to throw your laptop across the room, that is kind of wonderful[10]. But the fact that you care that much. The fact that a block of apparently meaningless symbols has somehow managed to grip your attention so completely that you are still here, still thinking, still trying, and that's rare. Most people spend huge chunks of their lives bored by the things they are doing. So if you have found something that is frustrating and difficult and maddening and still somehow compelling enough to keep pulling you back, that's a beautiful thing.

Remember that the feeling of finally solving it is really worth the storm of frustration and patience and mild existential crisis it takes to get there. (Mostly).

More than Just Codebreaking

If anything, this challenge teaches you more than just codebreaking.

Over the holidays especially, this challenge has been escapism in the best sense for me. There's something deeply comforting about knowing that no matter what else is going on, there's always a cipher waiting to be cracked, and a community of people trying, failing, thinking, posting, hinting, and occasionally losing their minds in roughly the same direction as you.

[10] Not the laptop part obviously

In many ways, the lessons you learn during the competition stay with you as well. Some of the most difficult things I've faced over the past few months were made more manageable because of what I learned while struggling with the challenges. Patience, persistence, and the willingness to keep trying different ideas even when the solution isn't obvious are skills that apply far beyond cryptography.

So work hard, care about it, get stuck, get annoyed, and keep going. But don't forget to enjoy it as well, because challenges like these are rare, and life would be much duller without them.

And with that, I'll leave you with the one piece of advice that seems to survive every bad idea, dead end, and mild existential crisis: DON'T PANIC.

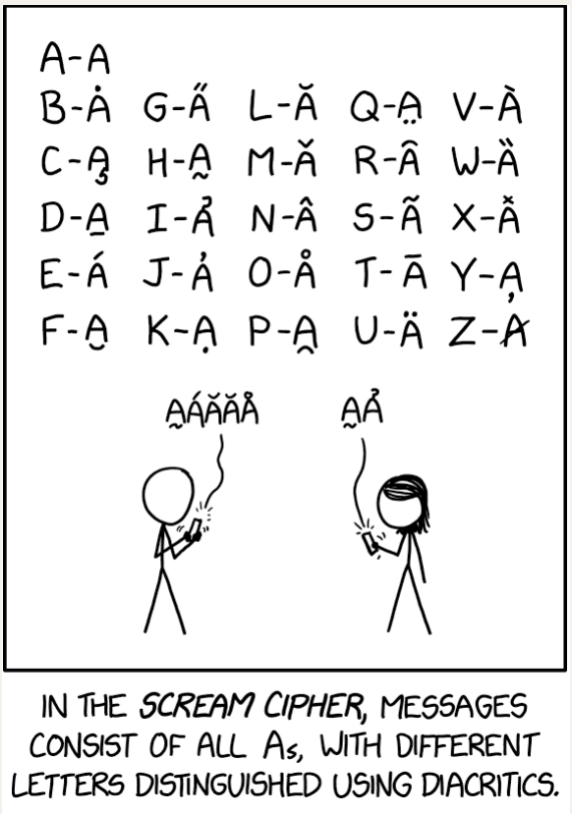

Figure 8.8: 3054: Scream Cipher