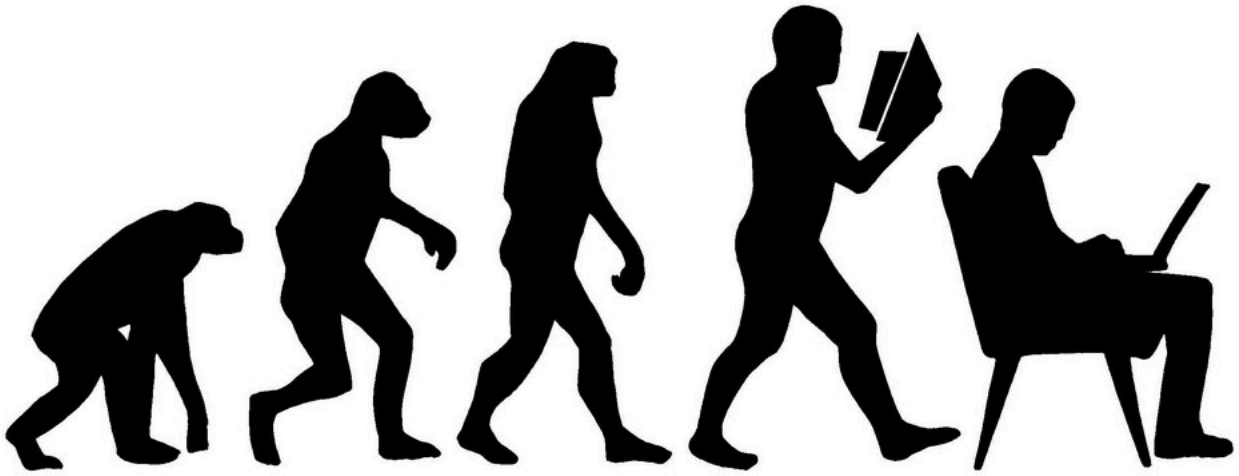
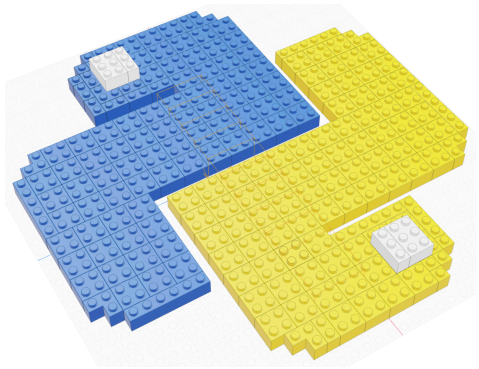
Time travel:

Let's learn from the history of Python packaging!

Kir Chou @ PyCon TW 2020

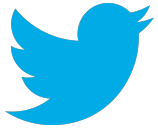




Kir Chou

- PyCon TW Speaker 2017, 2018, 2019
- PyCon JP Speaker 2019, 2020

- SDE @ Amazon Search



@k1rch0u

note35

kir.chou

Agenda

- Motivation
- History of Python software distribution
- Learn from history
- Challenge before community's python packaging solution
- Challenge after community's python packaging solution
- Takeaway
- Appendix

Motivation


One day, our builder Tool team announced…
Let’s migrate to idiomatic tools!

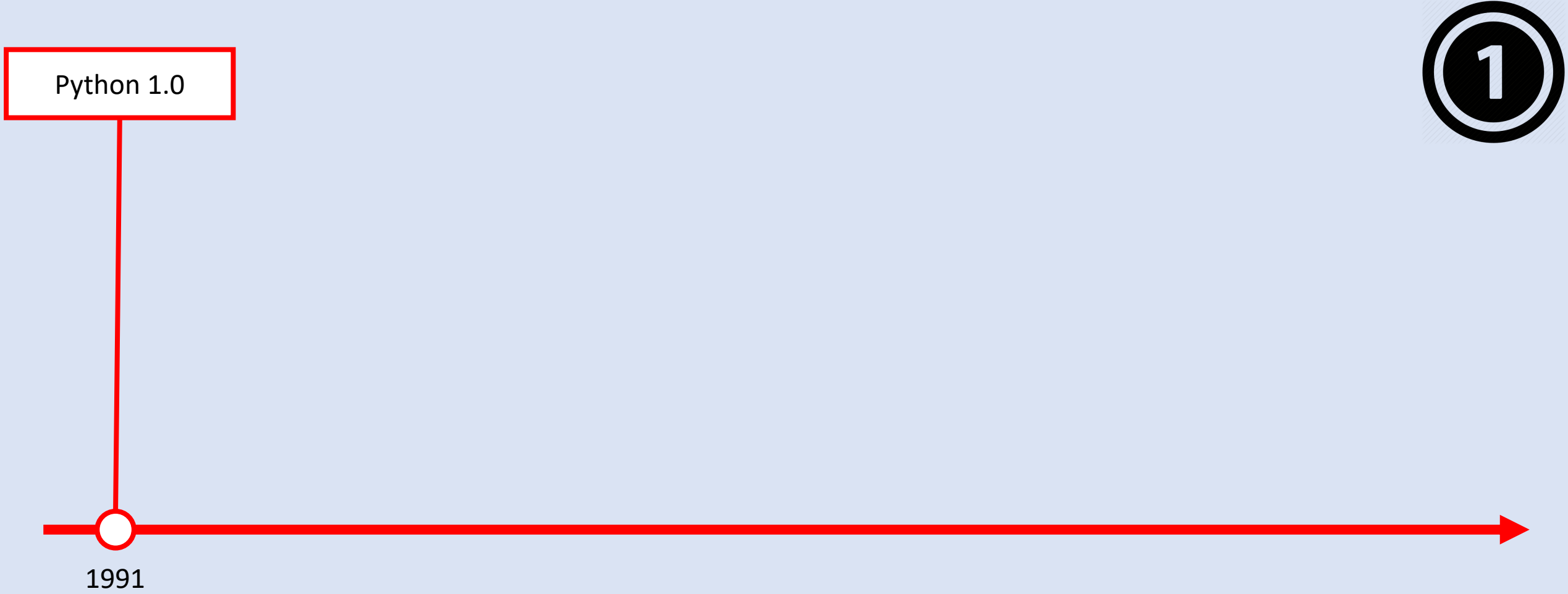
Great! But why?

PEP 566 -- Metadata for Python Software Packages 2.1

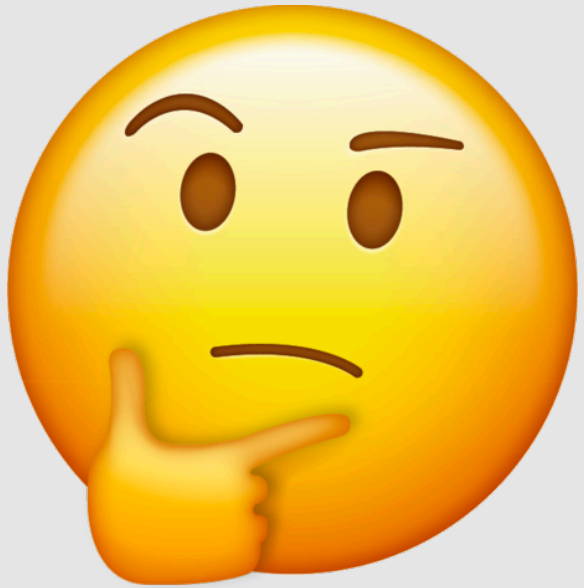
PEP:	566
Title:	Metadata for Python Software Packages 2.1
Author:	Dustin Ingram <di at python.org>
BDFL-Delegate:	Daniel Holth
Discussions-To:	distutils-sig <distutils-sig at python.org>
Status:	Final
Type:	Standards Track
Created:	1-Dec-2017

@di_codes

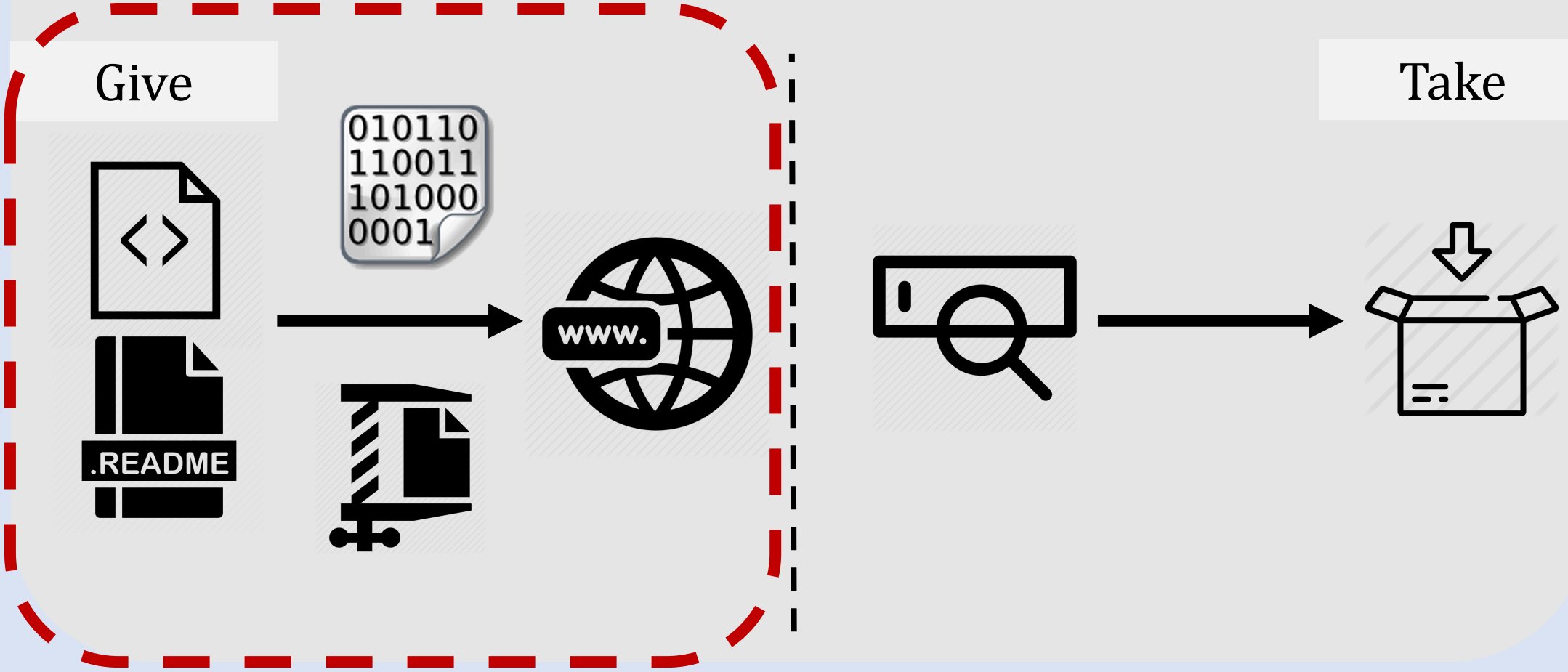
History of Python software distribution


Python 1.0
1
1991

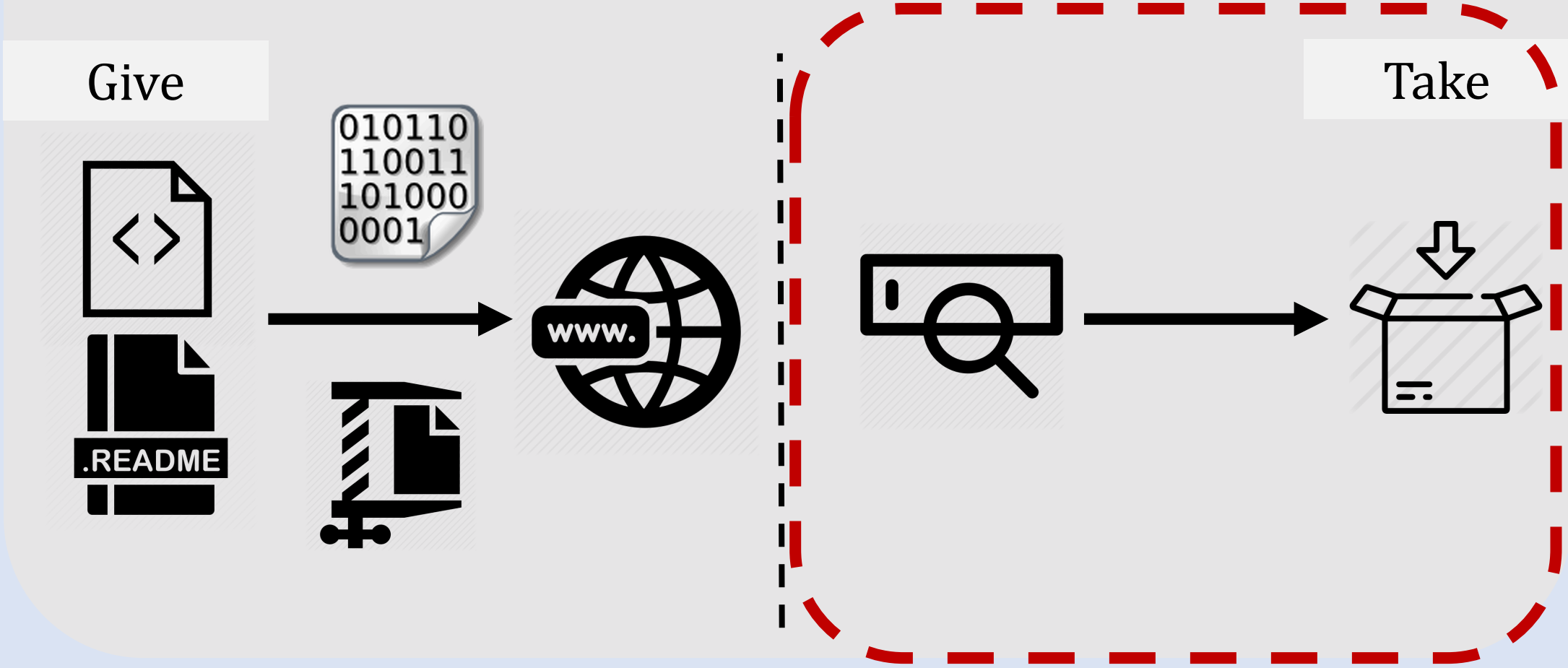
How to give/take the work to/from others?



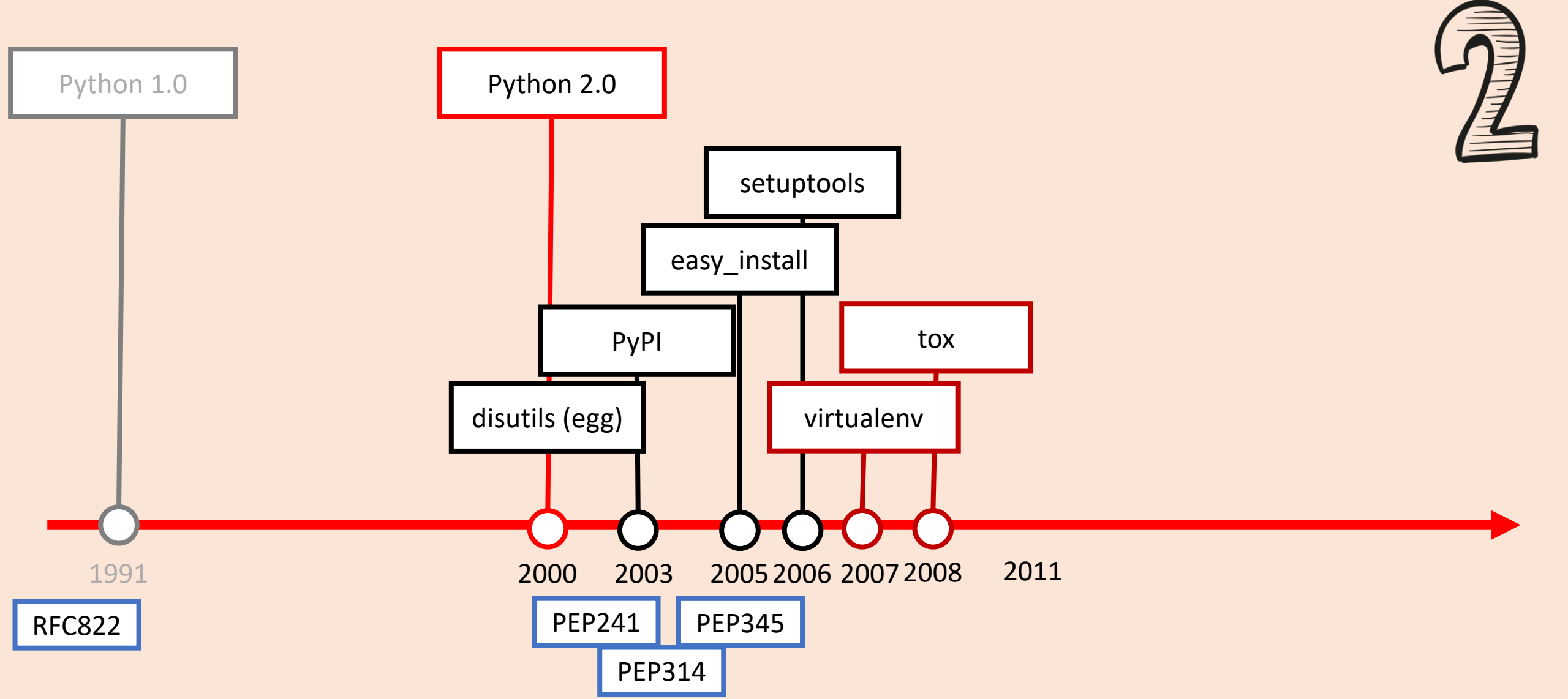
How to give/take the work to/from others?



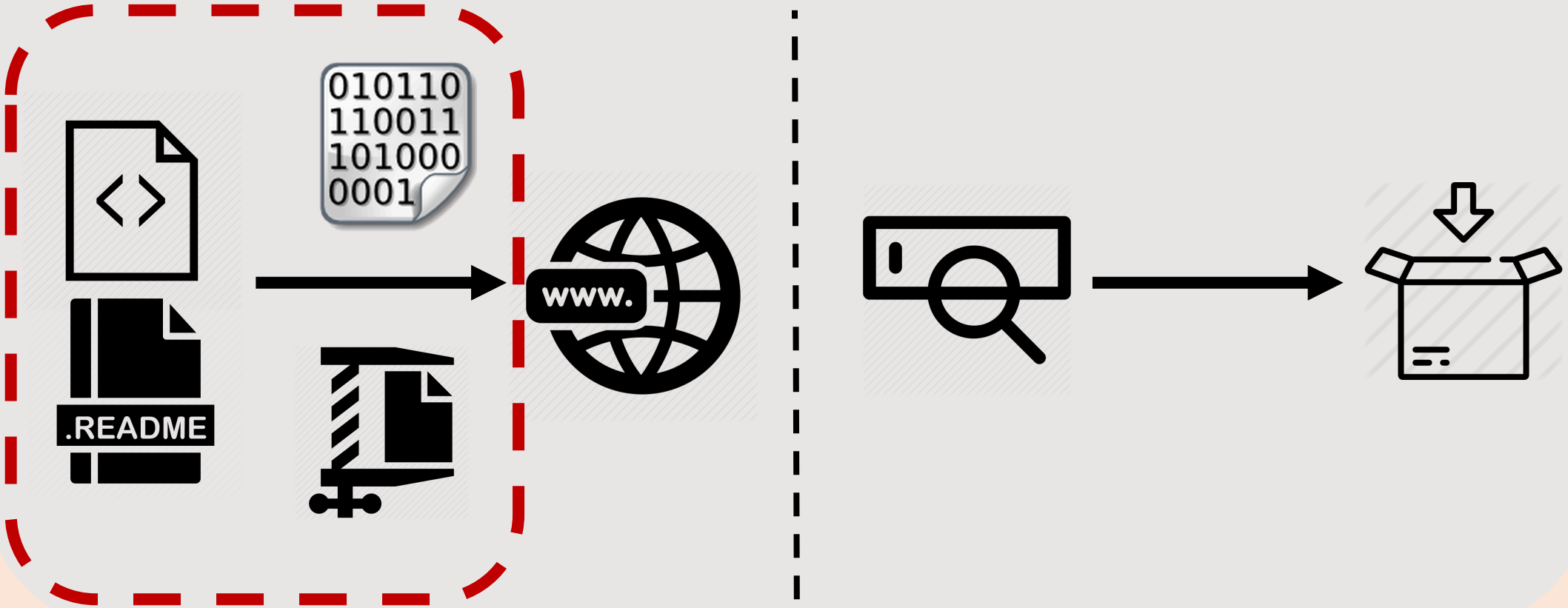
How to give/take the work to/from others?




10 YEARS LATER


Python 1.0
Python 2.0
setuptools
easy_install
PyPI
tox
disutils (egg)
virtualenv
1991
2000
2003
2005
2006
2007
2008
2011
RFC822
PEP241
PEP345
PEP314
PEP405

Metadata: PEP241 (2001), PEP314 (2003), PEP345 (2005)
Build tool: distutils, setuptools



```python
from setuptools import setup, find_packages

setup(
    name='helloworld',
    version='0.1',
    description="Hello World with setuptools",
    classifiers=[
      'Development Status :: 4 - Beta',
    ],
)
```

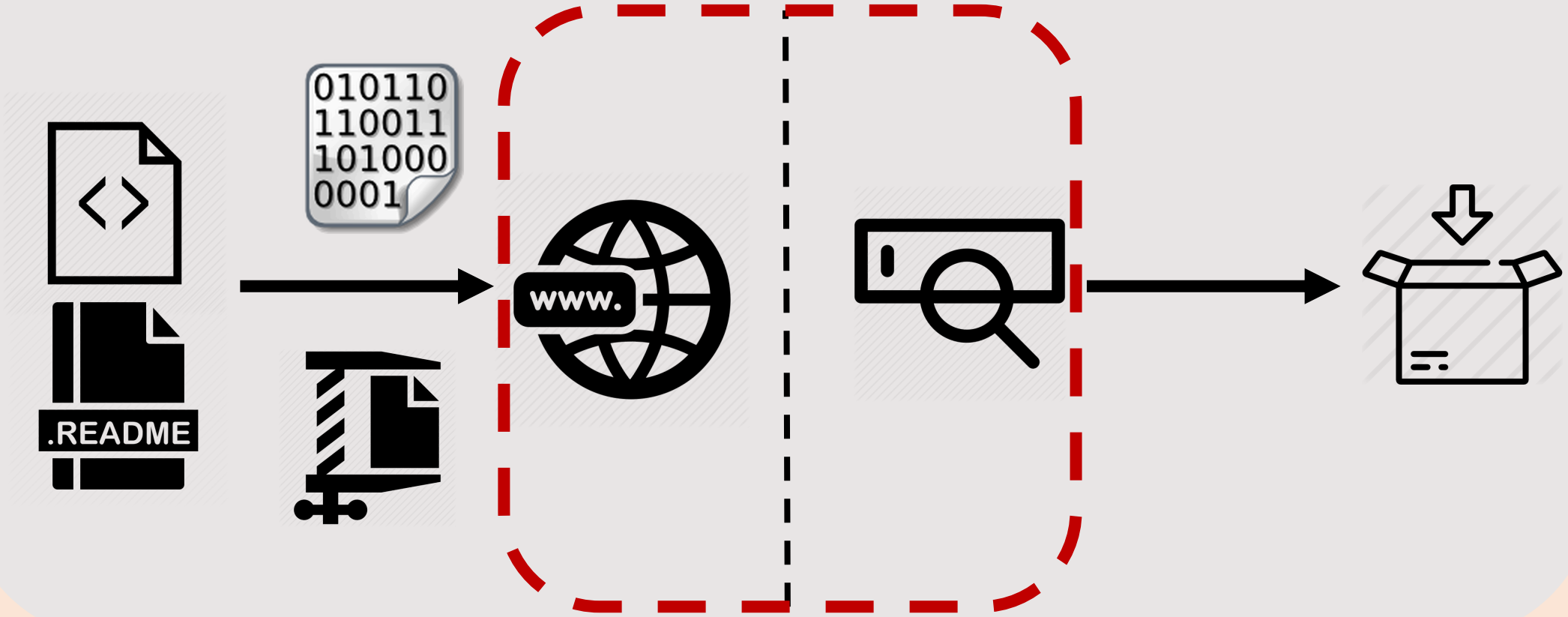
setuptools: setup.py

Metadata in EGG

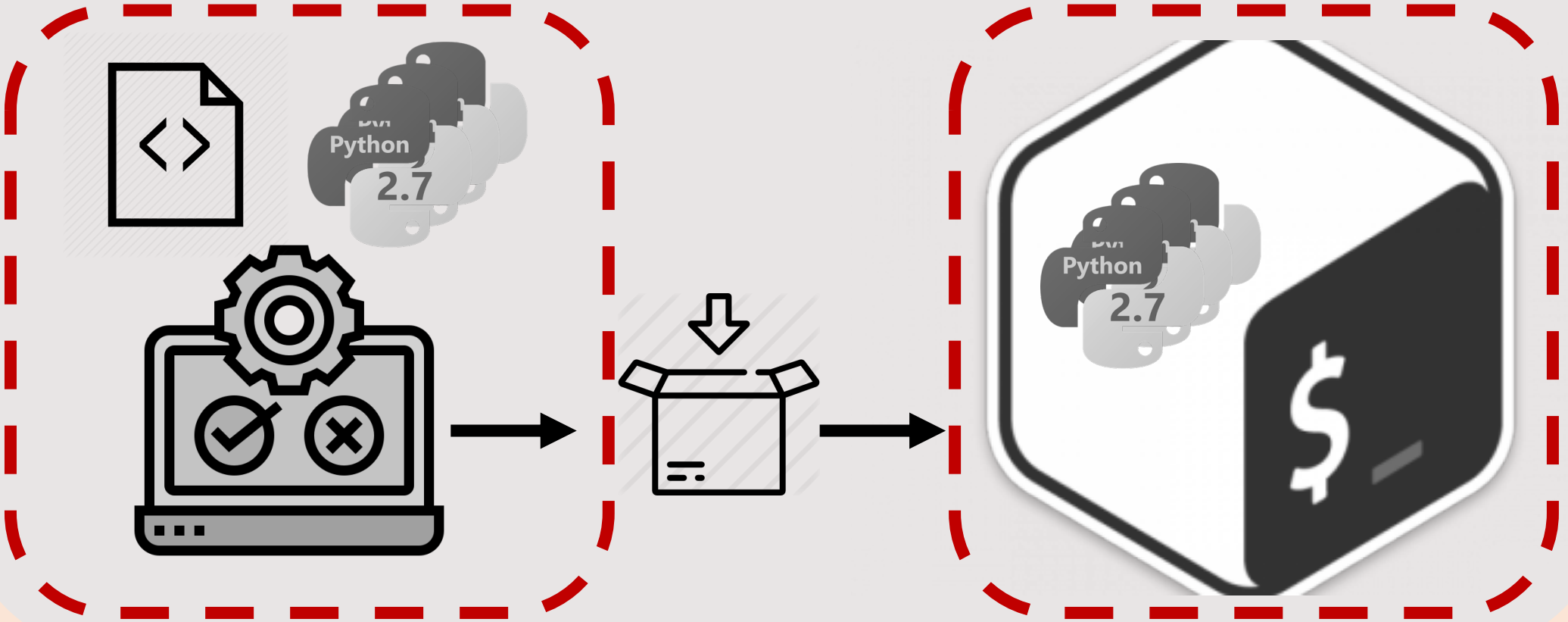
xxx.egg-info/PKG-INFO

```
Metadata-Version: 1.1
Name: helloworld
Version: 0.1
Summary: Hello World with setuptools
Home-page: UNKNOWN
Author: UNKNOWN
Author-email: UNKNOWN
License: UNKNOWN
Description: UNKNOWN
Platform: UNKNOWN
Classifier: Development Status :: 4 - Beta
```

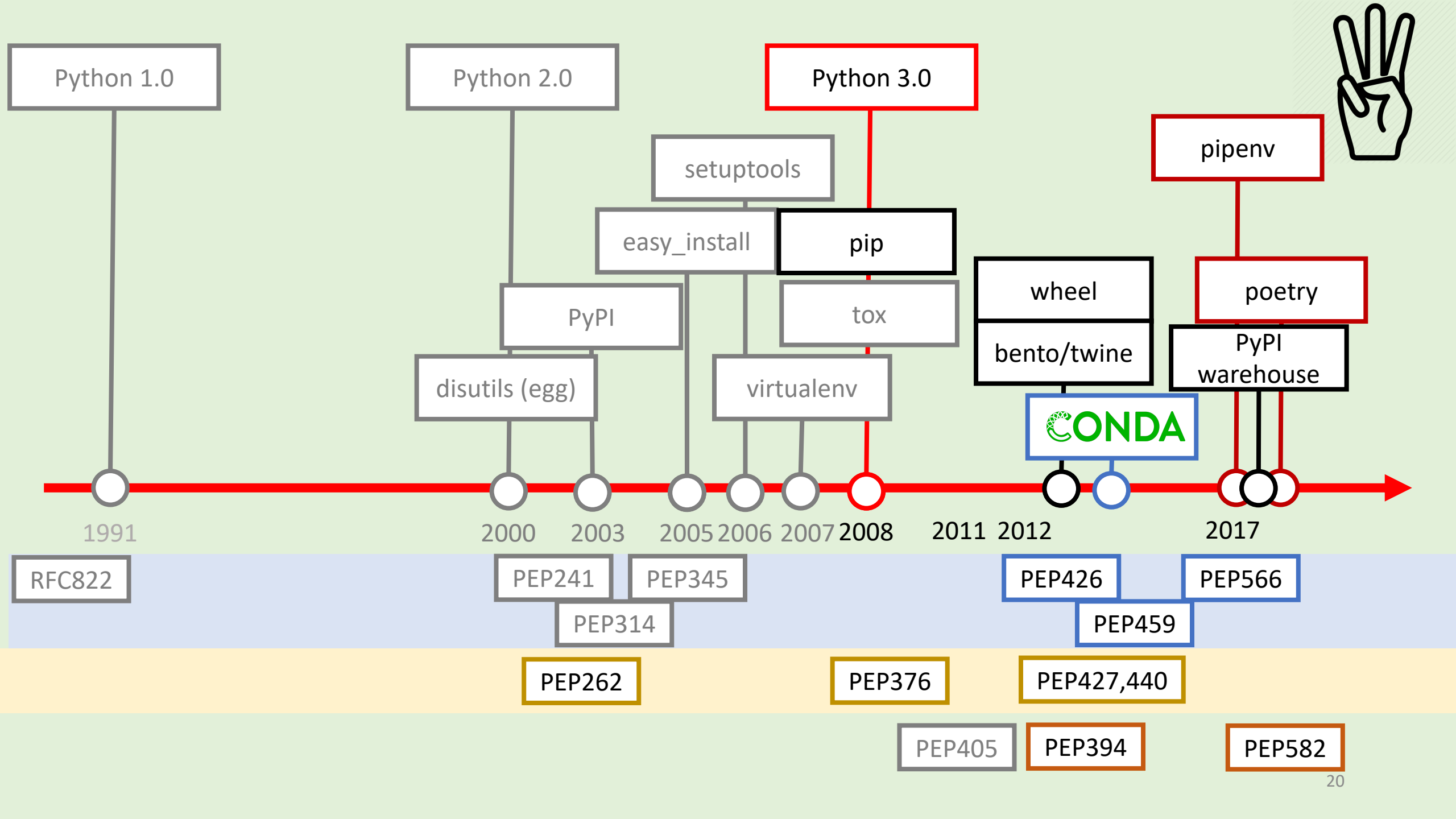
PyPI (Cheese Shop): PEP301 (2002/Nov)
Package Installer: easy_install



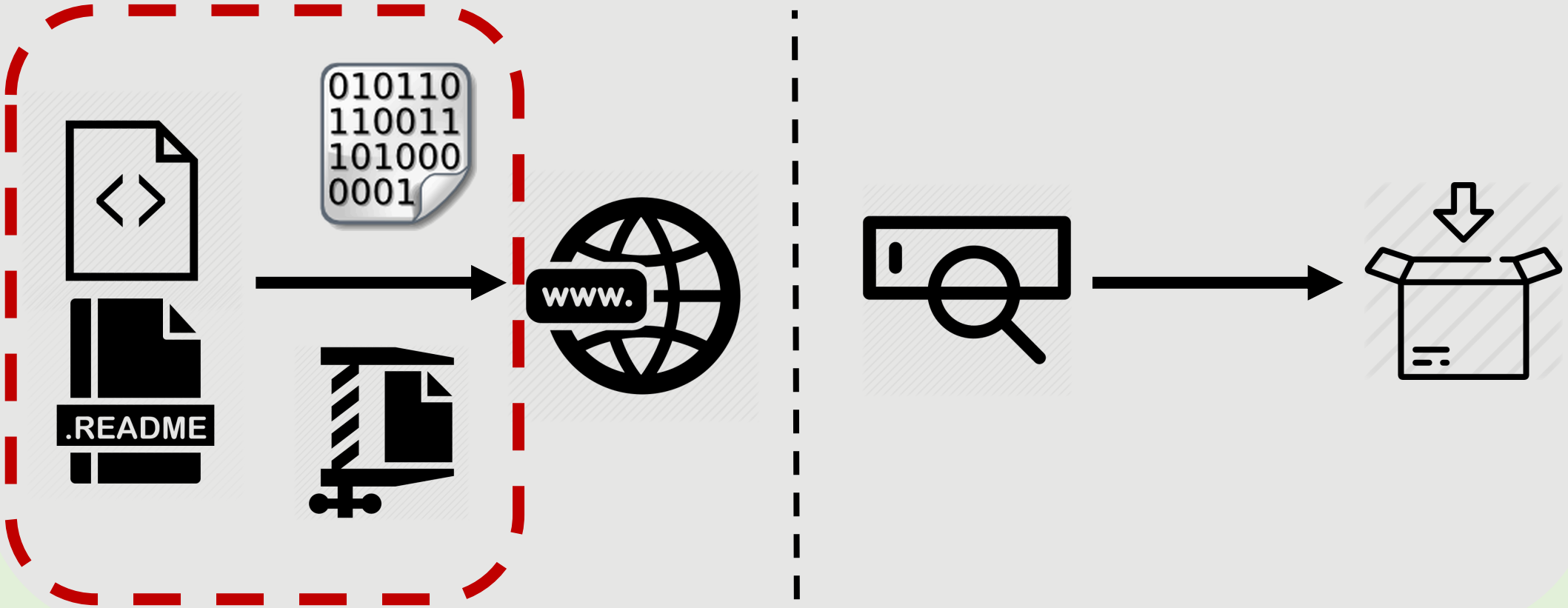
Isolated Environments (Build / Test / Release):
tox, virtualenv (2007), venv@PEP405 (2011)




10 YEARS LATER


Python 1.0
Python 2.0
Python 3.0
setuptools
easy_install
pip
pipenv
PyPI
tox
wheel
poetry
disutils (egg)
virtualenv
bento/twine
PyPI warehouse
CONDA
1991
2000
2003
2005
2006
2007
2008
2011
2012
2017
RFC822
PEP241
PEP345
PEP314
PEP426
PEP566
PEP459
PEP262
PEP376
PEP427,440
PEP405
PEP394
PEP582

Deferred Metadata 2.0: PEP426 (2012), PEP459 (2013)
Metadata 2.1: PEP566 (2017/Dec)



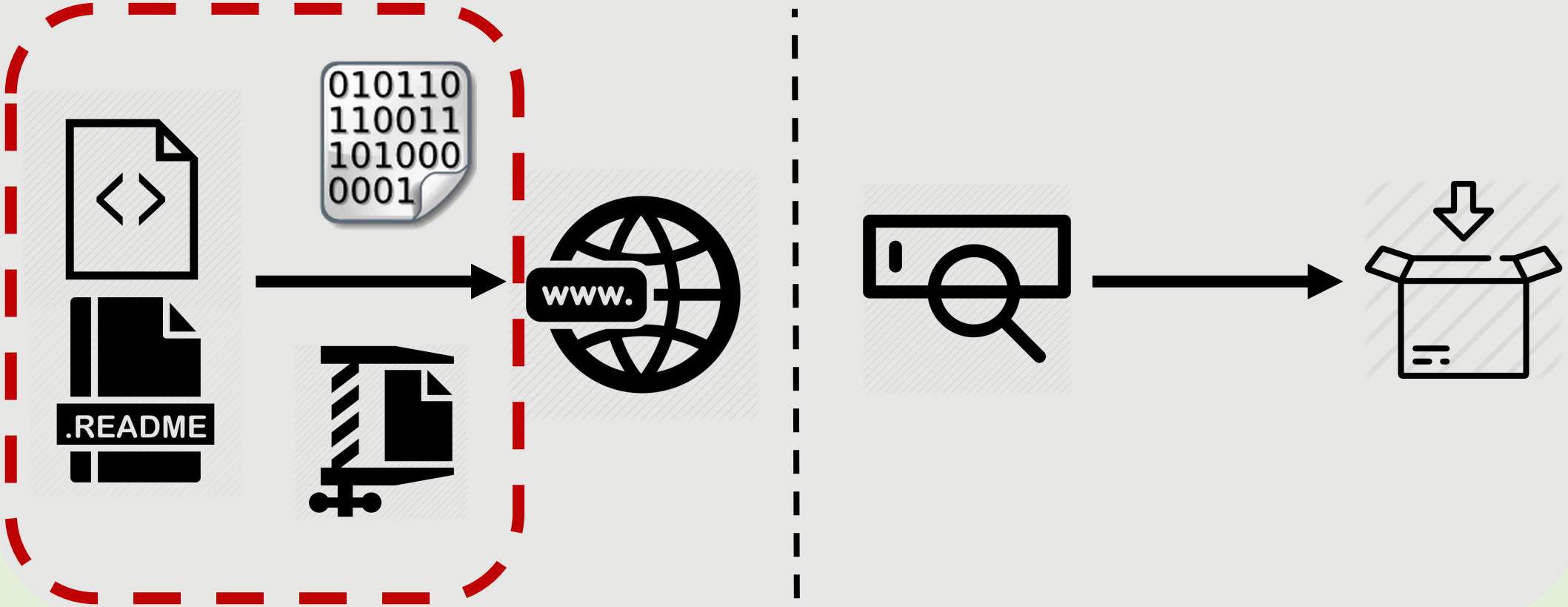
PEP566: pyproject.toml

```
[tool.poetry]
name = "Snake-DeepLearning"
version = "0.1.0"
description = ""
authors = ["Kir Chou"]

[tool.poetry.dependencies]
python = "^3.7"
numpy = "^1.18.3"
keras = "^2.3.1"
tensorflow = "^2.1.0"
tensorflow-estimator = "2.1"
# See README.md, pygame should be installed manually
# pygame = "^1.9.6"

[tool.poetry.dev-dependencies]

[tool.poetry.scripts]
game = "snake_deeplearning.game:main"
generate_data = "snake_deeplearning.training_data_generator:main_generate_data"
generate_model = "snake_deeplearning.training_data_generator:main_generate_model"
test_model = "snake_deeplearning.training_data_generator:main_apply_model"
test_genetic = "snake_deeplearning.training_data_generator:main_apply_genetic_algorithm"

[build-system]
requires = ["poetry>=0.12"]
build-backend = "poetry.masonry.api"
```

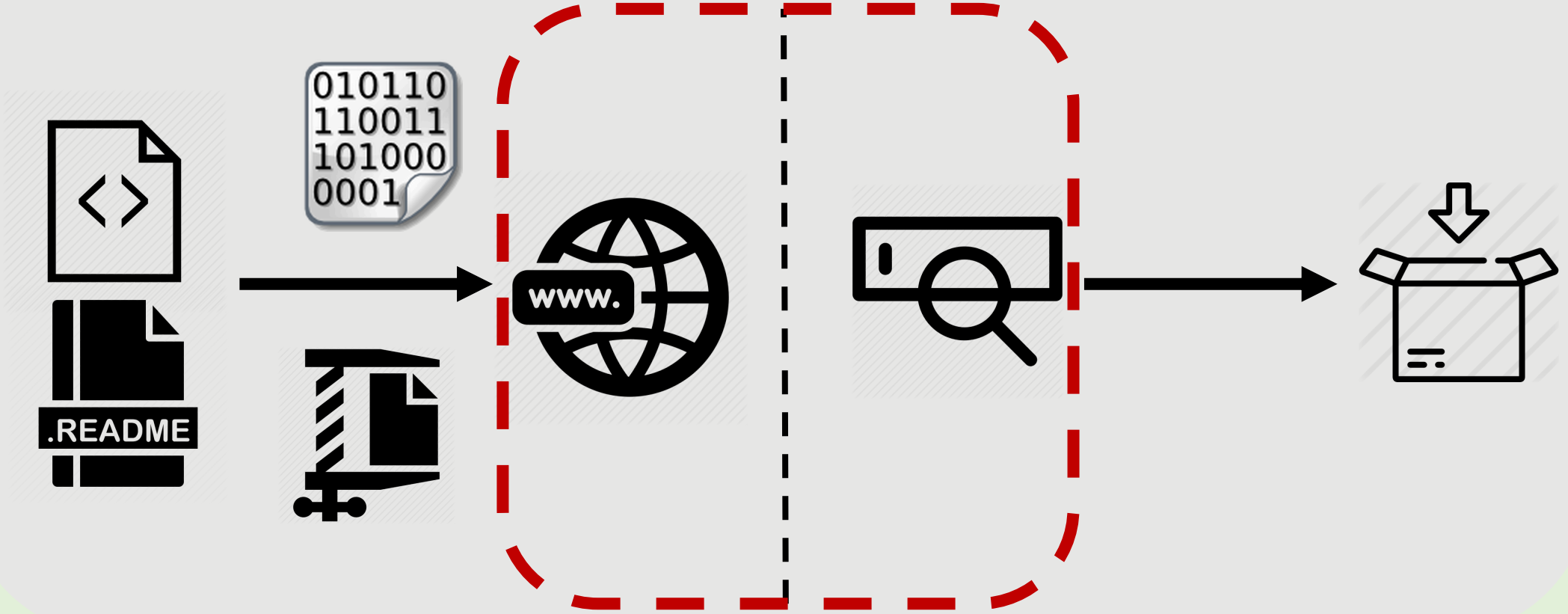
Version & Dependency:

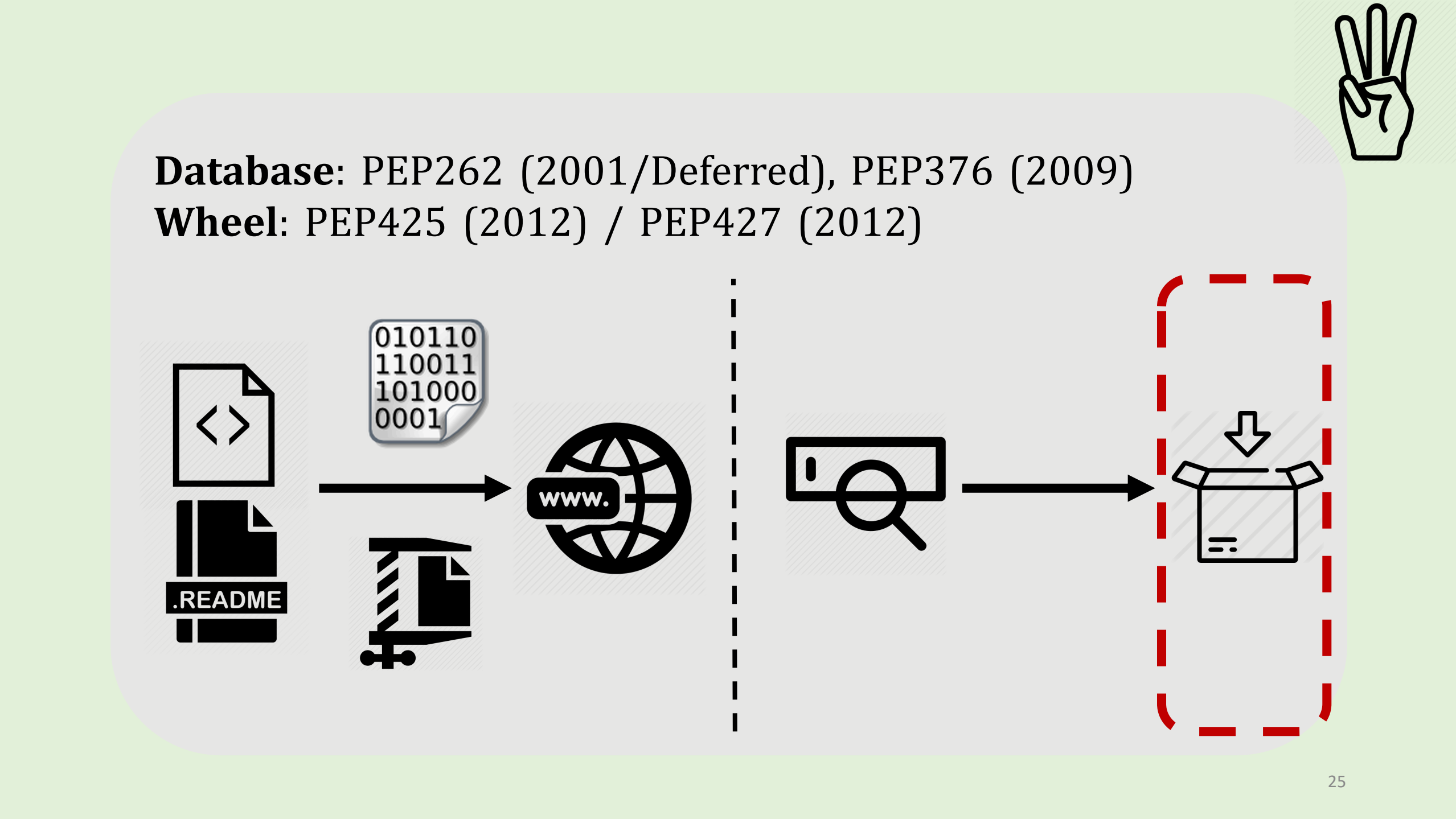
PEP386 (2009/Superseded), pip-tools/pip-sync (2012)

PEP440 (2013), PEP508 (2015)



Package installer (2008): pip
Publish Tool (2013): bento / twine / flit
PyPI (2018): PyPI warehouse




Database: PEP262 (2001/Deferred), PEP376 (2009)
Wheel: PEP425 (2012) / PEP427 (2012)
010110
110011
101000
0001
.README
www.

PEP376
Database of Installed Python Distributions

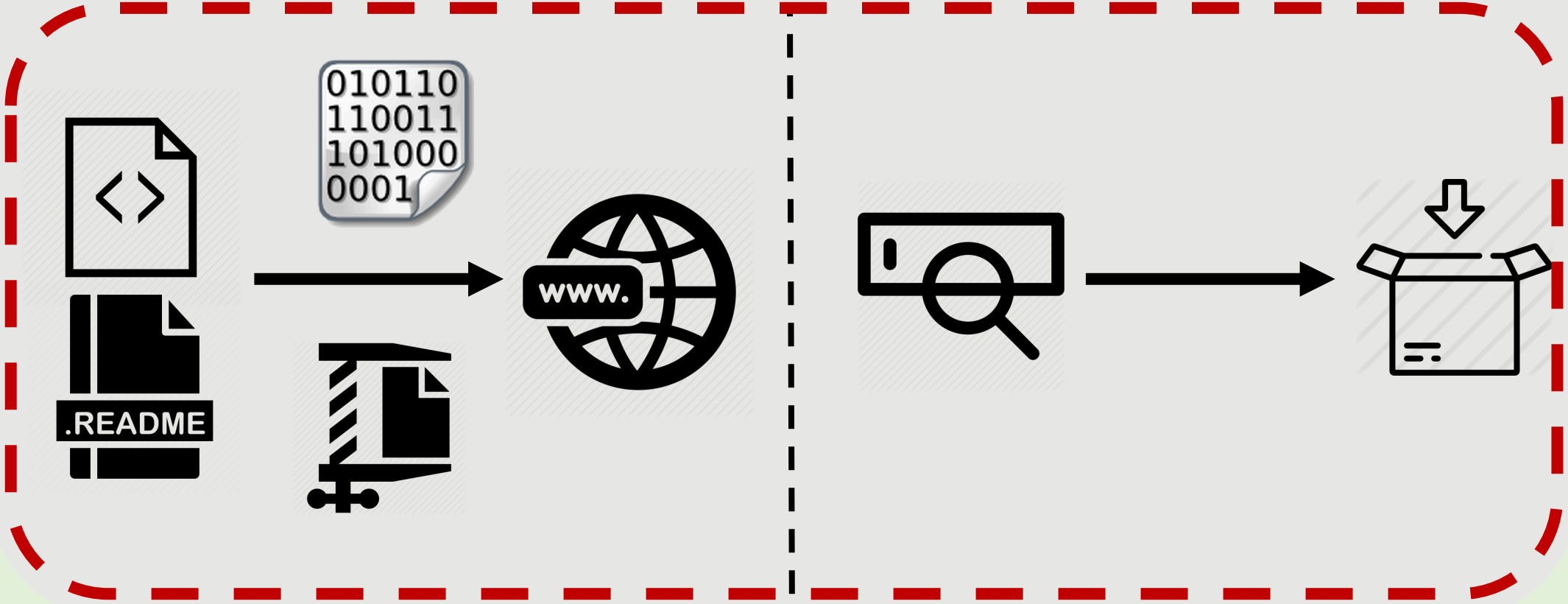
```
>>> import pkg_resources
>>> [k for k in dir(pkg_resources) if k.startswith('get')]
['get_build_platform', 'get_cache_path', 'get_default_cache', 'get_distributio
n', 'get_entry_info', 'get_entry_map', 'get_importer', 'get_platform', 'get_pr
ovider', 'get_supported_platform']
>>> pkg_resources.get_distribution('helloworld')
helloworld 0.1 (/Users/kirchou/test_lib/py38/lib/python3.8/site-packages/hello
world-0.1-py3.8.egg)
>>> pkg_resources.get_platform()
'macosx-10.14-x86_64'
```

xxx.dist-info/WHEEL

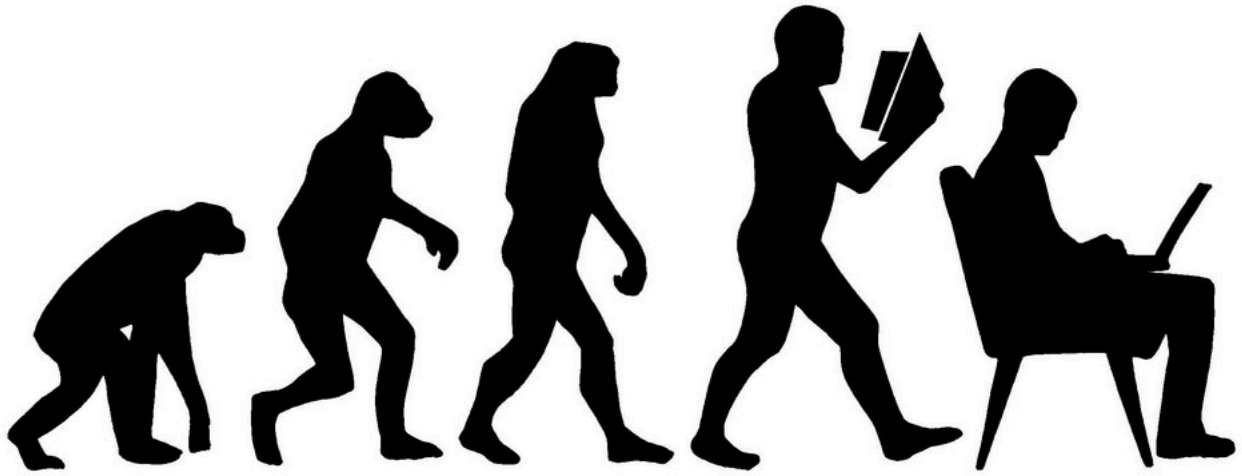
```
Wheel-Version: 1.0
Generator: bdist_wheel (0.34.2)
Root-Is-Purelib: true
Tag: py3-none-any
```

One For All: Conda / Pipenv / Poetry

PEP517 (2015), PEP518 (2016)

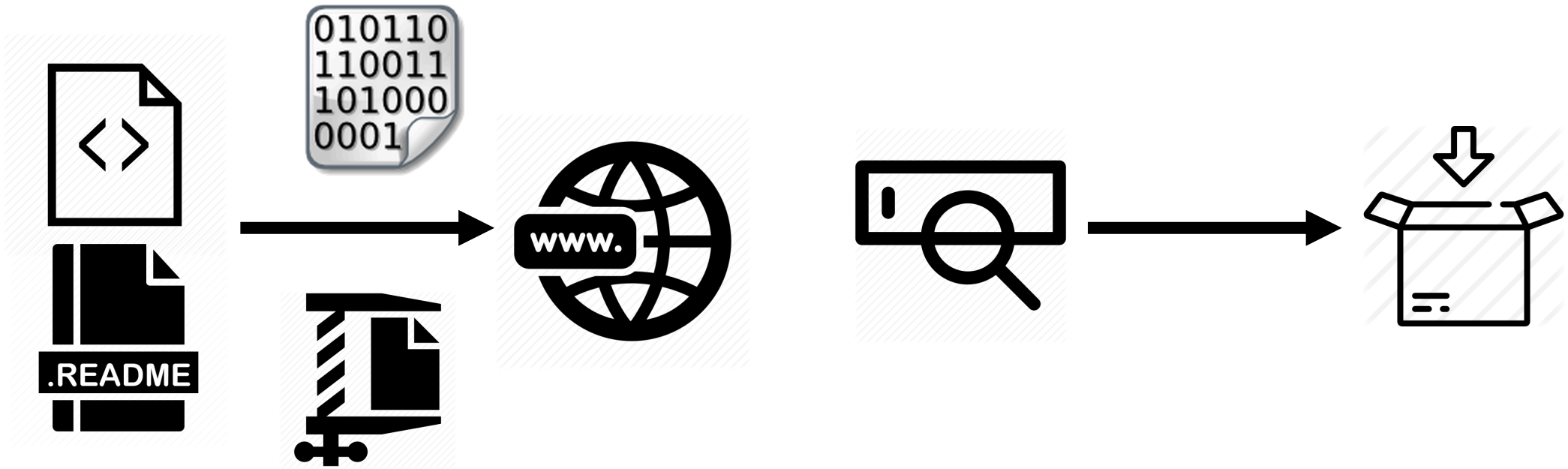


Learn from history



1. Metadata Schema

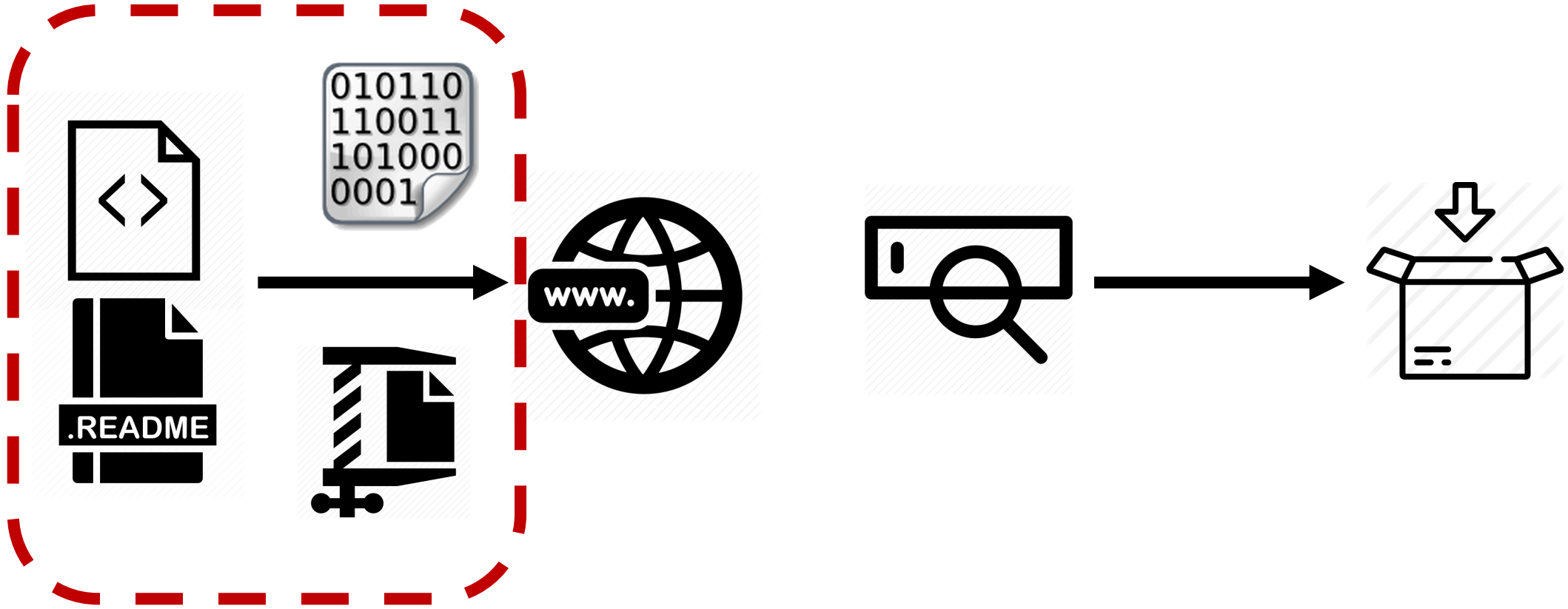
Evolution



Evolution

1. Metadata Schema

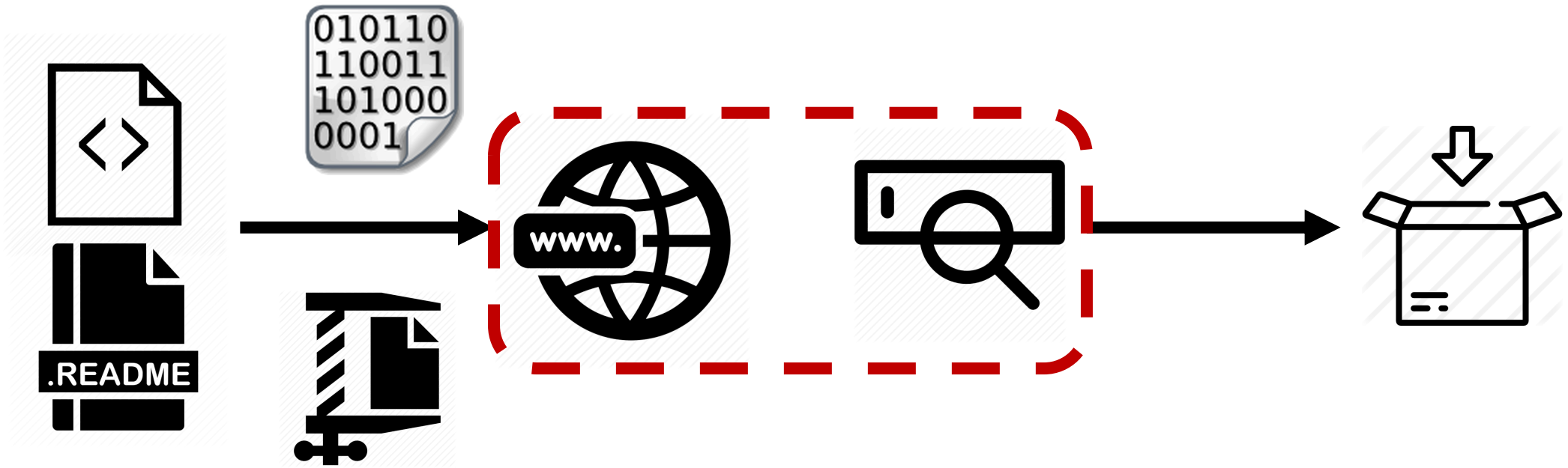
2. Build Tools



Evolution

1. Metadata Schema

2. Build Tools



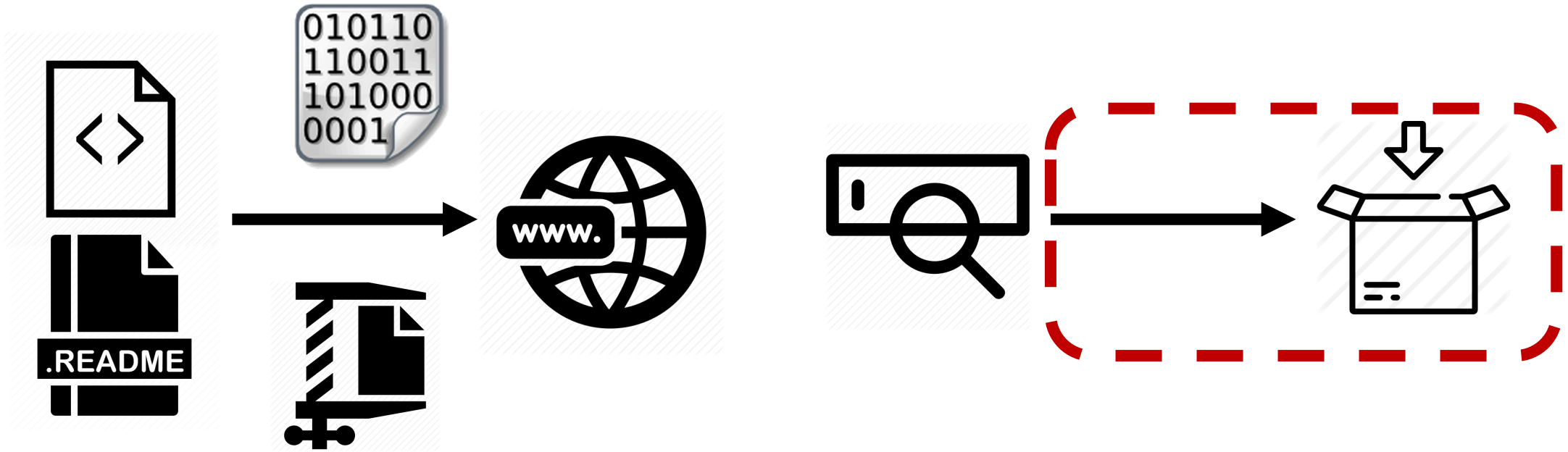
3. Publish Tools / Package Index

Evolution

1. Metadata Schema

2. Build Tools

4. Installer / Deployment Tools



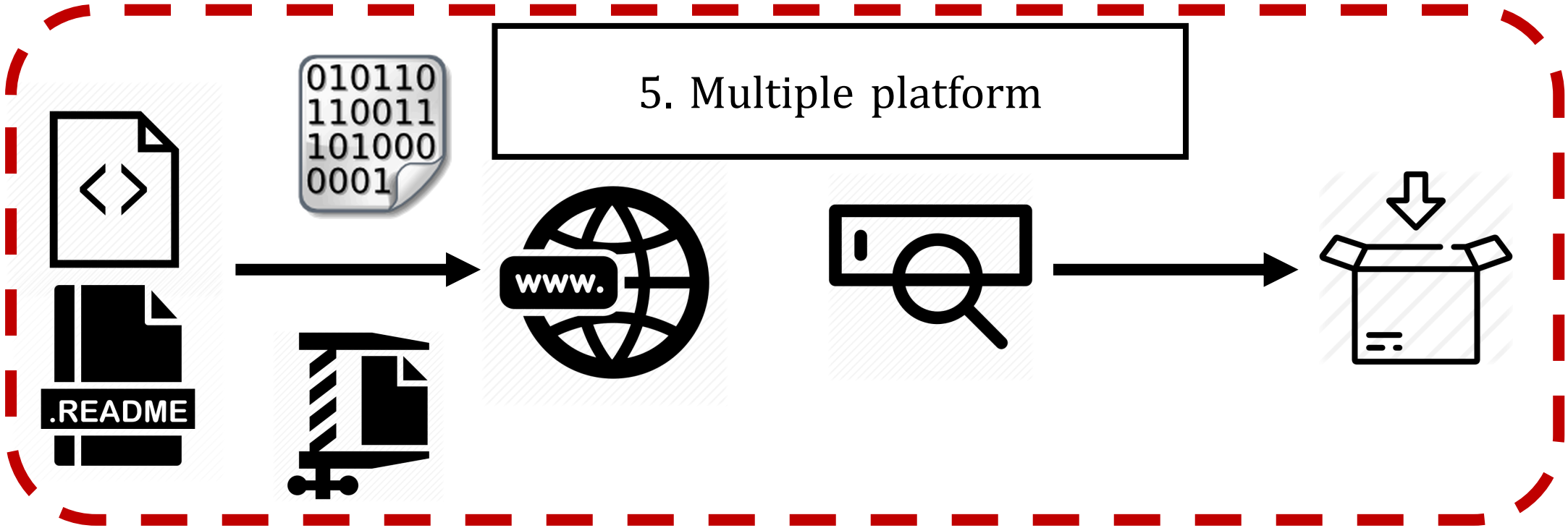
3. Publish Tools / Package Index

Evolution

1. Metadata Schema

2. Build Tools

4. Installer / Deployment Tools



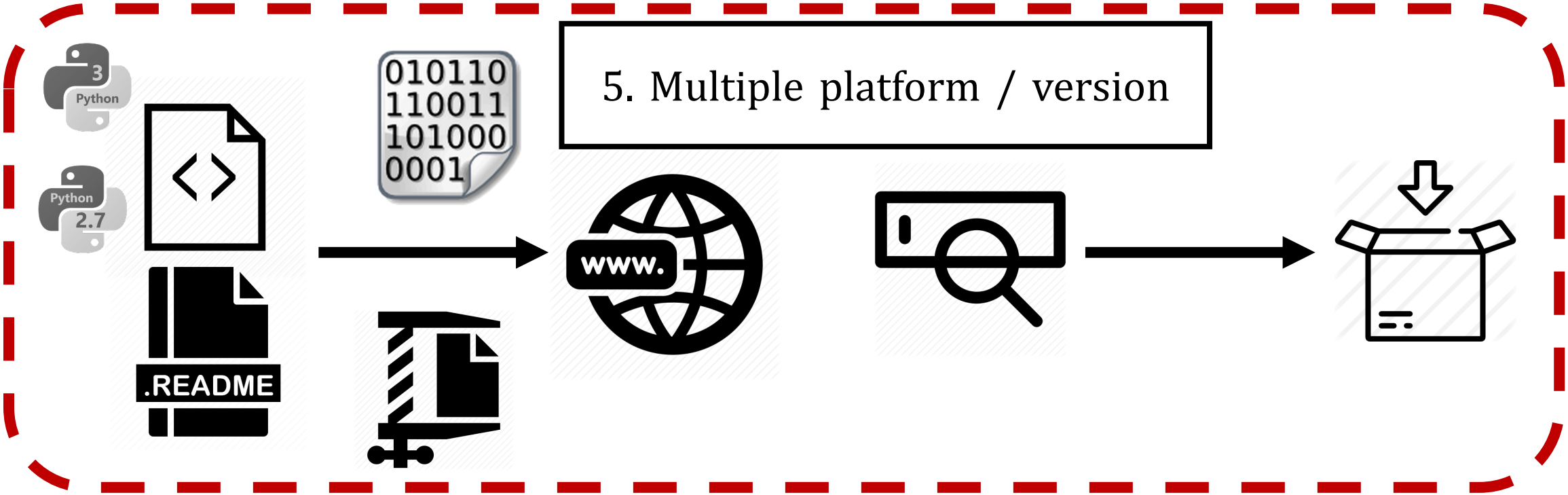
3. Publish Tools / Package Index

Evolution

1. Metadata Schema

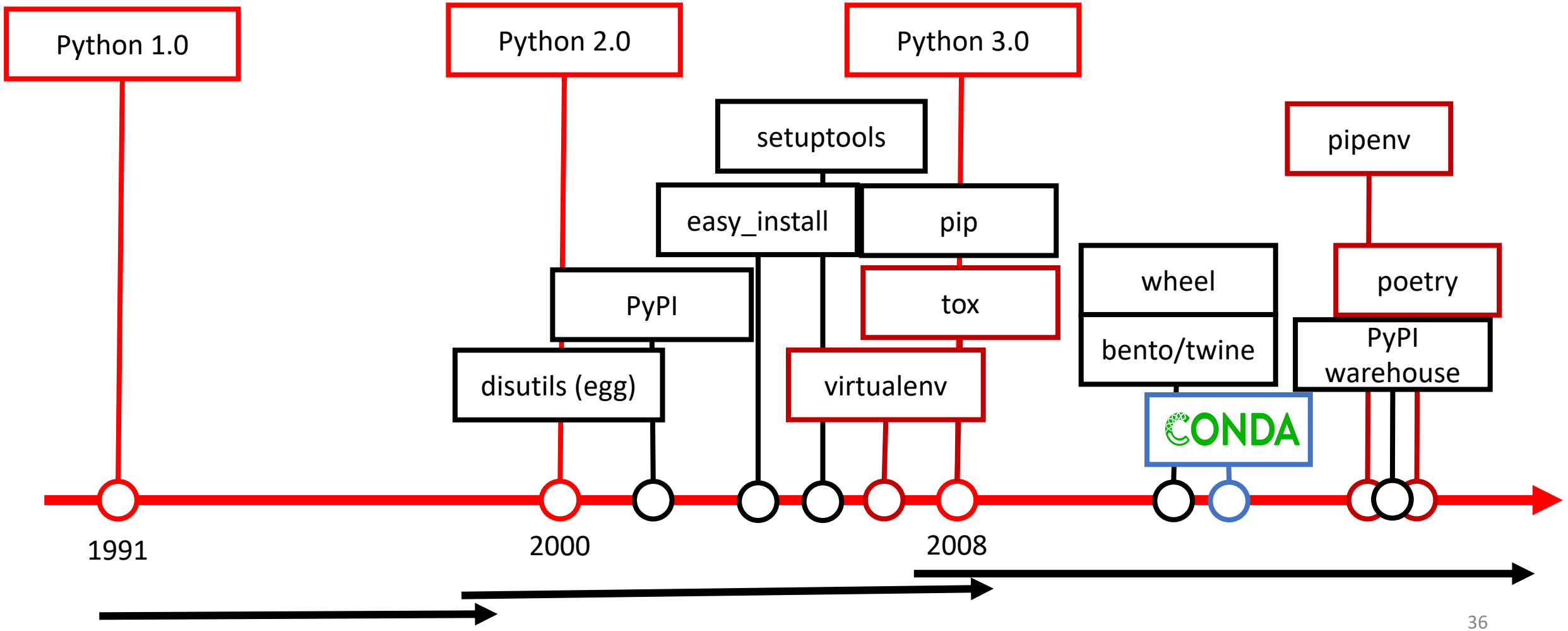
2. Build Tools

4. Installer / Deployment Tools



3. Publish Tools / Package Index

The pattern is repeated per generation

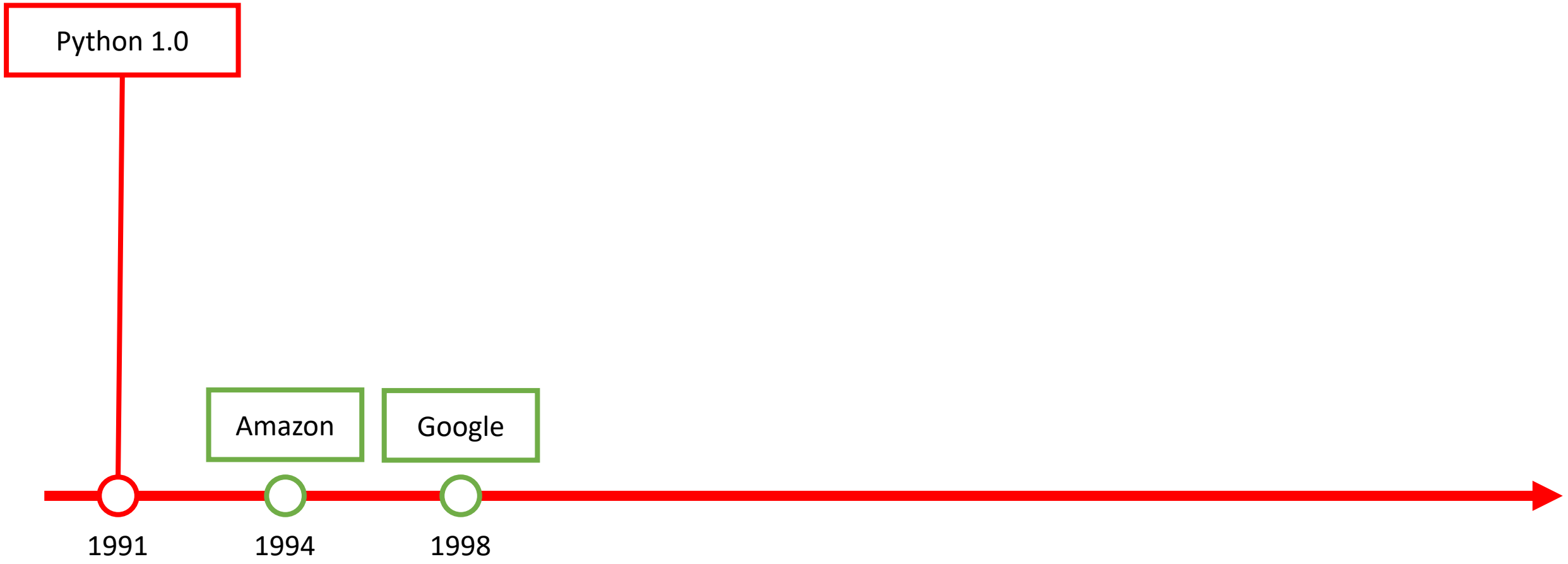


Challenge BEFORE community's python packaging solution

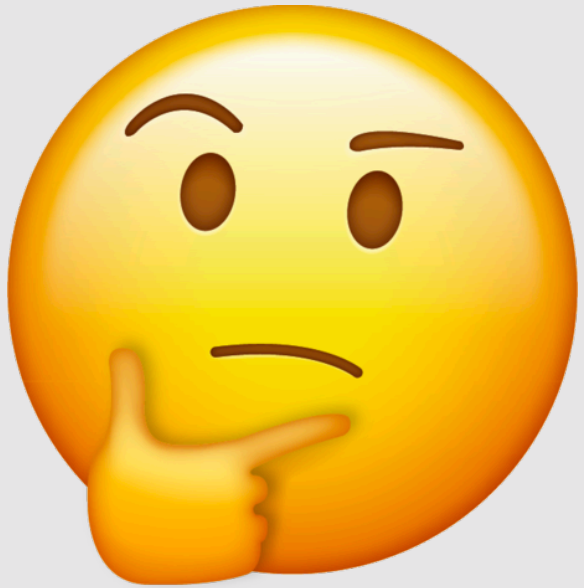
1994 Amazon

1998 Google

No Python packaging solution



How to give/take the work to/from others?



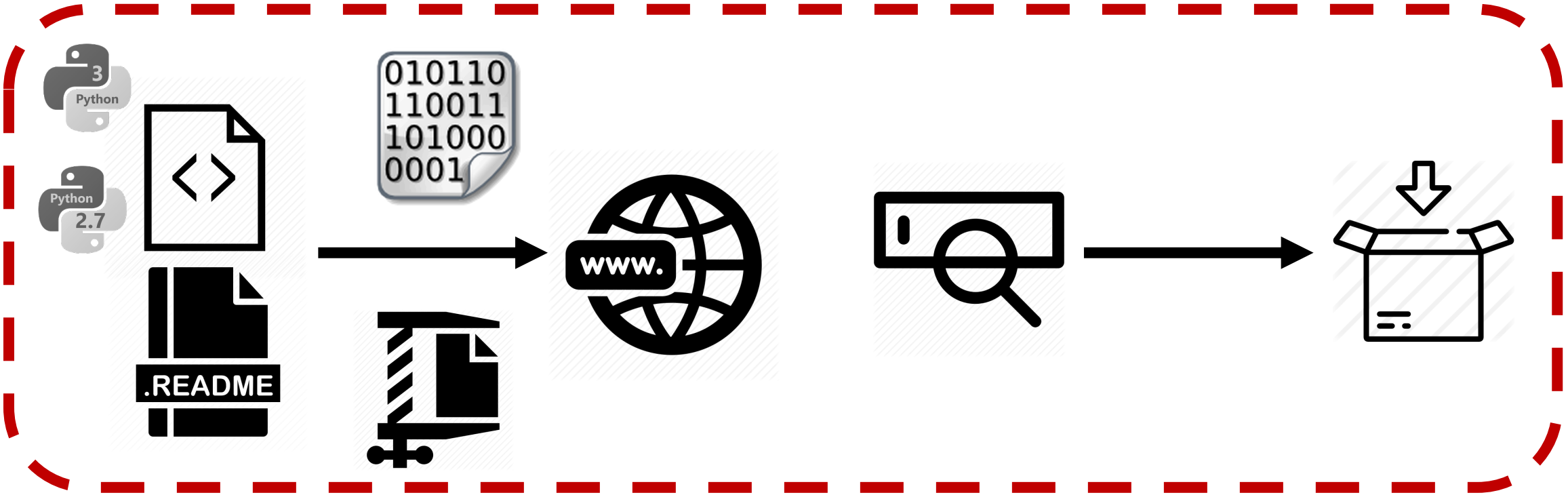
Solve same problems in history

- Build tool
  - build tool + dependency resolver
  - Support multiple Python version
- Publish tool / Private Package Registry
- Installer / Deployment tool
  - Support isolated environment
  - Support multiple platforms

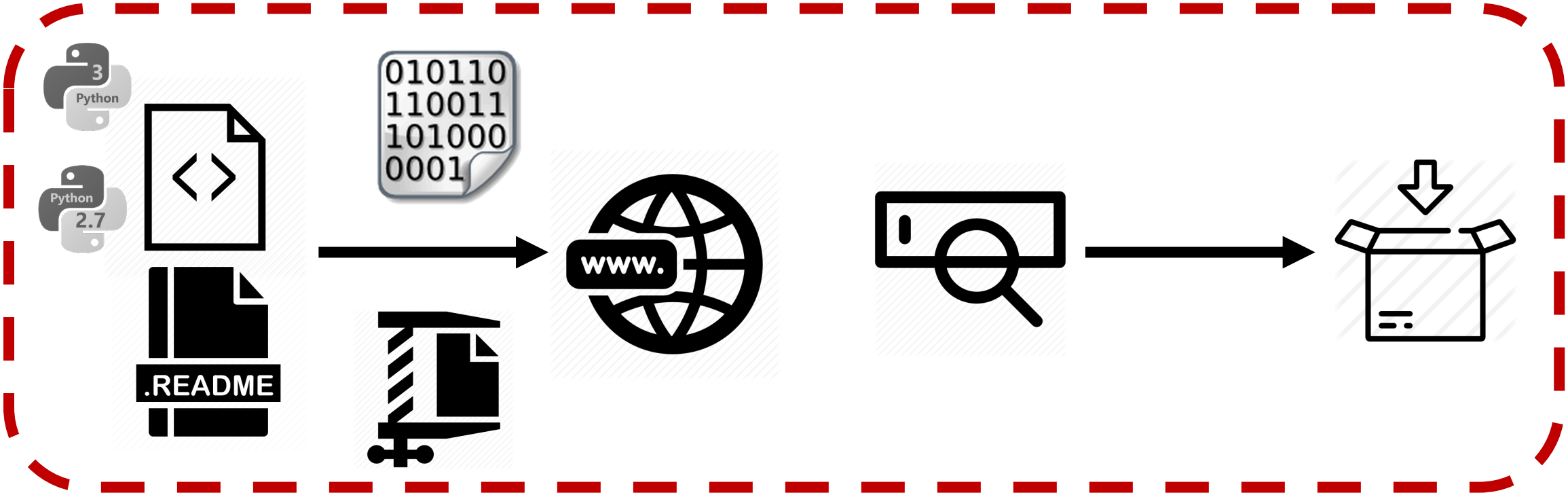
1. **Internal** Package Metadata

2. **Internal** Build Tools

4. **Internal** Deployment Tools



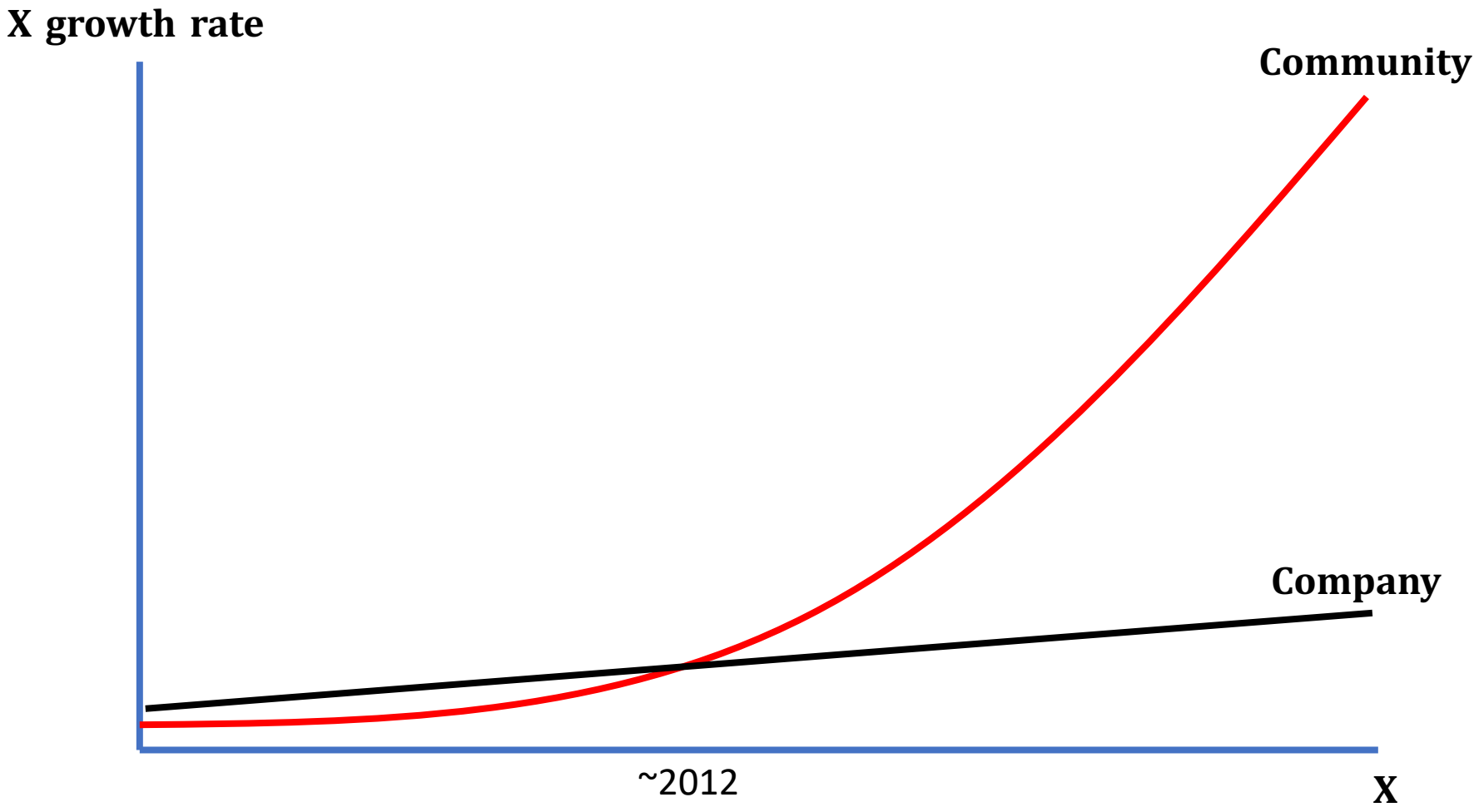
3. **Internal** Package Registry



3p packages are all handled by internal tools


BUILDER TOOL TEAM
imgflip.com

Community solutions grow faster now!



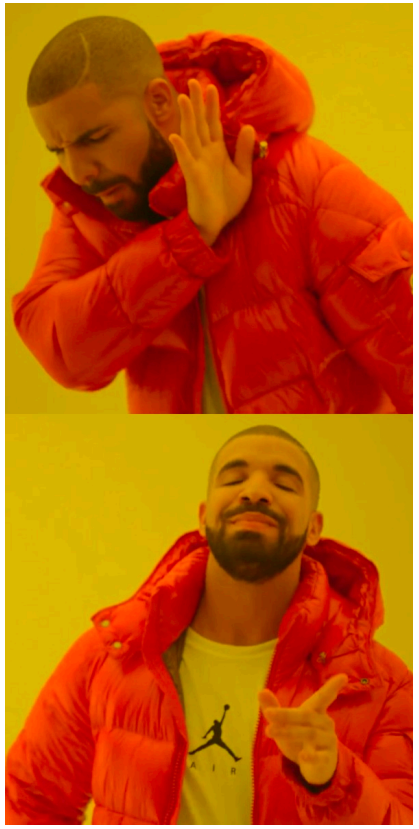
Packaging problem is mostly general!

- Build System
  - What is a Python package?
  - How does build tool resolve Python's dependencies?
  - How to build a Python package?
- Publish System / Private Package Registry
  - How to publish your Python package?
  - How to download other Python's dependencies?
- Deployment System
  - How to isolate your Python environment?
  - Which platform does the Python package support?

Shift to community solution

Challenge AFTER community's python packaging solution

1. Ecosystem



Community's solution → (one way) Company's solution

Community's solution ↔ (two way) Company's solution


THE BANK
FINE GOLD 999.9
NET WT 1000g

Cloud Services Business

Cloud Service Provider

Pay $

Developer

Adopt new community solution

Invent New solution

Sponsorship

One for all, all for one

Community

2. More standards ∝ More tools

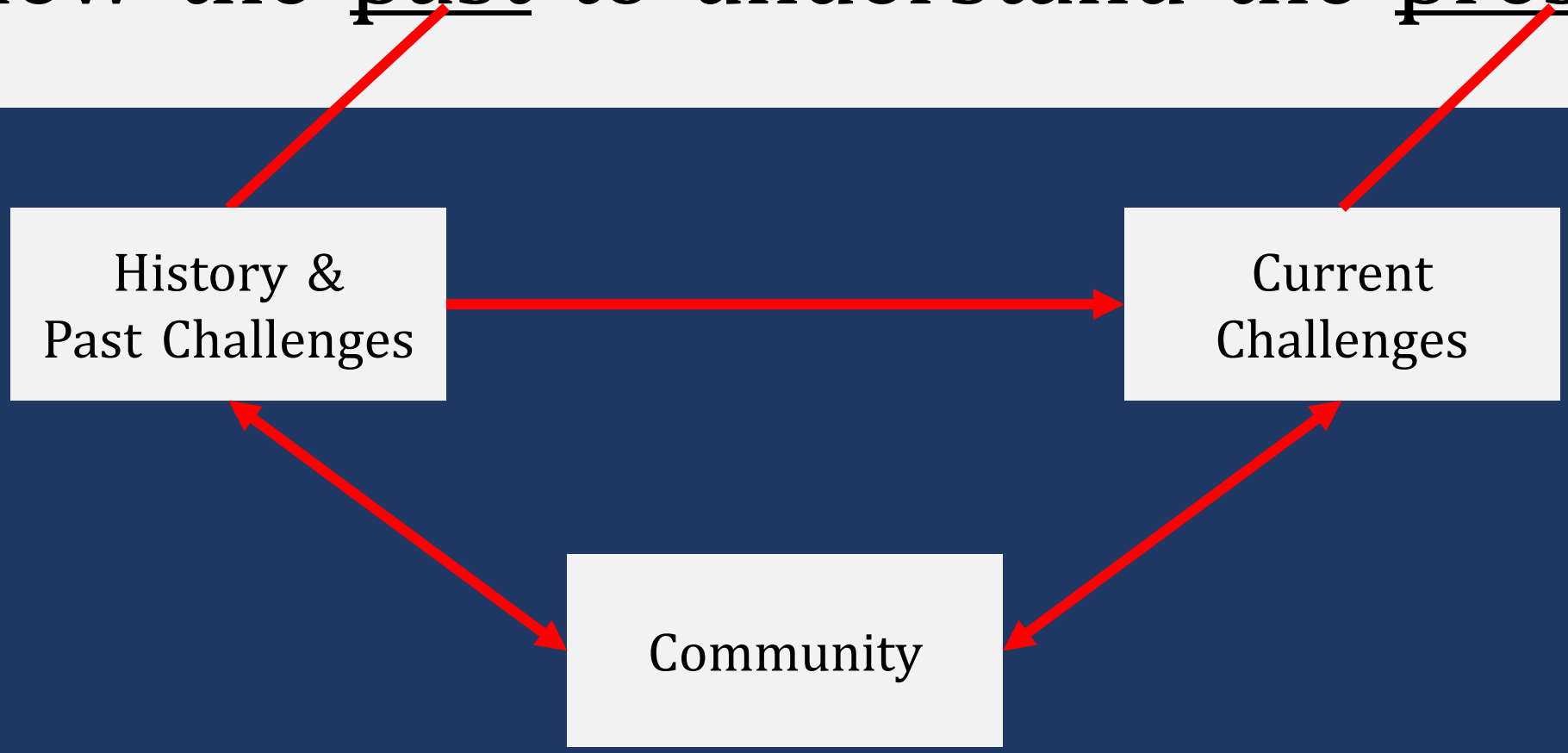

dephell
pipenv
poetry
setuptools
pip


flit
bento
JAKE-CLARK.TUMBLR

Join PyCon! Join the community!

Takeaway

Review the past to understand the present



Thanks!

Appendix

History

- Dustin Ingram @ SciPy 2018
  Inside the Cheeseshop: How Python Packaging Works
- History of packaging written by Martijn Faassen
- Kenneth Reitz @ PyCon 2018
  Pipenv: The Future of Python Dependency Management
- Clinton Roy @ Kiwi PyCon X (2019)
  The Packaging Lifecycle with Poetry
- History of wheel written by Daniel Holth

Packaging

- Dave Forgac @ PyOhio 2015
  Python Packaging from Init to Deploy

- Elana Hashman @ PyCon 2019
  The Black Magic of Python Wheel

- Official Document: Packaging binary extensions (2013)

Runtime (Virtual) Environment

- Carl Meyer @ PyCon 2011
  Reverse-engineering Ian Bicking's brain: inside pip and virtualenv

- Bernat Gabor @ EuroPython 2019
  Status quo of virtual environments

Why did 3 big cloud platforms provide **Package Registry** service JUST recently?

2019 May

2020 June

2020 July

Google Cloud