



# Python Incompatible Changes

Victor  
Stinner

# Victor STINNER

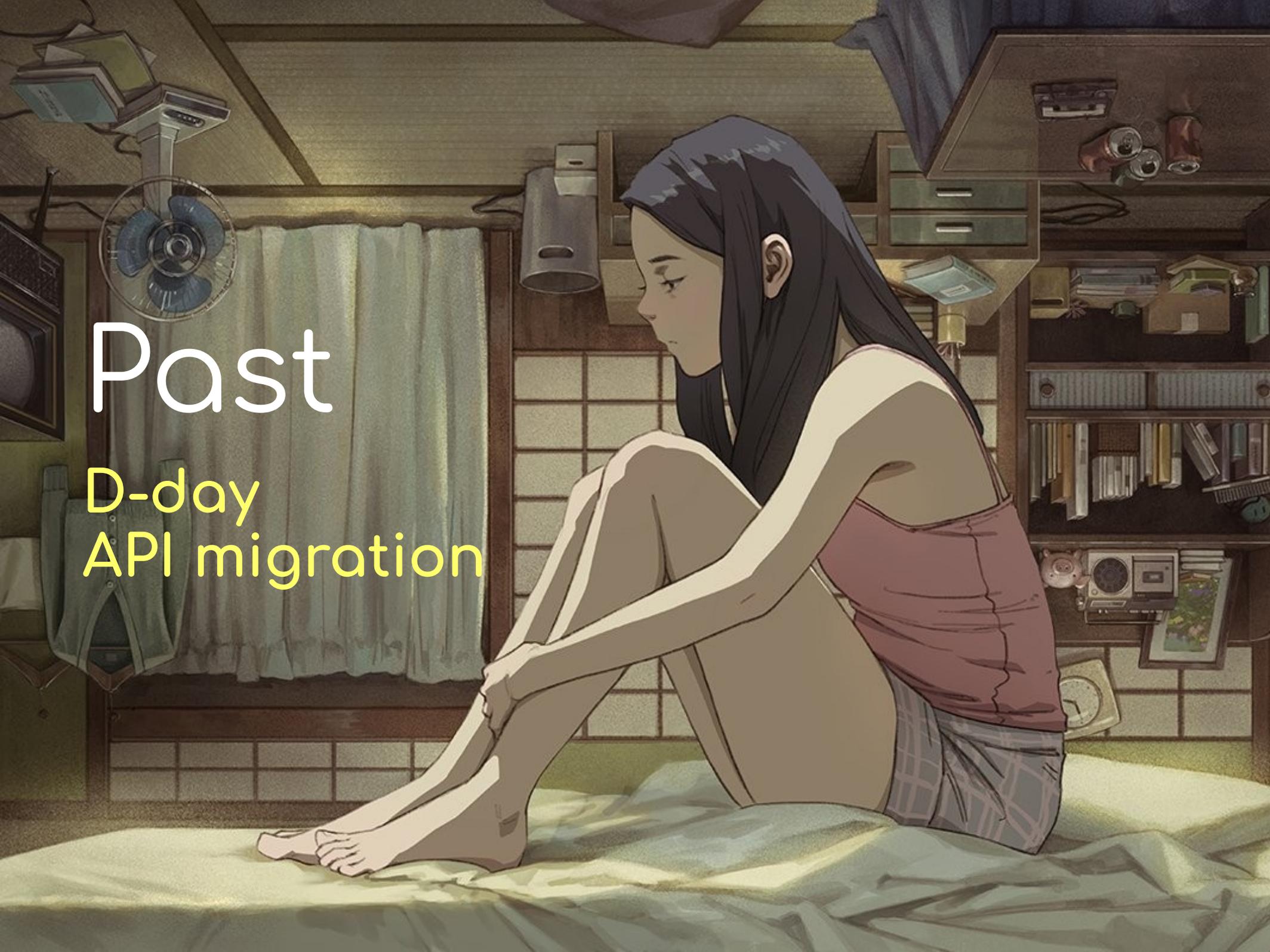
---



- Maintain Python **upstream** (python.org) and **downstream** (Fedora and RHEL) for Red Hat
- Python core developer since 2010
- Happy **Fedora** and **vim** user!
- Went through many Python incompatible changes since 2010..
- <https://twitter.com/VictorStinner>



Red Hat



Past

D-day  
API migration

A long time ago...  
in a galaxy far, far away  
There was **Python 2**.

Let's travel **15 years** in the past, before  
Python 3...

# Python 2 Unicode errors

---



- In the Python 2 era, **Django** started to become a real competitor of PHP frameworks and Ruby on Rails.
- One common issue in Django was handling **non-ASCII** characters
- French people are aware of these issues with non-ASCII first names! (Stéphane)
- In Python in general, handling **Unicode errors** was a frequently asked question.



Red Hat

# Python 2: bytes vs Unicode

---



- `"abc"` is a **bytes** string
- `"é"+u"ç"` raises a Unicode error
- Getting Unicode correctly is complicated in Python 2.



Red Hat

# Python 3 uses Unicode

---



- Python 3 fix the root issue: Unicode becomes a **first class citizen**
- **"abc"** is now a **Unicode** string
- Migrating Python 2 to Python 3 is painful: you have to handle **all** bytes vs Unicode **issues at once**



Red Hat

# Iterators and generators

---



- Python 2.2 introduces **iterators** (PEP 234) and **generators** (PEP 255)
- Python 2 **map()** and **zip()** return a **list**: consume more memory than a generator
- itertools **imap()** and **izip()** have to be used explicitly
- dict.**items()** returns a list,  
dict.**iteritems()** creates a generator



# Python 3 uses generators

---



- Python 3 `map()` and `zip()` create a generator
- `dict.items()` creates a generator
- Just use `list(dict.items())` to get a list



# Python 3

---



- Python evolved as Python 3 to get a **sane default behavior**
- Use Unicode
- Create generators
- Python 3 is “consistent” again
- .. but is **backward incompatible**



Red Hat

# Why incompatible changes?



PEP 20: “There should be **one** –and preferably only one– **obvious way** to do it.”

- **Consistent** coding style
- Easier to **teach**
- Easier to **review**



**Red Hat**

# Python 3 D-Day

---



- Python 3 initial plan was simple
- Everybody has to run **2to3** on their code at once to make it compatible with Python 3
- Problem: **dependencies**
- Problem: maintainers wanted to **keep Python 2 support**
- Problem: the migration took **10 years**



Red Hat

# Then comes the C API

---



- The C API is used by many third party C extensions to **extend Python**
- It's a big **part of Python success**
- Without C API, there is no Cython, **no numpy**, no pycopg2, etc.



**Red Hat**

# Python 3.11 C API changes

---



Python 3.11 optimizations changes multiple core C structures

- Py**Code**Object
- Py**Frame**Object
- Py**Thread**State

Problem: some C extensions directly access members of these structures



Red Hat

# Update C extensions

---



- `code` -> `f_code`  
→ **PyCode\_GetCode**(code)
- `frame` -> `f_back`  
→ **PyFrame\_GetBack**(frame)
- `tstate` -> `frame`  
→ **PyThreadState\_GetFrame**(tstate)
- Problem: These functions requires Python 3.11 or `#ifdef`





Present

Smoother  
API updates

# Python 3: six module

---



- six allows writing a **single code base** working with old and new Python
- **Add support** for the **new** Python 3, rather than removing support for the old Python 2
- New migration approach: **port incrementally** the code
- **D-day** approach **abandoned**: we learnt!



Red Hat

# Revert 3.10 changes

---



- Python 2.7 support just ended upstream, many projects still supported Python 2.7
- Revert **open()** “U” mode removal
- Revert **collections** ABC aliases removal
  - Removed again in Python 3.11
- These changes affected too many packages: give **more time to update**



Red Hat

# Revert 3.11 changes

---



- Revert unittest aliases removal
- Revert configparser removals
- Revert asyncore removal
- Changes postponed to Python 3.12
- Affected **too many packages**, it takes time to update them (fix, release)



Red Hat

# PEP 387

---



- Since **2020** (Python 3.9), one Python release per year (PEP 602)
- Backward compatibility policy (PEP 387) **updated** to **deprecate** for **2 years**:  
2 Python releases
- Example: **deprecate in 3.11** and 3.12,  
**remove in 3.13** (3 years)



Red Hat

# DeprecationWarning



- Python 2.7 (2010) hides them by default
- Python 3.7 shows them in the `__main__` module (PEP 565)
- `python -W default`: display once
- `python -W error`: raise an exception
- `python -X dev`: Python development mode



# Smooth deprecation

---



- **Add** new API
- Deprecate only in the **documentation**
- Emit **DeprecationWarning**
- Explain how to update existing code **without losing support** for old Python versions (**NEW!**)
- **Remove** the old API



Red Hat

# Code search

---



- `download_pypi_top.py`: **Download** source code of the **PyPI top 5,000** projects
- `search_pypi_top.py`: **Search** for code pattern with a regex
- **Help updating** affected projects  
<https://github.com/vstinner/misc/tree/main/cpython/>



**Red Hat**

# Ideal migration

---



- **Add** new API
- **Doc** and/or **tool** to update
- Identify and **update** affected projects
- Wait for **releases** of affected projects
- **Deprecate** old API
- **Remove** old API
- Take ~3-5 years



**Red Hat**

# Soft Deprecation

---



- PEP 387 updated
- Should no longer be used to write new code
- **No** scheduled the **removal**
- **No DeprecationWarning**
- Only in the documentation
- Still documented and tested



**Red Hat**

# Unmaintained projects

---



- Some core Python dependencies have a **single unavailable maintainer**
- Busy with work, life duties, get bored, sick, etc. (it's not only about bus)
- How to update projects when the maintainer doesn't reply?
- Problem of **funding** maintenance of these projects
- **Thankless work**



Red Hat

# Hidden projects

---



- Projects **behind closed doors**
- From short scripts to large applications
- Old projects no longer maintained
- Turnover
- **pyupgrade** and **upgrade\_pythoncap1.py**
- or: **keep** an **old Python** (security!)



Red Hat

# pythoncapi-compat



- C API: Provide **new functions** to **old Python**
- 2020: Project created
- **upgrade\_pythoncapi.py** script
- 2021: Python 2.7 support for Mercurial
- 2022: Python 3.11 functions
- 10 projects are using it
- <https://pythoncapi-compat.readthedocs.io>



**Red Hat**

# pythoncapi-compat



- Update your C extensions to **use new** Python 3.11 **functions**
- **Copy** pythoncapi\_compat.h file to get these functions on Python 3.10 and older
- Keep support with **Python 2.7**
- No need to update pythoncapi\_compat.h until you need a new function



**Red Hat**

# C API guidelines

---



- New functions must **not return borrowed references** but strong references
- New functions must **not steal references**
- Well define **ownership** rules and **lifetimes** of arguments and structure members
- It should ease supporting the C API on Python implementations other than CPython



Red Hat

# C API headers

---



C API now splitted in 3 categories (3 directories):

- **Limited** C API (stable ABI)
- **Public** C API
- **Internal** C API



**Red Hat**

# Fedora COPR

---



- My team tests **Python 3.13** since alpha1 to detect issues as soon as possible
- **Rebuild** Fedora Python **packages** (4000+) with Python 3.12
- Lot of work to **identify the root cause**, report issue to **upstream**, propose a fix
- **Collaboration** to get the fix merged and get a release



Red Hat

# What's new?

---



- **Proactively** search for affected projects
- **Document** how to update code without losing support with old Python versions
- Helping to **update affected projects** give a better **insight** on how to update code
- **Fedora** provides early feedback
- **pythoncapi-compat** for C API
- **Soft deprecation**



**Red Hat**



Future  
Better API

# Stable ABI

---



- Python 3.2 implements **Stable ABI** (PEP 384): the **limited C API**
- **Build once**, works all Python versions
- Python 3.10: Maintaining the Stable ABI (PEP 652), **tests** the ABI, better doc
- 2021: **CI** now fails on any ABI changes (not just the stable ABI)
- **Cryptography** and **PySide** binaries on PyPI use the stable ABI



Red Hat

# HPy project

---



- **New C API** designed in 2019 to be efficient on **PyPy** (and CPython)
- **ujson** made **3x faster** on PyPy with HPy
- **Universal ABI**: build one, work on all CPython and PyPy versions
- **numpy** WIP port to HPy
- <https://hpyproject.org/>



Red Hat

# Test next Python

---



- Test Python **nightly build** in your CI
- Bugs discovered earlier are easier to fix
- **Report** issues to CPython bug tracker
- If possible, start by testing Python **alpha** versions



Red Hat

# Q & A



# Sources

---



- Drawing by **Djamila Knopf**
- Python and Red Hat are registered trademarks



**Red Hat**