PRAGMATICODERS WWW.PRAGMATICODERS.COM

DATA ENGINEERING ROADMAP

“Don’t be overconfident. Don’t be insecure.”

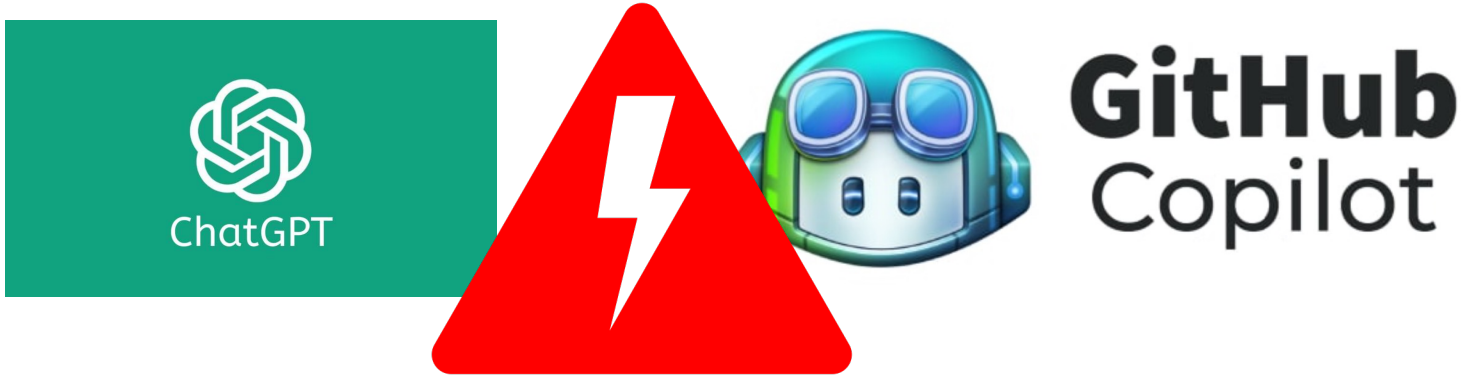
You can’t learn everything all at once. You CAN learn what you need to be successful, with patience and effort.

Tech Tools

Anaconda (conda), pip, Visual Studio Code (VSCode), PyCharm Community, Jupyter Notebook/Lab

Important because:

- **Gives you an environment to code in**
- **Can take advantage of intellisense/autocomplete**
- **Great support and communities that know so much about how to enhance and use these tools**



Over reliance on these tools will cause your skills to atrophy. If you get it to generate code, spend the extra time having the tool explain to you what it did. Using these skills too early is like giving a child a Ferrari/Lamborghini.

Learn Python – PART 1

1. Shotgun this video. Take simple notes, the point is to get through the video.

https://www.youtube.com/watch?v=rfscVS0vtbw&pp=ygUWZ2lyYWZmZSBhY2FkZW15IHB5dGhvbg%3D%3D

2. Do practice problems starting from Very Easy. Do not pass medium difficulty. For now, you just want to learn and practice quickly. You want quick feedback loops and application, so you learn and practice.

i. **https://edabit.com/challenges/python3**
ii. **https://leetcode.com/problemset/all/**

3. Follow a long with a website like this:

https://www.programiz.com/python-programming

https://www.educative.io/catalog/python

https://www.udemy.com/course/complete-python-bootcamp/

By step 3, you should have at least heard the material once or twice. Now, you can start taking notes and studying seriously. None of this should be brand new and you can really drill down on areas where you have gaps. I highly recommend this approach. If you do not know Python, everything else will be exponentially more difficult as you move down the roadmap.

Learn Python – PART 2

1. **Learn about Unit Testing. Right now, just learn and as you progress in the roadmap, think about ways you can incorporate what you learned. Do not focus on application yet, just theoretical knowledge and why it's important. Youtube for videos on Pytest and Unit Testing generally for now.**
   https://svitla.com/blog/testing-frameworks-for-python
2. **Python Libraries to Know:**
   1. **Requests –** know how it works and what it is for. Try to explain to yourself when this library would come in handy. Try finding other libraries in Python that do similar things and explain to yourself why/how they are similar and different.
   2. **Beautifulsoup** – This is for simple webscraping. Follow tutorials online and build at least one simple web scraper. Also learn about legalities of web scraping and keep this in mind when doing web scraping projects. You can get into a hot water if you scrape a website but never consult with your legal team at your workplace.
   3. **JSON** – Learn how to handle JSON data in Python.
   4. **Pandas –** Pandas is heavily relied on for data work. Learn as much as you can about Pandas and prepare to use this a lot. Don't go cheap on learning this, spend the time and energy to be proficient and understand the theory behind Pandas too.
   5. **Numpy –** Numpy is important for working with Data in ML and other areas. Learn a bit about the data structures and why this library is useful. When would you use Numpy instead of Pandas is a great place to start.
3. **Working with different file extensions/types**: csv, json, parquet, txt, xml, etc. –
   1. You must know how to convert between these, Pros and cons of using one versus another, how to read in these files effectively

HTML & CSS

1. Now you think I'm crazy.
2. It would help you to spend at least 1-2 hours on a crash course covering HTML and CSS.
3. Will be super useful for writing web scrapers to know at least the terminology:
    1. Element, tag, class, id, etc.
4. CSS:
    1. https://www.youtube.com/watch?v=1PnVor36_40&pp=ygUIaHRtbCBjc3M%3D
5. HTML:
    1. https://www.youtube.com/watch?v=XiQ9rjaa2Ow

6. That's it. In less than 1-hour you can become knowledgeable about SOME HTML and CSS. You're not building sites, you just want to know how they're built at the most rudimentary level (structure and design).

7. Javascript is out of scope although you may hear it mixed in. For now, just HTML and CSS.

Command Line

1. **A little command line is required:**

   https://www.youtube.com/watch?v=yz7nYlnXLfE&pp=ygUlaW50cm9kdWN0aW9uIHRvIGNvbW1hbmQgbGluZSBmb3IgZGF0YQ%3D%3D

2. **You should know:**
   1. **What command are**
   2. **What modifiers are**
   3. **How to google help with command line issues**
   4. **What PATH issues are in your operating system – this is the one that will cause you the most problems/difficulty. Study this well up front to save you hours of headaches and potentially embarrassment.**
3. **Commands to know at the minimum (these can be combined)**
   1. **Cat**
   2. **Cd**
   3. **Dir *or* Ls –** which one you use depends on the terminal/OS you are using
   4. **Mkdir**
   5. **Rm**
   6. **Cp**
   7. **Mv**
   8. **.**
   9. **~**

Git – do not overdo it

1. Watch these videos and try to find information online explaining to you why Git / version control is important. You probably already have some exposure to the concepts if you use Google Drive or OneDrive and can see version tracking of the files you have. If you edit a word doc, you can see the changes, when they happened and you can revert. Similar idea here.

a. https://www.youtube.com/watch?v=HVsySz-h9r4&pp=ygUMZ2l0IHR1dG9yaWFs
b. https://www.youtube.com/watch?v=CvUiKWv2-C0&pp=ygUMZ2l0IHR1dG9yaWFs

Data Structures & Algorithms

1. **Problems about waiting to learn this stuff:**
    1. **You will get used to writing slow and poorly thought out code if you wait until the end of the roadmap to think about this**
    2. **You will try to slam all of the information at once and will suffer in interviews. Usually roadmaps will put this toward the end. No! Learn about it early and then when it comes time for you to get serious, it's not brand new info. It's review. It's not even that hard, it's just less flexible at first than wildly coding whatever you want.**
2. **This book is great. Makes everything visual and is easy to get through. This is a conceptual book and is worth everything. See if you can find it online somewhere for cheap or free**
    1. **https://www.manning.com/books/grokking-algorithms**
3. **Another good book that is not insanely technical but covers everything you need to know with python examples**
    1. **https://www.amazon.com/Self-Taught-Computer-Scientist-Beginners-Science/dp/1119724414**

Learn SQL

1. **Learn SQL**
    a. **Database Concepts: tables, indexes, keys, queries, joins, normalization**
    b. **CRUD ops, aggregation, subqueries, window functions**
    c. **Learn about different flavors: mysql, sqlite, postgres, Microsoft sql**
    d. **Do SQL Practice Problems**
        i. **https://leetcode.com/studyplan/top-sql-50/**
    e. **Relational database design, entity relationship diagrams**

1. All of the SQL you need to be successful is in this one course. It's the best and I've gone through this course 2-3x. You're going to take the course, write notes and then do as many practice problems as you can. If you see yourself getting stuck on certain questions, you will find the section in the course you're stuck on, and review it again. These are not movies, these are courses. They are references, not stories. **There's no need to obsess about every single detail, you need to obsess on the concept so you can Google your way through a problem.**
    1. **https://www.udemy.com/course/the-complete-sql-bootcamp/**

2. **OLTP vs OLAP –**
    a. **why is this differentiation important?**
    b. **Which architecture should you use when?**

3. **Object Relation Models –**
    a. **https://www.fullstackpython.com/object-relational-mappers-orms.html**
    b. **you just need to know what this is, do not expect to implement this in the first year of working. If you end up needing to, just research.**
    c. **what are they for?**
    d. **why are they important?**
    e. **What technologies are out there for ORM?**
        a. **https://www.prisma.io/**

Learn Non-Relation DB/NoSQL Databases

1. Learn about non-relational database (non-RDBMS/noSQL)
    a. Keep it simple – learn MongoDB
        i. Do practice MongoDB practice problems
            1. https://www.w3resource.com/mongodb-exercises/
    b. Data Modeling – setting up your data into relational tables

Statistics

a. You're going to dedicate 1-2 week(s) to learning as much basic statistics as possible
b. Many people forget this stuff but need to know about it: sampling, distributions (normal, skewed, log, etc), central limit theorem (averages, median, etc in plain english…), t-tables, Z-scores, normalized data)
c. If you can only do 1, do option 1. If you can do both, do it! These courses should never be more than $20. If they're more than $20, leave the page and come back later. You'll see it's on sale for like $15-20. Weird marketing…
    a. Option 1: https://www.udemy.com/course/math-for-data-science-masterclass/
    b. Option 2: https://www.udemy.com/course/statistics-probability/

Math for Data Science Masterclass

Learn about probability, statistics, and more using the mathematics that are foundational to the field of data science.

Jose Portilla, Krista King

4.6 ★★★★½ (813)

16 total hours · 68 lectures · All Levels

Become a Probability & Statistics Master

Learn everything from Probability & **Statistics**, then test your knowledge with 600+ practice questions

Krista King

4.7 ★★★★½ (11,282)

15 total hours · 148 lectures · All Levels

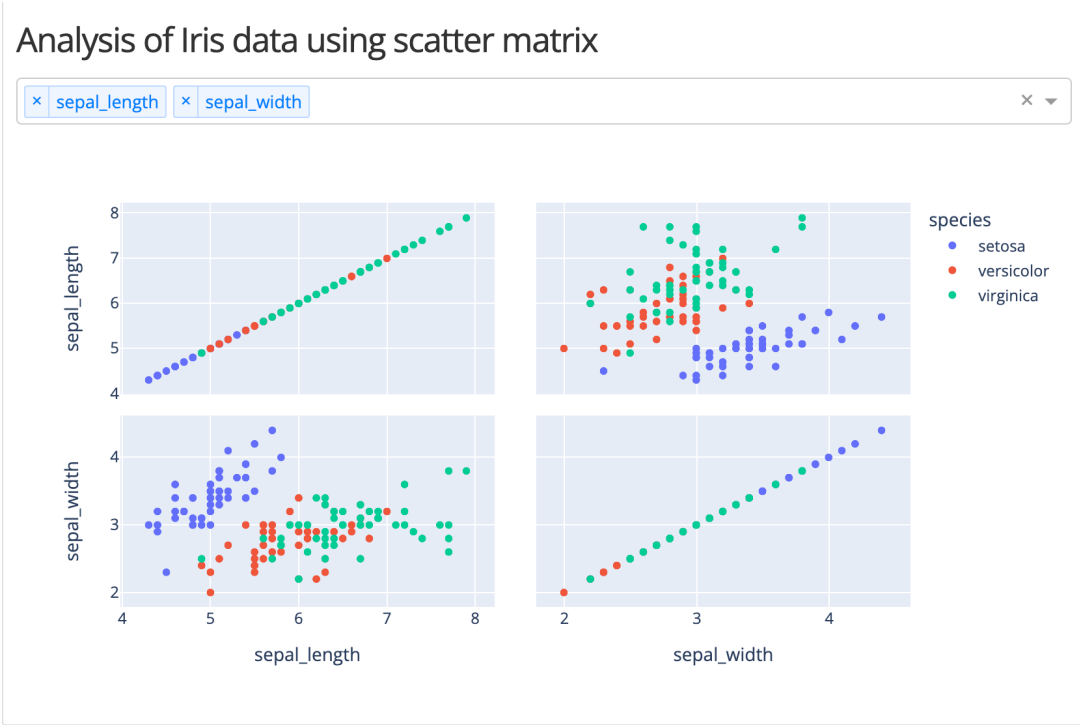
Bestseller

Data Cleaning

a. Handling Missing Values, Normalization, Min Max Scaling, Standardization
    i. You're going spend 2 days minimum dedicated to doing as much research as possible learning how to handle missing values and weird things in your data.
    ii. This will get you slaughtered in interviews if you don't know how to do this.
b. Units and Conversions –
    i. Converting units in data. Say one column is in grams and the other is in pounds. Would you convert these to have a common unit? Depends.
    ii. Research this and learn how pros handle units in data. It's not as straight-forward as you may think.
c. Krish Naik – amazing instructor. His whole channel is FIRE!!
    a. https://www.youtube.com/watch?v=P_iMSYQnqac
    b. https://www.youtube.com/watch?v=q-DyjA8ZmYM
d. Opinions and solution – Good to know his point of view. He answers it in a way you can also explain it during an interview.
    a. https://www.youtube.com/watch?v=f9AQy7p0QEo

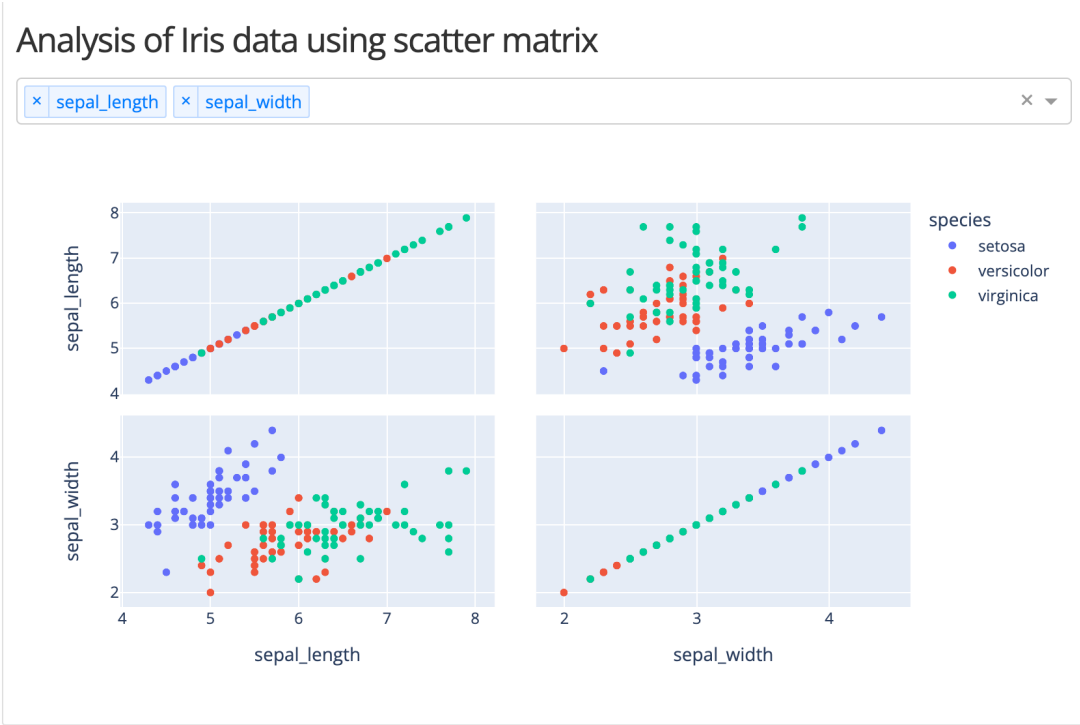
Data Visualization

a. **Now that you cleaned your data and know some statistics lets make some charts to explain things in the data**
b. **Matplotlib.pyplot (PLT), Seaborn, and PlotlyExpress are the go to for me**
    i. **You can spin up a chart in 2 seconds using ChatGPT or Github Copilot with these**
    ii. **Matplotlib.pyplot is when you want control of everything on the screen (it can get messy)**
        i. https://matplotlib.org/
    iii. **Seaborn is an abstraction of matplotlib and makes prettier charts, less control on its own, but can be modified by reading in PLT**
        i. **https://seaborn.pydata.org/**
    iv. **Plotly.express is a subset of plotly-dash. It makes nice simple charts that are modular and are interactive!**
        i. https://plotly.com/python/plotly-express/



Data Visualization

a. **Now that you cleaned your data and know some statistics lets make some charts to explain things in the data**
b. **Matplotlib.pyplot (PLT), Seaborn, and PlotlyExpress are the go to for me**
    i. **You can spin up a chart in 2 seconds using ChatGPT or Github Copilot with these**
    ii. **Matplotlib.pyplot is when you want control of everything on the screen (it can get messy)**
        i. https://matplotlib.org/
    iii. **Seaborn is an abstraction of matplotlib and makes prettier charts, less control on its own, but can be modified by reading in PLT**
        i. **https://seaborn.pydata.org/**
    iv. **Plotly.express is a subset of plotly-dash. It makes nice simple charts that are modular and are interactive!**
        i. https://plotly.com/python/plotly-express/

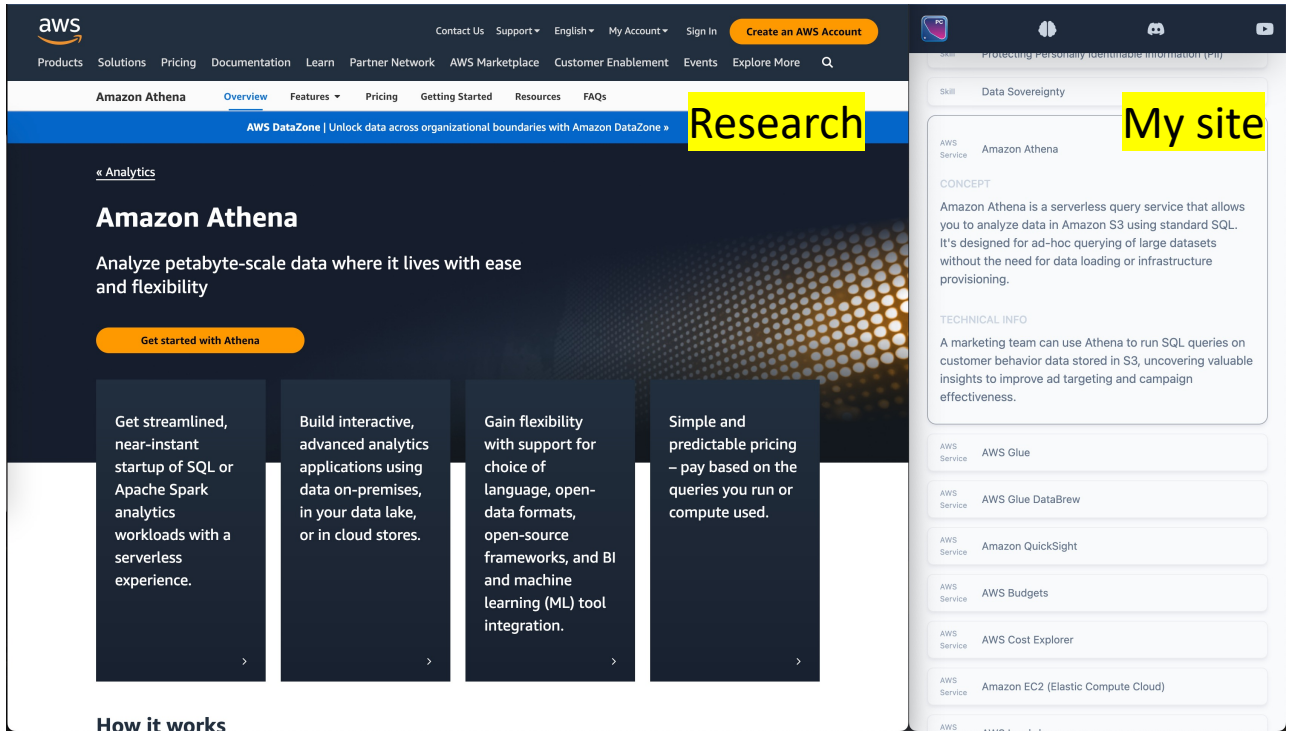


Job Orchestration and Pipelines

a. What is ETL and ELT?
    a. What's the similarities and differences?
    b. When do I do which?
    c. They sound the same… Why do companies specify them differently?
b. Learn about Apache Airflow
    i. Find/buy this book
        i. https://www.manning.com/books/data-pipelines-with-apache-airflow
    ii. What is it for?
    iii. How do I setup my own cluster locally on my computer?
    iv. Think about how this would work on the cloud (do 1-2 hours research on this topic, don't go overboard)
c. What is orchestration about?
d. Learn about CRON
    i. What is it for?
    ii. Is Airflow the only tool for this?

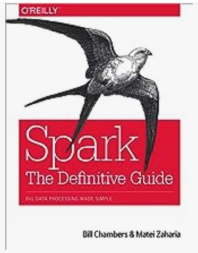
AWS

a. At the minimum, you need to know:
    a. How to use them:
        i. S3, EC2, RDS, Athena, Lambda, Costs + Budgeting
    b. What they are and what they are for:
        i. ECS, ECR, VPC, IAM
b. This can get overwhelming QUICKLY. Just go to my website and just read the flash cards. Then, look up each of the items and just research and take notes. This is FREE.
c. https://www.pragmaticoders.com/learn
d. Have my website on the right, as in the image, and look up more information once you've read the card.
e. From here you can look into the AWS cloud certs but for now just dip your toes via exposure.

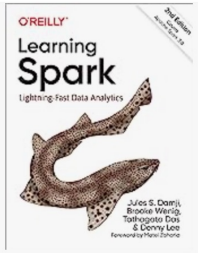


Learn Spark – Preferably Pyspark

1. Spark
    a. Spark has different language APIs: java, python, sql, and scala
        i. Just go with Python
        ii. It's similar to pandas so it shouldn't be terribly new info
        iii. Some of the syntax is different but it's not world changing
        iv. When you enter the realm of big data, you need to leave the idea of looping through each row at the door
        v. If you try to loop through 1 billion of rows data with a for loop… you're going to be waiting… a lot
        vi. Below are the two best books.
            i. Find them/buy them online – I cannot guarantee you how much better this is than anything out there to learn Spark.
            ii. Even Jose Portilla's content is not good, and everyone knows I love that dude's material.
            iii. One of them is more high level but still can get you going
            iv. The other is more detailed but not too much where it's a snore to read
            v. You can't go wrong, just get a preview before you drop money



Spark: The Definitive Guide: Big Data Processing Made Simple
by Bill Chambers and Matei Zaharia | Mar 8, 2018
4.5 ★★★★½ 416
Paperback
$43.61 List: $69.99
prime Two-Day
FREE delivery **Mon, Oct 30**
More Buying Choices
$28.56 (37 used & new offers)
Kindle
$15.53 to rent
$41.43 to buy
Available instantly



Learning Spark: Lightning-Fast Data Analytics
by Jules Damji, Brooke Wenig, et al. | Aug 25, 2020
4.5 ★★★★½ 269
Paperback
$43.99 List: $79.99
prime One-Day
FREE delivery **Tomorrow 10 AM - 3 PM**
More Buying Choices
$39.99 (10 used & new offers)
Kindle
$41.79 Digital List Price: $67.99
Available instantly

Databricks or Snowflake

1. Databricks or Snowflake –
    a. It's going to be likely your company will use one of these tools.
    b. I'd recommend learning databricks because it will force you to learn PySpark more.
    c. Learn everything you need about databricks here:
        a. https://www.databricks.com/learn/training/home
    d. Then learn delta lakes: why they are important, how to use them, and how they incorporate into AWS (S3 and Athena specifically)
    e. Once you enter the databricks ecosystem everything starts to unfold on you.

Shameless Plug

1. **Go on my website and click the little brain or just go straight to this URL: www.pragmaticoders.com/learn**
2. **Read those cards when you're bored and force yourself to learn.**
3. **I am this far along and I am still making myself learn new stuff. You're not going to be able to hang and get that dream job if you're caking it. Nobody wants to hire the minimum they can get. So make sure your minimum is well above the average. You took the stats course, so you know what I mean ☺**

4. **Good luck. Keep in touch. Have fun. Byeeeeee.**

Talk to me:
support@pragmaticoders.com