Tech Onboarding

Before You Code: Understanding Your Tools

Introduction

This short, guide is meant to give you a mental model for the tools you'll be using in your coding journey, before you dive in.

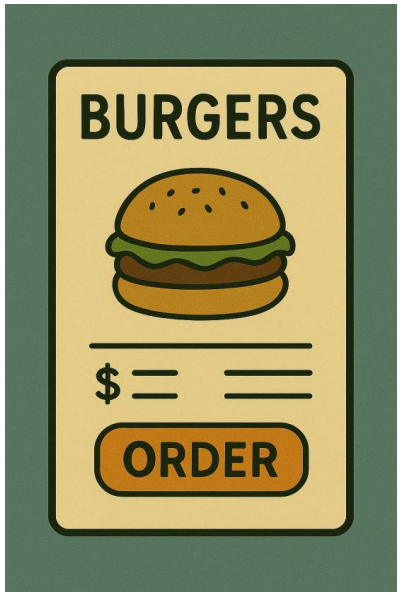
By the end of this, you'll have:

- A basic understanding of what Python, Conda, Terminal, Bash, Shell, and Git are.
- A clear idea of why these tools matter.
- A mental model of how they all fit together.

Let’s start with an Analogy!

Analogy

Imagine you and your family are hungry at home and craving some fast food burgers. Now, you've got a few options:



Order via the app

Order via the kiosk

Order via the cashier

Same Burger, Different Paths

- Whether you order through the app, the self-serve kiosk, or the cashier, you're still interacting with the same machine: the burger shop.
- Their job is to receive your order and send it to the backend (the kitchen).
- That's what a ***terminal*** is in coding, one way to talk to the machine.
- It doesn't make the burger, it simply sends your order to the kitchen, where the real work happens.

Terminals in the burger world:

Order via the app

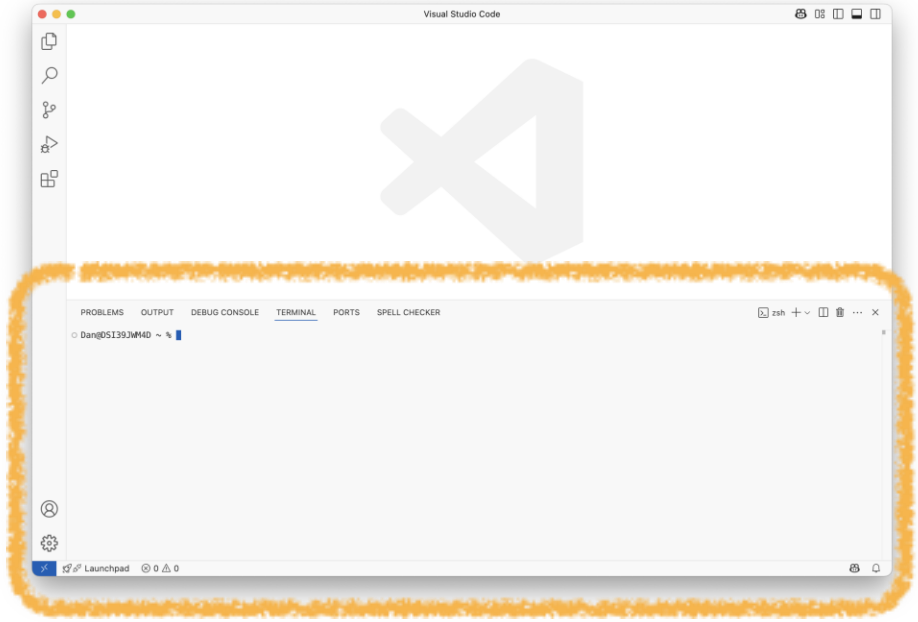
Order via the kiosk

Order via the cashier

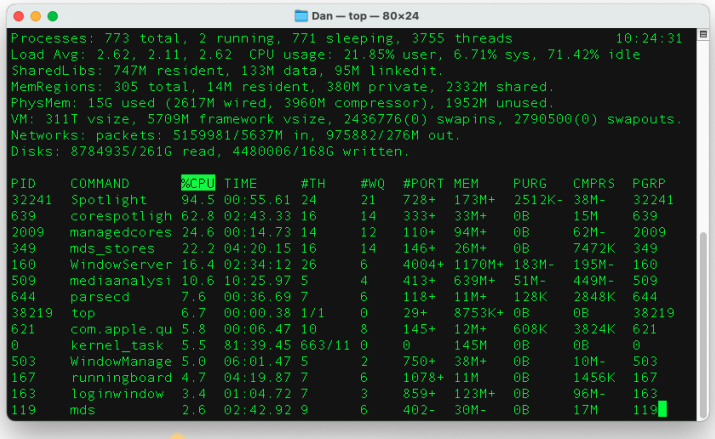
Terminal

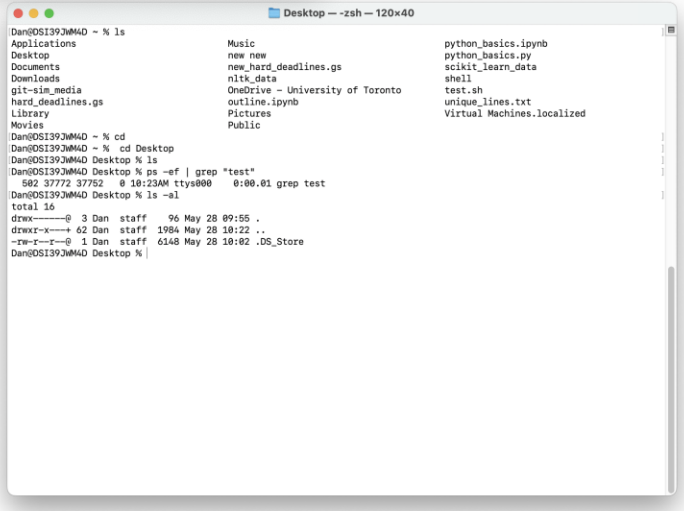
Terminals in the computer world:

A terminal built into Visual Studio Code



A terminal with a black background and green, stylized text





The Default terminal in macOS

Terminal

You can disregard what's written in the terminals above.

So let’s say we ordered the food, what now?

The "Runner" gets the Order

When you place an order at the terminal, it doesn’t go straight to the kitchen, it goes to the **Runner**.

The Runner:

- Tells the coffee person to make the drinks
- Tells the fries guy how many to drop
- Sends the burger order to the kitchen
- Gathers everything and sends it back to you

The "Runner" gets the Order

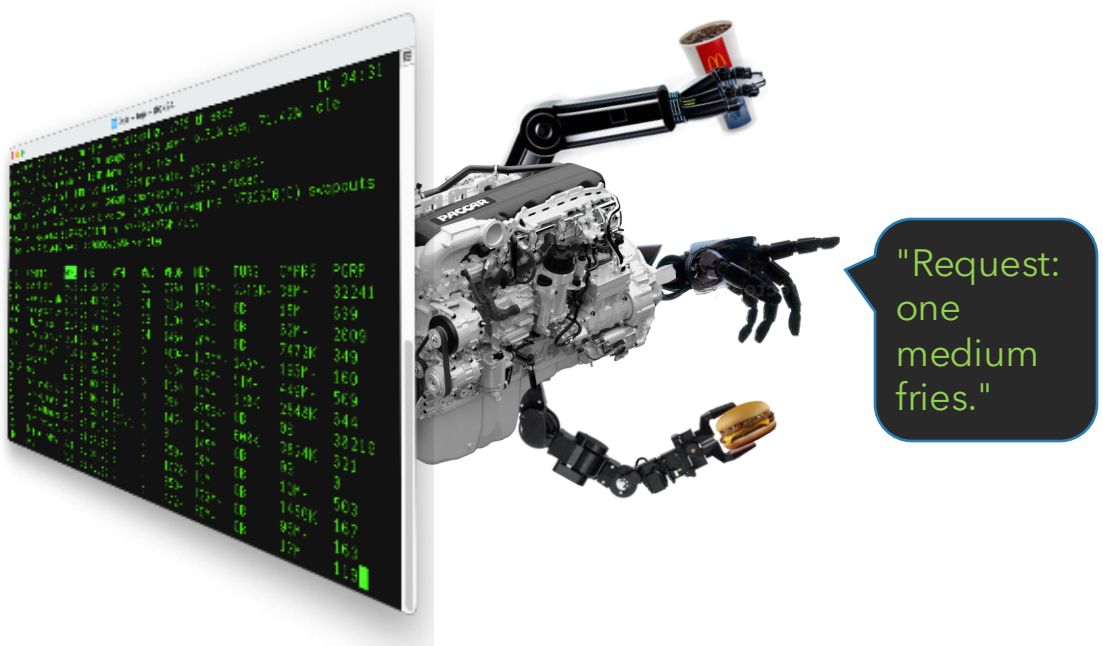
That engine you see? That’s the runner in the computer world! That’s the "Shell", hiding behind the terminal.

When you type a command, the terminal doesn’t magically know what to do, the Shell steps in and figures it out.

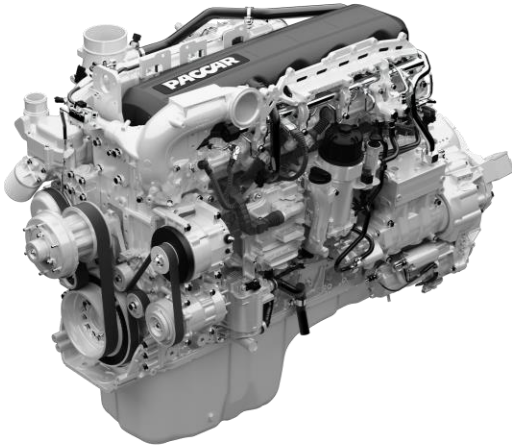
The Shell:

- Sometimes does the task itself (for built-in commands)
- Sometimes delegates the task to other programs
- Collects the result of the command
- Sends it back to the terminal so you can see the output

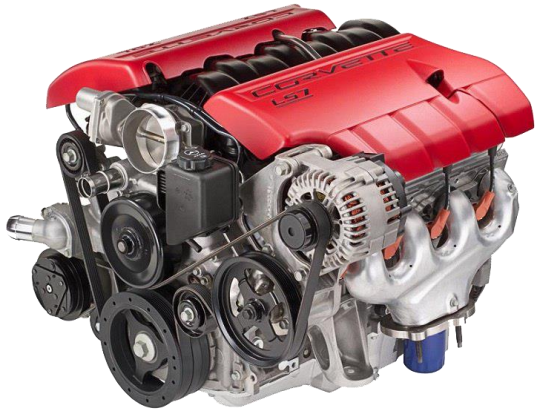
The Shell is your behind-the-scenes operator handling your requests line by line.



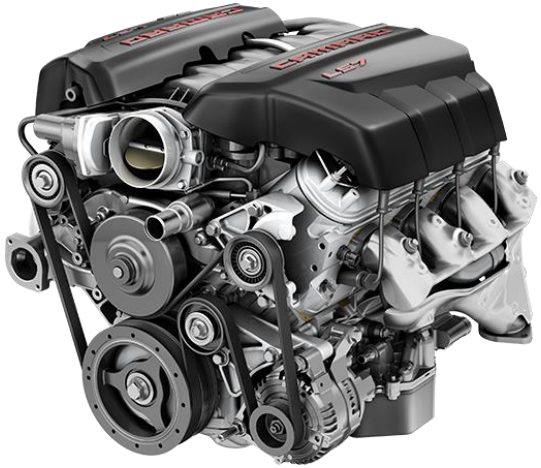
Different Shells



Bash



Zsh

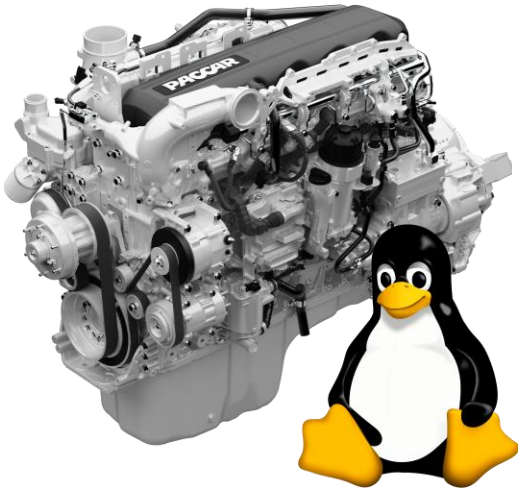


PowerShell

Command Prompt

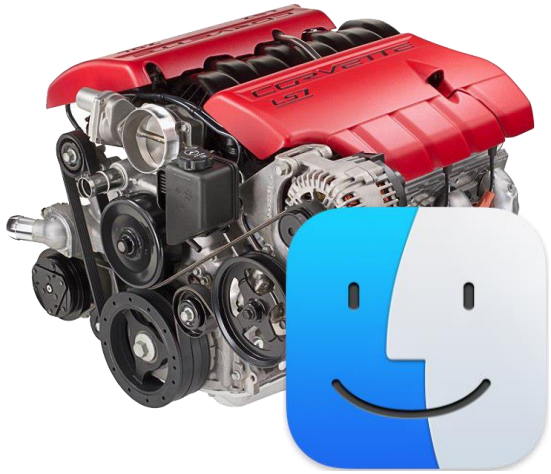
Mainly found in:

Linux



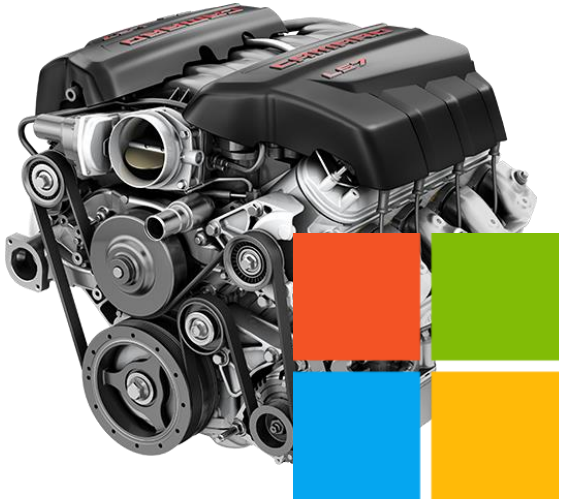
Bash

Mac OS



Zsh

Windows



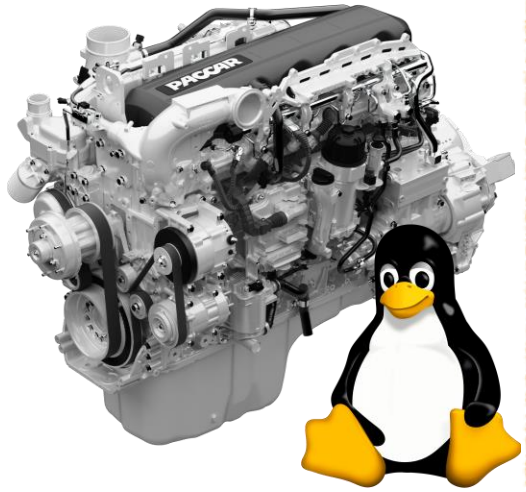
PowerShell

Windows

Command Prompt

Mainly found in:

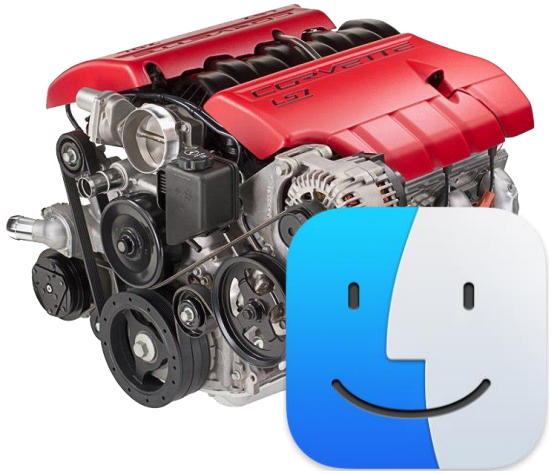
Linux



Bash

The Most Popular (What we're learning)

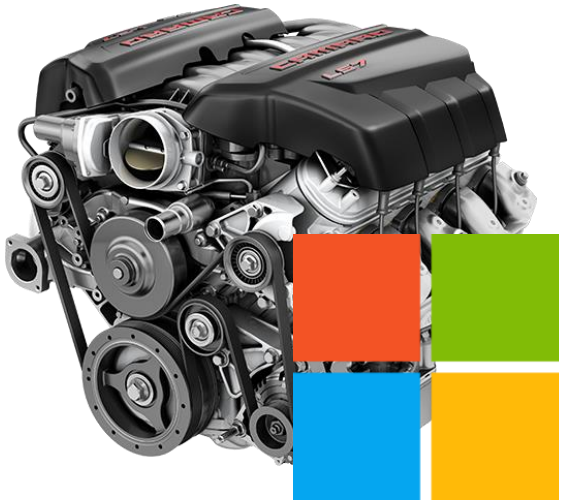
Mac OS



Zsh

Very Similar to Bash

Windows



PowerShell

Windows

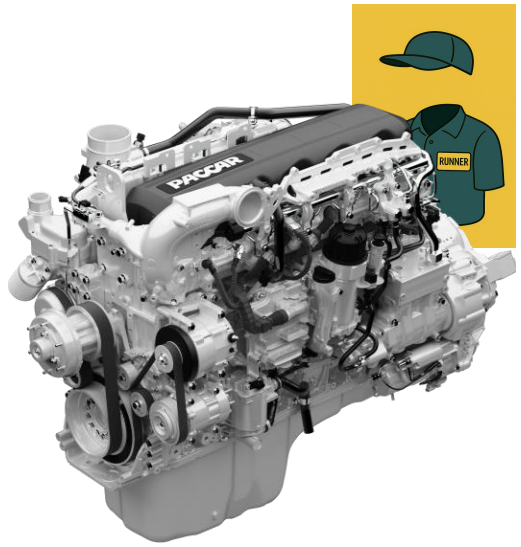
Command Prompt

Not that fun to use :(
BUT in this tutorial we will teach you how to set up Windows to be more like Bash :)

So:

Terminal

The Terminal is just the window, the interface where you type commands.



Shell

The Shell is the engine that takes those commands and makes sense of them.

Bash is one specific kind of shell, and the one we'll be using.

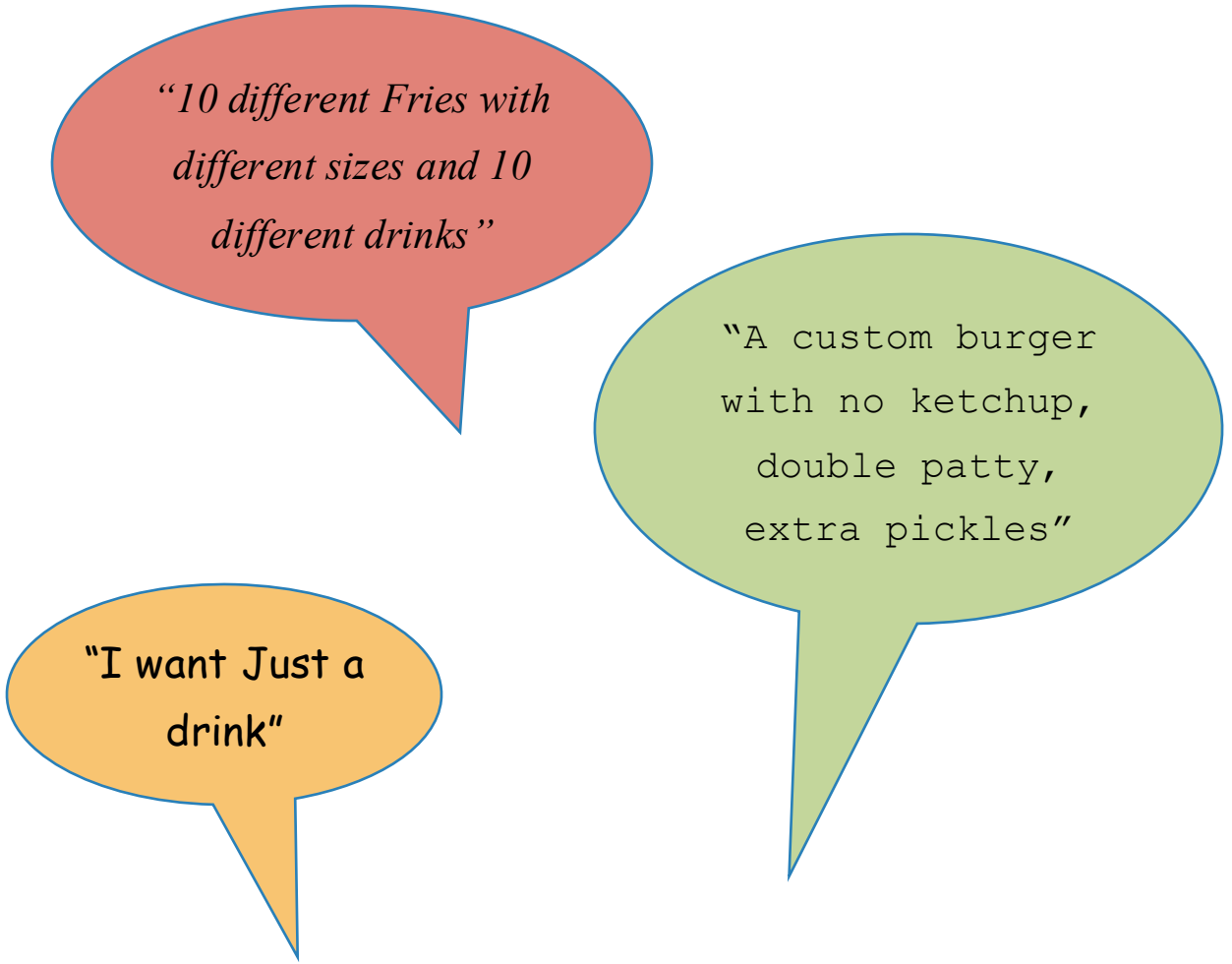
Whatever it needs to do

Next Up!

Let’s Make the Order!

Not all Orders Are the Same

To understand how orders get made, it helps to first look at the kinds of requests people make, from the simplest to the most custom.



Just a Drink

When a customer just asks for a drink, the runner doesn’t need to talk to the kitchen, they just grab it themselves and hand it over.

In the world of computers, this is like typing a single command directly into the terminal ("Make a new folder", "Show what’s inside a directory", "Do a quick calculation", etc.)

These are quick tasks. The Shell understands the request and takes care of it immediately, without needing help from anyone else.

Big Order of Simple Things

This time, the runner has a longer list, it's still made up of basic items, but there are a lot of them. Rather than remembering it all, you write it down and hand them the list.

In computing, this is like writing a **Shell Script,** a file that contains a series of terminal commands. Instead of typing them one by one, you hand the Shell the whole script, and it goes step-by-step to complete the task.

This is still the Shell doing the work, without you typing each step, with instructions you've written in advance.

Custom Order

This time, the runner doesn't just follow basic instructions, you've written a full **custom recipe** for a burger no one's seen before. The runner brings that recipe to the ***chef*** in the kitchen, who follows it step by step to make your exact custom burger.

In the world of computers, this is like writing a **program** in a real programming language, not just a list of commands.

You write the recipe (your code), then run it through the terminal.
The Shell sees it's a program file and passes it to the **interpreter or compiler** (the chef) who understands the language and turns it into something real.

Different Chefs

Different chefs speak different languages:

Speaks Python

Speaks JavaScript

Some are different kind of chefs:

Speaks Java

Python is the programing language we'll be using throughout the DSI Certificates

But the basic idea is the same:

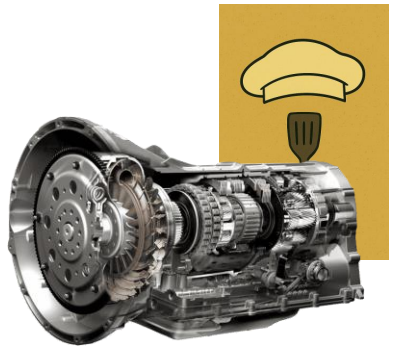
you give instructions → the chef brings them to life.

So:

A shell script that in it also wants you to call a Python script.

Shell

Performs the Shell tasks



Python Interpreter

Performs the Python tasks

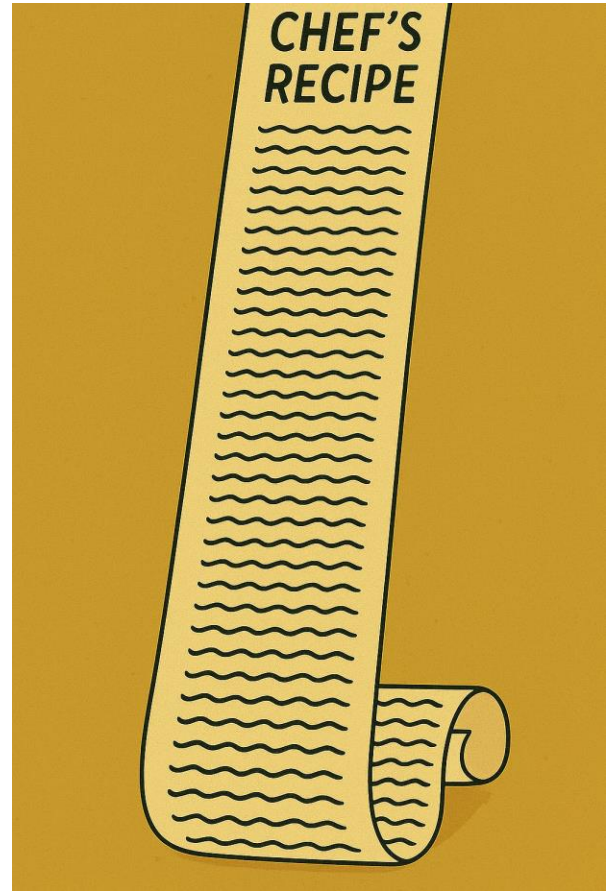
One Last Kind of Order

Now imagine a customer walks in and makes a very demanding request. They want a custom burger, but with ingredients your kitchen doesn't even have on-site.

Not only that, they want it made by a chef who speaks a very specific dialect of Python… version 3.9, no earlier, no later.

This might sound extreme, but in the world of software, this is actually one of the most common scenarios. Developers often need:

- Very specific tools
- Very specific versions
- And ingredients (dependencies) that aren't available by default



So what do we do?

Send the cashier to run around the city collecting ingredients and finding the perfect chef?

There has to be a better way…

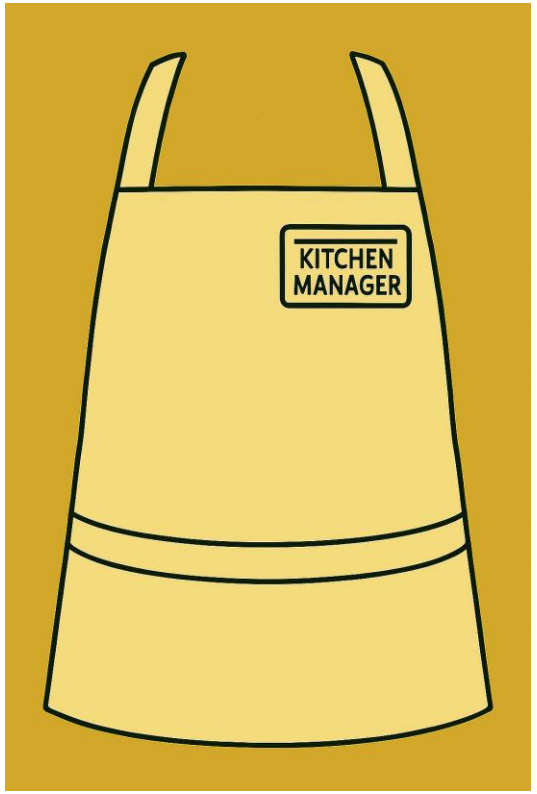
Meet the Kitchen Manager(s)

In any kitchen, someone has to make sure:

- The right ingredients are on hand
- The right chef is scheduled (and speaks the right language version)
- The kitchen is clean, isolated, and ready to cook

In some kitchens, one person does it all. In others, the responsibilities are split across multiple roles.

In software, this job is handled by tools like Conda, venv, pip, and more. Let’s talk options…



Options

venv + pip

The classic duo. venv sets up the kitchen, and pip does the shopping. Trusted, simple, and widely used, a solid go-to combo for many kitchens.

CONDA

Like a reliable kitchen manager, Conda sets up your cooking space (environment), shops for the exact ingredients (packages) you need, and makes sure your chef (Python) speaks the right version, so everything runs smoothly from prep to plate.

Note: Conda is the tool we'll be using throughout the DSI Certificates

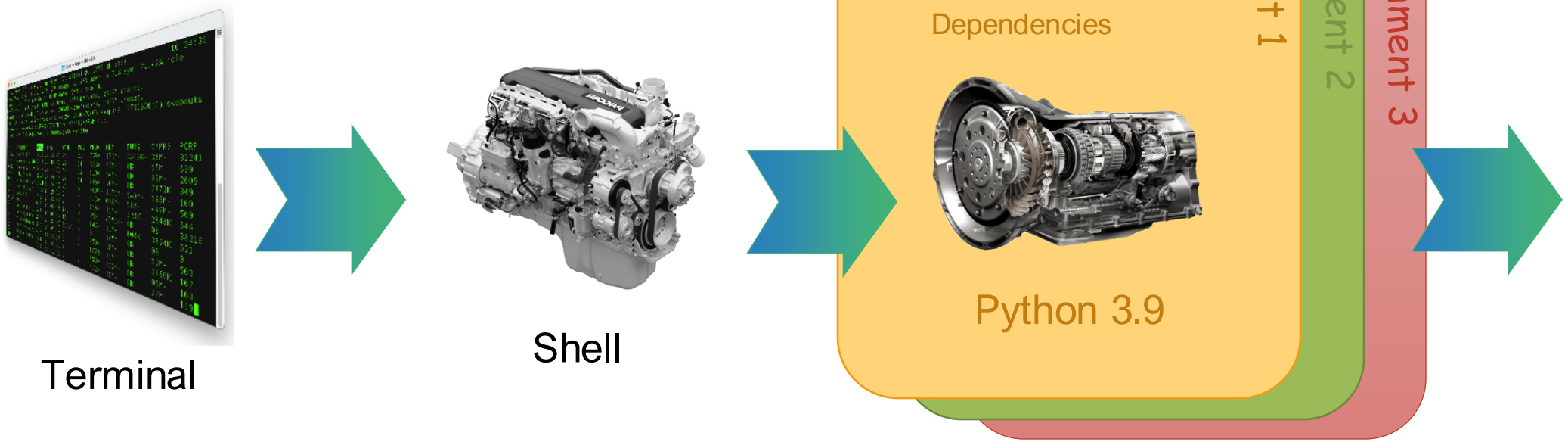
uv

The newest environment and package manager on the block. Fast, modern, and all-in-one, handles both setup and ingredient shopping in one streamlined process. Gaining popularity quickly.

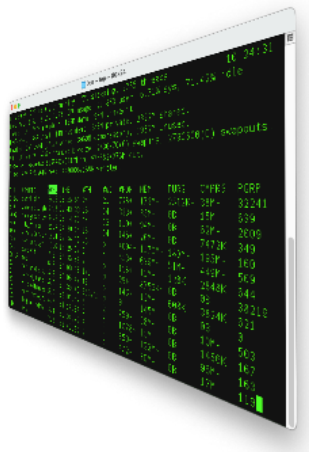
Same Kitchen, Different Menus

- At many fast food restaurants, the same restaurant serves breakfast in the morning and burgers in the afternoon, but the setup behind the counter is different for each.
- Conda helps manage this.
- It keeps track of what tools and ingredients (dependencies) are needed for each menu, like hash browns and eggs for breakfast, or buns and patties for lunch, and makes sure the right setup is ready before the crew starts working.
- Same restaurant. Different projects. Each needs its own environment. Conda handles the switch behind the scenes.

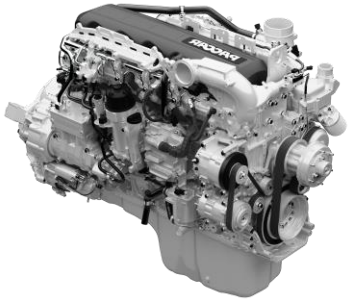
So in the world of computers:



So in the world of computers:

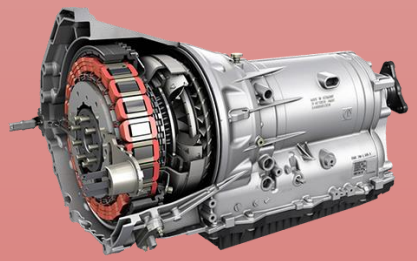


Terminal



Shell

Dependencies



Python 3.11

Environment 3

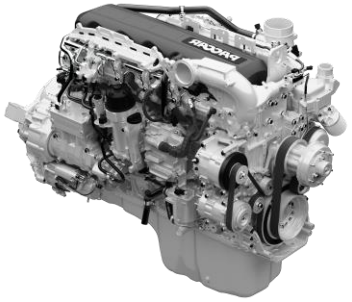
Environment 2

Environment 1

So in t
compu

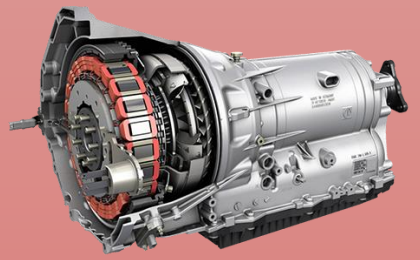
The kitchen manager switches from one environment to another, setting up the right Python version and dependencies for each project.

Terminal



Shell

Dependencies



Python 3.11

Environment 3

Environment 2

Environment 1

So in t
com

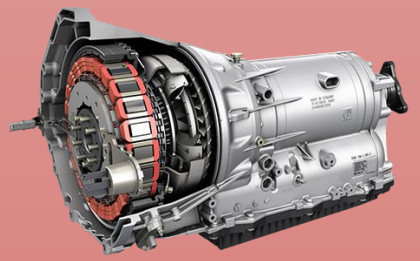
The kitchen manager switches
o
ght
ject.

This way, we can quickly switch between environments and work on different projects without hassle.

Terminal

Shell

Dependencies



Python 3.11

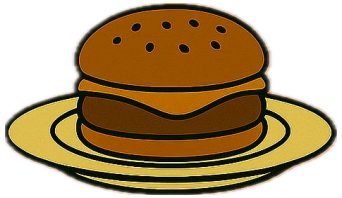
Environment 3

Environment 2

Environment 1

From Burger Fan to Burger Scientist

- With the system we have (terminal, shell, interpreter) you can absolutely make things. You can cook, code, and get results. And that's great!
- But… the difference between an average burger fan and a pro?
- The pro takes it seriously. They experiment! Two pickles today, one tomorrow. Less cheese. Try a chicken patty and a beef patty?
- They keep track of their experiments.
- And most importantly, they use a Flip Pad called *Git* to write it all down.

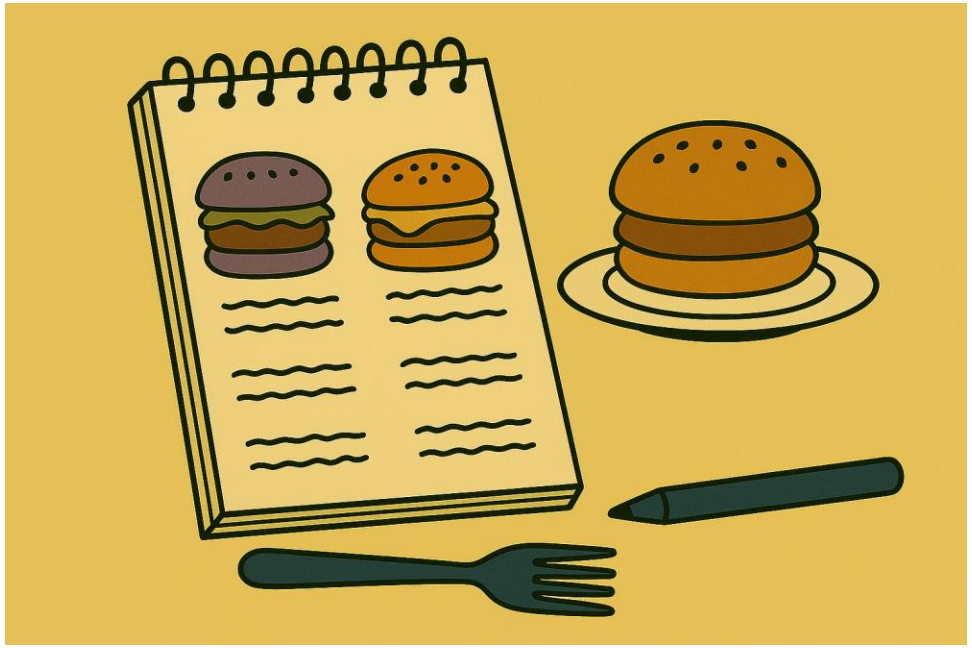


How an average burger fan eats

How a pro burger fan eats

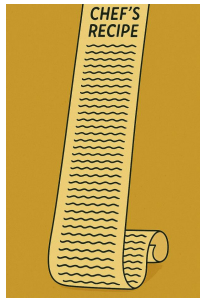
Git

- Git isn’t part of the kitchen, it doesn’t cook, run commands, or process your orders. Git is your Flip Pad, but smarter.
- It keeps a detailed, time-stamped record of everything you’ve tried:
  - Every version of your recipe
  - Every change and experiment
  - What worked, what didn’t, and when
- You can flip back, compare past ideas, and start something new (without losing what you had).
- Git helps you build with confidence, because it never forgets a single version.



So:

Flip Pad



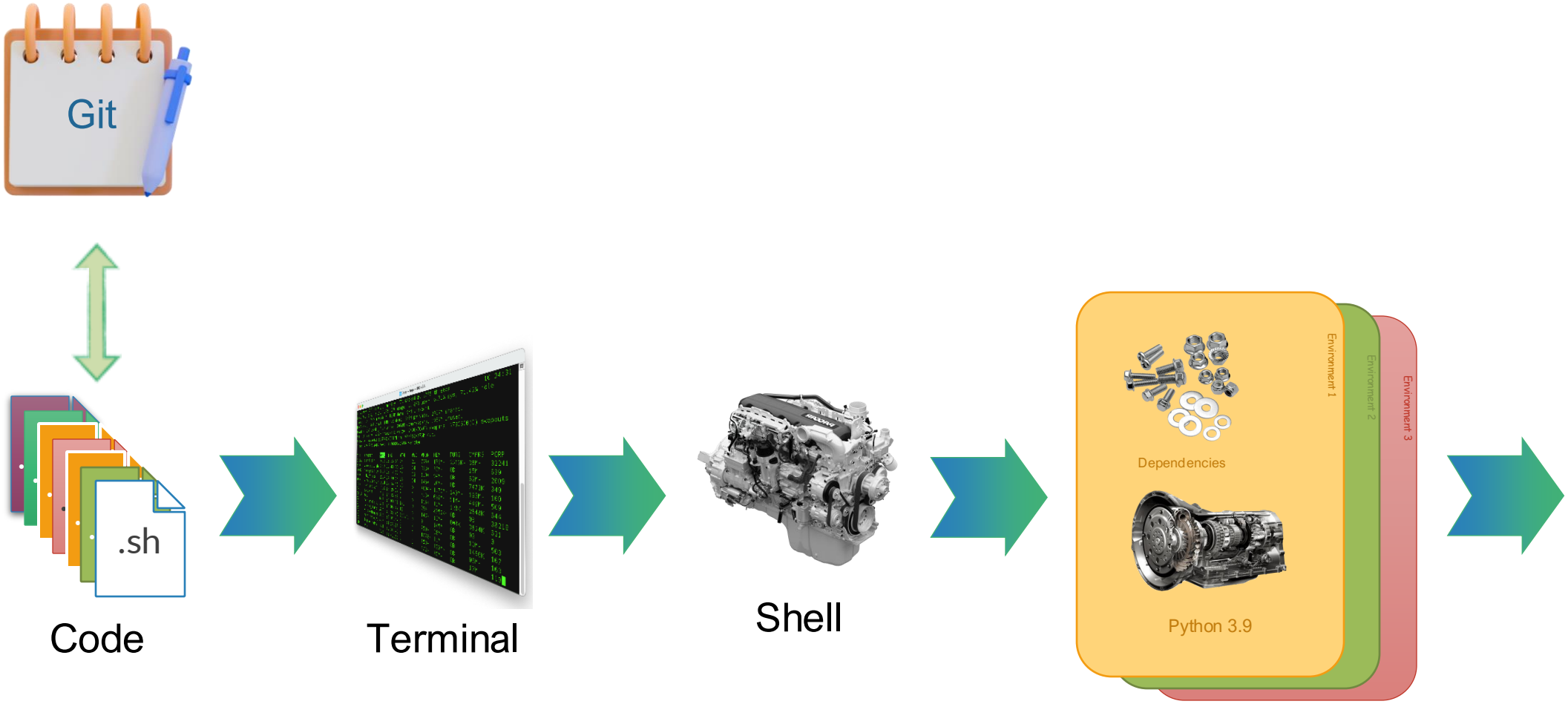
Ordering

Kiosk

Runner

Kitchen Manager
& Chef

So:

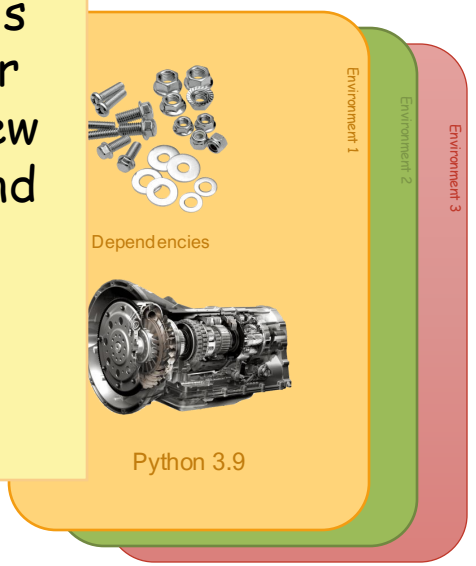

Git
.sh
Code
Terminal
Shell
Dependencies
Python 3.9
Environment 1
Environment 2
Environment 3

So:

Code

Of course, this is a very simplified version of how things work, but we're explaining it this way to help you build a clear picture in your mind. As you keep learning over the next few weeks, you'll start to refine that picture and understand each part in more depth.

Terminal



Now That I *Kind of* Understand…
What Now?

Setting up your Computer!

- First, take a quick self-test to check your understanding so far, helps confirm that you're ready for what's next:
  - *uoft-dsi.github.io/onboarding/interactive_problems.html#onboarding_test*
- Then, follow our step-by-step guide to install everything you'll need for the certificate: Git, VS Code, Conda, the DSI Conda Environment, and a few small tools.
- You'll be using these tools for the next 4 months, so getting set up now is key.
- You can find the full setup guide here:
  - *github.com/UofT-DSI/onboarding/blob/main/environment_setup/*