Main

Starlarkify Python flags

Please read Bazel Code of Conduct before commenting.

- **Authors**: Greg Estren (gregce@bazel.build)

- **Status**: **Implementing** (Sep 8, 2025)
- **Reviewers**:
  - Richard Levasseur (rules_python maintainer)
- **Created**: Sep 8, 2025
- **Updated**: Oct 23, 2025
- **Tracking issue**: https://github.com/bazel-contrib/rules_python/issues/3252

Overview

Replace all Python language flags built into Bazel with Starlark definitions owned by rule owners.

Python subtask of Starlarkify native Bazel flags.

This is an internal Bazel & rules_python cleanup change. It's mostly a no-op for end users. The goal is to give Python rule owners full control over Bazel's Python support.

"bazel_py" flags

These Python flags are defined in BazelPythonConfiguration.java.

Goal: delete `ctx.fragments.bazel_py` and supporting Bazel code.

Sources: BazelPythonConfiguration.java, builtin_exec_platforms.bzl

flag	type	default	references	exec value	decision
--python_top	Label	null	Starlark API	target value	Starlarkify
--python_path	String	null	Starlark API	target value	Starlarkify
--experimental_python_import_all_repositories	bool	true	Starlark API	target value	Starlarkify

Other builtin logic:

- BazelPyBuiltins.java: nothing flag related
- BazelPythonConfiguration.java

    - Fails build if `--python_path` isn't absolute. Move to Python `.bzl` logic. Challenge: OS-specific absolute path differences.
    - Fails build if `--python_top` is set with `--incompatible_use_python_toolchains`. Remove check and remove ability to not use toolchains.

Summary:

- Remove `--incompatible_use_python_toolchain`
- Starlarkify everything else.

"py" flags

These Python flags are defined in PythonOptions.java.

Goal: delete `ctx.fragments.py` and supporting Bazel code.

Sources: PythonOptions.java, builtin_exec_platforms.bzl

flag	type	default	references	exec value	decision
--build_python_zip	TriState	`auto`	Starlark reference	target value	Different default for Windows host machines. Use this or this or this?
--incompatible_remove_old_python_version_api	bool	true blazerc=true	none	target value	No-op since 2020. Remove.
--incompatible_allow_python_version_transitions	bool	true blazerc=true	none	target value	No-op since 2020. Remove.
--incompatible_py3_is_default	bool	true	trivial Bazel ref	target value	PY2-related. Remove.
--incompatible_python_disable_py2	bool	true blazerc=true	outdated Starlark ref?	target value	PY2-related. Remove.
--incompatible_py2_outputs_are_suffixed	bool	true	old Bazel ref	target value	PY2-related. Remove.
--python_version	Python Version	null	old Bazel refs	depends on PY2 vs. PY3	PY2-related. Remove.
--force_python	Python Version	null	old Bazel refs	**default**	PY2-related. Remove.
--host_force_python	Python Version	blazerc=PY3	old Bazel refs	target value	PY2-related. Remove.
--incompatible_disable_legacy_py_provider	bool	true blazerc=true	none	**default**	No-op since 2022. Remove, but check Bazel incompatible tracking bug status.
--incompatible_use_python_toolchains	bool	true	none	target value	Delete since flipped 2020.
--incompatible_default_to_explicit_init_py	bool	false	`default_to_explicit_init_p`	target value	Starlarkify

			`y`		
--python_native_rules_allowlist	Label	null blazerc set	`native_rules_allowlist`	target value	Starlarkify
--incompatible_python_disallow_native_rules	bool	false blazerc=true	`disallow_native_rules`	target value	Remove? Outdated?
--experimental_py_binaries_include_label	bool	false	`include_label_in_linkstamp`	target value	Starlarkify
--experimental_build_transitive_python_runfiles	bool	false blazerc=false	None	default	Graveyarded 2023. Delete.

Summary:

- Remove outdated no-op flags.
- Replace Tristate `--python_preload_swigdeps` with pure bool if possible.
- Remove PY2/3 checking logic if possible.
- Support `--host_*` semantics with pending Starlark API or short-term hack.
- Move `-python_is_target_config_internal_only_do_not_use` somewhere more generic.
- Starlarkify everything else.

Progress

Destination code:

- rules_python pre-existing Starlark flags - these are unrelated to native flags
- rules_python ctx.fragments calls
- no bazelbuild/ `ctx.fragments.py` or `ctx.fragments.bazel_py` references
- rules_python transition logic - has some programmatic complexity

Migration steps:

- Completed Mention migration on rules_python, bazelbuild
- Completed Define equivalent Starlark flags
- Completed Update `rules_python` to support both native and Starlark flags. Builds still read native flags.
- Completed Define flag aliases in `rules_python`'s `MODULE.bazel` so `--python_*` syntax continues working
- Completed Test that transitions still work when using Starlark flags
- Completed Test `rules_python` & bazelbuild CI works when using Starlark flags
- Completed Update `rules_python` to read from Starlark flags. This requires Bazel 9 or newer. Older Bazel versions will continue to use native flags.
- Not Started Remove `ctx.fragments.bazel_py`
- Not Started Remove `ctx.fragments.py`
- In Progress Remove native flags. Bazel 10+ now *must* use Starlark flags.
- Not Started Remove `BazelPythonConfiguration.java`
- Not Started Remove `PythonOptions.java`

- Not Started Update documentation if needed
- Not Started Starlarkify java tests or retain or remove as-is
- Not Started Check what's still needed in `src/main/java/com/google/devtools/build/lib/rules/python`

Documentation

Python flags aren't particularly documented in Bazel's core docs but any such references will be moved to rules_python.

Bazel's core docs mirror Python rule definitions. Flags will also be documented here.

Compatibility

Since this is an internal cleanup between `rules_python` and Bazel, we're trying to make it as much of a no-op to users as possible.

But there will be some differences because Starlark and built-in Bazel flags aren't exactly the same:

- Starlark flags don't support `$ bazel build //foo --flagname flagvalue`. This must be `$ bazel build //foo --flagname=flagvalue`.
- Starlark flags don't work with `ctx.fragments.py` or `ctx.fragments.bazel_py` Starlark APIs. The purpose of those APIs is to expose built-in Bazel logic to Starlark. We consider it an improvement to eliminate them. But if anyone besides `rules_python` has Starlark code that reads them it will fail.
- `$ bazel build //foo --some_python_flag` will no longer work if both:
  - the user hasn't loaded `rules_python` into their workspace
  - we've removed the native flag definition from Bazel.

  In that case, Bazel will emit a "No such flag" error.

  This is a no-op for anyone building Python since Bazel's Python support already requires loading `rules_python`. But it can trigger if someone has, say, a `genrule` that `select()`s on `--some_python_flag`.

  Bazel 9 will deprecate, but not remove, native flags. So this can't be an issue until Bazel 10.

There's enough skew that we'll gate Starlark Python flags behind two Bazel 9 incompatible flags:

- `--incompatible_remove_ctx_py_fragment` (#27079)
- `--incompatible_remove_ctx_bazel_py_fragment` (#27080)

We expect users to address failures with reasonable user-side cleanups. For example, anyone reading `ctx.fragments.py` can read the Starlarkified flags just as `rules_python` does.

Bazel 8.x and lower will continue to work as they do now with any supported `rules_python` version.

Overall, we don't anticipate major disruption.

Document History

Date	Description
Sep 8, 2025	First proposal
Oct 23, 2025	Progress update, updated backward compatibility details

Testing

Testing Starlark Python flags

Addendum tab for Starlarkify Python Flags

Overview

Moving Python flags from Bazel to `rules_python`requires changes to the following:

- `rules_python` .bzl files
- `rules_python`'s `MODULE.bazel`
- core bazel code

This tab describes how to test changes by properly staging all needed modifications.

Manual testing

1. Clone https://github.com/bazelbuild/bazel. This is the main testing workspace.
2. In this workspace, build a custom bazel@head:

```
Shell
$ bazelisk build //src:bazel
$ mkdir b
$ cp -f bazel-bin/bazel b/bazeldev
```

3. Clone https://github.com/bazel-contribu/rules_python into another directory.
4. Go back to the bazelbuild/bazel workspace and open its `MODULE.bazel`.
5. Add the following to `MODULE.bazel`:

```
Shell
local_path_verride(module_name = "rules_python", path =
"/path/to/rules_python"
```

   This may require updating other `bazel_dep` dependencies in the main `MODULE.bazel` since `rules_python` at head is newer than the released `rules_python`. I had to update `bazel_skylib` from `1.7.1` to `1.8.1`.

6. Define a flag in `rules_python/python/cofig_setting/BUILD.bazel`.
7. Open `rules_python/python/private/flags.bzl` and change the second `_POSSIBLY_NATIVE_FLAGS` tuple entry from `"native"` to `"starlark"`.

This forces Python to read the Starlark version of that flag even when `--incompatible_remove_ctx_py_fragment` and `--incompatible_remove_ctx_bazel_py_fragment` are unset.

8. Back in `bazelbuild/bazel`, create a simple python binary:

```
Shell
$ cat testapp/BUILD
load("@rules_python//python:py_binary.bzl", "py_binary")

py_binary(
    name = "py",
    srcs = ["py.py"],
)

$ cat testapp/py.py
print("Hello world!")
```

9. Test that the binary works:

```
Shell
$ b/bazelde run //testapp:py
```

10. Run rules_python and bazelbuild tests:

```
Shell
$ cd ../rules_python
$ bazelisk test //...

$ cd ../bazel
$ bazelisk test //...
```

See also

- https://github.com/bazelbuild/bazel/pull/27015 - "add flag_alias function"
- https://rules-python.readthedocs.io/en/latest/devguide.html
- https://github.com/bazel-contrib/rules_python/issues/3252
- https://github.com/bazel-contrib/rules_python/issues/3252#issuecomment-3304829822:

  "FYI: I've added a CI job that uses bazel rolling. It won't shorten the iteration cycle, but you'll be able to more commit changes here with more confidence they work and won't

regress. If they do, we'll see it in typical PRs. The CI job is non-blocking, so we can still merge, but it'll be much more visible."