Using Python on Android

- Adding Python to an Android app
- Building a Python package for Android

Python on Android is unlike Python on desktop platforms. On a desktop platform, Python is generally installed as a system resource that can be used by any user of that computer. Users then interact with Python by running a `python` executable and entering commands at an interactive prompt, or by running a Python script.

On Android, there is no concept of installing as a system resource. The only unit of software distribution is an "app". There is also no console where you could run a `python` executable, or interact with a Python REPL.

As a result, the only way you can use Python on Android is in embedded mode – that is, by writing a native Android application, embedding a Python interpreter using `libpython`, and invoking Python code using the `Python embedding API <embedding>`. The full Python interpreter, the standard library, and all your Python code is then packaged into your app for its own private use.

The Python standard library has some notable omissions and restrictions on Android. See the `API availability guide <mobile-availability>` for details.

Adding Python to an Android app

Most app developers should use one of the following tools, which will provide a much easier experience:

- Briefcase, from the BeeWare project
- Buildozer, from the Kivy project
- Chaquopy
- Flet
- pyqtdeploy
- Termux

If you're sure you want to do all of this manually, read on. You can use the `testbed app <Platforms/Android/testbed>` as a guide; each step below contains a link to the relevant file.

- First, acquire a build of Python for Android:
  - The easiest way is to download an Android release from python.org. The `prefix` directory mentioned below is at the top level of the package.
  - Or if you want to build it yourself, follow the instructions in `Platforms/Android/README.md`. The `prefix` directory will be created under `cross-build/{HOST}`.
- Add code to your `build.gradle <Platforms/Android/testbed/app/build.gradle.kts>` file to copy the following items into your project. All except your own Python code can be copied from `prefix/lib`:
  - In your JNI libraries:
    - `libpython*.*.so`
    - `lib*_python.so` (external libraries such as OpenSSL)
  - In your assets:
    - `python*.*` (the Python standard library)
    - `python*.*/site-packages` (your own Python code)
- Add code to your app to `extract the assets to the filesystem <Platforms/Android/testbed/app/src/main/java/org/python/testbed/MainActivity.kt>`.
- Add code to your app to `start Python in embedded mode <Platforms/Android/testbed/app/src/main/c/main_activity.c>`. This will need to be C code called via JNI.

Building a Python package for Android

Python packages can be built for Android as wheels and released on PyPI. The recommended tool for doing this is cibuildwheel, which automates all the details of setting up a cross-compilation environment, building the wheel, and testing it on an emulator.

□□□ #2
□ □□□ □□□ 2026-10-31 17:27:26 UTC
□ □□□ □□□ 2026-10-31 17:27:27 UTC