Python 2 to 3 migration

Big project edition

Dmitry Karpov
Software Engineer
Wargaming

Motivation

- Python2 EOL 2020
- Python3 features
- Hiring developers
- General performance
- Maintenance costs (subtle bugs)

Which Python?

At first look

2.0 … 2.5 … 2.7
3.0 … 3.8

Automated: low-level work
You: API decisions

methods -> function
Build-ins -> import function

Which Python?

At first look

2.0 … 2.5 … 2.7
3.0 … 3.8

Automated: low-level work
You: API decisions

methods -> function
Build-ins -> import function

Trove classificator

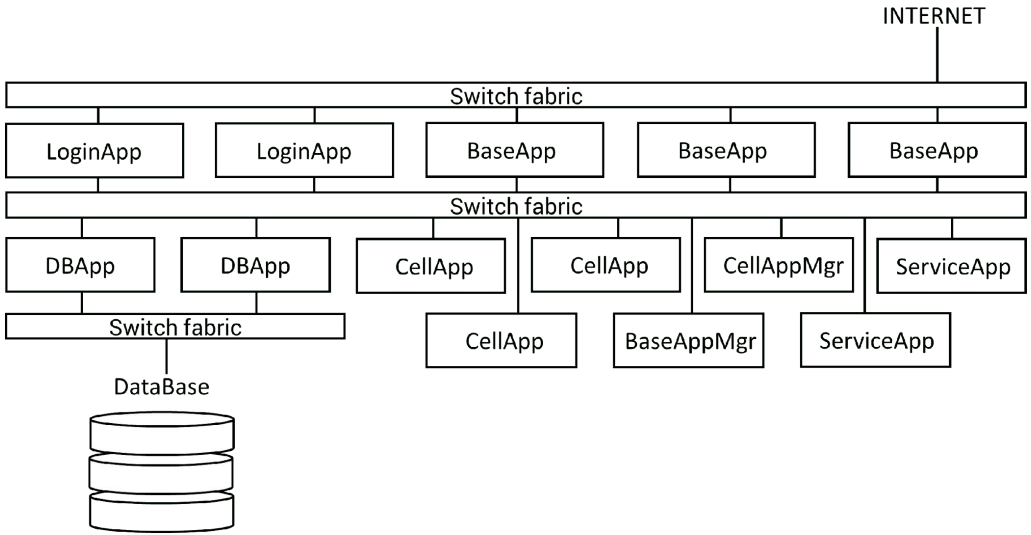
PEP 301

> 2.2.3

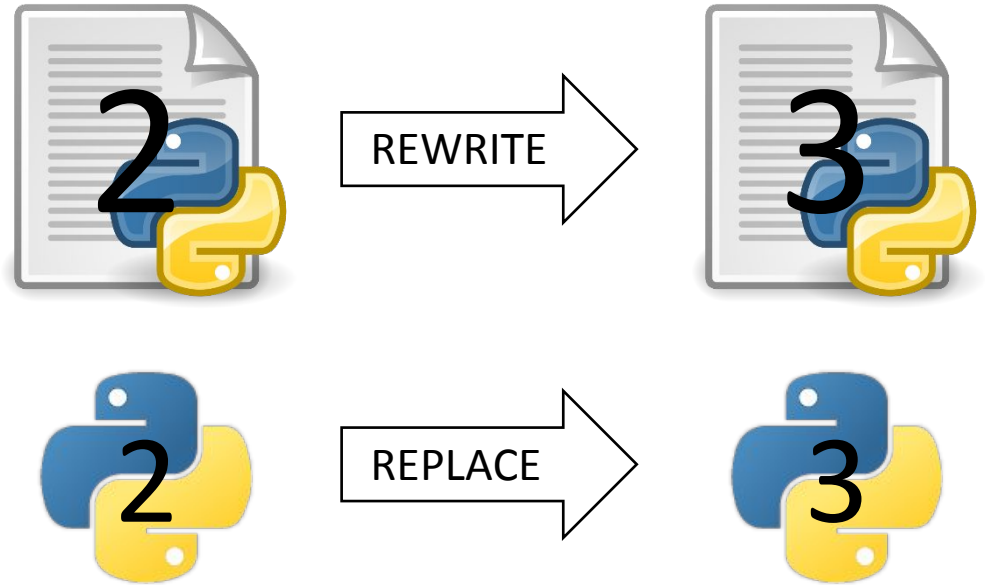
```
setup(
    name = "roundup",
    version = __version__,
    classifiers = [
        'Development Status :: 4 - Beta',
        'Environment :: Console',
        'Environment :: Web Environment',
        'Intended Audience :: End Users/Desktop',
        'Intended Audience :: Developers',
        'Intended Audience :: System Administrators',
        'License :: OSI Approved :: Python Software Foundation License',
        'Operating System :: MacOS :: MacOS X',
        'Operating System :: Microsoft :: Windows',
        'Operating System :: POSIX',
        'Programming Language :: Python',
        'Programming Language :: Python :: 2 :: Only',
        'Programming Language :: Python :: 2.8',  # PEP 404
        'Topic :: Communications :: Email',
        'Topic :: Office/Business',
        'Topic :: Software Development :: Bug Tracking',
    ],
    url = 'http://someaddress.com',
    ...
)
```

That’s it?

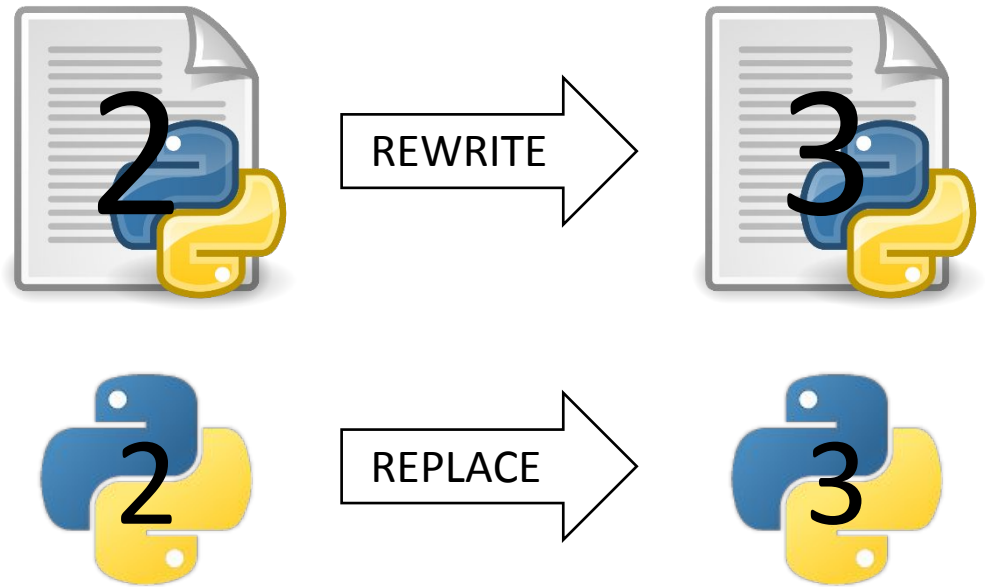
https://habr.com/ru/company/oleg-bunin/blog/418977/
https://en.wikipedia.org/wiki/BigWorld



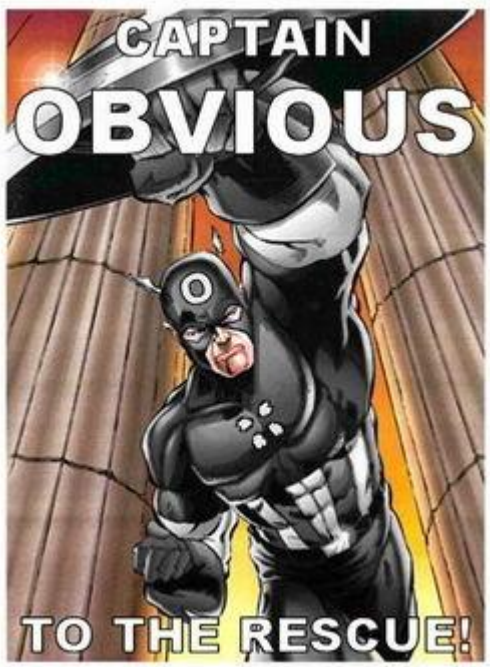
Small project



Small project



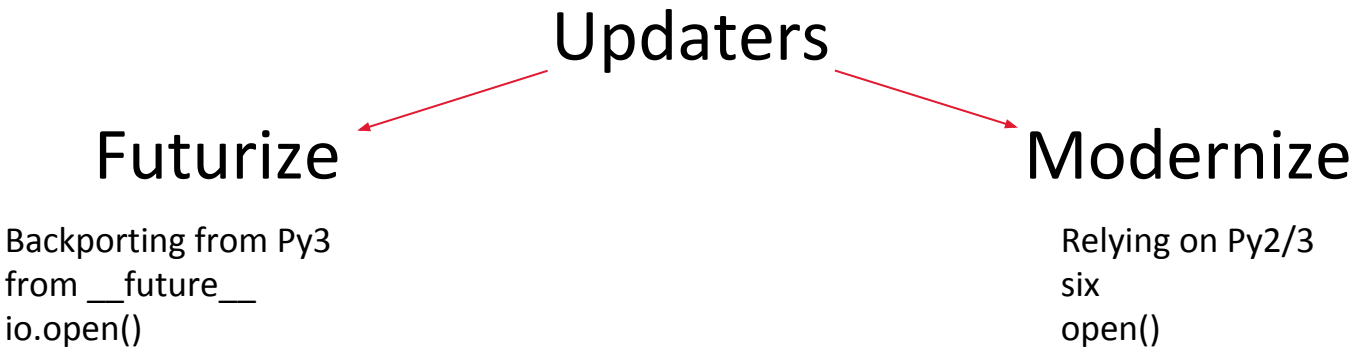
Many thanks to:



© memepedia.ru

How to migrate single application

https://docs.python.org/3/whatsnew/

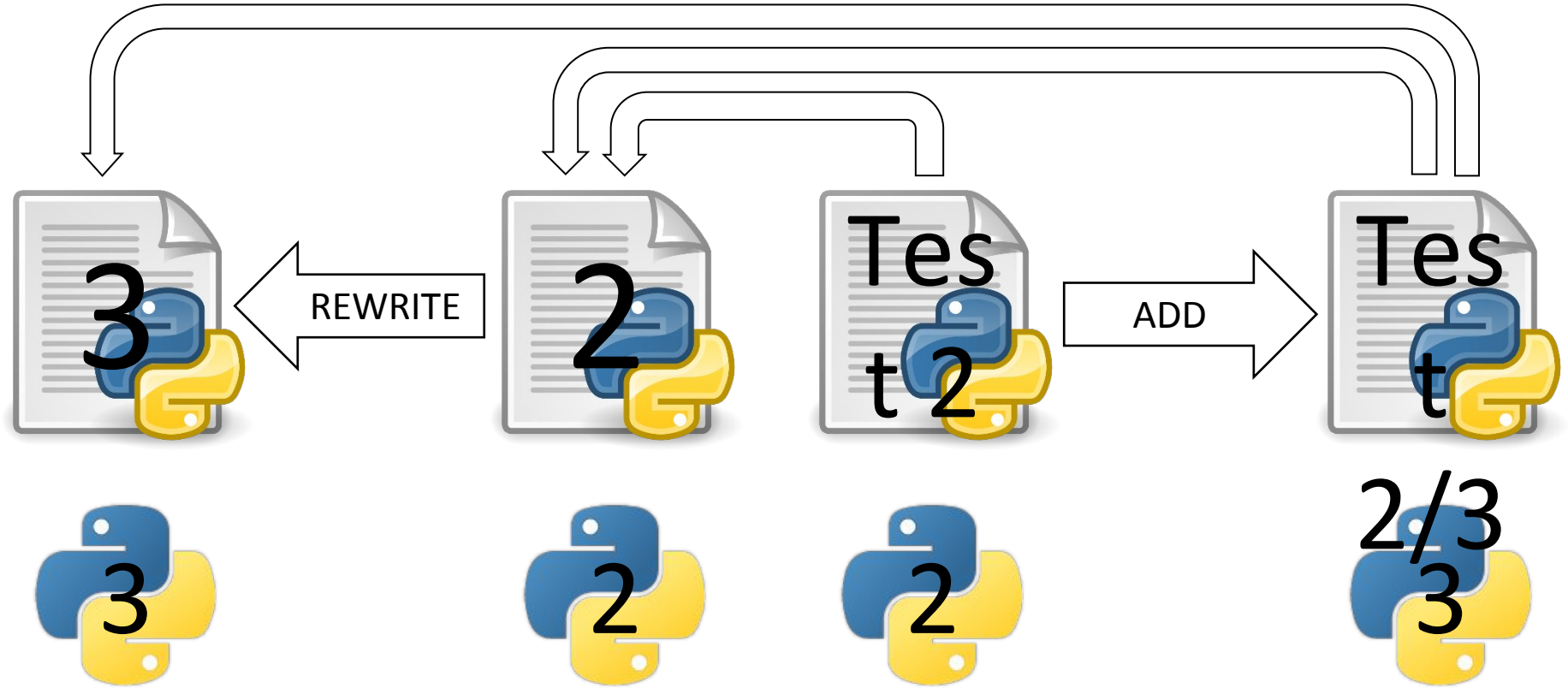


Backporting from Py3
from __future__
io.open()

Relying on Py2/3
six
open()

```
5 / 2 == 2  # Py2 True Py3 True False

class A:
  def __init__(self);
    pass
  def __div__(self, other):  # __floordiv__
    return other
a / 2  # Ok
a // 2  # Fails

b'123'  # Py3 Unicode strings by default

x = 1
[x for x in range(10)]
print x  # 9 in Py2 1 in Py3
```

Unit tests covered code base



Unit tests covered code base

Rule of thumb: 80% coverage

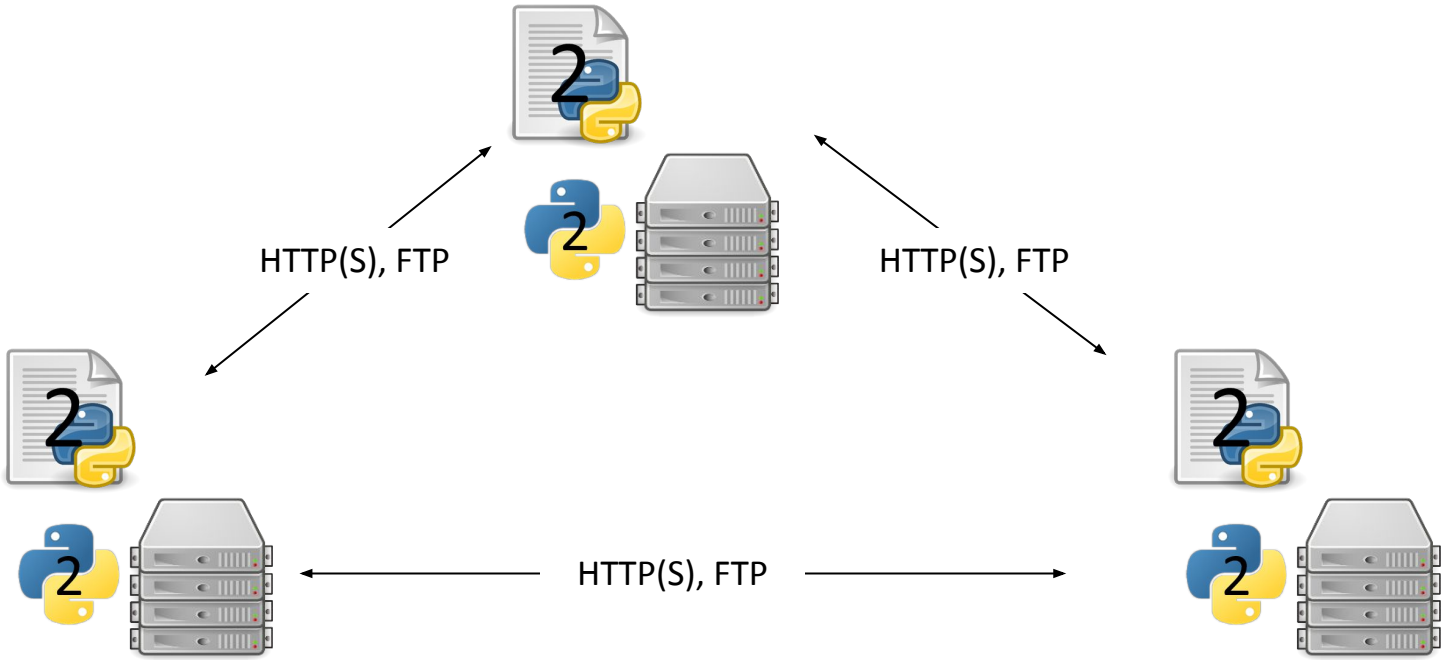
pip install coverage

>=2.6

http://python3porting.com/

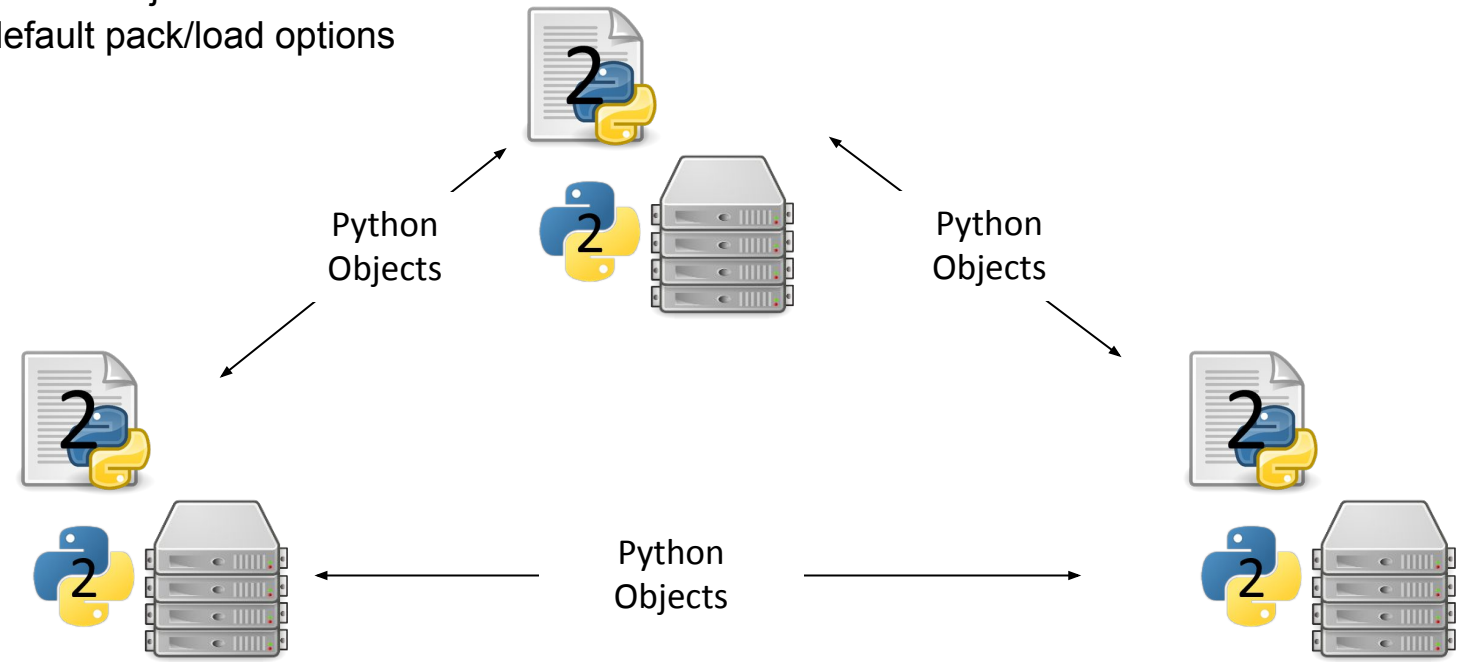
https://python-future.org/compatible_idioms.html

Services with no shared dependencies



Services with no shared dependencies shared Python objects

pickled objects
default pack/load options



Services with shared dependencies
shared Python objects

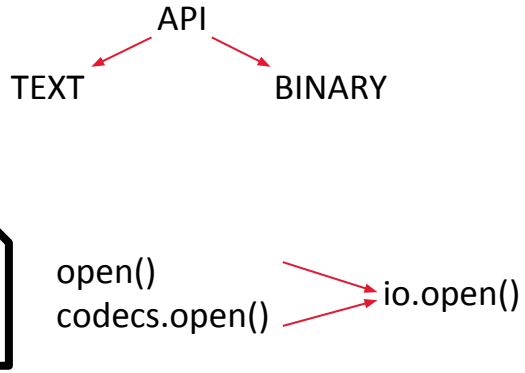
And infrastructure

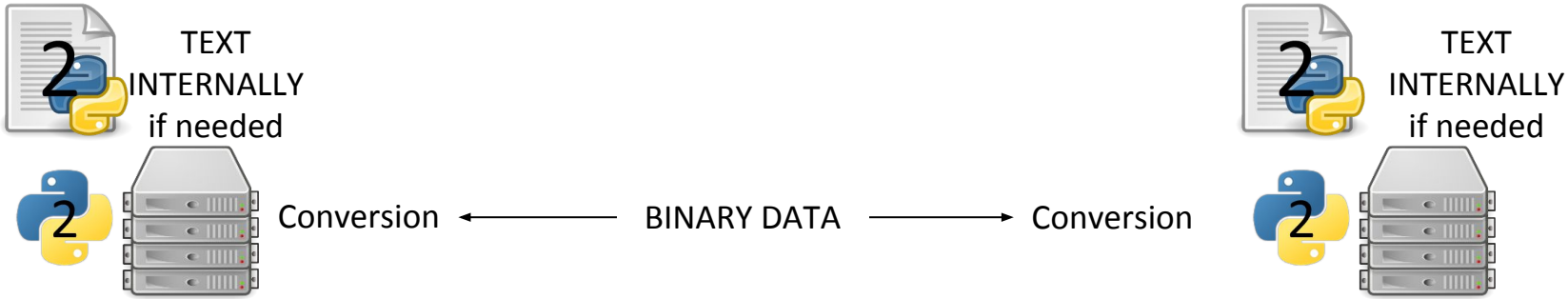
```python
import pickle

pickle.dump(obj, file, protocol=3)  # pickle.DEFAULT_PROTOCOL
                                    # Changed in version 3.0: The default protocol is 3.
                                    # Changed in version 3.8: The default protocol is 4.
```

```python
from kafka import KafkaProducer

producer = KafkaProducer(bootstrap_servers=['localhost:9092'],api_version=(0,1,0))
```

Text vs Binary

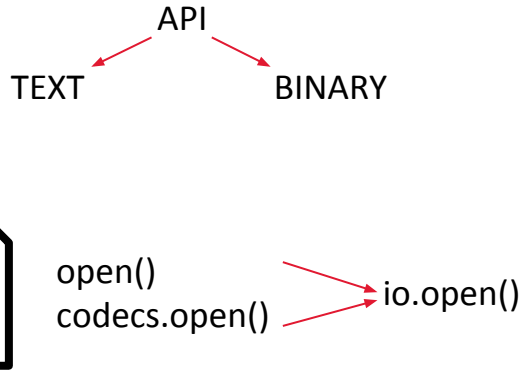
Text	Binary
	decode
encode	
format	
isdecimal	
isnumeric	





Text vs Binary

Text	Binary
	decode
encode	
format	
isdecimal	
isnumeric	

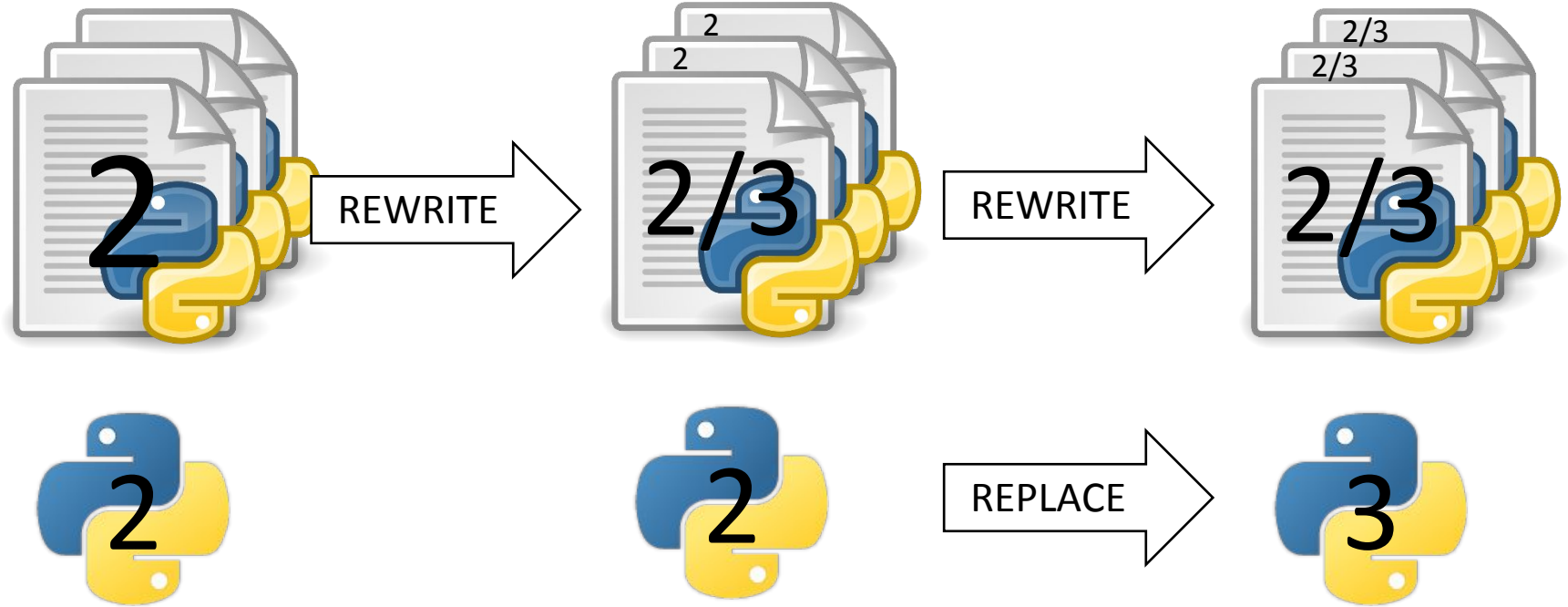



```
> bytes(5)
'5'
> b'123'[1]
'2'
```

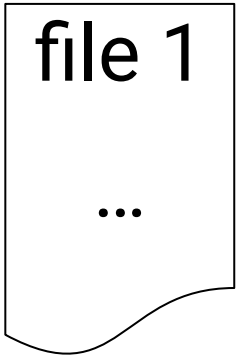
```
> bytes(5)
b'\x00\x00\x00\x00\x00'
> b'123'[1]
50
```

> python3 -bb

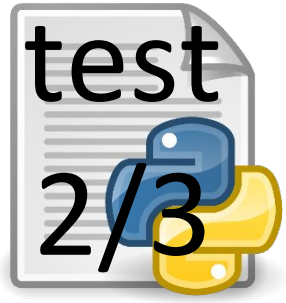
Big project



Big project with unit tests



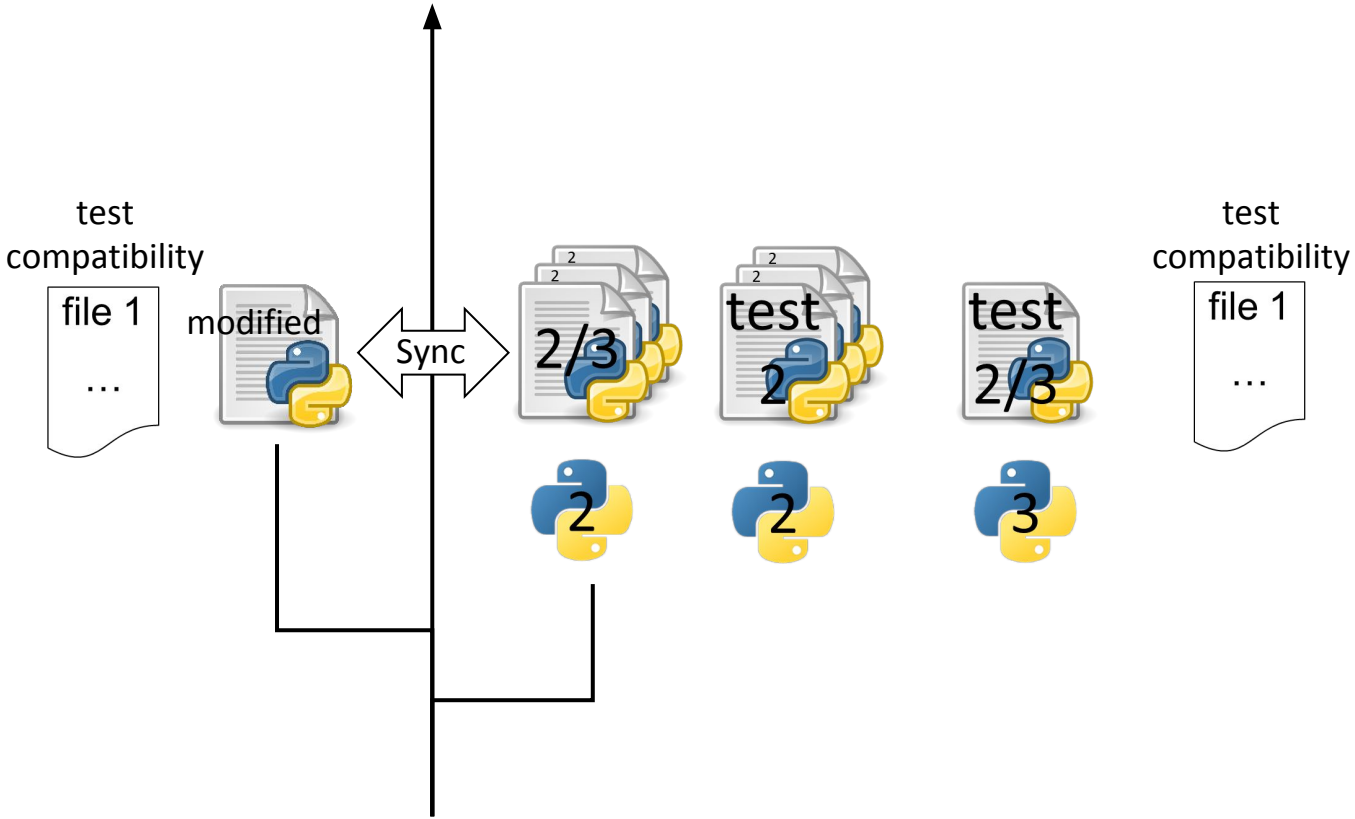
Big project with unit tests



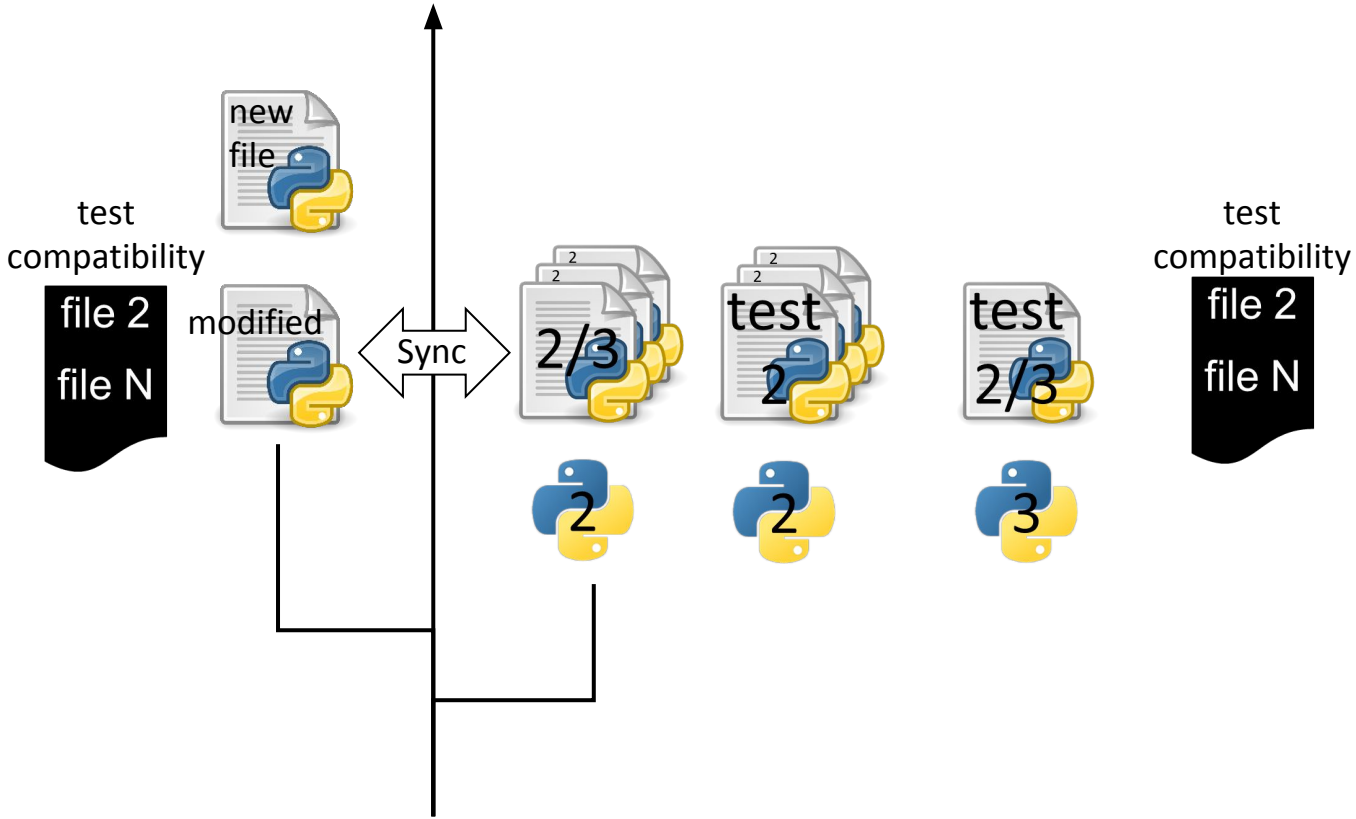
file 1

file N

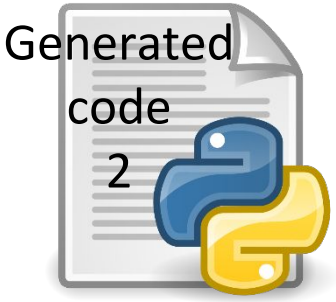
Version control



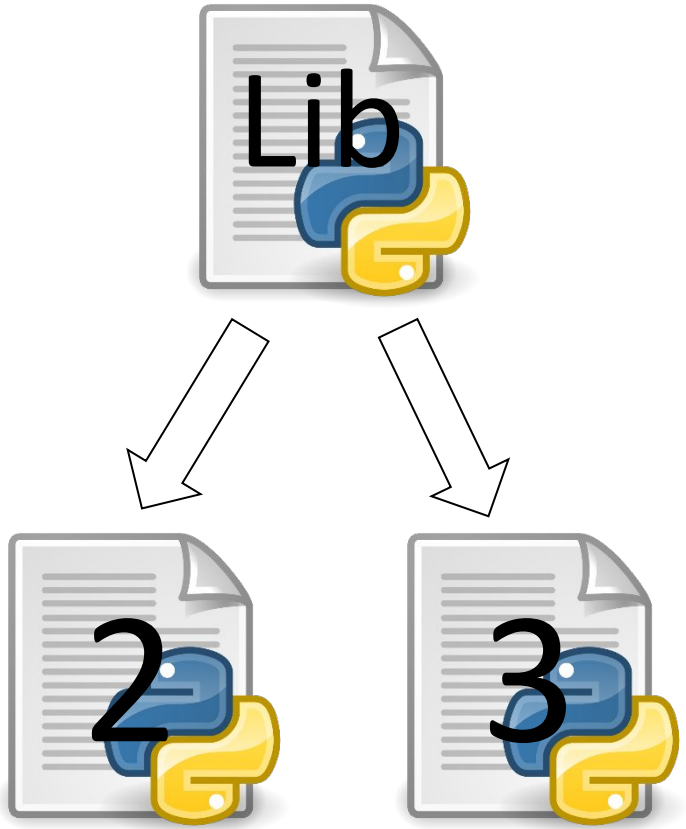
Version control



Code generation



Import lib



- Libs may have different versions for 2 and 3
  - lib for Python2
  - lib2 for Python3
- Libs may have different APIs for 2 and 3
  - lib.method(a, b, c) for Python2
  - lib.differentMethod(a, b) for Python3

Import lib

Feature detection vs Version detection

```
import sys

if sys.version_info[0] == 3:
    from importlib import abc
else:
    from importlib2 import abc
```

```
import sys

if sys.version_info[0] > 2:
    from importlib import abc
else:
    from importlib2 import abc
```

```
try:
    from importlib import abc
except ImportError:
    from importlib2 import abc
```

Import lib

Feature detection vs Version detection

```
import os
import scandir

os.walk()  # Python2 slow algorithm
os.walk()  # Python3 optimized algorithm

scandir.walk()  # Python2 optimized algorithm
```

```
try:
  import scandir.walk as walk
except ImportError:
  import os.walk as walk
```

pip install caniusepython3

Bindings

Options:

- CFFI
  - require users to learn domain specific language or API
  - preferred way is to work at the level of the API
- Cython
  - compiles a Python-like language to C
  - support for interfacing with C libraries
  - compatible with Python 2.6+ and 3.3+
- Python 3 APIs
  - rewriting the entire extension can be avoided
  - py3c 2/3 compatibility layer for C extention

Compatibility/Dependencies/CI

pip install caniusepython3

```
python -3
python -Werrors
```

pip install pylint

```
Pylint --py3k
```

pip install tox

Conclusions

Switching to Py3 is necessary update
Enforce you to close technical debt
Enforce you to update unit tests, CI, version control
Enforce you to do refactoring, optimizations
Depend on your project architecture
Make you to check every line of the project

Literature/Links

https://docs.python.org/3/whatsnew/
https://docs.python.org/3/howto/pyporting.html
http://python3porting.com/
https://python-future.org/compatible_idioms.html
https://habr.com/ru/company/oleg-bunin/blog/418977/

Many thanks to

CPython developers
Those, who share knowledge on 2/3
Colleagues of WG Engineering Team

Thank you! Any questions?

Dmitry Karpov
d_karpov@wargaming.net
karpov.dev