

WEEK 1 - PART 2

PROGRAMMER MENTALITY -

A GUIDE



LINNAEUS
UNIVERSITY
SUMMER 2024

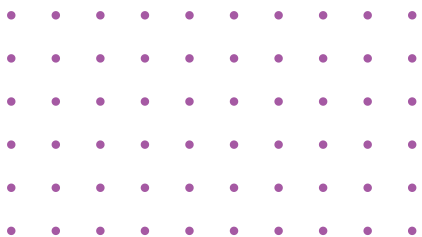
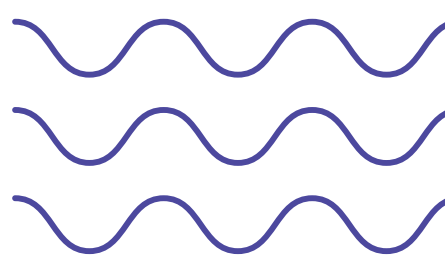
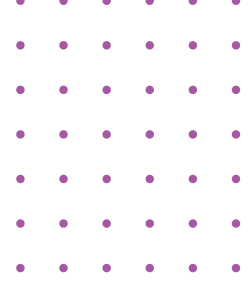


TABLE OF CONTENTS

3	mentality of a programmer	
	common misconceptions	3
5	mental skills to adopt	
	be curious and ask questions	5
	practice problem-solving	6
	adopt a growth mindset	7
8	practical advice	
	how to ask questions	8
	how to break down & solve problems	9
10	Chat GPT & Google - programming help	
	Google	10
	Chat GPT	11
12	asking questions on Slack	
	effectively communicate technical issues	12
	example	13



MENTALITY OF A PROGRAMMER

Programming's special nature lies in its ability to merge logic, creativity, and problem-solving to write code and create code that transforms ideas into functional solutions. Ideas need to be taken apart and broken down into practical components, and communicated to computers.

There are many preconceived notions about who writes code and what qualities or skills are required to be an effective coder.

COMMON MISCONCEPTIONS

Many believe that programming requires a special talent. **In reality, programming does not require innate talent; it is a skill that can be learned** with practice and dedication.

Success in programming comes from perseverance and continuous learning, not from a special aptitude.

Others think they can't program if they don't excel at math. Not true!

While mathematical skills can be beneficial, they are not a prerequisite for programming. Many programming tasks focus on logical thinking and problem-solving. These are skills used when solving math problems, which is why there is such a strong association between math and programming.

However, programming in most cases, and in most jobs, does not require advanced math skills.

In fact, when coding, language aptitude and reasoning often prove to be more crucial than advanced mathematical skills. **New research from the University of Washington finds that a natural aptitude for learning languages and reasoning skills is a stronger predictor of learning to program** than basic math knowledge or numeracy.



That's because writing code also involves learning a second language, the ability to learn that language's vocabulary and grammar, and how they work together to communicate ideas and intentions. Coding has a foundation in human language: Programming involves creating meaning by stringing symbols together in rule-based ways.



Language aptitude helps in grasping syntax, semantics, and the logic of programming languages, allowing for clearer communication of ideas and more effective problem-solving.


Some think that you have to know everything about computers to program. **But in fact, you don't need to be an expert in all aspects of computing to be a good programmer.** Understanding the basics of how computers work is helpful, and some degree of



computer literacy will make coding more enjoyable, speed up the learning process, and help solve issues. However, effective programming relies more on coding skills, problem-solving, and the ability to learn and adapt than an encyclopedic knowledge of computers.

There is also **the myth of the loner programmer**, writing code by themselves in a dark basement. The truth is, that **programming is most often a collaborative effort.** Many projects involve teamwork, communication, and sharing of code. Open-source communities, team-based development, and code reviews are integral elements of the programming process.





People often think that programmers are perfectionists. In reality, **perfectionism when coding can be counterproductive.** It's more important to focus on writing functional, maintainable code and iterating based on feedback.

Thinking you can write perfect code that works the first time you run it is unrealistic, and is going to contribute to a lot of unrealistic expectations and increase stress. Striving for perfection can lead to procrastination and hinder progress. Instead, it is important to internalize the importance of **iterative improvement and learning from mistakes**, fostering a healthier and more productive approach.

MENTAL SKILLS TO ADOPT




BE CURIOUS AND ASK QUESTIONS

Curiosity will drive you to learn and continuously evolve. It involves an eagerness to understand how things work, why certain solutions are used, and how different technologies or methodologies can be applied. **It's very easy to compare yourself with others, feel that your knowledge is lacking in comparison.** This can be a significant obstacle when it comes to asking questions, which is a crucial skill for learning how to program.

Most programmers have faced similar challenges at some point in their journey. By accepting that you will never know everything and embracing the learning process, you can overcome this barrier and fully benefit from asking questions.

Asking questions accelerates learning by providing clarification and allows you to grasp concepts more effectively and helps deepen your understanding of the material.





It also speeds up the problem-solving process. By asking for guidance, you can often resolve issues more quickly than by struggling alone.

Additionally, **asking questions fosters collaboration** with peers, mentors, and experts. It opens up avenues for sharing knowledge and learning from others' experiences. It also encourages collaborative environments, and contributes to **building a supportive community** where information and insights are freely exchanged. Lastly, it **encourages growth and normalizes the learning process**—asking questions shows that you're engaged and willing to learn, and it helps to create an environment where continuous improvement is valued and supported.

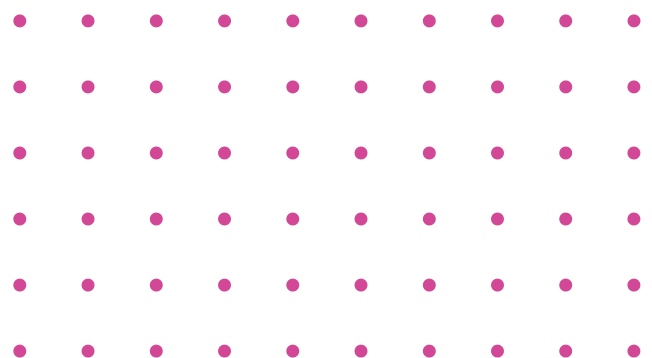
PRACTICE PROBLEM-SOLVING



Problem-solving skills involve the ability to dissect complex issues into smaller components, strategically analyze them, think critically, and find effective solutions. This often requires a blend of analytical thinking and creativity.

At its core, problem-solving is about **breaking down a problem into manageable parts and understanding each aspect individually**. By isolating the components of a challenge, you can tackle each one systematically, making it easier to identify the root cause and address it effectively. This approach not only simplifies the problem but also allows you to develop targeted solutions.

Creativity enhances problem-solving by encouraging innovative approaches and out-of-the-box thinking. **Sometimes, traditional methods may not be sufficient to address a problem, and creative solutions can lead to more efficient or novel outcomes.** Combining creativity with analytical skills allows you to explore unconventional solutions that might offer significant advantages.



Effective problem-solving also involves **iterative testing and refinement**. As you implement solutions, it's essential to test them thoroughly, gather feedback, and make necessary adjustments. This iterative process helps in refining the solution and ensuring that it effectively addresses the problem.

ADOPT A GROWTH MINDSET

It can be said that programming is a job for people who don't mind “feeling dumb”.

More than anything, programming is about solving problems.

This means that in this field, you will continuously face complex issues that may require several attempts and failures to resolve.

This is not an indication of intellectual failure or a lack of intelligence; it is simply a part of the process that everyone experiences.

In programming, not only will you need to tackle problems and explore various solutions before finding the right one, but the field itself is also continuously evolving. As technology advances, you must be willing to return to the proverbial school bench and learn new skills.

FIXED MINDSET

Intelligence is static

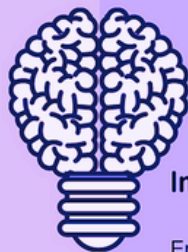
Avoid challenges

It's too hard

Expect reward without effort

Ignore feedback

Threatened by success of others



GROWTH MINDSET

Intelligence can be developed

Embrace challenges

I can train my brain.

Effort is a path to mastery

Learn from feedback

Inspired by success of others

Imposter syndrome is a well-known issue in tech, and it's understandable given that the core of the profession involves trying, failing, and trying again. **However, recognizing that this iterative process is a normal part of the profession—rather than a reflection of your abilities—can help you understand that experiencing failure is not unusual.** It does not diminish your intelligence or truly say anything about you as a programmer - **it's simply the nature of the job.**

Many people struggle at the beginning, but it's not a high IQ that determines success as a programmer. Instead, **it's the willingness to “feel dumb”, embrace the learning process, persist through challenges, and foster resilience that truly leads to success.**

PRACTICAL ADVICE

HOW TO ASK QUESTIONS

Be Specific: Frame your questions clearly and specifically. This helps others understand exactly what you need and provide precise answers.

Do Your Homework: Before asking, make an effort to research the issue on your own. This shows that you've put in some effort and helps you ask more informed questions.

Be Respectful: Show respect and gratitude. Acknowledge others' time and expertise, and be open to different perspectives and solutions.

Follow Up: After receiving answers, follow up with any additional questions or clarifications to ensure you fully understand the solution.

HOW TO BREAK DOWN & SOLVE PROBLEMS

When faced with a complex problem, it helps to **break it down into smaller, more manageable parts.**

Understand the Problem: Start by making sure you fully understand what you're trying to solve.

Ask yourself:

What is the main issue or goal?

What are the key requirements or needs?

Are there any specific constraints or limitations?

Divide the Problem: Break the problem into smaller pieces.

If it is still hard to grasp, keep breaking down the problem until it is manageable. This makes it easier to handle and solve. Think of it like slicing a big task into smaller, more actionable tasks. Identify the main parts or steps involved. Determine what each part needs to achieve.

Plan Your Approach: Decide how you will tackle each part of the problem.

This involves: Figuring out how the smaller tasks will fit together.

Planning the steps or actions needed to address each part.

Take Action: Start working on each part of the problem according to your plan: Focus on one small task at a time. Use the approach you planned to handle each piece.

Review and Adjust: After working on your solution, check to see if it's working as expected: Test each part to ensure it's functioning correctly. Make adjustments or improvements as needed. Don't be afraid to change approach if necessary!

Combine and Finalize: Once all parts are working, bring them together: Ensure all the smaller pieces fit together smoothly. Check that the combined solution meets the overall goal.

Breaking down problems into their smallest components will help you get started, get familiar with the problem, and help you systematically tackle each part of the task, making it manageable.

CHAT GPT & GOOGLE - PROGRAMMING HELP

GOOGLE

While asking questions is always a good idea, and should not be discouraged, most professional programmers will tell you that most solutions to a problem they encountered was achieved by googling the issue.

Googling problems is a vital skill for any coder. When you encounter issues or need to understand concepts, searching online can quickly provide solutions, documentation, or relevant discussions. Using search engines effectively can lead you to forums, tutorials, and articles that address similar challenges faced by others. It's important to frame your queries clearly and explore multiple sources to gather a well-rounded understanding. It is a skill to learn how to phrase the problem accurately and to identify which part of the issue needs to be addressed.



Practicing googling issues not only helps in solving immediate problems but also enhances your knowledge and coding skills over time.

CHAT GPT

This course will encourage you to use ChatGPT extensively. Think of ChatGPT as your personal 24/7 tutor who never needs to eat or sleep and is always available for your questions.

Feel free to copy and paste text or code you don't understand and ask ChatGPT for explanations.

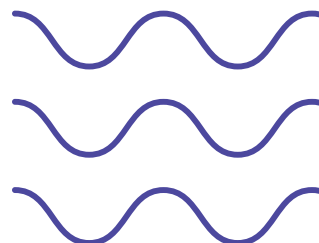
Knowing how to use ChatGPT to explain concepts, provide examples, or help you get started will give you a significant advantage. This technology is becoming increasingly prevalent, and the more familiar you are with it, the better you'll be at leveraging its capabilities. Practicing with ChatGPT will enhance your problem-solving skills and prepare you for future applications of this tool. People will increasingly rely on tools like ChatGPT in the future, and getting comfortable with it early on will give you an edge!

Using ChatGPT to cheat is not the goal here—relying on this tool to bypass learning will ultimately backfire.

It will be quite embarrassing when you start working and realize you lack the knowledge and skills you claim to have. Instead, use ChatGPT as a resource to enhance your understanding and support your learning journey.

One must critically evaluate the answers received from ChatGPT, and to do that effectively, a solid background knowledge is essential. Just as translators sometimes make mistakes, blindly copying and pasting from ChatGPT can lead to issues, even though it may occasionally work. With more knowledge, you'll be better equipped to distinguish between code that can be copied and pasted—code you immediately understand but may not have thought of initially—and code that needs to be customized or replaced. **To make accurate assessments, you need to understand the code**, follow its logic, and understand how to use it in the context of other code you have written.

Remember, while ChatGPT can accelerate learning, it cannot replace the need for fundamental knowledge.



ASKING QUESTIONS ON SLACK

EFFECTIVELY COMMUNICATE TECHNICAL ISSUES

You must assume that the people you are asking have 0 background knowledge about your circumstances. They might not even be using the same operating system as you! Say what it is you want to accomplish.

Describe what you have done so far - be as specific as possible.

Take a screenshot of your code, and if possible, your error. Explain what the goal is, and what happens instead.

EXAMPLE

Person A

Hello everyone! I'm having a problem with a code snippet I'm trying to write in Python. I'm trying to open a file and write something in it, but it doesn't seem to be working as expected. Can someone help me?

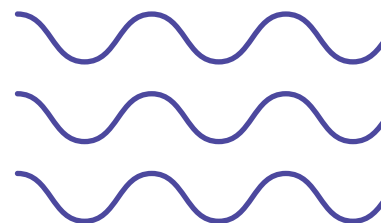
Person B

Of course! I can try to help you. Can you share the code you have written so far and tell me what you're trying to achieve?

Person A

I'll send you a screen shot

```
filename = "myfile.txt"
file = open(filename, "w")
file.write("Hello, world!")
file.close()
```





Person B

Thank you for sharing the code! The goal of this code seems to be to open a file named "myfile.txt" and write the text "Hello, world!" in it. Am I understanding it correctly? Which operating system are you using?

Person A

Person A: Yes! But when I run the code, nothing happens, and no file is created. I'm using Windows.

Person B

There could be several reasons for that. First, we need to make sure we have the proper permissions and that the file ends up in the right place. Which operating system are you using?

Person A

Yes! But when I run the code, nothing happens, and no file is created. I'm using Windows.

Person B

Great! To troubleshoot, you can try adding a complete file path, such as C:/Users/Username/myfile.txt, to ensure that the file is created in the right location. You can also check if there are any error messages displayed when you run the code.

Person A

I'll give it a try and let you know if it works or if I encounter any problems. Thank you for your help!

Person B

No problem! Keep us updated and don't hesitate to ask if anything else comes up. Good luck!

