

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, UV, 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



```
Dan — top — 80x24
Processes: 773 total, 2 running, 771 sleeping, 3755 threads    10:24:31
Load Avg: 2.62, 2.11, 2.62  CPU usage: 21.85% user, 6.71% sys, 71.42% idle
SharedLibs: 747M resident, 133M data, 95M linkedit.
MemRegions: 305 total, 14M resident, 380M private, 2332M shared.
PhysMem: 15G used (2617M wired, 3960M compressed), 1952M unused
VM: 311T vsz, 5709M framework vsz, 2436776(0) swapins, 2790500(0) swapouts.
Networks: packets: 5159981/5637M in, 975882/276M out.
Disks: 8784935/261G read, 4480006/168G written.

#PID  COMMAND           %CPU  TIME    #TH   #WQ  #PORT  MEM      PURG    CMPRS  PGRP
32241 Spotlight          94.5  00:55.61 24    21    728+   173M+   2512K-  38M-   32241
639   corespotlight     62.8  02:43.33 16     14    333+   33M+    0B      15M-   639
2009   managedcores      24.6  00:14.73 14     12    110+   94M+    0B      62M-   2009
349   mds_stores        22.2  04:20.15 16     14    146+   26M+    0B      7472K  349
160   WindowServer      16.4  02:34:12 26     6     4004+  1170M+  183M-  195M-   160
509   mediaanalysis     10.6  10:25.97 5      4     413+   639M+   51M-   449M-   509
644   parsecd           7.6   00:36.69 7      6     118+   11M+    128K    2848K  644
38219 top                 6.7   00:00.38 1/1    0     29+    8753K+  0B      0B      38219
621   com.apple.qu...   5.8   00:06.47 10     8     145+   12M+    608K    3824K  621
0     kernel_task       5.5   81:39.45 663/11 0      0      145M    0B      0B      0
503   WindowManage      5.0   06:01.47 5      2     750+   38M+    0B      10M-   503
167   runningboard      4.7   04:19.87 7      6     1078+  11M     0B      1456K  167
163   loginwindow       3.4   01:04.72 7      3     859+   123M+   0B      96M-   163
119   mds                 2.6   02:42.92 9      6     402-   30M-    0B      17M    119
```

```
Desktop — -zsh — 120x40
Dan@OS139JMM4D ~ % ls
Applications      Music              python_basics.ipynb
Desktop           new_new            python_basics.py
Documents         new_hard_deadlines.gs  scikit_learn_data
Downloads         nltk_data          shell
git-sia_media     OneDrive - University of Toronto  test.sh
hard_deadlines.gs outline.ipynb       unique_lines.txt
Library           Pictures            Virtual Machines.localized
Movies           Public

Dan@OS139JMM4D ~ % cd Desktop
Dan@OS139JMM4D Desktop % ls
Dan@OS139JMM4D Desktop % ps -ef | grep "test"
502 37772 37752  0 10:23AM ttys000  0:00.01 grep test
Dan@OS139JMM4D Desktop % ls -al
total 16
drwx-----@ 3 Dan  staff   96 May 28 09:55 .
drwxr-xr-x 62 Dan  staff  1984 May 28 18:22 ..
-rw-r--r--@ 1 Dan  staff  6148 May 28 18:02 .DS_Store
Dan@OS139JMM4D Desktop %
```

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

Terminal

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

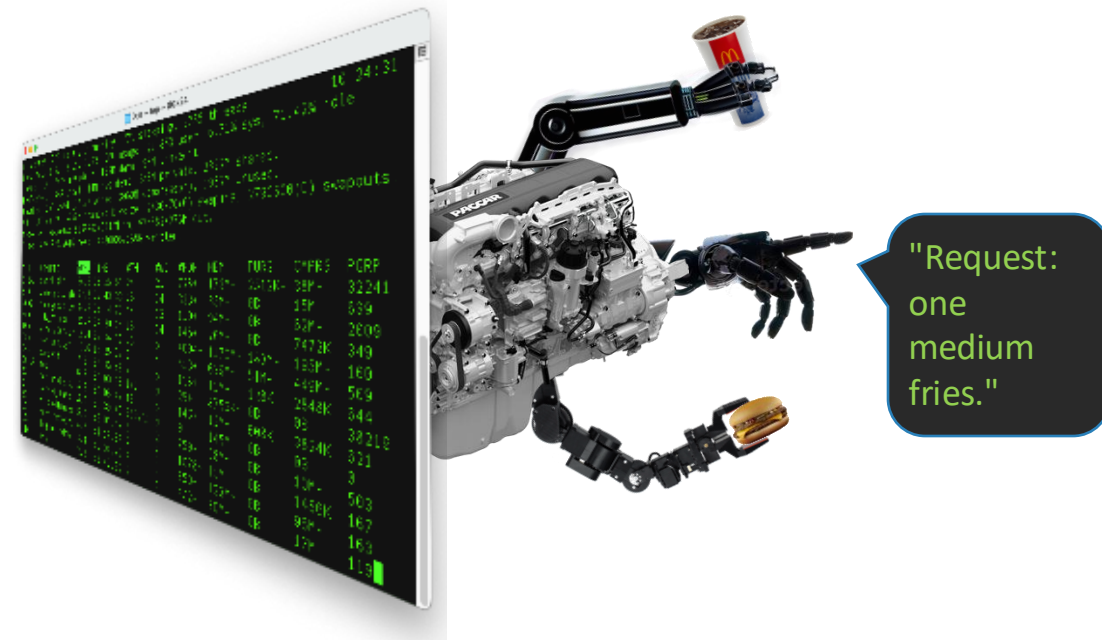
That engine you see? That's the runner in the computer world! That's the "Shell", hiding behind the terminal.

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

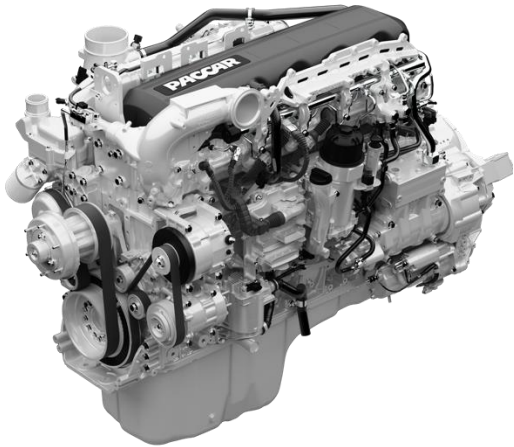
The Shell:

- 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

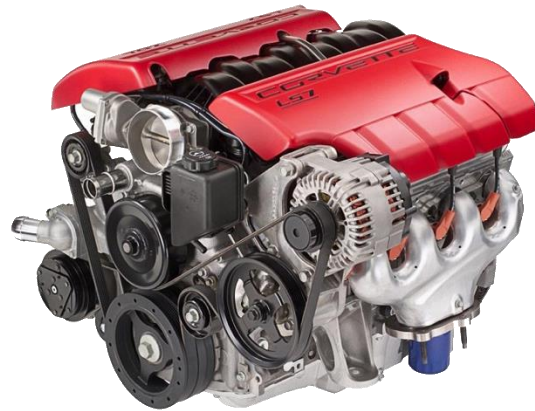
The Shell is your behind-the-scenes operator handling your requests line by line.



Different Shells



Bash



Zsh



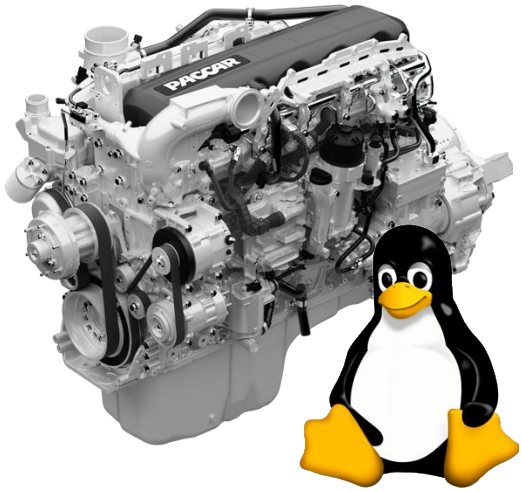
PowerShell



Command Prompt

Mainly found in:

Linux



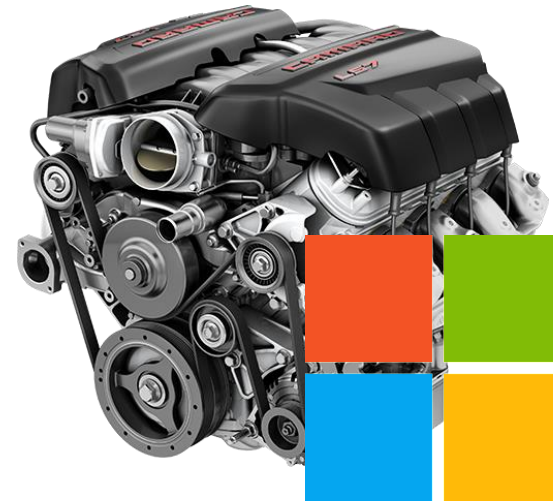
Bash

Mac OS



Zsh

Windows



PowerShell

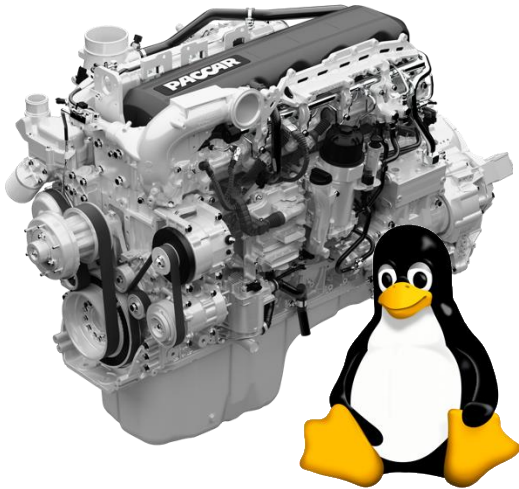
Windows



Command Prompt

Mainly found in:

Linux



Bash

The Most Popular (What we're learning)

Mac OS



Very Similar to Bash

Zsh

Windows



Not that fun to use :(

BUT in this tutorial we will teach you how to set up Windows to be more like Bash :)

PowerShell

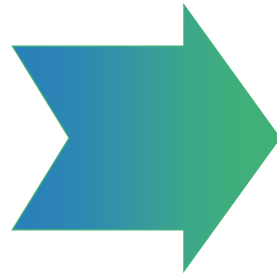
Command Prompt

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.



"10 different Fries with different sizes and 10 different drinks"



"A custom burger with no ketchup, double patty, extra pickles"



"I want Just a drink"

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.




*“10 different Fries with
different sizes and 10
different drinks”*

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.



"A custom burger
with no ketchup,
double patty,
extra pickles"

Different Chefs

Different chefs speak different languages:



Speaks
Python



Speaks
JavaScript

Some are different kind of chefs:



Speaks
Java

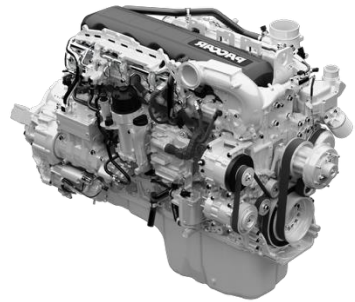
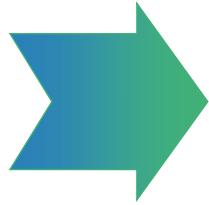
Python is the
programming language
we'll be using
throughout the DSI
Certificates

But the basic idea is the same:
you give instructions → the chef brings them to life.

So:



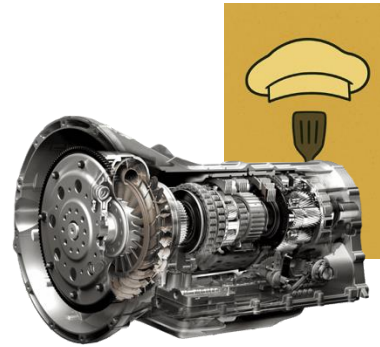
A shell script that in it also wants you to call a python script.



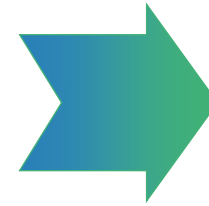
Shell



Performs the
Shell tasks



*Python
Interpreter*



Performs the
Python tasks

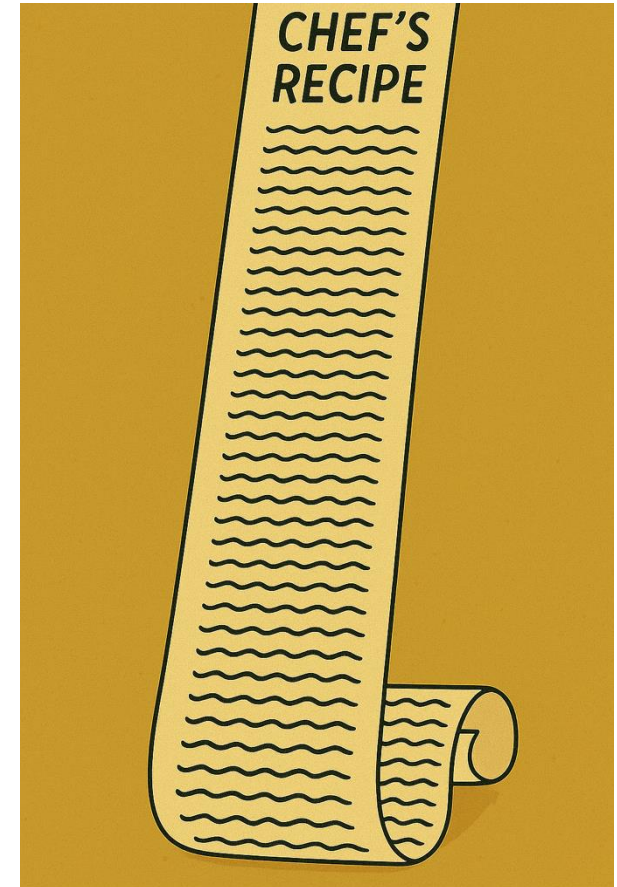
One Last Kind of Order

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 3.11, 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 UV, venv, pip, and more. Let's talk options...



Options

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

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.

UV



The newest tool on the block. Fast, modern, and all-in-one, it creates the cooking station (environment) and brings the ingredients (packages) in one simple step. It follows the same ideas as venv and pip but does them faster and more smoothly. Gaining popularity quickly.

Different Projects need Different Tools

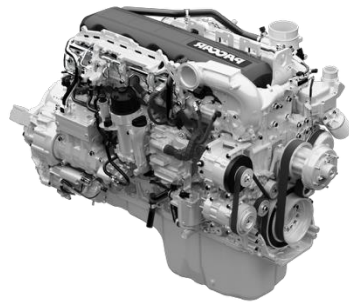
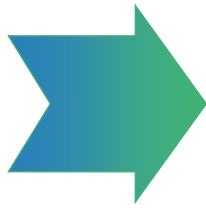
- Different fast-food restaurants need different kitchen appliances and different ingredients.
- Like a kitchen manager, UV helps manage this.
- It keeps track of what tools (packages) and ingredients (dependencies) are needed and makes sure the right setup is ready before the chef starts working.
- Same Computer. Different projects. Each needs its own environment. UV handles the switch behind the scenes.



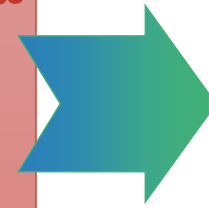
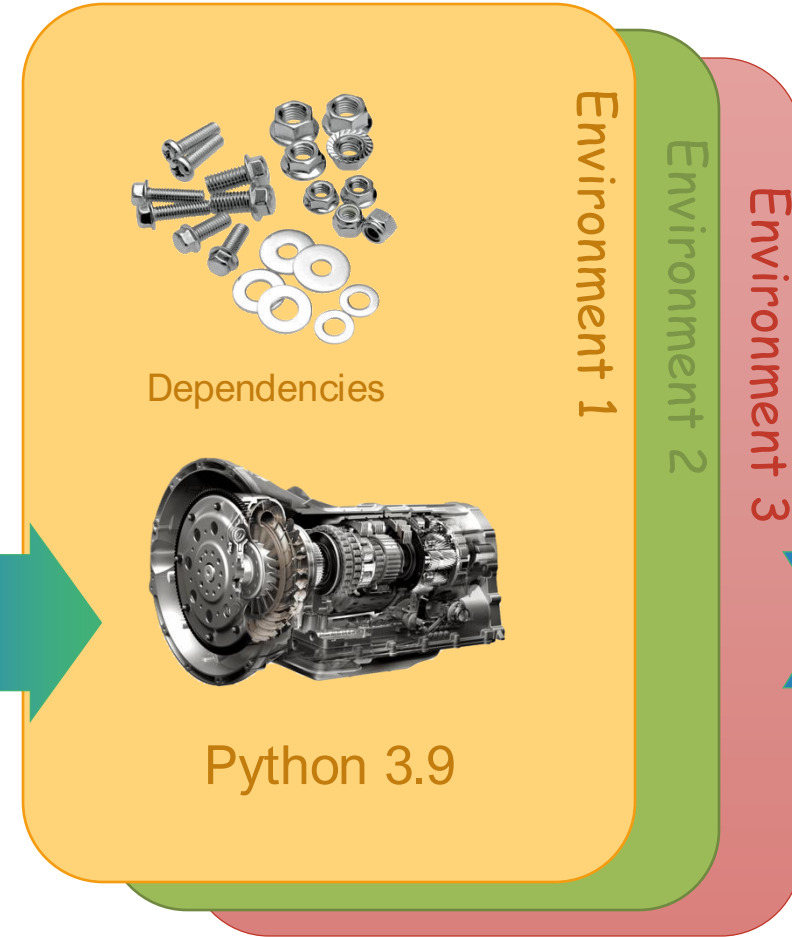
So in the world of computers:



Terminal

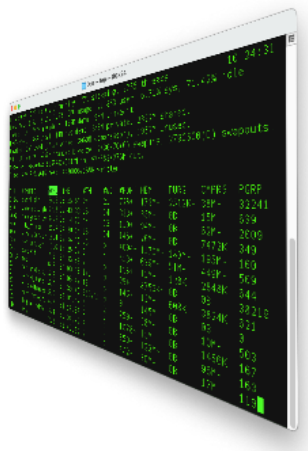


Shell

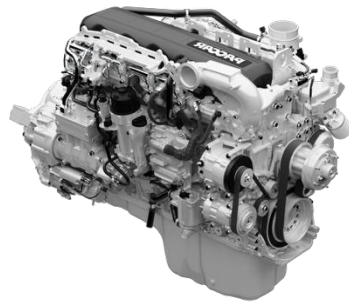
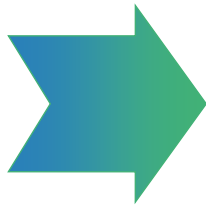


So in t compu

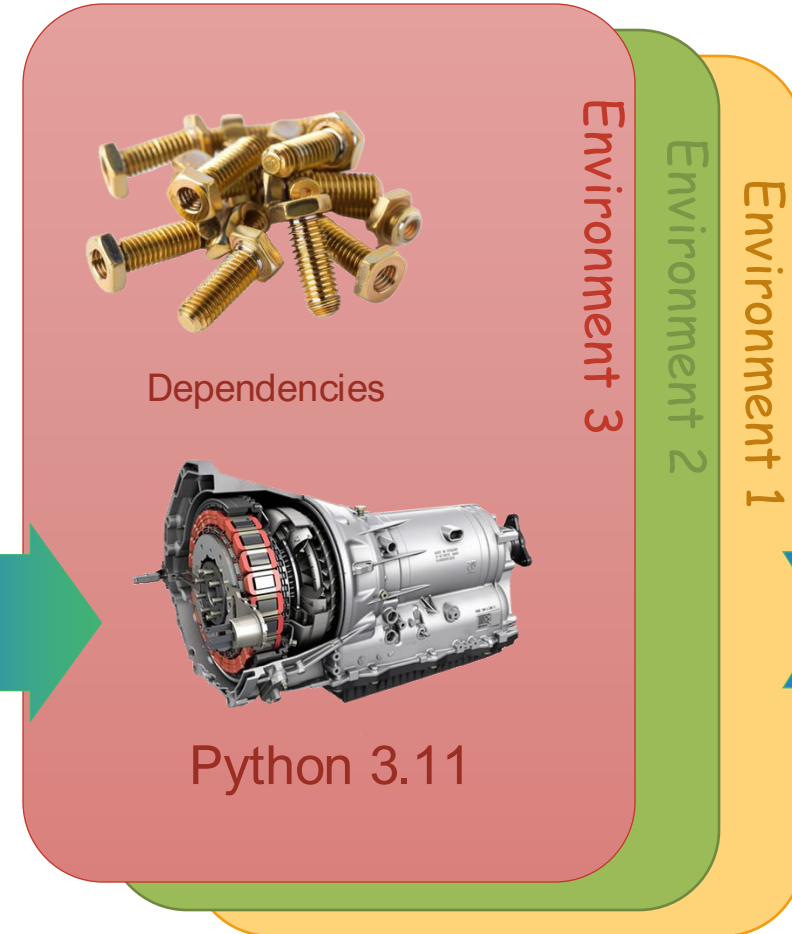
This way, we can quickly switch
between environments and
work on different projects
without hassle.



Terminal

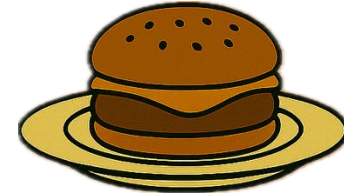


Shell



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

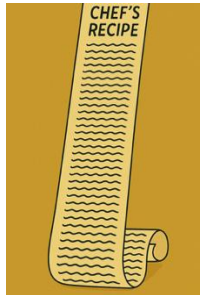
Git

- 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.

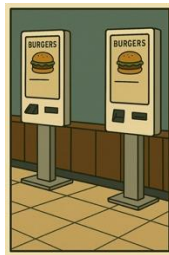


So:

Flip Pad



Ordering



Kiosk



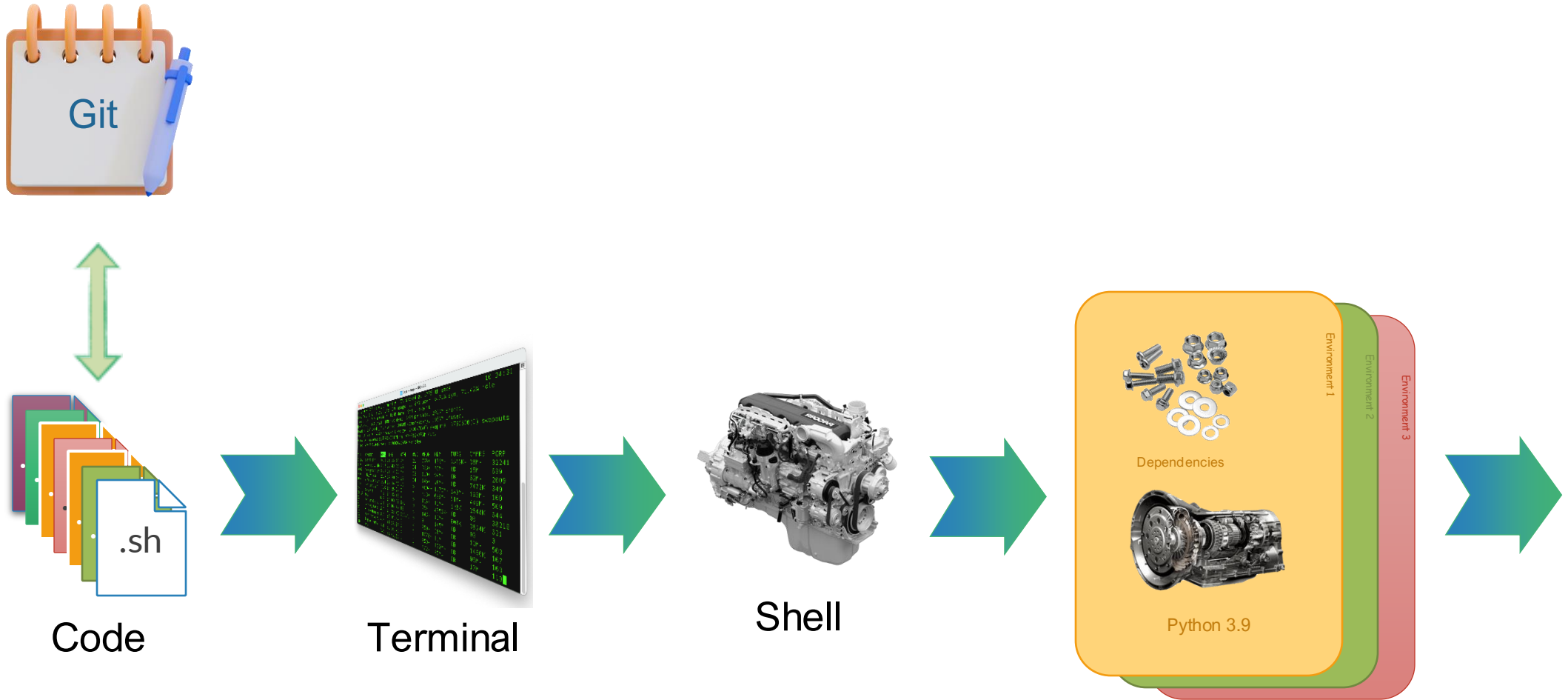
Runner



Kitchen Manager
& Chef



So:



So:

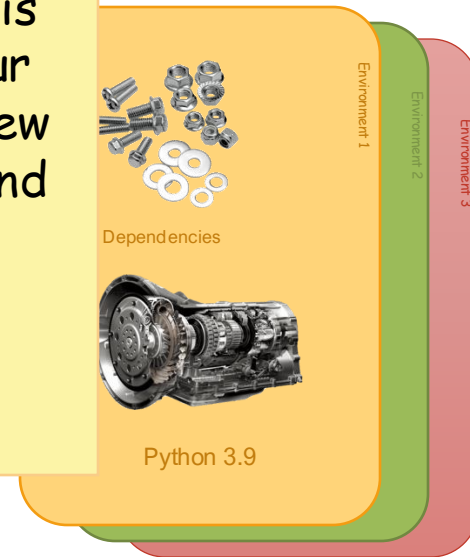


Code



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:
 - https://github.com/UofT-DSI/onboarding/blob/main/environment_setup/tech_onboarding_test.md
- Then, follow our step-by-step guide to install everything you'll need for the certificate: Git, VS Code, and UV. You'll be using these tools for the next little while, so getting set up now is key.
- You can find Git, VS Code and UV setup guide here:
 - https://github.com/UofT-DSI/onboarding/blob/main/environment_setup/
- For repo specific instructions (packages, activation, etc.), always refer to that module's [SETUP.md](#) file.

