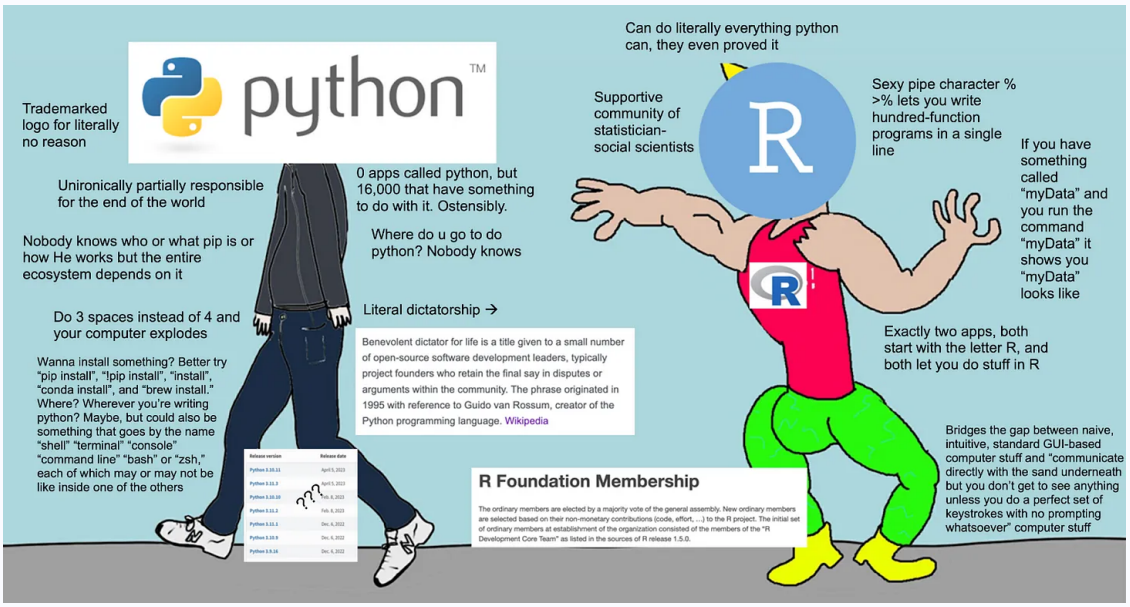
"Tf is a kernel?"

Professional developer Agus Covarrubias tries to untangle my brain

AARON BERGMAN
APR 30, 2023

Intro

R has a special place in the nerdiest corner of heart. And Python—for reasons that are beginning to seem more like real features of the language ecosystem and less like figments of my imagination—does not.



My meme elicited strong emotions on Twitter

Unfortunately for me and my brethRen, the relative abundance of python code on the internet from which pseudo-proto-AGI-chatbots like chatGPT can learn—and thus later emulate (for lack of a better word)—has made me wary of leaning too hard on my not-really-all-that-strong aesthetic preference.

That's my legible-sounding reason for trying (again) to learn python, anyway. My gradual discovery that every ~serious~ software engineer regards R as an civil engineer might regard legos (*yes, you* *could* *build the bridge out of legos but, like, c'mon man*) pushes in the same direction, as does the fact that learning virtually any technical, digital skill is way, way, easier now than it was a year ago before the launch of chatGPT and its crazy younger brother.

"Easier" doesn't mean "easy," though, at least for me and at least when it comes to what I'd call the *infrastructure* surrounding the literal words and symbols that chatGPT so competently spits out, all neat and tidy in an aesthetic little black code box with a copy button bow tied on top.

```python
from pytube import YouTube
from moviepy.editor import *
import os

# Define function to download and convert video to mp3
def convert_to_mp3(url):
    # Download YouTube video
    yt = YouTube(url)
    stream = yt.streams.filter(only_audio=True).first()
    stream.download()

    # Convert video to mp3
    video_path = f"{yt.title}.mp4"
    audio_path = f"{yt.title}.mp3"
    video = None

    try:
        video = VideoFileClip(video_path)
        audio = video.audio
        audio.write_audiofile(audio_path)
    except Exception as e:
        print(f"Error converting {yt.title} to MP3: {str(e)}")
    finally:
        # Close video and audio streams if they were opened
        if video:
            video.close()
        if audio:
            audio.close()

    # Delete original video file
    os.remove(video_path)
```

Screengrab from the giving chatGPT the arbitrary simple prompt “python to download audio from youtube.” If you haven’t, try for yourself

Describing what I mean by this could be a lengthy post on its own, but right now, I just want to share a conversation that I suspect might be helpful for anyone else stuck in (or, more likely, ‘moving weirdly slow through’) similar terrain: a sort of pre-actual-programming purgatory. Because, despite doing everything right, *nothing goddamn works*.

So without further ado, a lightly edited conversation with professional developer and fellow member of EA Twitter, Agus Covarrubias (Twitter):

Conversation

Agus (or “**a**stute”)**:** [...] feel free to ask any questions!

Me (or ‘**m**istaken’): About python in general: any reason I shouldn’t install the 100 or 300 or 1000 most common libraries just so I never have to think about it again? Like why isn’t this standard?

A: It isn't standard because dependencies can conflict between them, and you need to keep dependencies updated. You're better off just learning to use a modern tool for managing python environments, like poetry (or conda, if that's your tune)

M: Yea so just import the right ones right? And no way I’m not adding any more complexity, nothing ever works as is

A: basically, my workflow (and the workflow of most python developers) is that when you create a project, you create a new environment. You just add the dependencies to the environment as you need them. So in my case, I just poetry add <dependency> whenever I need something. For example, for this project, Poetry auto-defined these dependencies:

```
[tool.poetry.dependencies]
python = "<3.12,⩾3.10"
squigglepy = "^0.24"
numpy = "^1.24.2"
ipykernel = "^6.22.0"
pandas = "^2.0.0"
matplotlib = "^3.7.1"
seaborn = "^0.12.2"
scipy = "^1.10.1"
```

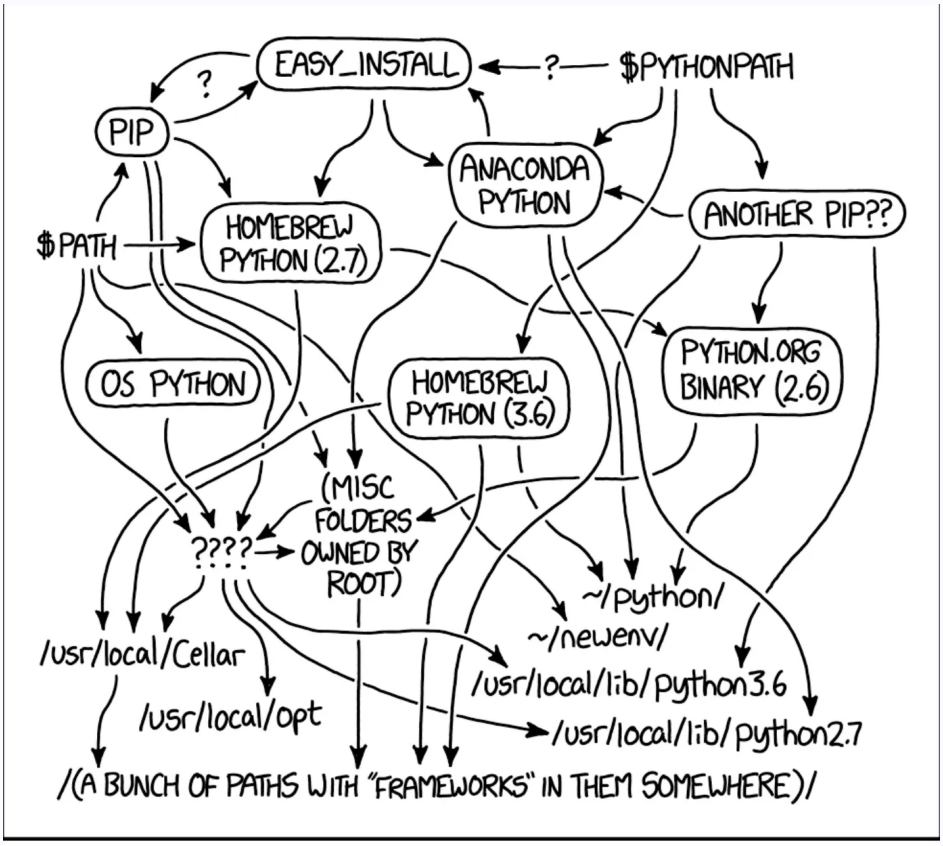
By doing this (having a defined environment) you avoid a huge amount of headaches associated with conflicting dependencies and versioning. It's a habit I've built by force

M: pip install *

A: If you do pip install all the time, then you're going to last a couple of months before you have to reinstall python 😭

M: Wait how else do u get the package?

A: Using Poetry or Conda!



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

You can also use pip if you're using something like venv, but it's hard to use

M: Poetry is just an English noun to me and conda is just for when pip install is weird for some reason but idk what it is in the slightest. And tf is brew?

A: ok so...

M: Plan A is pip, plan B is brew, plan C is conda, plan D is give up

Plan E is a weird long zsh command starting with sudo that chatGPT cooked up for me

A: You have general-purpose package managers like brew and port (in macOS), and then you have python package managers. The default Python package manager is pip (which stands for pip install packages✨) which is pretty good if you want to install things globally, that is to your entire computer

However, whenever you actually need to build things and share your work with other people, most python developers don't install things to the global environment. But rather, create an environment (a collection of packages) for a particular project, and then install all the packages they need to that project environment

Traditionally, that used to be done with tools like venv + pip, but nowadays, there are modern tools like Poetry or Conda, which try to make your life easier and handle all the hard stuff for you. Poetry is my favorite, because you only need to learn two commands, and once you start using it you can make everything from disposable projects to publishable python libraries with it

M: Fuck no pip cannot stand for pip install packages unless we're getting theological

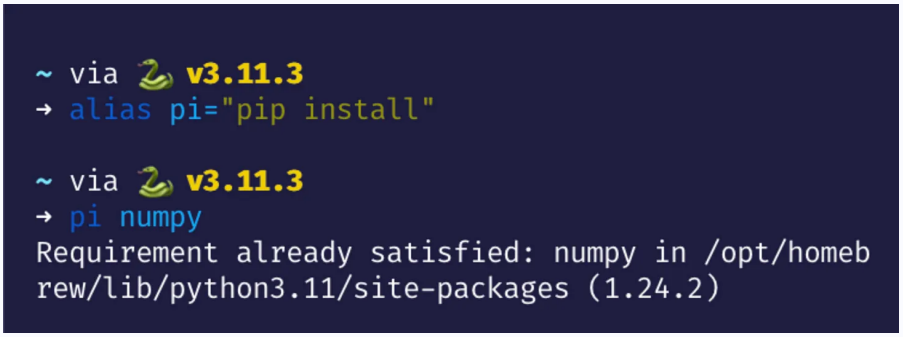
A: it actually does

M: ok what did the first pip stand for

A: it's a recursive backronym. The first pip didn't stand for anything, which is why it's a backronym, but the backronym is recursive

M: Ok fine that does make sense. But then the command should just be "pi" not "pip install." Who do I submit a complaint to

A: I have great news for u:



M: Tf is via

Tf is the snake doing?

A: JDFKGKJDFG

I've tweaked my terminal so it shows the activated versions of different programming languages

M: ah. What's a terminal? I'm only slightly joking

A: the thing that you type commands to. like, i'm pretty sure that's close to the technical definition

M: Ok then what about shell command line and console

A: ok so that is where it gets messy

M: Yeah I've gathered, they're all the same 😭

A: basically, the terminal used to be the physical interface which you used to interact with a computer. Nowadays terminal really means terminal emulator, which is a program that emulates that textual interface. A console is another word for virtual terminal. And then shell is the program that answers what you write, or rather, the program that coordinates how to answer whatever you write

M: Oy vey. What language is it when I'm talking to the thing underneath bash? called "base" I think

A: There's nothing underneath bash. bash is bash

M: Fake news. What if I uninstall bin/bash and then open the terminal app

A: depending on what computer you have you'll either ruin your computer or nothing will happen

M: Lmao

A: If you have an up to date mac, you probably think you're using bash when you're actually not. you're probably using something called zsh which is made to look like bash while being actually better

bash will still be installed for compatibility reasons, but AFAIK most apps no longer use it

if you want to know what you're using you can type echo $SHELL on your terminal

M: Does everything that happens on (in?) my literal MacBook Pro a zsh command? And for any set of things that happens is there a zsh command that would do the same thing - like some crazy long nonsense that would open Word, move the window around the monitor, and write my name in comic sans?

A: In some very specific Linux computers, kinda, but most modern computers have something called an init process that actually orchestrates opening the most important programs (things like the window manager), which can spawn other programs, like apps, on their own.

M: ok so I just opened the textedit app by moving and clicking my cursor. What's at the bottom of that if not zsh? some sort of init thing actually opens the app I assume

A: you click the cursor over the textedit icon, the mouse click is received by the kernel

M: tf is a kernel

A: Basically the program that helps software interact with the physical input/output components in your computer like keyboard, mouses and displays

M: might fuck around and get a cs degree via this chat

A: anyway, after the kernel has started, the window manager can say “hey, the user clicked over a program called the dock, let's let the dock know that they have been clicked!” Then the dock gets the click and says “hey, the click is over the textedit app. And then the window manager tells the kernel, "hey, the user requested opening this app, please do it yourself" and then the program gets opened -

M: no no no no anthropomorphizing I already know all that, well kinda

A: - zsh doesn't intervene because every step in that communication between processes happens through something called System Calls, which is how processes and the kernel can communicate between each other, by telling the CPU to interrupt normal program execution for a sec so the kernel can get summoned for some clock cycles and can do what the system calls mandate

So every time processes need to communicate, they tell the CPU to stop normal execution for a second and pass an instruction (alongside some information) to the kernel, the supervising process for the entire system

Bash is just a process which makes doing these system calls very easy. Like you just type some textual commands and bash decides what system calls to do based on that info -

M: Wait stop. This is getting out of hand. I need to read the book you just wrote and lots of wikipedia or else its just questions all the way down. I feel like ur answers/at least parts of this convo are worth publishing

A: I think these kinds of things have been collected in many places, but you had some very particular questions in sequence

M: Yeah but people hate reading textbooks, they like reading convos

Comments

Write a comment...