Questions

1. The image is itself derived from buildpack-deps:bookworm. What is this derived from? Can you trace and list all of the layers used in the container?

To trace the layers used in the container I searched for the FROM … line (first line) in the Dockerfiles and then searched for such Dockerfile to get the next layer. Here is a Linux tree-like (Linux tree command) structure of the layers used in the container.

```
python:3.13 (Dockerfile)
|-- buildpack-deps:bookworm (Dockerfile)
    |-- buildpack-deps:bookworm-scm (Dockerfile)
        |-- buildpack-deps:bookworm-curl (Dockerfile)
            |-- debian:bookworm
```

2. Given that the container image is based on Debian, why do you think the Dockerfile is downloading the source code on line 28 of the Dockerfile? Why not just install the Debian package with apt?

There are several reasons why the source code is being downloaded:

- Having access to the source code enables developers to make modifications to the code.

- Debian's package manager (apt) provides a stable version of Python but most of the time this are outdated versions, so if you want a specific or the latest version you have to download the source code.

- Using apt, the Python version could vary depending on when the image is built.

3. Docker will build the Python interpreter when it builds the image. Python is built using make. Can you see the make invocation(s) in the Dockerfile? What line(s) does it appear on?

make is invoked in several lines: line 54, line 61 and line 66. Two of them are for compilation and one for installation.

make -j … is used for compilation in lines 54 and 61, and it performs parallel compilation using all the available CPU cores (number obtained from nproc).

make install is used to call the install target on the Makefile, used to install python into the system (always done after compilation).