We have nearly one million lines of Python 2 code in production — and now?

Michael Howitz, gocept
Created in May/June 2020.
Introduction

Michael Howitz

• since 2003
  * at gocept gmbh & co kg.
  * in Halle (Saale), DE
  * Python developer
Audience question
Python 2 End Of Life Survey by ActiveState

How prepared do you feel for Python 2 EOL?
- Not prepared
- Somewhat prepared
- Highly prepared

Survey results in Q4/2019
- Not prepared: 31%
- Somewhat prepared: 19%
- Highly prepared: 50%
Agenda

• Discussion of approaches
• Introduction into union.cms
• Doing the migration
• Tips and tricks
Discussion of approaches

- Sunset the application
- Replace by something off the shelf
- Don’t migrate
- Start over / Fade out
- Move to Python 3
Possible approaches

Sunset the application
Possible approaches

Replace by something off the shelf
Possible approaches

Don’t migrate

Image by Lynn Greyling from Pixabay
Possible approaches

Start over

Image by Lynn Greyling from Pixabay
Possible approaches

Move to Python 3

Image by Marcus Bouvin from Pixabay
union.cms

👉 Migrate to Python 3
Steps towards running on Python 3
I. General preparation

Migration steps

• Know your tools: Python 2 and 3 and the differences
• Have a good test coverage.
• Type annotations might help.
II. Clean up code and tests

Migration steps

• Make sure all tests run and pass
• Find out which code parts are still used, delete the rest
III. Bring dependencies to Python 3

Migration steps

- List all Python package dependencies
- Check each dependency for Python 3 compatibility
- Update to newest versions of dependencies
- Replace or port not yet Python 3 compatible ones
- Create a dependency tree of your packages

*tl.eggdeps*
IV. Migrate the code

Migration steps

• Run `python-modernize` (package is called `modernize`).
• Run `pylint --py3k`
• Fix the tests.
  * `pytest-quarantine`
  * Make sure, the tests still run on Python 2.
• Mark code with `# PY2` resp. `# PY3`. 
V. Switch to Python 3

Migration steps

• Run instance against empty database.
• Maybe migrate the data.
• Test the application in depth from end to end.
• Rollout on Python 3.
• Drop Python 2 support.

* pyupgrade
• Use metrics to compute the effort
  * pylint warning: 30 min
  * doctest: 20 min
  * Python module: 5 min
  * modernized Python module: 2 min
  * base effort per package: 135 min
• Plan the migration
• Rollout often
Tips and tricks

- Target to a recent Python 3 version
- No long running branches
- Always do small steps
- Take the time it needs
• Python 2 End Of Life Survey from ActiveState
  * https://www.activestate.com/resources/white-papers/python-2-eol-survey-results/

• union.cms consumers
  * https://www.dgb.de
  * https://www.verdi.de

• My employer
  * https://gocept.com/index_en.html

• Migration helpers
  * https://pypi.org/project/mypy/
  * https://pypi.org/project/modernize/
  * https://pypi.org/project/pylint/
  * https://pypi.org/project/pytest-quarantine/
  * https://stxnext.com/blog/2019/04/16/python-3-migration-guide/
  * https://portingguide.readthedocs.io
  * https://pypi.org/project/pyupgrade/
  * https://pypi.org/project/tl.eggdeps/