F-strings by default!

Eric V. Smith
eric@python.org

F-strings have been very popular, but …

- Added in Python 3.6.
- I believe they are one 3.x's "killer features", accelerating its adoption: I have several clients who cited them as a motivating reason to move to 3.6 (I'm of course biased).
- But, I still see errors where normal strings are assumed to be f-strings: "value is {value}" shows up in output every now and again. This usually happens to me while making wholesale code changes.
- So, let's make all strings f-strings by default! This was originally proposed when f-strings were first added, but it got shot down, mostly because of the "brace doubling" workaround.

Proposal

- Add a “p” prefix. These are the same as 3.8’s regular unprefixed strings.
- Allow f-strings to be doc strings , if they contain no expressions.
- Add a “from __future__ import all_fstrings” (or whatever). I’ll call this “f-mode”.
- The only change in f-mode is that these are treated as f-strings:
    - "": unprefixed strings.
    - f"": f-strings.
    - r"": raw strings (but see further discussion).
- Just like in 3.6+, b"" binary strings are never f-strings.

Problem areas

- Just turning on f-mode interpreter-wide would definitely break some existing code, so we need some sort of per-module switch.
- Places where p-strings would commonly be required:
    - Calls to str.format() – ouch! Note that there’s really no need to use string literals with .format() anymore: just use f-strings.
    - Regular expressions that use braces. This is probably reasonably common.
    - logging.Formatter(style='{'), but I don’t see this used very often (in fact, I’ve never seen it used except in documentation or on SO).
- Yet another string prefix.

Non-problems

- Performance, for the most part: f-strings with no expressions already become regular strings at compile time. There’s a special case for doc strings, which is why they would need to be changed to allow expression-less f-strings.
- It’s opt-in by the __future__ statement, so it won’t affect modules that don’t opt in.
- For existing strings without braces, no one would notice a difference.

What about raw strings?

- Because raw f-strings exist in 3.6+, I originally thought that unadorned raw strings in f-mode must be treated as f-strings.
- You could say that r"" strings now are treated as rp"", and that if you really want an rf"" string you have to explicitly say so, even when/if f-mode is enabled by default.
- But this seems like a special case: is it worth it? Maybe so, it seems like the most reasonable use.

Would we ever really enable f-mode by default?

- If not, then __future__ seems like wrong way to turn it on.
- But we’ve said we don’t want pragmas, and this needs to be on a module-by-module opt-in basis during a long transition period.

Next steps

- Good idea, or bad idea?
- PEP is almost complete.
- Implementation is straightforward.